Human-Robot Collaborative Learning Methods

Thesis submitted in partial fulfillment
of the requirements for the degree of
"Doctor of Philosophy"

by
Uri Kartoun

Submitted to the Senate of
Ben-Gurion University of the Negev

2007

BEER - SHEVA

Human-Robot Collaborative Learning Methods

Thesis submitted in partial fulfillment
of the requirements for the degree of
"Doctor of Philosophy"

by
Uri Kartoun

Submitted to the Senate of
Ben-Gurion University of the Negev

Approved by the advisor	Prof. Helman Stern	____________
Approved by the advisor	Prof. Yael Edan	____________
Approved by the Dean of The Kreitman School of Advanced Graduate Studies		____________

2007

BEER - SHEVA

This work was carried out under the supervision of

Prof. Helman Stern

Prof. Yael Edan

In the

Department of Industrial Engineering and Management

Faculty of Engineering Sciences

Acknowledgments

Very special thanks go to my supervisors **Prof. Helman Stern** and **Prof. Yael Edan**. Without their motivation and encouragement I would not have completed this research. With their valuable insights and advices Helman and Yael are educators who truly made a difference in my life.

It has been their tutelage enthusiasm that inspired me to turn to the field of machine learning and robotics. Their infinite patience, understanding and professional guidance led me steadily and securely through the research and the entire period of the Ph.D. thesis, and for this I owe them my deepest gratitude.

I would also like to thank the *Department of Industrial Engineering and Management* whose members over the past decade have worked to promote excellence in teaching. Specifically I would like to thank **Prof. Joachim Meyer** and **Prof. Vladimir Gontar**.

I also owe many thanks to the people at the *Department of Industrial Engineering and Management* I've had the pleasure to work for over the past years, **Prof. Ehud Menipaz**, **Prof. Nava Pliskin**, and **Prof. Gadi Rabinowitz**.

I would like to express my appreciation to **Dr. Sigal Berman** whose expertise understanding added considerably to my graduate research. I appreciate her vast knowledge and skills in robotics.

Appreciation also goes to **Dr. Amir Shapiro** from the *Department of Mechanical Engineering* at the *Ben-Gurion University of the Negev* for his support and discussions.

I must express my sincerest gratitude to my friend **Dr. Juan Wachs** for his suggestions, exchanging of knowledge and helpful skills.

My thanks to **Prof. Mark Nagurka**, my colleague from *Marquette University*, who with patience and good humor exposed me to new ideas.

I would like to gratefully acknowledge the technical support of **Nissim Abuhatzira**, **Rubi Gertner**, and **Yossi Zehavi** from the *Department of Industrial Engineering and Management* and to **Uri Naim** from *KNT Engineering*.

My thanks also to **Mouki Cohen** and **Ariel Plotkin** who provided computing support services during the course of the research.

I am grateful for mathematical suggestions, comments, and contributions from the following friends and colleagues: **Prof. Israel David**, **Dudi Gabay**, **Shlomo Kashani**, **Prof. Ephraim Korach**, **Dr. Ofer Levi**, **Nir Naor**, **Dr. Luba Sapir**, **Prof. Edna Schechtman**, **Uri Shaham**, and **Dr. Dvir Shabtay**.

I would also like to thank my friends in the *Telerobotics Lab*: **Amit Gil**, in the *Intelligent Systems Lab*: **Yoash Chasidim**, **Nir Friedman**, **Olga Grechko**, and **Oren Shapira**, at the *Computer Integrated and Manufacturing Lab*: **Hadasa Blum**, **Dr. Ofir Cohen**, **Keren Kapach**, **Dr.**

Alexander Krein, **Dr. Shahar Laykin**, **Greg Piltz**, **Yael Salzer**, and **Gil Shayer**, and from the *Ergonomics Lab*: **Noam Ben Asher**, **Avinoam Borowsky**, **Pnina Gershon**, and **Dr. Tal Oron**.

I am greatly indebted to the office staff for all the instances in which their assistance helped me along the way, **Iris Abramovitz**, **Albina Bulgaro**, **Nilly Manin**, **Ayelet Mark**, **Ilanit Sarder**, and **Dorit Vazana**.

Although they were not directly involved in the project, I would also like to acknowledge **Mor Atlas**, **Dr. Peter Bak**, **Moran Beeri-Goshen**, **Dr. Avital Bechar, Boris Efros**, **Paul Erez**, **Haim Moyal**, **Dr. Danny Ratner**, and **Dr. Moshe Zofi**.

I recognize that this research would not have been possible without the financial assistance of the *Paul Ivanier Center for Robotics Research and Production Management*, *Ben-Gurion University of the Negev* and by the *Rabbi W. Gunther Plaut Chair in Manufacturing Engineering*.

Special thanks go also for the people I worked with while I was an informatics fellow at the *Institute for Medical Informatics* at the *Washington Hospital Center* (currently Microsoft), **Dr. Craig Feied**, **Dr. Mark Smith**, **Dr. Jonathan Handler**, **Dr. Michael Gillam**, **Robert Irving**, and **Peter Delate**.

A person who I owe her many thanks is **Simone Gil**, my good friend from the *Biology Department* who shared with me many evenings full with delightful discussions.

In conclusion I would like to thank my family for the support they provided me through my entire life and in particular over the period of accomplishing this work. To my grandmother, **Zila Sawitsky**, my parents, **Aviva** and **David Kartoun**, my sister **Talia**, her husband **Ran** and their first child **Yoav Diskin**. In particular I owe my father, **David**, my eternal gratitude for all of his support.

Uri Kartoun
Ben-Gurion University of the Negev
Beer Sheva, 2007

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where," said Alice.
"Then it doesn't matter which way you go," said the Cat.
"…so long as I get somewhere," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."[1]

[1] Carroll L., "Alice's Adventures in Wonderland", 1865.

Table of Contents

List of Appendixes

List of Figures

List of Tables

Acronyms[1]

RL	Reinforcement learning
SVMs	Support vector machines
HO	Human operator
DP	Dynamic programming
CQ(λ)	Collaborative *Q(λ)*
HRI	Human-robotic interfaces
OSH	Optimal separating hyperplane
PCA	Principle component analysis
RBF	Radial basis function
NN	Neural network
SA	Semi-autonomous mode
A	Autonomous mode
S	State space
A	Action space
$s_t \in S$	State at time step t
$s_{t+1} \in S$	State at time step $t+1$
$a_t \in A$	Action at time step t
$a_{t+1} \in A$	Action at time step $t+1$
$r(s_t, a_t)$	Reward at time step t
Λ	A minimum acceptable performance threshold above which the human is called to intervene
ω	A maximum acceptable performance threshold in terms of mean number of steps to reach a goal.
N	Number of most recent learning episodes
S_i	Indicates whether a policy was successful
L_{ave}	A moving average learning performance measure in terms of rewarded policies
T_{ave}	A moving average learning performance measure in terms of number of steps required to reach a goal
α	Learning rate
γ	Discount factor
$e(s_t, a_t)$	Eligibility trace
λ	Eligibility trace factor
δ	Temporal difference error
Q_c	Collaborative learner
$W \times W$	Grid world dimensions
T	Shaking time of a bag
O	Number of items that fell from a bag during a shaking operation
$\overline{R}$	A reward threshold for a successful episode
R_i	the reward achieved for the i^{th} learning episode
W_j	Weight measured on a digital scale at time j
w	Weight of one inspected object
c	A positive constant to adjust the reward
N_{t_o}	Convergence to optimality performance measure
K	Number of learning processes in a multi-agent system (*e.g.*, robots).
$N_{t_{no}}$	Convergence to near optimality performance measure
L_i	Number of steps a learning agent performs at the i^{th} trial and gets a reward

[1] To improve the flow of the reading the abbreviations were not always used.

Abstract

To accelerate the use of robots in everyday tasks they must be able to cope with unstructured, unpredictable, and continuously changing environments. This requires robots that perform independently and learn both how to respond to the world and how the world responds to actions the robots undertake.

One approach to learning is reinforcement learning (RL), in which the robot acts via a process guided by reinforcements from the environment that indicate how well it is performing the required task. Common RL algorithms in robotic systems include Q and its variation $Q(\lambda)$-learning, which are model-free off-policy learning algorithms that select actions according to several control policies. Although Q and $Q(\lambda)$ learning have been used in many robotic applications, these approaches must be improved. Their drawbacks include: (i) extremely expensive computability, (ii) large state-action spaces, and (iii) long learning times (until convergence to an optimal policy).

This thesis presents a new collaborative learning algorithm, denoted the $CQ(\lambda)$ algorithm, that is based on the $Q(\lambda)$-learning algorithm. The $CQ(\lambda)$-learning algorithm was developed, tested and applied for two frameworks: (i) learning by multiple agents, and (ii) learning by human-robot systems. In the **first** framework, collaboration involves taking the maximum of state-action values, *i.e.*, the Q-value, across all learning agents at each update step. In the **second** framework, two levels of collaboration are defined for a human-robot learning system: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - a human operator (HO) guides the robot to take an action or a policy and the robot uses the suggestion to replace its own exploration process. The key idea here is to give the robot enough self awareness to adaptively switch its collaboration level from autonomous (self performing) to semi-autonomous (human intervention and guidance). This awareness is represented by a self test of its learning performance. The approach of variable autonomy is demonstrated in the context of an intelligent environment using mobile and fixed-arm robots.

Extensive experimentation with different robotic systems in a variety of applications demonstrated the strengths and weaknesses of the algorithm. Applications specifically developed for testing the $CQ(\lambda)$-learning algorithm are demonstrated in the context of an intelligent environment using a mobile robot for navigation and a fixed-arm robot for the inspection of suspicious objects. The results revealed that $CQ(\lambda)$ is superior over the standard $Q(\lambda)$ algorithm. The suggested learning method is expected to reduce both the number of trials needed and the time required for a robot to learn a task.

Methodology

The $CQ(\lambda)$-learning algorithm was developed, tested and applied for two frameworks:

1) Learning by multiple agents

The $CQ(\lambda)$ learning algorithm for multiple agents is based on a state-action value of an agent or learning process is updated according to the best performing agent; collaboration is in taking the maximum of state-action values, *i.e.*, the Q-value, across all learners at each update step. By applying this method, the Q value for a collaborative learner will be the best value.

2) Learning by human-robot systems

In this framework, two levels of collaboration are defined for a human-robot learning system: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - a HO guides the robot to take an action or a policy and the robot uses the suggestion to replace its own exploration process. The key idea here is to give the robot enough self awareness to adaptively switch its collaboration level from autonomous (self performing) to semi-autonomous (human intervention and guidance). Here it is assumed that the learning system consists of one robot and one HO. The robot learns to perform a task by using a standard $Q(\lambda)$-learning function. During its learning it keeps measuring its learning performance. This is done by defining Λ, a minimum acceptable performance threshold above which the human is called to intervene. The measure Λ is compared with L_{ave}, a moving average learning performance measure over the last N most recent learning episodes considered. Based on this learning performance threshold, the robot switches between fully autonomous operation and the request for human intervention. The procedure is repeated M times where M (set *a-priori*) is the maximal number of learning episodes.

A theoretical analysis is presented for both frameworks. For the multiple agents' framework where a system consists of one collaborative agent, Q_C and several independent Q-learners, it was shown mathematically that the learning function of the collaborative agent converges faster than those of the independent agents. For the human-robot framework, a theoretical discussion was conducted to show that collaborative Q-learning is a special case of Q-learning and therefore will also converge to optimal solution with probability one.

The $CQ(\lambda)$-learning algorithm presented in this thesis was evaluated in three systems that were specially developed in this thesis:

1) Navigation of multiple robots (simulation)

The system consists of several mobile robots represented in a simulation model. The robots learn to navigate a two dimensional world that contains undesirable areas choosing the optimum path to reach a target. A learning system consists of one collaborative robot denoted Q_c and one or more independent $Q(\lambda)$-based learners. The $CQ(\lambda)$ algorithm is applied to autonomous mobile robot navigation where several robot agents serve as learning processes with the objective of choosing the optimum path to reach a target.

System performance was evaluated using the following measures: (i) $N_{t_{no}}$ - convergence to near optimality - mean of the last N path lengths, and (ii) N_{t_o} - convergence to optimality - number of learning episodes required to perform a policy optimally and repeat it an infinite number of times.

2) Human-robot collaboration for a bag shaking task

The system consists of a fixed-arm six degrees of freedom Motoman UP-6 robot, a bag that contains objects, a digital camera that provides visual feedback from the robotic scene to the HO interface, an inspection platform on which the inspected bag is manipulated and a digital scale for measuring rewards. The robot's learning task is to observe the position of the bag located on an inspection surface, grasp it and to learn how to shake out its contents in minimum time by interacting with the environment and by acquiring suggestions from a HO.

System performance was evaluated using: (i) average time to complete emptying the contents of a bag, (ii) average cumulative reward, *i.e.*, measures learning improvement, and (iii) human intervention rate, *i.e.*, a measure that represents the percentage of human interventions out of the total number of learning episodes; the lower it is, the more autonomous the robot is. Three reward functions were used to evaluate the learning system as follows: (i) linear reward function, (ii) cumulative-based reward function, and (iii) events-based reward function.

3) Human-robot collaboration for a mobile robot navigating a two dimensional world

The system consists of an Evolution Robotics ER-1 mobile robot equipped with a laptop and a camera. The robot task is to learn to navigate toward a target location in a two-dimensional world. The robot is located remotely from the HO. Under pre-defined system conditions, the robot decides to ask for human advice and guidance or to navigate autonomously. Learning is achieved by interaction with the environment and by acquiring suggestions from the HO. The purpose of the learning system is to let the robot start navigating from any starting location in the world and reach the target using the shortest path while avoiding undesirable areas.

System performance was evaluated using the following measures: (i) mean number of steps to optimally reach target, (ii) mean number of steps to feasibly reach target, and (iii) percent of human interventions - measuring how frequently a human collaborated with the robot.

Analysis and Results

Acceleration of learning using collaboration between several learning agents was demonstrated in simulations of the mobile robot navigation task. Fifty simulation runs showed an average improvement of 17.02% while measuring the number of learning episodes required reaching definite optimality and an average improvement of 32.98% for convergence to near optimality by using two robots compared with the $Q(\lambda)$ algorithm. Significant statistical difference was indicated for both convergence to optimality and convergence to near optimality while comparing two robots; the first uses $CQ(\lambda)$ and the second uses $Q(\lambda)$. While using three robots; the first uses the $CQ(\lambda)$ and the second and third use $Q(\lambda)$, it was found that there is no statistical significant differences in both convergence to optimality and convergence to near optimality while comparing the $Q(\lambda)$-based robots' learning performance. Statistical significance differences were found in both convergence to optimality and convergence to near optimality while comparing the $CQ(\lambda)$-based robot to the other two. Additionally, no statistically significant differences were observed for either convergence to optimality or convergence to near optimality using a $CQ(\lambda)$-based robot learning in either two robot or three robot environments. Superiority of the $CQ(\lambda)$ algorithm over the $Q(\lambda)$ was demonstrated for both setups. Further, it was shown that collaboration of an agent with additional two learning agents has no significant advantage over collaboration with only one learning agent.

From a learning rate[1] perspective it was shown that the independent agents show a better improvement of learning than the collaborative agent for both of the experimental setups described; the lower the learning rate the higher the improvement of learning is. Although the collaborative agent learns faster than the independent agents and reaches an optimal solution faster, the independent agents' improvement of learning is better. This is of course reasonable since the independent agents learn less efficiently than the collaborative agent at early stages of learning and since all agents (collaborative and independent) converge eventually to the same optimal solution (after many episodes) then the independent agents must "catch up" with the collaborative agent.

For the bag shaking task results showed that learning was faster when the HO was asked to intervene in the robot activity. Using a linear reward function, comparing $CQ(\lambda)$ with $Q(\lambda)$ over 25 learning episodes, indicated an improvement of 45.5% in the average reward while a HO intervened in 86.7% of the trials. Using a cumulative-based reward function and comparing $Q(\lambda)$-learning with $CQ(\lambda)$, the average time to complete emptying the contents of a bag decreased by 16.6% and the average reward achieved increased by 25.76%. The human intervention rate for the $CQ(\lambda)$-learning

[1] The learning rate parameter determines how significantly an agent improves. The reader should make a distinction between the learning rate evaluation performance measure which indicates the improvement of learning and α, the RL learning rate parameter.

experiment was 30%. For the events-based reward function, comparing $Q(\lambda)$-learning with $CQ(\lambda)$, the average time to complete emptying the contents of a bag decreased by 34.3% and the average reward achieved increased by 30.04%. The human intervention rate for the $CQ(\lambda)$-learning experiment was 20%. For all three reward functions either for $Q(\lambda)$ or $CQ(\lambda)$, the robot starts experiencing the environment by performing a random shaking policy over the X, Y, and Z axes. Intuitionally, vertical shaking would be best, but experiments determine that policies of shaking most of the time over the Y axis and with small number of actions over the X axis were the most effective. This policy caused the bag sometimes to become entangled, also due to the fact that most of the plastic bag weight is concentrated most of the time in one place. Possible explanation might be the type of the plastic bag knot; pulling it sideways makes it loose faster. Further, hypothetically, if a human would require shaking the bag, she could have seen visually the servo feedback to determine the optimal time to pull it up in a horizontal strategy, an ability that the robot system used here does not have.

To interpret the results achieved and to show that there were no subjective influences, a physical model of opening a plastic bag knot by a robot was developed. The model explains the results achieved for all three experimental setups. It was shown that acceleration is developed over time; thereby it is worthwhile to open the bag by activating forces continuously while holding locations as far as possible over the Y axis. Ideally, it is desirable to accelerate the robot arm at an acceleration that is close to the gravitational acceleration downwards and to oscillate it over the Y axis for overcoming of most of the friction forces.

For the mobile robot navigation task, significant improvements in comparison with the $Q(\lambda)$ algorithm (learning with no human intervention) were achieved by using the $CQ(\lambda)$ algorithm. In particular, for feasible and optimal solutions, improvements of 23.07% and 18.56% respectively were achieved for by using a collaboration threshold of $\Lambda = 8$ using $\gamma = 0.99$ and $\lambda = 0.75$ while a HO was asked to intervene in 30% of the robot navigational trials. In three variable autonomy experiments when the robot learned the environment, human collaboration rate decreased, as expected, with an increase in Λ. For the best significant improvement using $\gamma = 0.99$ and $\lambda = 0.75$, the combination of high γ with high λ values that achieved the highest learning performance can be explained due to choosing values of λ large enough to allow longer sequence of values of state-action pairs to be updated while keeping the computational solution to be achieved in reasonable time. In other experiments for various values of discount factors and eligibility traces, no consistency was found in achieving a solution that fits all of human-robot threshold collaboration levels. This may be attributed to cases when a human is collaborating with a robot to accelerate its learning performance. Here the intervention may impair the ability of the robot to explore the environment autonomously because its exploitation was enhanced on the account of less exploration by the human.

Conclusions

The main contribution of this work is in developing a new learning method. The proposed algorithm, denoted $CQ(\lambda)$ algorithm, enables collaboration of multiple agents in the learning process. Collaboration can expedite the learning by exploiting human intelligence and expertise.

Extensive experimentation with different robotic systems in a variety of applications demonstrated the strengths and weaknesses of the $CQ(\lambda)$-learning algorithm. Specific applications developed to serve as a test-bed for testing the $CQ(\lambda)$-learning algorithm were demonstrated in the context of an intelligent environment using a mobile robot for navigation and a fixed-arm robot for suspicious bags inspection. Results revealed the superiority of the $CQ(\lambda)$ over the standard $Q(\lambda)$ algorithm in the context of acceleration of learning performance of robotic systems.

Key words: Reinforcement learning, Robot learning, Human-robot collaboration

Publications

<u>Journal Papers (in preparation)</u>:

1. **Kartoun U.**, Stern H., Edan Y., A Human-Robot Collaborative Reinforcement Learning Algorithm.
2. **Kartoun U.**, Stern H., and Edan Y. Collaborative Reinforcement Learning Algorithm applied in Multi-Robot Framework.
3. Shapiro A., **Kartoun U.**, Stern H., Edan Y. Physical Modeling of a Bag Knot in a Robot Learning System.

<u>Reviewed Conference Papers</u>:

4. **Kartoun U.**, Stern H., Edan Y., Human-Robot Collaborative Learning System for Inspection. *IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 8 - Oct. 11, Taipei, Taiwan, 2006, Finalist for the Best Student Paper Competition (top 5 papers).

5. **Kartoun U.**, Stern H., Edan Y., Feied C., Handler J., Smith M. and Gillam M., Vision-Based Autonomous Robot Self-Docking and Recharging, *ISORA 2006 11th International Symposium on Robotics and Applications, World Automation Congress (WAC 2006)*, Budapest, Hungary, July 24-27, 2006.

6. **Kartoun U.**, Stern H. and Edan Y., Bag Classification Using Support Vector Machines, *Applied Soft Computing Technologies: The Challenge of Complexity Series: Advances in Soft Computing, Springer Berlin / Heidelberg*, ISBN: 978-3-540-31649-7, pp. 665-674, 2006.

7. **Kartoun U.**, Stern H., Edan Y., Feied C., Handler J., Smith M. and Gillam M., Collaborative $Q(\lambda)$ Reinforcement Learning Algorithm - A Promising Robot Learning Framework, *IASTED International Conference on Robotics and Applications (RA 2005)*, Cambridge, U.S.A., October 31 - November 2, 2005.

<u>Conference Papers:</u>

8. **Kartoun U.**, Stern H., Edan Y., Human-Robot Collaborative Learning of a Bag Shaking Trajectory, *The Israel Conference on Robotics (ICR 2006)*, Tel Aviv University, Faculty of Engineering, June 29, 2006.

9. **Kartoun U.**, Stern H. and Edan Y., Virtual Reality Telerobotic System, *e-ENGDET 2004 4th International Conference on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University Yorkshire, U.K., 2004.

10. Edan Y., **Kartoun U.** and Stern H., Cooperative Human-Robot Learning System using a Virtual Reality Telerobotic Interface, *Conference on Advances in Internet Technologies and Applications*, Purdue University, West Lafayette, Indiana, U.S.A., 2004.

1. Introduction

Chapter Overview

Chapter one describes the problem addressed in this work and lays out the research objective and its significance followed by a statement of the contributions and innovations.

1.1 Problem Description

To expand the use of robots in everyday tasks they must be able to perform in unpredictable and continuously changing environments. Since it is impossible to model all environments and task conditions in a rigorous enough manner, robots must learn independently how to respond to the world and how the world responds to actions they take.

One approach to robot learning is reinforcement learning (RL) [Watkins, 1989; Peng and Williams, 1996; Sutton and Barto, 1998; Ribeiro, 2002]. In RL the robot receives positive/negative rewards from the environment indicating how well it is performing the required task. The robot learning goal is to optimize system responses by maximizing a reward function. This is achieved through gaining experience and through direct interaction with the robot's environment. Under uncertainty, the robot may fail to make the correct associations between the observed states and chosen actions that lead to higher rewards. Moreover, certain problems are often too memory intensive to store the large values for each state. Another disadvantage is that RL-based approaches require substantial interaction with the environment to test large numbers of state-action values until an effective policy is determined. One approach to overcome this problem is to avoid storing all state-action pairs, instead to compute them dynamically as the need arises [Touzet, 2004]. For RL tasks an optimal policy is usually found by striving for a goal and attaining rewards. However, this policy is useless when the goal state changes, which means that the policy learned for one problem cannot be used for other problems [Park and Choi, 2002]. Additional disadvantage of RL tasks concern the slow convergence toward satisfactory solutions.

The disadvantages of autonomous learning robotic systems involve the large state-space typical of most robotic environments. In response to some of the requirements, several authors have used RL. The RL-based learning algorithm Q-learning [Watkins, 1989], and its variation $Q(\lambda)$ [Peng and Williams, 1996], an incremental multi-step Q-learning algorithm that combines one-step Q-learning with eligibility traces, have been used in many robotic applications [Zhu and Levinson, 2001; Kui-Hong *et al.*, 2004; Broadbent and Peterson, 2005; Dahmani and Benyettou, 2005]. The learning process entailed in learning robot systems must be accelerated to reduce the heavy computational costs, and to reduce failures. Collaboration of a robot with a human is essential to minimize the amount of time required by a robot to accomplish a learning task [*e.g.*, Papudesi and Huber, 2003;

Papudesi *et al.*, 2003; Mihalkova and Mooney, 2006]. This can be overcome via human intervention whose guidance can decrease the number of learning episodes and accelerate convergence to reach a satisfactory solution for a task. The involvement of superior human intelligence in the learning procedure will affect the learning agent's behavior.

The learning algorithms Q and $Q(\lambda)$ require no human intervention, and as such the agent is placed in an unknown environment and explores it independently with the objective of finding an optimal policy. One drawback to this approach is the large amount of interaction required between the robot and the environment until an effective policy is determined. One possible solution to this problem includes guiding the agent using rules that suggest trajectories of successful runs through the environment [Driessens and Džeroski, 2004; Mihalkova and Mooney, 2006]. [Mihalkova and Mooney, 2006] suggest a RL-based framework denoted "relocation". At any time during training an agent can request to be placed in a different state of the environment. In the "relocation" approach the agent accrues a cost for each relocation event, and thus seeks to limit the number of relocations. Requiring minimal human involvement, the "relocation" approach comprises two agent conditions: (i) "in trouble" - although the agent learns from the negative experience, the actions taken represent a poor choice, thereby leading to a waste of time-steps in a part of the state-space that is unlikely to be visited during optimal behavior; and (ii) "bored" - if the Q-values are being updated by only small increments, the agent is not learning anything new in the current part of the environment. When updating a particular Q-value does not change that value, the agent must relocate to a part of the environment with the greatest probability of changing the Q values.

A central issue in human-robot collaboration involves adjustable autonomy levels, including the determination of whether and when human intervention is required. A learning task performed by a robot must be designed such that it considers how to achieve cooperation via appropriate degrees of sharing and trading between human and robot. Sheridan [Sheridan, 1987] describes a ten-level formulation of robot autonomy, a perspective of relating the degree of robot autonomy to human control. On the one hand, to ensure that highest-quality decisions are made, a robot should transfer control and collaborate with a human operator (HO) when it has superior decision-making expertise. On the other hand, interrupting a user may cause delays or the acquisition of information that is not necessarily beneficial; thus such transfers of control should be minimized.

Fong *et al.*, 2001, describe key issues that must be addressed:

(i) *Robot capability of detecting when it should request help and when it must solve problems on its own.* In this thesis, collaboration with an HO is triggered when a robot reports that its learning performance is low. Then the human must intervene and suggest alternative solutions. Collaboration between the robotic learning process and the HO is essential when robot autonomy fails or an acceleration in learning is desired, but as long as the robot learns policies autonomously and adapts to new states, human-robot collaboration is unnecessary.

(ii) *Robot capability toward self-reliance and safe operation.* In this work safety concerns were taken into account for the robot applications described (*e.g.*, the fixed-arm robot is placed inside a security cage and it is capable of recognizing when its gripper hits an obstacle. If a human opens one of the cage's doors or a collision occurred, the robot is immediately and automatically disabled).

(iii) *Human and robot communication dialogue.* Linguistic-like human interfaces enhanced with real-time visual feedback were developed.

(iv) *Robot adaptation to various users with different skills, knowledge, and experience.* In the applications suggested, the focus is on human expert operator behavior.

In an effort to reduce the long learning times of the $Q(\lambda)$ algorithm, this thesis presents a collaborative $Q(\lambda)$ denoted $CQ(\lambda)$, which accelerates learning. The collaborative algorithm integrates the experience of several agents (*e.g.*, robot, human), and was applied on two real robotic test-bed applications integrating two learning agents each: a robot and a human working cooperatively to achieve a common goal.

1.2 Research Objectives

The fundamental research objective of this work is to develop a new reinforcement learning algorithm, denoted the $CQ(\lambda)$-learning algorithm, for improving the learning performance of robotic systems through human collaboration.

1.3 Research Significance

"As robots move into our natural environment, it is easy to envision situations that afford the need for efficient task learning and collaboration" [Breazeal *et al.*, 2004]. This thesis provides an important step toward realizing that goal. Robot learning requires novel algorithms for learning to identify important events and find efficient action policies. The robot does not have a teacher who can tell him which actions are optimal in every situation that arises. Therefore, by using RL it can independently improve its behavior policy.

Collaboration between a robot and a human during learning is essential, since humans have superior intelligence and skills such as perception, intuition and awareness, to direct policy adjustments in the most beneficial direction. Furthermore, in most robotic applications the environment is unpredictable and unstructured. Thus, if part of the environment is familiar to a HO or if he has some expertise regarding how to perform a task efficiently with a robot, then the operator's intervention in the robot learning process will overcome the uncertainty in the environment. This results in a reduction in the number of learning episodes, thereby accelerating convergence to achieve a satisfactory solution for a task and overcome long learning times.

Learning algorithms can be improved by transferring acquired knowledge between related tasks or learning processes [*e.g.*, Matarić, 1997; Wang *et al.*, 2003; Papudesi and Huber, 2003; Papudesi *et al.*, 2003; Mihalkova and Mooney, 2006]. The ability of a robot to acquire knowledge that it learned or to benefit from knowledge achieved via collaboration with another agent or a human accelerates the entire process of learning. The development of learning algorithms enhanced by collaboration with a HO leads to a learning task solution significantly faster than if performed by a single agent only. Collaboration here is similar to a learning application as described in [Thomaz and Breazeal, 2006], where human reward signals can be treated as an "interactive rewards interface" in which humans can give rewards to a learning agent by rewarding a whole world state.

The proposed development of a RL Q-learning based algorithm denoted $CQ(\lambda)$ (collaborative $Q(\lambda)$) accelerates learning in systems comprising multiple learning agents or designed for human-robot interaction, thus overcoming the main criticism of the RL approach, *i.e.*, long training periods.

1.4 Research Contributions and Innovations

Collaboration between a learning process and a human advisor is important and essential in many tasks to reduce the amount of time required for a robot to accomplish a learning task. This research provides the necessary tools both to improve performance and to reduce the learning times required by robotic systems via the development of human collaboration learning methods.

A new human-robot collaborative reinforcement learning algorithm, $CQ(\lambda)$ (collaborative $Q(\lambda)$), is developed for accelerating learning in robotic tasks. Two frameworks of the $CQ(\lambda)$-learning algorithm are described: (i) learning for multiple agents where learning agents can enter into their learning functions both the information that arrives from the environment and that which arrives from other learners that exist in the system, and (ii) learning for human-robot systems where a robot agent learns both from rewards it receives from in its environment as well as from human suggestions and guidance. Within the framework of the first approach, collaboration is effected by taking the maximum of state-action values, *i.e.*, the Q-value, across all learners at each update step [Kartoun *et al.*, 2005]. In the second approach, two levels of collaboration are defined for human-robot systems: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - the HO suggests actions and the robot incorporates this knowledge into its memory [Kartoun *et al.*, 2006 (a); Kartoun *et al.*, 2006 (b)]. The robot learns efficient policies applied in certain environments and the human assists the robot when needed. This variable autonomy approach is demonstrated in the context of an intelligent environment using a mobile and a fixed-arm robots. Evaluating robot performance during two different tasks revealed the superiority of the $CQ(\lambda)$ over the standard $Q(\lambda)$ algorithm.

The $CQ(\lambda)$ algorithm is based on the $Q(\lambda)$-learning algorithm, but it allows for the collaboration of more than one agent in the environment [Kartoun *et al.*, 2005; Kartoun *et al.*, 2006 (a); Kartoun *et al.*, 2006 (b)].

The main contribution of this work is in developing a new learning method. The suggested method was tested and found to reduce the number of trials a robot requires to learn a task. Furthermore, it enabled reaching solutions that could not be achieved via a single learning process alone - a time consuming procedure. Another contribution is the development of performance measures that quantify learning performance. The $CQ(\lambda)$-learning approach was demonstrated in the context of an intelligent environment using mobile and fixed-arm robot applications.

2. Scientific Background

Chapter Overview

This chapter reviews the literature of the relevant research topics. In particular, reinforcement learning is discussed in detail. Furthermore, current human-robot collaboration and robot learning applications are presented.

2.1 Introduction

Characterized by direct interaction with a real world, sensory feedback, and complex control systems, robotics is one of the most challenging applications of machine learning techniques [Kreuziger, 1992]. Several applications of learning include **(i) world model and elementary sensor-based actions** [*e.g.*, Kerr and Compton, 2003]: learning object properties (*e.g.*, geometry), world exploration (*e.g.*, finding objects, determining/detecting obstacles), learning elementary actions in the world (*e.g.*, effects of actions), learning elementary actions with objects (*e.g.*, manipulation of an object) and learning to recognize/classify states in the internal world model; **(ii) sensors** [*e.g.*, Harvey *et al.*, 2003]: learning how to classify objects based on image data, learning sensor strategies/plans (*e.g.*, how to monitor an action to ensure its correct execution or how to determine certain states of the real world); **(iii) error analysis** [*e.g.*, Scheffer and Joachims, 1999]: learning error recognition, error diagnosis, and error repairing rules; **(iv) planning** [*e.g.*, Theocharous and Mahadevan, 2002]: improvement (speed-up) of planning module (*e.g.*, planning macros, control rules), learning action rules or plans (*i.e.*, how to solve a sub-task in principle), learning relationships between typical task classes and related action plans (*e.g.*, generalized action plan for a set of tasks), learning at the task level (*e.g.*, which geometrical arrangements/action plans satisfy certain functional specifications).

Robot systems must be able to operate in environments with a potentially large variety of unfamiliar objects, materials, and lighting conditions, thus complicating perceptual and manipulation tasks unless a significant number of domain-dependent techniques are used. Imaging sensors provide a large amount of information and therefore they are widely employed in robot applications [Unger and Bajcsy, 1996]. Vision-based grasping and retrieval of objects are skills important in many tasks [*e.g.*, Coelho *et al.*, 2001], and a robotic system that is capable of perceiving pertinent target object features and that can select a viable grasp approach for a robotic arm can perform many useful functions [Unger and Bajcsy, 1996]. Possible scenarios for such a system range from the handling of explosive materials in dangerous environments to the assistance of people with physical disabilities in household and rehabilitation environments.

2.2 Neural Networks

Although the use of NN self-learning techniques allows autonomy, robustness to noise and to errors in data [Ziemke, 1998], disadvantages include long training times, the requirement of many training examples, and the internal reasoning process is not transparent. Three-layer neural networks are universal classifiers in that they can classify any labeled data correctly if there are no identical data in different classes [Young and Downs, 1998; Abe, 2001]. In training multilayer neural network classifiers, usually network weights are corrected so that the sum of squared errors between the network outputs and the desired outputs is minimized. But since the decision boundaries between classes acquired by training are not directly determined, classification performance for the unknown data, *i.e.*, the generalization ability, depends on the training method and it degrades substaintially when the amount of training data is small and the overlap among classes is rare [Shigeo, 2001].

2.3 Support Vector Machines

One popular supervised learning method, support vector machines (SVMs) has emerged in recent years as a successful pattern recognition method [Vapnik, 1998]. SVMs have exhibited superior performance in various applications including text categorization [Goertzel and Venuto, 2006; Lee *et al.*, 2006; Lin *et al.*, 2006], face detection [Jiuxian *et al.*, 2006; Shavers *et al.*, 2006], and content-based image retrieval [Dube *et al.*, 2006; Djordjevic and Izquierdo, 2007].

Among the advantages of SVMs is their maximization of generalization, and they do not confront situations with local minima as in NN-based systems. Disadvantages include the difficulty in extending them to multi-class systems and the long training times involved. SVMs are based on a statistical learning theory developed by [Vapnik, 1995; Vapnik, 1998] that minimizes classification errors of training data and unknown data. In SVMs, the n-class classification problem is converted into n-two-class problems, and in the i^{th} two-class problem the optimal decision function that separates the i^{th} class from the remaining classes is determined. In classification, if one of the n decision functions classifies an unknown datum into a definite class, then it is classified into that class. If more than one decision function classifies a datum into definite classes, or no decision functions classify the datum into a definite class, then the datum is unclassifiable. Another limitation of SVMs is the long training times. Since SVMs are trained by solving a quadratic programming problem with the number of variables equal to the number of training data, training is slow for a large number of training data. Another limitation of SVMs is that when the input environment changes in time, accuracy decreases because the weights are fixed, thus preventing it from adapting to the changing environment.

2.4 Reinforcement Learning

In reinforcement learning (RL), by receiving rewards and punishments the learning system receives feedback in terms of "good" or "bad." The advantage is that a detailed model of the problem and a training set are not required. Instead, the system learns to find an optimal policy by experiencing negative situations (*e.g.*, robot hitting an obstacle) and positive ones (*e.g.*, robot reaching a target). This is different from an NN where learning is achieved by getting feedback for every time step. For RL, often reinforcements are not available until, for example, a goal is achieved, *i.e.*, typically after a possibly long sequence of actions.

In this work, RL involves learning through direct experimentation [Peng and Williams, 1996; Smart, 2002]. It does not assume the existence of a teacher that provides training examples. Instead, experience is the only teacher. The learner receives signals (reinforcements) from the process via indications about how well it is performing the required task. These signals are usually associated with some dramatic condition, - *e.g.*, accomplishment of a subtask (reward) or complete failure (punishment), and the learner's goal is to optimize its behavior based on some performance measure (maximization of a reward function). It learns the associations between observed states and chosen actions that lead to rewards or punishments, *i.e.*, it learns how to assign credit to past actions and states by correctly estimating costs associated with these events [Ribeiro, 2002]. RL algorithms can model actions with non-deterministic outcomes and can learn optimal policies from non-optimal training sets. The disadvantages of RL algorithms include the dependency on a real valued reward signal for each transition. Additionally, convergence can be slow and space requirements can be very large and computationally expensive.

2.5 Robot Learning

Nowadays, robots are migrating from factory production lines and into our everyday lives. Unlike stationary and pre-engineered factory buildings, an everyday environment, such as an office, museum, hospital, or home, is an open and dynamic place where robots and humans can co-exist and cooperate. The office robot, Jijo-2 [Asoh *et al.*, 2001], was built as a test-bed for autonomous intelligent systems that interact and learn in the real world. Jijo-2's most notable properties are its communication and learning skills: it can communicate with humans through a sophisticated Japanese spoken-dialogue system, and it navigates using models that it learns by itself or through human supervision. Self learning is accomplished via a combination of a microphone array, a speech recognition module, and a dialogue management module. Supervised learning occurs using statistical learning procedures in which the robot applies what it learns from landmarks or features in its environment to construct useful navigation models.

Modern robot systems entail increasing intelligence and autonomy requiring new and powerful man-machine interfaces [Längle *et al.*, 1996]. For example, a robot's capability to autonomously recover from error situations corresponds with how well the robot can dynamically adjust its activity during execution of an action [Längle *et al.*, 1996]. Längle *et al.*, 1996, provide a natural language explanation for the error recovery of an autonomous mobile robot named KAMRO.

It is stated in [Nehmzow and Walker, 2005] "a mobile robot interacting with its environment can be described as an analog computer, taking environmental, morphological and task-related data as input, and computing behavior as output." While navigating in an environment, robot tasks include [Howard, 1999] **(i) localization -** determining the robot's location; **(ii) mapping -** building a model of the environment; and **(iii) planning -** planning the robot's movements. Robots have an inherent uncertainty about the state of their environments due to sensor limitations, noise, and the unpredictability of the real-world environment [Howard, 1999]. Learning to navigate in realistic environments requires novel algorithms for identifying important events and planning efficient action policies.

Carreras *et al.*, 2002, propose a Neural Q-learning approach designed for on-line learning of simple and reactive robot behaviors. In this approach, the Q function is generalized by a multi-layer neural network allowing the use of continuous states and actions. The algorithm uses a database of the most recent learning samples to accelerate and guarantee convergence. Each Neural Q-learning function represents an independent, reactive, and adaptive behavior that maps sensorial states to robot control actions. A group of these behaviors constitutes a reactive control scheme designed to fulfill simple missions. Another on-line learning example is given in [Bakker *et al.*, 2006], who propose a quasi on-line RL method: while a robot is exploring its environment, a probabilistic model of the environment is parallelly built in the background as new experiences present themselves, and the policy is trained concurrently based on this model.

[Wang *et al.*, 2006], suggest a modified RL algorithm for a multi-fingered hand for solving the problem of how an arm-hand robot approaches objects before grasping. Learning is divided into two phases, heuristic learning and autonomous learning. In the first phase of learning, the heuristic search is utilized to help the robot reach the goal quickly. The action selection of the robot is guided by the heuristic function of A^* search, which, via rewards, can make the robot move toward a goal. The learning system uses these rewards to update a Q table. Once the table has been modified enough to effectively control the robot, the second learning phase starts. In this phase, the robot is trained using a standard RL learning method, which impels the robot to find the local optimal policy.

Navigation learning by a miniature mobile robot equipped with vision capabilities using several RL-based algorithms is described in [Bhanu *et al.*, 2001]. Comparison between the Q and $Q(\lambda)$

algorithms for a 6 × 6 maze show only a few significant differences between the two learning algorithms. Both begin to converge on the shortest path at approximately the same number of trials. Overall, the $Q(\lambda)$ algorithm requires fewer actions during the entire experiment, which suggests that it is faster in finding the shortest path. [Kui-Hong *et al.*, 2004], demonstrate two mode Q-learning on a humanoid robot in a 17 × 17 maze for improving Q-learning performance. A RL algorithm for accelerating a real mobile robot's acquisition of new skills is described in [Martínez-Marín and Duckett, 2005]. The algorithm speeds up Q-learning by applying memory-based sweeping [Touzet, 2003], and it was tested for a docking task within an image-based visual servoing framework on an ActivMedia PeopleBot mobile robot. A solution for robotic docking based on neural and reinforcement is presented in [Weber *et al.*, 2004]. The solution was partially achieved by training a value function unit and four motor units via RL.

[Kollar and Roy**,** 2006], suggest an approach to trajectory control for a mobile robot performing exploration. RL was used to learn the best trajectory for a robot tracking its position in a 35 × 35 world with an Extended Kalman filter. The control problem was to generate a motion trajectory for the robot from its current estimated pose to a destination position (or sequence of destinations). It was shown that the reinforcement learner successfully generated motion trajectories that minimized the posterior covariance of the robot in contrast to a standard hand-tuned controller that minimized distance.

[Kretchmar, 2002], investigates the problem of multiple reinforcement-learning agents attempting in parallel to learn the value function of a particular task for the n-armed bandit. A parallel RL solution is suggested to avoid statistical overload from, for example, the information of an agent with correspondingly greater accumulated experience than the other agents. To overcome this problem each agent keeps track of two sets of parameters: (i) one set for the actual, independently experienced trials of a particular agent, and (ii) an additional set for combined trials among all other agents. The agents share accumulated experience by keeping separate parameters for their own independent experience and for the combined experience of all other agents. Additionally, the agents can compute an estimate (general Q value which is a weighted combination of a particular agent with all the other agents) based upon global experience. This estimate is computed from a weighted average of the agent's own independent experience and the accumulated experience of all other agents. Results show that as agents are added, learning is accelerated because there is a larger pool of accumulated experience upon which to base future estimates. The experiment with ten parallel agents (the largest number of agents in any of Kretchmar's experiments) learns the fastest. [Ambrym-Maillard *et al.*, 2005], present a method to parallelize the $TD(\lambda)$ algorithm to reduce computation times for various learning tasks. An extension to Kretchmar's work is presented, in which agents share their experience by averaging their value functions. This is done for multi-state episodic tasks using the *TD*(λ) algorithm to generalize function approximators. Experiments using the same *TD*(λ) algorithm were conducted on three kinds of problems, the pendulum problem; the cart-pole problem, and the swimmer problem, and they showed that using more than an average of seven processes in no way reduces computation times.

Experiments on a group of four foraging mobile robots learning to map their conditions to corresponding behaviors was conducted by [Matarić, 1997]. The learning algorithm of the robots consists of reward functions that combine individual conditions of a robot (such as, "grasped a puck", "dropped puck away from home") and collaborative conditions, *i.e.*, how close the robots are to each other. Individually, each robot learns to select the behavior with the maximum value for each condition, in this case to find and take home the most pucks. An evaluation of groups of three and four robots found that interference was a detriment; in general, the greater the number of robots learning at the same time, the longer it took for each individual to converge. Additionally, [Matarić, 1997] found that while measuring the "percent of the correct policy the robots learned in 15 minutes, averaged over twenty trials," the use of heterogeneous reward functions resulted in better performance.

2.6 Collaborative Learning

To build a semi-autonomous collaborative control system, Fong *et al.*, 2001, describe four key issues that must be addressed. First, the robot must have self-awareness [Fong *et al.*, 2001]. This does not imply that the robot needs to be fully sentient, merely that it be capable of distinguishing the conditions under which it should ask for help and those under which it has to solve problems on its own. Second, the robot must be self-reliant. Since the robot cannot always rely on the human to be available or to provide accurate information, it must be able to maintain its own safety. Specifically, the robot should be capable of avoiding unnecessary hazards. Third, the system must support dialogue. That is, the robot and the human need to be able to communicate effectively with each other. Each participant must be able to convey information, to ask questions, and to judge the quality of responses received. To an extent, traditional teleoperation has dialogue (*i.e.*, the feedback loop), but the conversation is limited. With collaborative control, dialogue is two-way and requires a richer vocabulary. Finally, the system must be adaptive. By design, collaborative control provides a framework for integrating users with varied skills, knowledge, and experience. As a consequence, the robot must be able to adapt to different operators and to adjust its behavior accordingly, *e.g.*, asking questions based on the operator's capacity to answer.

Human-robot interaction (HRI) can be defined as the study of humans, robots, and the ways they influence each other. Sheridan notes that one of the challenges for HRI is to provide humans and robots with models of each other [Sheridan, 1997]. In recent years, much effort has focused on developing robots that work directly with humans, as assistants or teammates [Nourbakhsh *et al.*, 1999; Baltus *et al.*, 2000]. Crucial aspects for the human-robot cooperation include simulation, distribution, robot autonomy, behavior descriptions, and natural human-machine communication [Heguy *et al.*, 2001]. An experimental environment called EVIPRO (Virtual Environment for Prototyping and Robotic) was developed allowing the assistance of autonomous robots while carrying out a teleoperation mission [Heguy *et al.*, 2001]. In this project, man-machine cooperation to carry out teleoperated missions in a system using virtual reality and adaptive tools was studied. The goal for the human users and the autonomous robots was to achieve a global task in a virtual environment. This project used both virtual reality and behavior simulation technologies. Thanks to virtual reality, the project could have a natural, intuitive interface and mix different information to increase user perception. Behavior simulation tools were used to help a human user via autonomous robots. Affordable commercial simulators are now available for practicing tasks such as threading flexible endoscopes down a virtual patient's throat or manipulating long surgical instruments [Sorid and Moore, 2000].

[Clouse, 1996] introduces the "Introspection Approach" (IA) in which a learning RL agent determines when to ask a training agent for aid. In IA, an automated *Q*-learner relies on on-line trainer-suggested actions for given situations. When the trainer is asked to intervene, the task state is changed, *i.e.*, the suggested action is executed by the learner as it had chosen with its own policy. The goals here are to: (i) maximize the impact of the trainer's instruction to allow the learner to develop its decision policy quickly, and (ii) minimize the trainer's usage while simultaneously minimizing the training time. Experiments including graph-traversal in the form of two-dimensional mazes were performed. The learner's objective was to traverse a maze optimally from top-left cell to the bottom-right cell. In the experiments performed, automated trainers with varying levels of proficiency instructed the learning agents. When two extreme *Q*-values were sufficiently close, the learner asked for aid. "Sufficiently" is defined by examining the minimum and maximum values and comparing the result to a pre-defined width parameter. The width's value determines how conservative the learner is; the larger it is, the learner asks for aid more frequently. The results showed that the same number of trainer's responses produced a faster learning than letting the learner to ask aid randomly. Thus, guidance received via IA is more informative than random guidance.

[Blumberg *et al.*, 2002] describe autonomous animated dog training using an RL-based approach involving human interaction to promote real-time learning for synthetic characters. Their approach simplifies the learning task for characters by (i) exploiting predictable regularities, (ii) allowing the use of supervisory signals, and (iii) allowing training by humans. The approach presented, denoted "clicker training," entails training to recognize and use acoustic patterns as cues for actions as well as to synthesize new actions from novel paths. The work describes RL techniques in which a dog learns to maximize reward (*e.g.*, scratching the dog's head) and allows it to make maximal use of supervisory signals. The described "clicker training" technique has three steps: (i) making an association between the sound of a toy clicker and a food reward, (ii) marking behaviors by users, and (iii) subsequently treating for performing a behavior more frequently. Several learning tasks involving a human are described, such as (i) learning to relate to a new percept-action pair (*e.g.*, recognizing the phrase "sit"), (ii) a demonstration of luring the dog through a novel trajectory - when rewarded, this lured trajectory is added to the action space as a new action, and (iii) shaping - shaking a paw.

[Hellström, 2005] addresses the problem of making intelligent robots that learn reactive behaviors from demonstrations. His paper describes experiments conducted with a Khepera robot equipped with eight IR sensors for obstacle avoidance. To control the robot, a rule base that included a set of stimuli-response pairs was generated, demonstrating the required behavior. The experiments performed (the road sign problem [Linåker and Jacobsson, 2001] and mimicking the behavior of a light-avoiding cockroach) demonstrate the power of using association rules to model reactive behaviors.

[Aminaiee and Ahmadabadi, 2006] developed a team Q-learning approach to the distributed object pushing task. In the proposed approach, the required individual skills for single-robot object pushing are learned first using a fuzzy RL method. Then the robots learn how to coordinate their actions to push the object cooperatively. Such an RL method consists of two steps: each robot learned to push the object by controlling its arm to move a defined point on the object to its corresponding desired goal or final position; then the robots learned to cooperate with each other in pushing the object with the goal of avoiding a block in the system through maximization of their individual average rewards.

[Lockerd and Breazeal, 2004] describe a collaborative process enabling a robotic learner to acquire concepts and skills from human examples. During teaching the robot is required to perform tasks based on human instructions. It executes the tasks, and by incorporating feedback its hypothesis space is converged. With the Q-learning approach, the robot learns a button pushing task.

[Bowling and Veloso, 2003] describe GraWoLF, a general-purpose, scalable, multi-agent learning algorithm that combines gradient-based policy learning techniques with the WoLF ("Win or Learn Fast") variable learning rate. The algorithm was applied to an adversarial multi-robot task with simultaneous learning. They showed that learning does considerably improves performance relative to the starting policies.

[Gu and Hu, 2005] present a cooperative RL algorithm of multi-agent systems denoted as the "leader-following Q-learning algorithm." The algorithm is based on a Markov or stochastic game, in which there are multiple stages and each stage is a static Stackelberg game. [Kretchmar, 2002] investigates the problem of multiple RL agents attempting to learn the value function of a particular task in parallel for the n-armed bandit task. A parallel RL solution is suggested to overcome the problem of statistical overload from the information presented by an agent with correspondingly more accumulated experience than the other agents. Another multi-agent learning system is the Cobot; [Isbell *et al.*, 2001], describe a multi-user chat software agent, the Cobot, that collects social statistics and reports them to users. The human-computer interaction application is RL-based and its action selection is determined by using multiple resources of human rewards. Cobot can initiate actions such as proposing conversation topics and introducing users by learning their individual and commercial preferences. Cobot is rewarded or punished via explicit verbal feedback from users (*e.g.*, the verbs hug and spank). Because Cobot can be understood as running a large number of separate RL processes in parallel with a different state-action space for each process, linear function approximation was used. Results indicated that repeated user feedback for a non-uniform set of preferences pays off with a corresponding policy. In particular, for a specific group of users, Cobot learned that its presence causes a significant shift toward its preferences, *i.e.*, Cobot responds to his dedicated users.

[Rosenstein *et al.*, 2005], describe an HRI (human-robot interface) that supports both adjustable autonomy and hierarchical task selection. With adjustable autonomy, a computer switches among several control modes ranging from full supervision to full autonomy. With hierarchical task selection, the interface allows an operator to easily solve a high-level task autonomously or else to guide a robot through a sequence of lower-level subtasks that may or may not involve autonomous control. [Yanco *et al.*, 2005], define sliding scale autonomy as the ability to create new levels of autonomy between existing, pre-programmed autonomy levels. The sliding scale autonomy system shows the ability to dynamically combine human and robot inputs using a small set of variables such as user and robot speeds, speed limitations, and obstacle avoidance.

In [Wang *et al.*, 2003], a variable autonomy approach is used. User commands serve as training inputs for the robot learning component, which optimizes autonomous control for its task. This is achieved by employing user commands for modifying the robot's reward function. Using the potential of learning from reinforcement and human rewards illustrate the changes in user reward and Q-value functions, accordingly [Papudesi and Huber, 2003; Papudesi *et al.*, 2003]. The task was to learn how to optimally navigate to a specific target in a two-dimensional world with obstacles. Similarly, [Thomaz and Breazeal, 2006] describe a new RL-based approach for providing reward signals by human. The signals depend not only on past actions but also on future rewards called "Future Directed Rewards." The experimental platform described is a learning game platform called "Sophie's Kitchen" that was developed for investigating how human interaction changes the learning process for baking a cake. One feature of "Sophie's Kitchen" is called the "Interactive Rewards Interface," in which humans can give rewards using a standard mouse. For teaching the agent, this reward can be given in two ways: (i) rewarding a whole state of the world and (ii) rewarding a state of a particular object. This distinction was made to determine whether people prefer to communicate feedback about particular aspects of a state rather than an entire world state. Results achieved indicate that, in general, people assumed that specific rewards are future directed or guidance for the agent, *i.e.*, what people want the agent to do next.

Although Q-learning and $Q(\lambda)$ were used in many robotic applications [*e.g.*, Touzet, 2003; Menegatti *et al.*, 2004; Broadbent and Peterson, 2005; Dahmani and Benyettou, 2005; Zhu and Levinson, 2005; Asadpour *et al.*, 2006], the issue of accelerating learning is still significant. It includes accelerating of learning toward finding an optimal or close to optimal solution.

2.7 Summary

Machine learning methods applied on robotics involve the development of statistic-based algorithms and techniques that allow them to perform tasks optimally. Significant machine learning methods have been reviewed in this section. Several of the methods mentioned were applied in real robot applications that will be described later in this work. Major types of learning techniques are summarized in Table 2.1 [Zimmerman and Kambhampati, 2001; Nordlander, 2001], including the possible models to which each method is applicable and the corresponding advantages and disadvantages.

Table 2.1 Learning algorithms comparison

Learning Algorithm	**Models**	**Advantages**	**Disadvantages**
Neural Networks	Discrete, real and vector-valued functions	Robust to noisy, complex data and errors in data. Very flexible in types of hypotheses they can represent. Bears some resemblance to a very small human brain. Can adapt to new data with labels. Do not have to fulfill any statistical assumptions, and are generally better at handling large amounts of data with many variables. Fast.	Long training times are common, learned target function inscrutable. Many training examples required. Very difficult to understand their internal reasoning process.
Support Vector Machines	Discrete, real and vector-valued functions	Maximization of generalization ability. No local minima.	Extension to multi-class problems is not straightforward. Long Training Time.
***Q*-Learning**	**Control policy to maximize rewards**	**Actions modeled with non-deterministic outcomes, optimal policy learned from non-optimal training sets, facilitates life-long learning.**	**Depends on a real valued reward signal for each transition.** **Convergence can be slow.** **Space requirements can be huge.**

Significant works related to reinforcement learning applied in robot learning are summarized in Table 2.2:

Table 2.2 Summary of "state of the art" robot learning related works

Method	Application	Reference
Statistical learning procedures	Interactive office robot (Jijo-2)	Asoh *et al.*, 2001
Virtual reality and behavior simulation	Cooperative assistance in teleoperation (EVIPRO)	Heguy *et al.*, 2001
RL-based approach involving human interaction	Human teacher to guide exploration during learning	Clouse and Utgoff, 1992, Clouse 1996
	Animated dog	Blumberg *et al.*, 2002
Learning reactive behaviors from demonstrations	Khepera robot for obstacle avoidance	Hellström, 2005
Neural Q-learning	Learning of reactive robot behaviors	Carreras *et al.*, 2002
Q and $Q(\lambda)$ learning	Mobile robot navigation	Bhanu *et al.*, 2001
Q learning	Humanoid robot navigation	Kui-Hong *et al.*, 2004
	Vision-guided mobile robot	Martínez-Marín and Duckett, 2005
	"Relocation" of mobile robots	Mihalkova and Mooney, 2006
	Flight control	Motamed and Yan, 2006
Q learning and human instructions	Robot button pushing task	Lockerd and Breazeal, 2004
Gradient-based policy learning	Multi-robot learning algorithm	Bowling and Veloso, 2003
	Motor primitive learning for baseball	Peters and Schaal, 2006
Leader-following Q-learning algorithm	Stackelberg game	Gu and Hu, 2005
Multiple RL agents	Mobile robots learning a foraging task	Matarić, 1997
	Multiple agent RL	Bagnell, 1998
	n-armed bandit task	Kretchmar, 2002
	Inverse RL with evaluation	Freire da Silva *et al.*, 2006
Multi-agent learning	Cobot: a social RL agent	Isbell *et al.*, 2001
	Cooperative learning via combining decision trees	Asadpour *et al.*, 2006
	Distributed object pushing task	Aminaiee and Ahmadabadi, 2006
HRI and adjustable autonomy	Robot guiding	Rosenstein *et al.*, 2005
HRI and sliding scale autonomy	Robot speed control and obstacle avoidance	Yanco *et al.*, 2005
HRI and Q-learning	Button pushing task	Lockerd and Breazeal, 2004
	Mobile robot navigation	Papudesi *et al.*, 2003; Papudesi and Huber, 2003;
HRI and variable autonomy	Modifying mobile robot reward function	Wang *et al.*, 2003
Human-computer interaction and future directed rewards	Sophie's Kitchen	Thomaz and Breazeal, 2006
Hierarchical RL	Quadruped robot obstacle negotiation	Honglak *et al.*, 2006
Quasi on-line RL	Mobile robot navigation	Bakker *et al.*, 2006
Two-stages RL algorithm	Multi-fingered robotic hand	Wang *et al.*, 2006
RL for control	Trajectory control for a mobile robot	Kollar and Roy, 2006

3. Methodology

Chapter Overview

This chapter describes the methods used in this research. Definitions and notations for the systems developed are described first. The following sections present the $CQ(\lambda)$ learning algorithm applied on the systems, after which the performance measures and experiments performed for each system are described.

3.1 Introduction

The $CQ(\lambda)$ algorithm was developed to overcome the expensive computation, and the long learning times entailed in both the Q and its variation $Q(\lambda)$-learning algorithms. The new algorithm enables collaboration of learning of several agents (*e.g.*, robots) in the environment. Through collaboration the number of learning episodes required to perform a task can be decreased by taking advantage of human intelligence and expertise.

The $CQ(\lambda)$-learning algorithm was developed, tested and applied for two frameworks: (i) learning by multiple agents and (ii) learning by human-robot systems. In the first framework, collaboration involves taking the maximum of state-action values, *i.e.*, the Q-value, across all learning agents at each update step. In the second framework, two levels of collaboration are defined for a human-robot learning system: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - a human operator (HO) guides the robot and the robot replaces its own exploration process. The key idea here is to give the robot enough self awareness to adaptively switch its collaboration level from autonomous (self performing) to semi-autonomous (human intervention and guidance).

Three systems, specially developed for this thesis, were used to evaluate the $CQ(\lambda)$-learning algorithm presented in this thesis: (i) navigation of multiple robots (simulation), (ii) human-robot collaboration for a bag shaking task, and (iii) human-robot collaboration for a mobile robot navigating a two dimensional world. For each learning system described below, different definitions and notations are presented. Two frameworks of the $CQ(\lambda)$-learning algorithm are then described, including descriptions of the performance measures used and the experiments performed.

3.2 Problem Definitions and Notations

3.2.1 Multiple Mobile Robot Navigation

The system consists of several mobile robots represented in a simulation model. The robots learn to navigate a two dimensional world that contains undesirable areas choosing the optimum path to reach a target. A learning system consists of one collaborative robot denoted Q_c and one or more independent $Q(\lambda)$-based learners. The $CQ(\lambda)$ algorithm is applied to autonomous mobile robot navigation where several robot agents[1] serve as learning processes with the objective of choosing the optimum path to reach a target. The robot's state is represented by its location in a $W \times W$ grid world. The state of the robot at time step t, $s_t \in S$, is defined by: $s_t = (x_k, y_l)$ where $k \in (1, 2, ..., W)$ and $l \in (1, 2, ..., W)$. An action, $a_t \in A$, taken at each state is traveling north, west, south, or east. Rewards are defined as $r(s_t, a_t)$. If the robot reaches the target, the reward is positive. If it passes through an undesirable area, the reward is negative. Otherwise, the reward is zero. The system was implemented in simulation to avoid interference of hardware limitations such as robot obstacle avoidance and localization.

3.2.2 Bag Shaking Experiment with a Fixed-Arm Robot

The system comprises of a fixed-arm six degrees of freedom Motoman UP-6 robot, a bag that contains objects, and a platform on which the inspected bag is manipulated. The learning task is to observe the position of the bag located on an inspection surface, grasp it, and learn how to shake out its contents in minimum time by interacting with the environment and by using suggestions acquired from a HO. It is assumed that the number of items in the bag is known in advance. Robot states are denoted $s_t \in S$ defined as its gripper location in a three-dimensional grid. The performance of the task is a function of a set of actions, $a_t \in A$, for each physical state of the system. An action, a_t, consists of a robot movement over its X, Y or Z axes from a state s_t to state s_{t+1}. Three reward functions were used to evaluate the learning system as follows: (i) linear reward function, (ii) cumulative-based reward function, and (iii) events-based reward function.

3.2.3 Navigation of a Mobile Robot

The system consists of an Evolution Robotics ER-1 mobile robot equipped with a laptop and a camera. The robot task is to learn to navigate toward a target location in a two-dimensional world. The robot is remotely located relative to the HO, and it uses environmental sensing capabilities for recognizing undesirable areas[2] on its way to the target. Under pre-defined system conditions, the robot decides to ask for human advice and guidance or to navigate autonomously. Learning is achieved by interaction with the environment and by acquiring suggestions from the HO. The purpose of the learning system is to let the robot begin navigating from any starting location in the world and reach the target using the shortest path while avoiding undesirable areas. An optimal route

[1] The terms agent and robot will be used interchangeably.
[2] Undesirable area - an area where a robot can physically pass through but it is not recommended.

is defined as the shortest route that the robot navigates most efficiently, *i.e.*, move toward the target and not away, while avoiding undesirable areas.[1] If the robot travels inefficiently (but still reaches the target), the route is defined as feasible. State-space, state, action, and reward definitions are identical to those described in Section 3.2.1.

3.3 Robot Learning Algorithms

Two robotic frameworks were developed for testing the $CQ(\lambda)$-learning algorithm: (i) learning with multiple agents, and (ii) learning in human-robot collaborative systems. In the **first** framework (3.3.1), collaboration involves taking the best state-action values, *i.e.*, the Q-value across all learners at each update step. In the **second** framework (3.3.2), two levels of collaboration are defined for a human-robot learning system: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - a human operator (HO) guides the robot to take an action or a policy and the robot uses the suggestion to replace its own exploration process. The robot adaptively switches its collaboration level from autonomous (self performing) to semi-autonomous (human intervention and guidance) based on its learning performance.

3.3.1 CQ(λ) -Learning for Multiple Agents

The $CQ(\lambda)$ learning algorithm for multiple agents is based on a state-action value of an agent or learning process is updated according to the best performing agent; collaboration is in taking the maximum of state-action values, *i.e.*, the Q-value, across all learners at each update step [Kartoun *et al.*, 2005]. By applying this method, the Q value for a collaborative learner will be the best value.

3.3.2 CQ(λ)-Learning for Human-Robot Systems

For human-robot systems, it is assumed that the learning system consists of one robot and one HO. The robot learns to perform a task by using a standard $Q(\lambda)$-learning function. While learning, it continuously measures its learning performance by defining Λ[2], a minimum acceptable performance threshold above which the robot requests human intervention. The measure Λ is compared with L_{ave}, a moving average learning performance measure over the last N most recent learning episodes[3] considered (3.1).

---

[1] The shortest route that the robot navigates most efficiently is in terms of path length remaining. It is calculated manually after accomplishing the experiments and is used for results evaluation and analysis.
[2] The minimum acceptable performance threshold. Above this value, the human is called to intervene. This threshold value is determined based on empirical tests.
[3] The term "learning episode" is similar to the term "learning trial".

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) / N \tag{3.1}$$

where n is the current learning episode, $i = n-N, n-N+1, n-N+2, ...n-1$, and $S_i \in \{0,1\}$ [1] indicates whether a policy was successful for the i^{th} episode. Based on this learning performance threshold, the robot switches between fully autonomous operation and the request for human intervention. The threshold for a successful episode is defined as $\overline{R}$ (3.2).[2]

$$S_i = \begin{cases} 1 & if \quad R_i > \overline{R} \\ 0 & \quad else \end{cases} \tag{3.2}$$

where R_i is the reward achieved for the i^{th} learning episode. Λ [3] is defined as a minimum acceptable performance threshold, which is compared to the average performance, L_{ave}. If the robot performance fails below the threshold (3.3), the robot switches between fully autonomous operation and semi-autonomous operation and requests human intervention. The procedure is repeated M times where M (set *a-priori*) is the maximal number of learning episodes.

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) / N < \Lambda \tag{3.3}$$

3.4 Performance Measures

3.4.1 Multiple Mobile Robot Navigation

System performance was evaluated using the following measures:

1) $N_{t_{no}}$ - convergence to near optimality[4] - mean of the last N path lengths. This measure determines how close to optimality the current solution is.
2) N_{t_o} - convergence to optimality - number of learning episodes required to perform a policy optimally and repeat it an infinite number of times.

3.4.2 Bag Shaking with a Fixed-Arm Robot

System performance was evaluated using the following performance measures: (i) average time to complete emptying the contents of a bag, (ii) average cumulative reward, *i.e.*, it measures the

[1] A policy is considered as successful if the reward achieved is higher than a predefined value.
[2] The threshold value for a successful episode is determined based on empirical tests.
[3] The minimum acceptable performance threshold. This threshold value is determined based on empirical tests.
[4] Convergence to near optimality requires definition of how near to optimality is sufficient [Kaelbling *et al.*, 1996]. That's the measure for a predefined level of performance after a given time.

improvement in learning, and (iii) human intervention rate, *i.e.*, a measure that represents the percentage of human interventions out of the total number of learning episodes; the lower it is, the more autonomous the robot is.

Robot learning experience is achieved through direct experience with the environment according to rewards, and it is based, after performing a shaking policy, on the number of items that fell from the bag and when they fell. Three reward functions were used to evaluate the learning system as follows (see also Appendix VI):

1) Linear reward function:

$$R_n = c \cdot O \tag{3.4}$$

where R_n is the reward at learning episode n, O is the number of items that fell from a bag during a shaking operation and c is a positive constant to adjust the reward values achieved.

2) Cumulative-based reward function:

$$R_n = c \cdot \left(\sum_{i=0}^{T} \left(\frac{W_j}{t_j} \right) \right) \tag{3.5}$$

where W_j is the current weight measured by a digital scale at time t_j (increments of 0.25 second), *T=min{Fixed Horizon Time*[1], *Amount of Time when all Objects Fell*[2]*}* is the time of shaking, c is a positive constant to adjust the reward values achieved and R_n is the reward for learning episode n.

3) Events-based reward function:

$$R_n = c \cdot \left(\sum_{i=0}^{T} \left(\frac{\Delta(t_j)\left(\frac{W_j - W_{j-1}}{w} \right)}{t_j} \right) \right), \qquad \Delta(t_j) = \begin{cases} 0, & \text{if no items fell} \\ 1, & \text{if an item(s) fell} \end{cases} \tag{3.6}$$

[1] *Fixed Horizon Time* is the time it takes the robot to perform a pre-defined number of state-action transitions (it was set to 100 state-action pairs).
[2] The amount of time when all objects fell.

where R_n is the reward at learning episode n, W_j is the current weight measured by a digital scale located under the inspection surface at time t_j (increments of 0.25 second) when the j event occurred (an event is defined as the falling of one or more objects). Dividing the weight differences by t_j effectively increases the reward for items that fall early. w is the weight of one object (a constant value). W_{j-1} is the weight measured by the scale when the pervious (for the first event, $W_{j-1}=0$). *T=min{Fixed Horizon Time, Amount of Time when all Objects Fell}* is the shaking time. The value $\frac{W_j - W_{j-1}}{w}$ represents the number of objects that fell at time t_j and is rounded toward the closest integer value to eliminate scale inaccuracy. The positive constant c is used to adjust the reward values achieved.

3.4.3 Navigation of a Mobile Robot

System performance was evaluated using the following measures:

1) Mean number of steps to optimally reach target - in each learning episode the robot starts from a random state and tries navigating toward the target. If the robot navigates without passing through an undesirable area and does not travel inefficiently (*e.g.*, moves away from the target), the route is defined as optimal.[1]
2) Mean number of steps to feasibly reach the target - if the robot navigates without passing through an undesirable area but travels inefficiently (but still reaches the target), the route is defined as feasible.
3) Percent of human interventions - measures frequency of human collaboration with the robot.

3.5 Experiments

3.5.1 Multiple Mobile Robot Navigation

The simulated system was evaluated using two experimental setups. To evaluate $CQ(\lambda)$ performance, several hypotheses entailing a total of fifty simulation runs[2] were conducted for each setup. The two setups were designed as follows:

[1] The shortest route that the robot navigates most efficiently is in terms of path length remaining. It is calculated manually after accomplishing the experiments and is used for results evaluation and analysis.

[2] One simulation run contains 100 learning episodes. Each learning episode consists of placing the robot at a starting location in the environment. The robot explores the environment. A learning episode ends when the robot reaches the target.

1) Experimental setup I:

This setup contains two robot agents, a collaborative robotic agent denoted as Q_c learns according to the $CQ(\lambda)$ algorithm, *i.e.*, learns both from interaction with the environment and from gathering knowledge from an independent learning robot. The independent agent learns according to the traditional $Q(\lambda)$ algorithm and does not gain knowledge from the collaborative learner. The following parameters were set: $\alpha_1 = \alpha_2 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = 0.99$, and $\lambda_1 = \lambda_2 = 0.5$.[1]

2) Experimental setup II:

This setup contains three agents where Q_c is a collaborative robot agent that learns according to the $CQ(\lambda)$ algorithm, *i.e.*, gathers knowledge from other independent $Q(\lambda)$ learners and from interaction with the environment. The other two agents learn independently according to the $Q(\lambda)$ algorithm by interacting with the environment. The setup was set with the following parameters: $\alpha_1 = \alpha_2 = \alpha_3 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = \gamma_3 = 0.99$, and $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$.

3.5.2 Bag Shaking Experiment with a Fixed-Arm Robot

A system consisting of three experimental setups using different reward functions was developed for testing the $CQ(\lambda)$-learning algorithm. In all experiments performed, the first trial consisted of a random shaking policy over the robot's X , Y and Z axes. The systems acquired rewards achieved based on interaction with the environment while $\gamma = 0.9$, $\lambda = 0.5$, and $\alpha = 0.05$ were set. To balance between exploration and exploitation (*e.g.*, [Guo *et al.*, 2004; Meng *et al.*, 2006]), ε-greedy action selection with $\varepsilon = 0.1$ was used. The experimental setups are described:

1) Experimental setup I:

In this experimental setup rewards were measured manually with a standard timer using (3.4). When system learning performance was low, HO was asked to intervene and suggest various speeds and adjacent state distances over the X , Y and Z axes of the robot gripper. For this experimental set up, 75 learning episodes were separated into three stages: (i) training - during the first ten runs the robot performs shaking policies autonomously, (ii) collaboration - this stage consists of forty shaking policies with human intervention allowed, but whether the human is activated to intervene is based on the system learning performance, and (iii) testing - for measuring the efficiency of the human collaboration, the robot performed 25 policies using the original shaking parameters defined in the training stage with no human intervention. To compare $CQ(\lambda)$ with the $Q(\lambda)$-learning

[1] The RL parameters, α (learning rate), γ (discount factor), and λ (eligibility trace), are described in Section 4.1.

algorithm, a second experiment was designed. The experiment consisted of 25 learning episodes in which the system learned according to the standard $Q(\lambda)$-learning algorithm with no human intervention.[1]

2) Experimental setup II:

In this setup, a digital scale was used to automatically measure the rewards using a cumulative reward function (3.5). Similar to the first experimental setup, when system learning performance is low, the human is asked to intervene and suggest various speeds and adjacent state distances over the X, Y and Z axes of the robot gripper. $Q(\lambda)$-learning is compared with $CQ(\lambda)$, running each of them for fifty learning episodes.

3) Experimental setup III:

Similar to the second experimental setup, a digital scale was used to automatically measure the rewards. The reward function used here is based on events, *i.e.*, the occurrence of falling objects (3.6). When system learning performance is low, the human is asked to intervene directly in the system Q table. This is done by using an interface designed to take control of the different swing weights over the robot X, Y and Z axes. $Q(\lambda)$-learning is compared with $CQ(\lambda)$, running each of them for fifty learning episodes.

3.5.3 Navigation of a Mobile Robot

Four experiments using several robot autonomy modes were performed. Each experiment included sensitivity analysis of adjusting various values of the γ and λ RL parameters. In the first experiment, to compare the $Q(\lambda)$ and the $CQ(\lambda)$ algorithms, the robot autonomously learned the environmental surroundings with no human intervention (according to the standard $Q(\lambda)$ algorithm). In the remaining three experiments, the robot learned the environmental surroundings semi-autonomously using guidance gained from human intervention (using the $CQ(\lambda)$ algorithm). Different levels of human intervention corresponding to three robot learning threshold values, Λ, were defined and the robot switched its activity from semi-autonomous navigation to full autonomy. The performance sensitivity of different λ values (0.25, 0.5, 0.75) was tested for each combination of values of γ (0.9, 0.95, 0.99). Each combination of an autonomy level, γ and λ, consisted of fifty

[1] To compare between the algorithms, 25 learning episodes were taken into account for each one of them. For the *CQ(λ)* algorithm, the learning episodes consist of the ten training learning episodes and the first 15 learning episodes of the collaboration stage. This results a total of 25 learning episodes. For the *Q(λ)* algorithm an additional 25 learning episodes with no human intervention where considered.

learning episodes, *i.e.*, in each experiment, the robot was placed randomly in one of the world states and tried to navigate toward a target state.

4. Robot Learning Algorithms

Chapter Overview

The objective of the new $CQ(\lambda)$ algorithm is to accelerate learning. Two $CQ(\lambda)$-based learning frameworks are presented: (i) learning with multiple agents, and (ii) learning with human-robot systems. This chapter describes the algorithm including convergence and performance analysis.

4.1 Introduction

It is stated in [Ribeiro, 2002] "nearly all RL methods currently in use are based on the Temporal Differences (TD) technique [Sutton, 1988]. The fundamental idea behind it is prediction learning: when the agent receives a reinforcement, it must somehow propagate it backwards in time so that states leading to that condition may be associated with a prediction of future consequences. This is based on an important assumption on the process' dynamics, called the Markov condition: the present observation must be a conditional probability on the immediate past observation and input action. In practical terms, this means that the agent's sensors must be "good" enough to produce correct and unambiguous observations of the process states."

The basic assumption in Markov Decision Processes is that any state s_{t+1} occupied by an agent is a function only of its last state and action: $s_{t+1} = f(s_t, a_t)$ where $s_t \in S$ and $a_t \in A$ are the state and action, respectively, at time step t [Ribeiro, 2002]. In Q-learning, an algorithm specific to Markov systems, the system estimates the optimal action-value function directly and then uses it to derive a control policy using the local greedy strategy [Watkins, 1989]. It is stated in [Broadbent and Peterson, 2005] "Q-learning can learn a policy without any prior knowledge of the reward structure or a transition model." Q-learning is thus referred to as a "model-free approach" where Q values can be calculated directly from the elementary rewards observed. Q is the system's estimate of the optimal action-value function [Smart and Kaelbling, 2000]. It is based on the action value measurement $Q(s_t, a_t)$, defined in (4.1).

$$\begin{aligned} Q(s_t, a_t) &= E[r(s_t, a_t) + \gamma V^*(s_{t+1})] = \\ &= r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1}) \end{aligned} \tag{4.1}$$

where $V^*(s_{t+1})$ is the optimal expected cost and the γ parameter is the discount rate that describes how foreseeing the agent is; small values of γ (*e.g.*, close to zero) make the agent giving immediate events higher significance. Equation (4.1) represents the expected discounted cost for taking action

a_t when visiting state s_t, and following an optimal policy thereafter. From this definition and as a consequence of Bellman's optimality principle [Bellman and Kalaba, 1965], (4.2) is derived.

$$Q(s_t,a_t) = r(s_t,a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t,a_t) \max_a Q(s_{t+1},a_t) \tag{4.2}$$

The essence of Q-learning is that these characteristics (maximum operator inside the expectation term and policy independence) allow an iterative process for calculating an optimal action. The first step of the algorithm is to initialize the system's action-value function, Q. Since no prior knowledge is available, the initial values can be arbitrary (*e.g.*, uniformly zero). Next, at each time step t, the agent visits state $s_t \in S$ and selects an action $a_t \in A$. Then it receives from the process the reinforcement $r(s_t,a_t) \in R$ and observes the next state s_{t+1}. The procedure continues by updating the action value $Q(s_t,a_t)$ according to (4.3) which describes a Q-learning one step.

$$Q_{t+1}(s_t,a_t) = (1-\alpha)Q_t(s_t,a_t) + \alpha[r(s_t,a_t) + \gamma \hat{V}_t(s_{t+1})] \tag{4.3}$$

where $\hat{V}_t(s_{t+1}) = \max_{a_t \in A}[Q_t(s_{t+1},a_t)]$ is the current estimate of the optimal expected cost $V^*(s_{t+1})$ and α is the learning rate which controls how much weight is given to the immediate reward, as opposed to the old Q estimate. The greater α, the more the state-action value tends toward new information. High values of α make learning faster, but ending up receiving slightly lower rewards. The process repeats until a stopping criterion is met. The greedy action $\mathrm{argmax}_{a_t \in A}[Q_t(s_{t+1},a_t)]$ is the best the agent performs when at state s_{t+1}. For the initial stages of the learning process, however, actions are chosen randomly to encourage exploration of the environment. Under some reasonable conditions (the rewards are bounded and the learning rate is in the range of zero to one) [Watkins and Dayan, 1992], by iteratively applying (4.3), convergence to the optimal value function is guaranteed [Smart and Kaelbling, 2000].

A generalization of Q-learning, represented by $Q(\lambda)$ [Peng and Williams, 1996] uses eligibility traces, $e(s_t,a_t)$: the one step Q-learning is a particular case with $\lambda = 0$ [Glorennec, 2000]. The Q-learning algorithm learns quite slowly because only one time step is traced for each action [Wang *et al.*, 2003]. To boost learning, a multi-step tracing mechanism, the eligibility trace, is used in which the Q values of a sequence of actions are updated simultaneously according to the respective lengths of the eligibility traces [Zhu and Levinson, 2005]. λ represents the eligibility decay rate. The greater λ is, the longer the sequence of values of state-action pairs updated.

Although the convergence of $Q(\lambda)$ is not assured for $\lambda > 0$, experience shows that learning is faster [Glorennec, 2000]. Several action selection policies are described in the literature for RL where the greedy policy (*e.g.*, [Nason and Laird, 2004; Natarajan and Tadepalli, 2005]) is always to choose the best action. Other policies (*e.g.*, "softmax" [Bakker *et al.*, 2006] or "ε-greedy" [Sutton and Barto, 1998]) are stochastic and based on choosing a suboptimal policy to explore the state-action space.

4.2 CQ(λ)-Learning for Multiple Agents

The proposed $CQ(\lambda)$ learning algorithm aimed to accelerate learning in a system composed of multiple learning agents. The following assumptions are considered:

- The agents have an identical state representation of the environment.
- The agents have full communication to transfer value functions.
- The agents perform the same task, sequentially.
- Initial value functions for all learners are underestimated.
- A reward is given to an agent individually when it accomplishes the task.

In contrast with [Matarić, 1997], where the learning algorithms of multiple robots consist of reward functions that combine individual conditions of a robot, the $CQ(\lambda)$ learning algorithm for multiple agents is based on a state-action value of an agent or learning process is updated according to the best performing agent; collaboration is in taking the maximum of state-action values, *i.e.*, the Q-value, across all learners at each update step [Kartoun *et al.*, 2005]. By applying this method, the Q value for a collaborative learner will be the best value. As opposed to parallel RL [*e.g.*, Kretchmar, 2002; Grounds and Kudenko, 2006; Grounds and Kudenko, 2007] a sequential learning is performed here. In parallel RL, several agents learn the value function of a particular task in parallel, *i.e.*, each agent gains a different learning experience while the agents learn the task simultaneously. The agents share accumulated experience by keeping separate parameters for their own independent learning episodes and the combined experience of all other agents. In sequential learning each agent performs the learning task on its turn while the other agents are idle or assigned for a different task. Only when an agent completed a learning episode (a stopping condition was met), the next agent is allowed to perform the learning task. When all agents complete a learning episode, the first agent is allowed to perform the task again. A sequential learning implementation for real robots is more practical than the parallel one. Parallel implementation for a system in which there are more agents (real robots) than the number of states is not feasible to be designed. This is due to the physical dimension of the robots. If parallel implementation is desired for only small number of robots, then aspects such as collision avoidance should be taken in account. This also involves designing a different reward mechanism (additional rewards should be supplied for each robot for cases of encountering another robot agent rather than a fixed obstacle). Further, the complexity of such a dynamic environment should be taken into account since the state-space will not be deterministic anymore.

Initialize $Q_i(s,a)=0$ where Q is a matrix of size |S|x|A| and set eligibility trace $e_i(s,a)=0$ for all (s,a) $i \in \{1, 2, \ldots K\}$
where K is the number of learning processes ($e.g.$, robots).

Repeat (for each learning episode):
- **Set** initial state s_{i_t} and pick initial action a_{i_t}.
- **Repeat** (for each step t of an episode):
  - **Repeat** (for each learning process i):
    - **Take** action a_{i_t}, observe reward r_{i_t} and the next state $s_{i_{t+1}}$.
    - **Choose** $a_{i_{t+1}}$ for $s_{i_{t+1}}$ using a certain policy ($e.g.$, softmax).
    - $a^*_{i_{t+1}} \leftarrow \arg\max_{a_{i_{t+1}}} Q_i(s_{i_{t+1}}, a_{i_{t+1}})$
    - $\delta_{i_t} \leftarrow r_{i_t} + \gamma_i Q_i(s_{i_{t+1}}, a^*_{i_{t+1}}) - Q_i(s_{i_t}, a_{i_t})$
    - $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow e_{i_t}(s_{i_t}, a_{i_t}) + 1$
  - **For** (s_{i_t}, a_{i_t}):
    - $Q_i(s_{i_t}, a_{i_t}) \leftarrow \max_{i\in K}[Q_i(s_{i_t}, a_{i_t})] + \alpha_{i_t}\delta_{i_t}e_{i_t}(s_{i_t}, a_{i_t})$
    - **If** $a_{i_{t+1}} = a^*_{i+1}$, **Then** $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow \gamma\lambda e_{i_t}(s_{i_t}, a_{i_t})$
      Else $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow 0$
  - $s_{i_t} \leftarrow s_{i_{t+1}}$; $a_{i_t} \leftarrow a_{i_{t+1}}$

Until $i = M$ where M is the maximal number of learning episodes.

where |X| = cardinality of X.

Fig. 4.1 *CQ(λ)*-learning pseudo code for multiple agents

$$Q_i(s_{i_t}, a_{i_t}) \leftarrow \max_{i\in K}[Q_i(s_{i_t}, a_{i_t})] + \alpha_{i_t}\delta_{i_t}e_{i_t}(s_{i_t}, a_{i_t}) \tag{4.4}$$

In equation (4.4), δ_{i_t} is the temporal difference error that specifies how different the new value is from the old prediction, and $e_{i_t}(s_{i_t}, a_{i_t})$ is the eligibility trace that specifies how much a state-action pair should be updated at each time step. When a state-action pair is first visited, its eligibility is set to one. Then at each subsequent time step it is reduced by a factor $\gamma\lambda$. When it is subsequently visited, its eligibility trace is increased by one [MackWorth *et al.*, 1998].

Fig. 4.2 demonstrates a flowchart describing a system that consists of one collaborative agent and multiple independent agents. The collaborative agent learns both from interaction with the environment and from knowledge it gathers from all the independent agents. Each one of the independent agents learns the environment according to the standard $Q(\lambda)$-learning algorithm. $CQ(\lambda)$-learning is not limited for only one collaborative agent, and if desired, it is possible to develop a system that consists of a combination of many $CQ(\lambda)$ learners and many independent $Q(\lambda)$ learners. Another option is to develop a system that consists of only $CQ(\lambda)$ learners. If the assumptions described above are addressed in such systems (in particular to underestimate value functions for all agents during initialization), then there is no risk of overestimation of the agents' value functions.

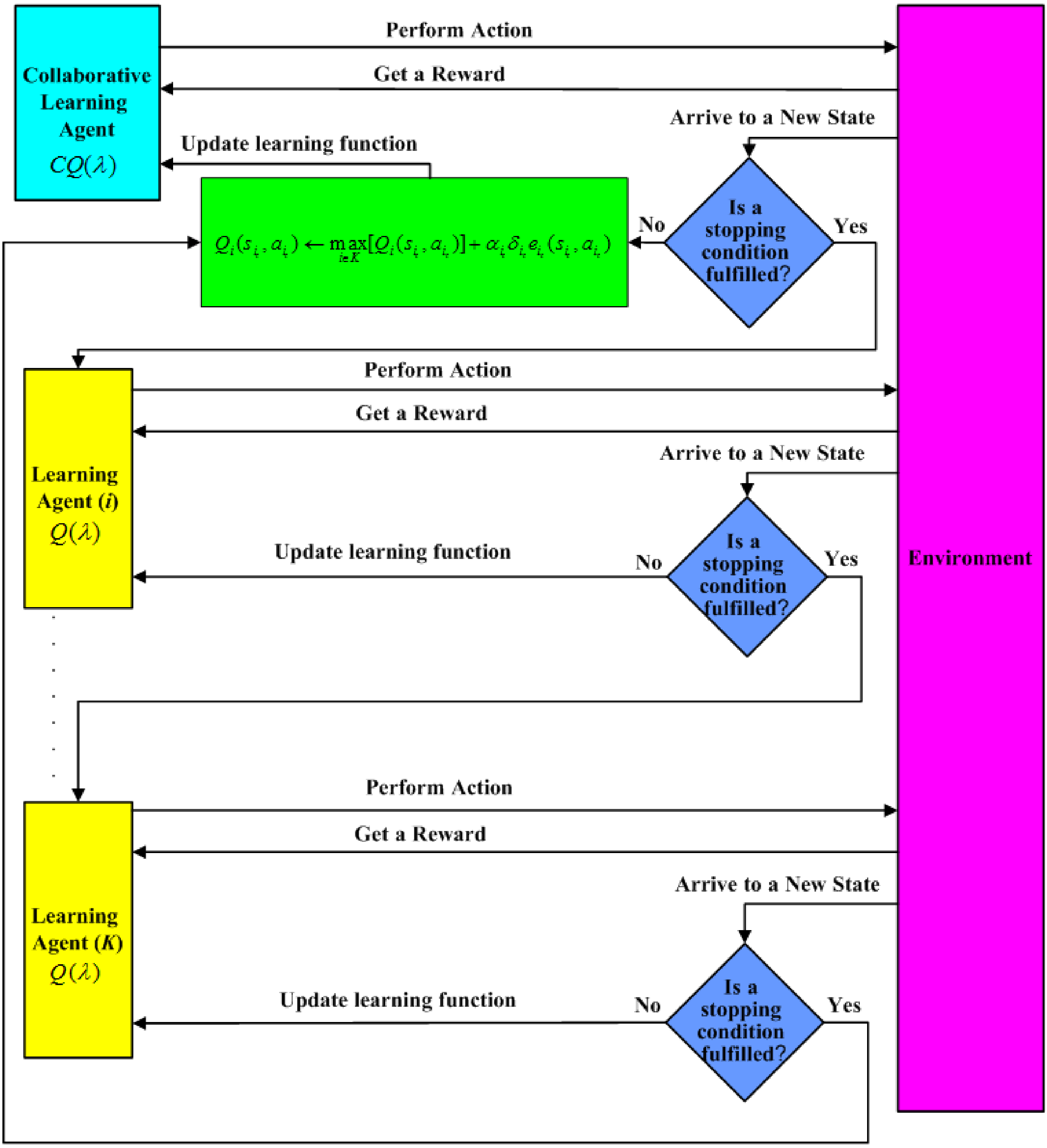


Fig. 4.2 Flowchart for multiple agents *CQ(λ)*-learning with one collaborative learning agent

4.3 CQ(λ)-Learning for Human-Robot Systems

In human-robot systems, when applying $CQ(\lambda)$-learning, the robot learning function acquires state-action values achieved from policies suggested by a human operator (HO) (Fig. 4.3). In this case, a moving average learning performance measure, L_{ave}, is defined over the last N most recent learning episodes (4.5).

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) / N \tag{4.5}$$

where n is the current learning episode, $i = n-N, n-N+1, n-N+2, \ldots n-1$. S_i, a scaler in the range [0, 1] calculated over the last N most recent learning episodes, indicates whether a policy was successful for the i^{th} episode or not. The threshold for a successful episode is defined as $\overline{R}$ (4.6).[1]

$$S_i = \begin{cases} 1 & if \quad R_i > \overline{R} \\ 0 & else \end{cases} \tag{4.6}$$

where R_i is the reward achieved for the i^{th} learning episode. Λ [2] is defined as a minimum acceptable performance threshold, which is compared to the average performance, L_{ave}. If the robot performance fails below the threshold (4.7), the robot switches between fully autonomous operation and semi-autonomous operation and requests human intervention. The procedure is repeated M times where M (set *a-priori*) is the maximal number of learning episodes.

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) / N < \Lambda \tag{4.7}$$

Two levels of collaboration are defined: (i) autonomous - the robot decides which actions to take, acting autonomously, *i.e.*, the robot updates its state-action values according to the standard $Q(\lambda)$ learning algorithm, and (ii) semi-autonomous - the robot requests collaboration with the HO. The HO then suggests an action or a policy and the robot uses the suggestion to replace its own exploration process, *i.e.*, a collaborative $Q(\lambda)$ ($CQ(\lambda)$-learning) is performed. Human-robot collaboration is unnecessary as long as the robot learns policies autonomously and adapts to new states. The HO is required to intervene and suggest alternative policies if the robot reports that its learning performance

---

[1] The threshold value for a successful episode is determined based on empirical tests.

[2] The minimum acceptable performance threshold. Above this value, the human is called to intervene. This threshold value is determined based on empirical tests.

is low (4.7), *i.e.*, the robot switches its learning level from autonomous (self performing) to semi-autonomous (acquiring human guidance) based on its learning performance. Fig. 4.4 shows the $CQ(\lambda)$ -learning algorithm pseudo code for a robot and human.

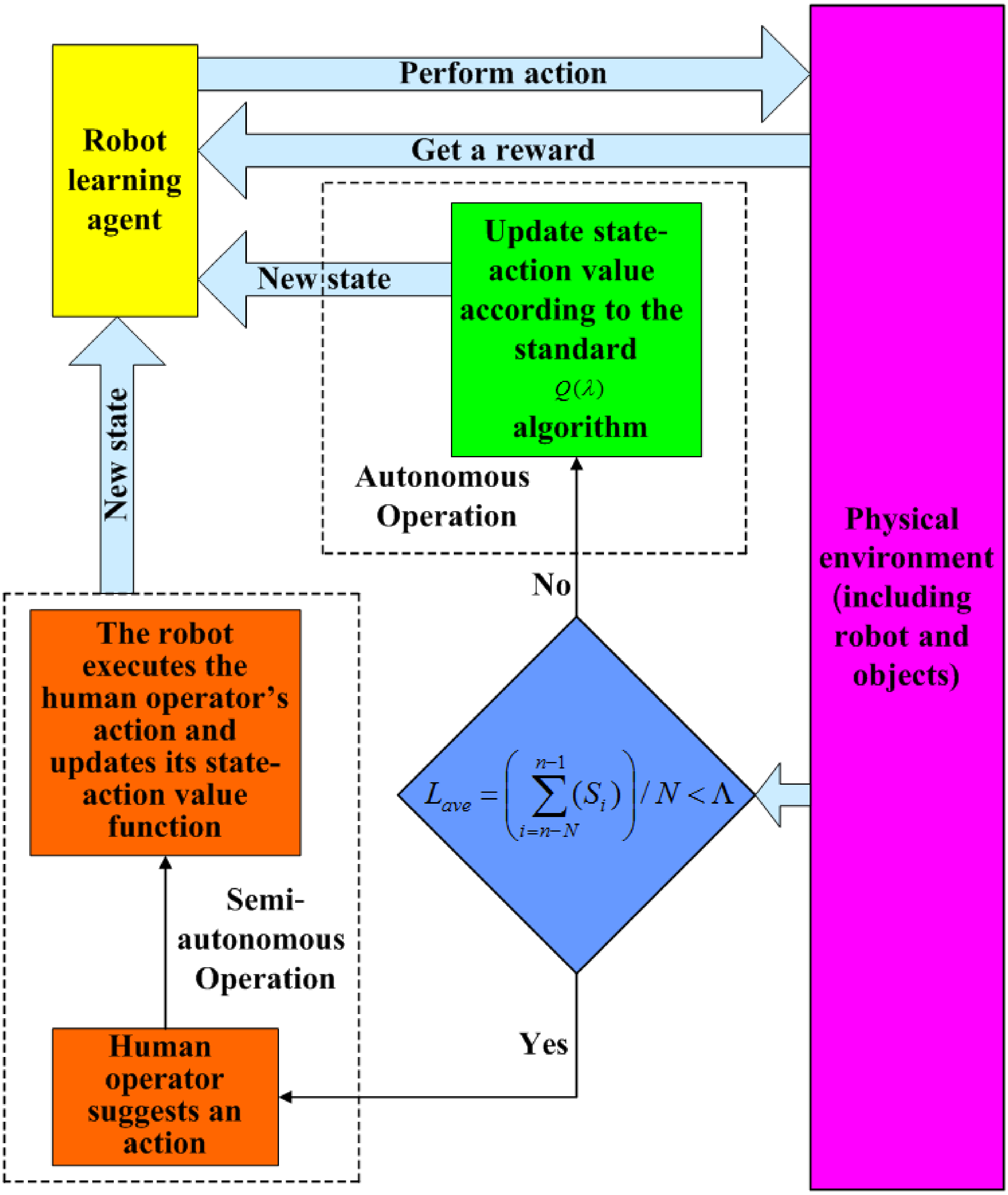


Fig. 4.3 Flowchart for robot and a human operator *CQ(λ)*-learning

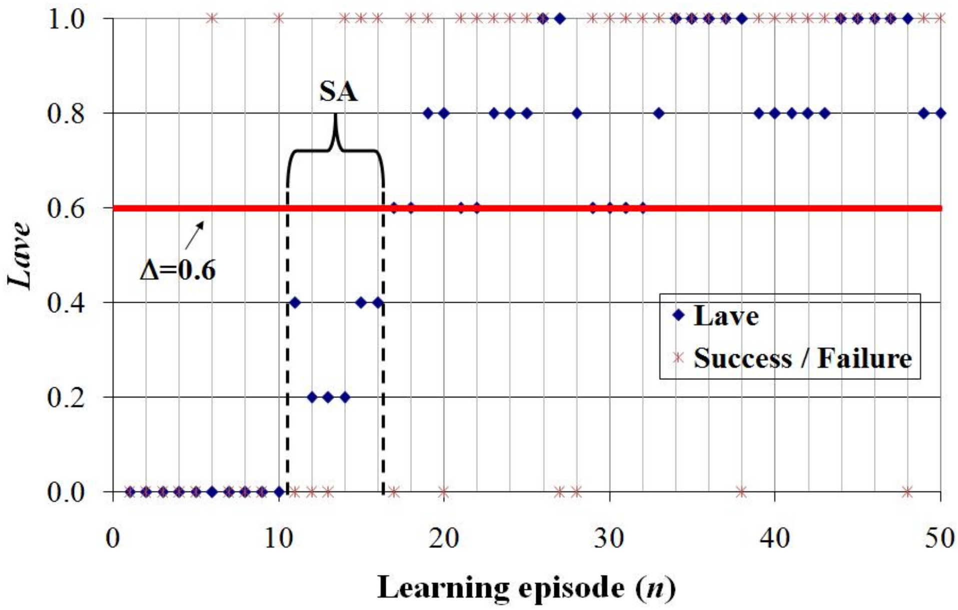
Initialize $Q(s,a)=0$ and eligibility trace $e(s,a)=0$ for matrices of size $|S|x|A|$ for the robot learning process.
Set $n = 1$ for the first learning episode.
Repeat (for each learning episode):
    If $L_{ave} < \Lambda$ use *Action Selection I*,
    Else, use *Action Selection II*.
    Set learning agent to initial state s_t and pick initial action a_t.
    Repeat (for each step of episode):
        Take action a_t, observe reward r_t and the next state s_{t+1}.
        Action Selection I: human operator selects an action a_{t+1} using a subjective maximal function.
        Action Selection II: choose a_{t+1} from s_{t+1} using any action selection rule (*e.g.*, greedy, ε-greedy, softmax, etc.).
        $a^*_{t+1} \leftarrow \arg\max_{a_t} Q(s_{t+1}, a_{t+1})$
        $\delta_t \leftarrow r_t + \gamma Q(s_{t+1}, a^*{t+1})$ - $Q(s_t, a_t)$
        $e_t(s_t, a_t) \leftarrow e_t(s_t, a_t) + 1$
        For all (s_t, a_t):
            $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \delta_t e_t(s_t, a_t)$
            If $a_{t+1} = a^*_{t+1}$, **Then** $e_t(s_t, a_t) \leftarrow \gamma\lambda e_t(s_t, a_t)$
                Else $e_t(s_t, a_t) \leftarrow 0$
        $s_t \leftarrow s_{t+1}$; $a_t \leftarrow a_{t+1}$
    Until $i = M$ where M is the maximal number of learning episodes.

where |X| = cardinality of X.

Fig. 4.4 *CQ(λ)*-learning pseudo code for a human and a robot

An example of changing system autonomy is shown in Fig. 4.5. The example presents variable autonomy that consists of two system levels: (i) autonomy, and (ii) semi-autonomy - acquiring human guidance and suggestions. The robot switches its learning level based on its learning performance. Here, the robot switches its learning level from semi-autonomous (acquiring human guidance) to autonomous (self learning) based on its learning performance. In the example, the acceptable learning performance threshold was set to $\Lambda = 0.6$.[1] The robot keeps measuring its learning performance by averaging its last N[2] most recent learning episodes and comparing this value, L_{ave}, to Λ. In Fig. 4.5, an L_{ave} greater than Λ indicates that the robot's learning performance is high, and the robot learns autonomously. Otherwise, human intervention is allowed, *i.e.*, semi-autonomous learning is performed.

[1] The minimum acceptable performance threshold. Above this value, the human is called to intervene. This threshold value is determined based on empirical tests.
[2] N was set to five.



SA - Semi-autonomous mode

Fig. 4.5 Example of moving learning performance average

A variation of L_{ave}, donated as T_{ave}, may be used. T_{ave} is defined as the learning performance of a system in terms of the average number of steps to reach a goal and get a reward. This is different from L_{ave}, where it is desired to maximize the value of L_{ave}. Instead, *minimal* values of T_{ave} indicate a good learning performance. In this case, the measure ω[1] is compared with the average number of steps to reach the goal, T_{ave}, over the last N most recent learning episodes (4.8).

$$T_{ave} = \left(\sum_{i=t-N}^{t-1} (L_i) \right) / N \tag{4.8}$$

where t is the current learning episode, $i = t-N, t-N+1, t-N+2, ...t-1$, and L_i is the number of steps a learning agent performs at the i^{th} episode. The HO is required to intervene and suggest alternative policies if the robot reports a large average number of steps to reach the goal (4.9).

$$T_{ave} = \left(\sum_{i=t-N}^{t-1} (L_i) \right) / N > \omega \tag{4.9}$$

[1] The maximum acceptable performance threshold in terms of mean number of steps to reach a goal. Above this value, the human is called to intervene. This threshold value is determined based on empirical tests.

4.4 Convergence and Superiority Discussion

Since there is no convergence proof for the $Q(\lambda)$-learning algorithm [Sutton, 1999; Glorennec, 2000], it is claimed here that a convergence proof for $CQ(\lambda)$ where $\lambda > 0$ is also unobtainable. [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994] proved that Q-learning will converge to an optimal policy under certain conditions and showed that if every state-action pair is visited an infinite number of times, Q-learning converges to a unique set of values that define an optimal policy. Since a proof of convergence exists for the Q-learning algorithm, the main scope of this section is to prove the existence of convergence for the $CQ(\lambda)$-learning algorithm for the case where $\lambda = 0$, *i.e.*, $CQ(0)$.

For $CQ(0)$ two cases are considered:

1) A system that consists of one learning agent (a robot) and one human [Kartoun *et al.*, 2006 (a); Kartoun *et al.*, 2006 (b)]:

The robot learns to achieve a goal using Q-learning and measures its learning performance. Q-learning [Watkins, 1989] works by successively improving its evaluations of the quality of particular actions at particular states [Watkins and Dayan, 1992]. [Watkins and Dayan, 1992] prove a convergence theorem for Q-learning and show that it converges to the optimum action-values with probability one as long as all actions are repeatedly sampled in all states and the action-values are represented discretely. [Jaakkola *et al.*, 1994] provide a rigorous proof of convergence for a single Q-learner using techniques of stochastic approximation theory via a new convergence theorem. The Q-learning algorithm given by $Q_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t(s_t, a_t) + \gamma V_t(s_{t+1})]$ converges to the optimal $Q^*(s_t, a_t)$ values if the following conditions apply:

1) The state and action spaces are finite.
2) $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$ uniformly over s_t and a_t with probability one.
3) $Var\{r(s_t, a_t)\}$ is finite.
4) If $\gamma = 1$ all policies lead to a cost free terminal state with probability one.

The proof is based on the observation that the Q-learning algorithm can be viewed as a stochastic process to which techniques of stochastic approximation are applicable. The proof from [Jaakkola *et al.*, 1994] for essential lemmas and theorems is presented in Appendix II. For the $CQ(0)$ human-robot case, the only difference to standard Q-learning is that the exploration policy is changed, it is sometimes determined by the human. The basic Q-learning convergence proof applies as long as the human does not systematically prevent the use of certain actions in particular states. In other words, as long as the autonomous operation still guarantees that every action is executed infinitely often in every state (a condition of the standard convergence proof is met), the convergence proof directly extends to the human-robot interaction case.

In $CQ(0)$, after each learning episode the learning performance is compared with Λ, the minimum acceptable performance threshold above which the human intervention is requested. Since $CQ(0)$-learning is a case of Q-learning where a human is required to intervene in a learning agent activity, if a learning system performance is indicated to be higher than Λ then the system learns a task using pure Q-learning, the convergence of which was already been proved [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994]. For the $CQ(0)$ case where the human is asked to intervene (when system learning performance is low) and his suggestions/selections of actions are not necessarily optimal, $CQ(0)$ will also converge to an optimal solution. Convergence is achieved since the human activities, whether optimal or not, can be considered explorative (actions that have not been tested enough and potentially can produce better solutions). The learning agent then uses these activities to exploit its environment. Of course if the human intentionally and consistently chooses the worst possible actions, the algorithm will converge as well, but slower. The human is assumed to be an expert, and therefore will select beneficial actions. It is reasonable to consider the human for this $CQ(0)$ case as a greedy decision maker at times of human intervention, as opposed to "softmax" [Bakker *et al.*, 2006] or "ε-greedy" [Sutton and Barto, 1998]. Since Q-learning was proved to converge regardless of the action-selection method, $CQ(0)$ will converge to an optimal solution if every state-action pair is visited infinitely often as well. Furthermore, $CQ(0)$ is a special case of Q-learning and therefore, will also converge with probability one.

2) A multiple-agent learning system [Kartoun *et al.*, 2005]:

For this case, the $CQ(0)$ algorithm objective is to accelerate learning in a system composed of multiple learning agents. Learning is based on the state-action value of a collaborative learner being updated according to the maximal value within all other independent learning processes state-action values exist in the learning system (including itself). $CQ(0)$ superiority is based on the proof of convergence for a single agent Q learner [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994]. On the one hand, the problem of multiple agents simultaneously adapting is in general non Markov because each agent provides an effectively non stationary environment for the other agents. Hence, the existing convergence guarantees do not hold, and in general, it is not known whether any global convergence will be obtained, and if so, whether such solutions are optimal [Tesauro and Kephart, 1999]. On the other hand, the superiority for multiple Q-learners ($CQ(0)$) over the standard Q can be demonstrated as described below.

Given a system that consists of $i \in \{1, 2, ...K\}$ learning agents where Q_c is a collaborative learner ($i=1$) and $K-1$ ($K>1$) Q-learners, based on one Q-learner convergence proof [Jaakkola *et al.*, 1994], for each Q-learner $i \in \{1, 2, ...K\}$ (assuming a learning rate of α such $\alpha = \alpha_1 = \alpha_2 =,... = \alpha_K$ and a discount factor γ such $\gamma = \gamma_1 = \gamma_2 =,... = \gamma_K$):

$$Q_{i_{t+1}}(s_{i_t}, a_{i_t}) = (1-\alpha)Q_{i_t}(s_{i_t}, a_{i_t}) + \alpha[r_{i_t}(s_{i_t}, a_{i_t}) + \gamma V_{i_t}(s_{i_{t+1}})] \tag{4.10}$$

$$Q_{i_{t+1}}(s_{i_t}, a_{i_t}) = Q_{i_t}(s_{i_t}, a_{i_t}) + \alpha\left[r_{i_t}(s_{i_t}, a_{i_t}) + \gamma\left(\max_{a_{i_{t+1}}} Q_{i_t}(s_{i_{t+1}}, a_{i_{t+1}})\right) - Q_{i_t}(s_{i_t}, a_{i_t})\right] \tag{4.11}$$

For the collaborative $CQ(0)$ learner (Q_c), the update rule is as follows:

$$\begin{aligned} Q_{c_{t+1}}(s_{c_t}, a_{c_t}) &= Q_{c_t}(s_{c_t}, a_{c_t}) \\ &+ \alpha\left[r_{c_t}(s_{c_t}, a_{c_t}) + \gamma\left(\max_{i \in 1, 2, ..., K} [\max_{a_{i_{t+1}}} Q_{i_t}(s_{i_{t+1}}, a_{i_{t+1}})]\right) - Q_{c_t}(s_{c_t}, a_{c_t})\right] \end{aligned} \tag{4.12}$$

or:

$$\begin{aligned} Q_{c_{t+1}}(s_{c_t}, a_{c_t}) &= Q_{c_t}(s_{c_t}, a_{c_t}) \\ &+ \alpha[r_{c_t}(s_{c_t}, a_{c_t}) + \gamma\left(\max\left[\max Q_{c_t}(s_{c_{t+1}}, a_{c_{t+1}}), \max_{a_{i_{t+1}}, i \in 2, ..., K} Q_{i_t}(s_{i_{t+1}}, a_{i_{t+1}})\right]\right) - Q_{c_t}(s_{c_t}, a_{c_t})] \end{aligned} \tag{4.13}$$

(4.13) can be written:

$$\begin{aligned} Q_{c_{t+1}}(s_{c_t}, a_{c_t}) &= Q_{c_t}(s_{c_t}, a_{c_t}) \\ &+ \alpha\left[r_{c_t}(s_{c_t}, a_{c_t}) + \gamma\left(\max_{a_{i_{t+1}}}[\hat{Q}^*_{c_t}, \hat{Q}^*_{i_t}]\right) - Q_{c_t}(s_{c_t}, a_{c_t})\right] \end{aligned} \tag{4.14}$$

where $\hat{Q}^*_{c_t}$ is an estimator for an optimal Q_c. To maintain superiority of Q_c over standard Q learning agents, $i \in \{2, ...K\}$, (4.15) must hold for all states.

$$r_{c_t}(s_{c_t}, a_{c_t}) + \gamma\left(\max_{a_{i_{t+1}}}[\hat{Q}^*_{c_t}, \hat{Q}^*_{i_t}]\right) - Q_{c_t}(s_{c_t}, a_{c_t}) \leq$$
$$r_{i_t}(s_{i_t}, a_{i_t}) + \gamma\left(\underset{a_{i_{t+1}}}{\hat{Q}^*_{i_t}}\right) - Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.15}$$

Since it is assumed that the agents have an identical state representation of the environment (4.16) holds for any state.

$$r_{c_t}(s_{c_t}, a_{c_t}) = r_{i_t}(s_{i_t}, a_{i_t}) \tag{4.16}$$

Thereby (4.15) can be written as (4.17):

$$\gamma\left(\max_{a_{i_{t+1}}}[\hat{Q}^*_{c_t}, \hat{Q}^*_{i_t}]\right) - Q_{c_t}(s_{c_t}, a_{c_t}) \leq \gamma\left(\underset{a_{i_{t+1}}}{\hat{Q}^*_{i_t}}\right) - Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.17}$$

To assure that (4.17) holds, two constants C_1 and C_2 are defined such as $C_1 = \gamma\left(\max_{a_{i_{t+1}}}[\hat{Q}^*_{c_t}, \hat{Q}^*_{i_t}]\right) - Q_{c_t}(s_{c_t}, a_{c_t})$ and $C_2 = \gamma\left(\underset{a_{i_{t+1}}}{\hat{Q}^*_{i_t}}\right) - Q_{i_t}(s_{i_t}, a_{i_t})$ such that $C_1 \geq 0$ and $C_2 \geq 0$.

<u>Two cases are possible</u>:

<u>Case I</u>:
During an iteration:

$$\hat{Q}^*_{c_t} < \hat{Q}^*_{i_t} \tag{4.18}$$

where (4.17) is written as (4.19):

$$\gamma\hat{Q}^*_{i_t} - Q_{c_t}(s_{c_t}, a_{c_t}) \leq \gamma\hat{Q}^*_{i_t} - Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.19}$$

or:

$$Q_{c_t}(s_{c_t}, a_{c_t}) \geq Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.20}$$

Fix some positive constant Δ_1:

$$Q_{c_t}(s_{c_t}, a_{c_t}) = Q_{i_t}(s_{i_t}, a_{i_t}) - \Delta_1 \tag{4.21}$$

If during the iteration, $Q_{i_t}(s_{i_t}, a_{i_t}) > Q_{c_t}(s_{c_t}, a_{c_t})$, then $Q_{i_t}(s_{i_t}, a_{i_t})$ is reduced in the amount of at least Δ_1. This is a sufficient condition to to assure that (4.17) holds, *i.e.*, $CQ(0)$ is superior.

<u>Case II</u>:

During an iteration:

$$\hat{Q}^*_{c_t} \geq \hat{Q}^*_{i_t} \tag{4.22}$$

where (4.17) is written as (4.23):

$$\gamma\hat{Q}^*_{c_t} - Q_{c_t}(s_{c_t}, a_{c_t}) \leq \gamma\hat{Q}^*_{i_t} - Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.23}$$

or:

$$\gamma(\hat{Q}^*_{c_t} - \hat{Q}^*_{i_t}) \leq Q_{c_t}(s_{c_t}, a_{c_t}) - Q_{i_t}(s_{i_t}, a_{i_t}) \tag{4.24}$$

Using Δ_1 from Case I, results (4.25):

$$\gamma(\hat{Q}^*_{c_t} - \hat{Q}^*_{i_t}) \leq \Delta_1 \tag{4.25}$$

Fix some constant Δ_2:

$$\Delta_2 = \gamma(\hat{Q}^*_{c_t} - \hat{Q}^*_{i_t}) \tag{4.26}$$

During any iteration, keeping $\Delta_2 \leq \Delta_1$ is a sufficient condition to assure that (4.17) holds, *i.e.*, $CQ(0)$ is superior.

5. Multiple Mobile Robot Navigation

Chapter Overview

$CQ(\lambda)$-learning algorithm is implemented in a simulation consisting of several robots.[1] Such a version of collaborative learning involving the navigation of robots is based on maximizing the state-action values of a collaborative agent by using interaction with the environment and by gathering information from other agents that exist in the system.

5.1 Introduction

This section presents the implementation of the $CQ(\lambda)$-learning algorithm enabling learning agents to acquire knowledge from each other. Acquiring knowledge learned by an agent via collaboration with other agents is expected to accelerate learning performance. In implementing the $CQ(\lambda)$ algorithm, the approach described in [Smart, 2002] is applied: the learning rate (a parameter that controls how much weight is given to the current reward, as opposed to the old Q estimate) of each agent $i \in \{1, 2, ...N\}$ for each state-action pair, $\hat{\alpha}_{i_{(s_t,a_t)}}$, is reduced adaptively over time. This is done independently for each state-action pair using the number of times it has been updated previously, $c_{i_{(s_t,a_t)}}$. The effective learning rate at time step t, $\alpha_{i_{(s_t,a_t)}}$, is then determined (5.1).

$$\alpha_{i_{(s_t,a_t)}} = \frac{\hat{\alpha}_{i_{(s_t,a_t)}}}{c_{i_{(s_t,a_t)}}} \tag{5.1}$$

Smart's idea is based on the principle that the more a certain state-action pair is selected, the less it is modified in response to any particular experience.

In [Sutton and Barto, 1998] it is stated "although ε-greedy action selection is an effective and popular means of balancing exploration and exploitation [*e.g.*, Guo *et al.*, 2004; Meng *et al.*, 2006], one drawback is that when it explores it chooses equally among all actions. Hence, it is as likely to choose the worst action as it is the next-to-best action. The obvious solution is to vary the action probabilities as a graded function of estimated value." One way to do that is to choose action a_t with a probability that depends on the value of $Q(s_t, a_t)$. This is known as "softmax" action selection. A common method is to use a Gibbs or Boltzmann distribution (example is shown in Appendix VIII), where the probability of choosing action a_t at state s_t is proportional to Q. This probability is denoted $P(a_t, s_t)$ (5.2).

[1] The terms agent and robot will be used interchangeably.

$$P(a_t \mid s_t) = \frac{e^{Q(s_t, a_t)/T}}{\sum_{a_{t+1} \in A} e^{Q(s_t, a_{t+1})/T}} \tag{5.2}$$

where T is a positive parameter that specifies how randomly values should be chosen. When T is high, all actions have the same likelihood of being chosen. As T decreases, the highest valued actions are more likely to be chosen, and at the limit $T \rightarrow 0$ the best action is always chosen [MackWorth *et al.*, 1998].

5.2 Task Definition

Several robots learned to navigate a two dimensional world that contains undesirable areas[1] with the objective of choosing the optimum path to reach a target. The agents learned the environment sequentially (Fig. 4.1 and Fig. 4.2), *i.e.*, a collaborative agent, Q_c, performs a learning episode, then another independent $Q(\lambda)$ agent performs a learning episode etc. After all agents perform one learning episode the collaborative agent uses both the experience it gained from the environment and from the other agent(s), and then a new sequence of learning episodes begins. The task's goal for the collaborative robot agent is to decrease its learning time relative to that of independent $Q(\lambda)$-learning agents [Kartoun *et al.*, 2005].

5.3 Experimental Setup

Robot states include robot locations in an 11 × 11 two dimensional world (Fig. 5.1). Robot learning agents $i \in \{1, 2, ...K\}$ can navigate the world where Q_c is a collaborative learner ($i = 1$) and $K - 1$ ($K > 1$) are $Q(\lambda)$ agents. Q_c learns via both: (i) acquiring experience and interaction with the environment and (ii) acquiring experience from the other $Q(\lambda)$ learners that learn the environment independently according to the standard $Q(\lambda)$ learning algorithm; this is done by taking the maximal Q value of all agents that exist in the system. An agent's interaction with the environment is performed by getting rewards and punishments, *i.e.*, if an agent reaches the target, the reward is +1, and if it passes through an undesirable area, the reward is -1. A learning episode describes the placing of an agent at a starting state in the environment and letting it reach the target.

[1] Undesirable area - an area where a robot can physically pass through but it is not recommended.

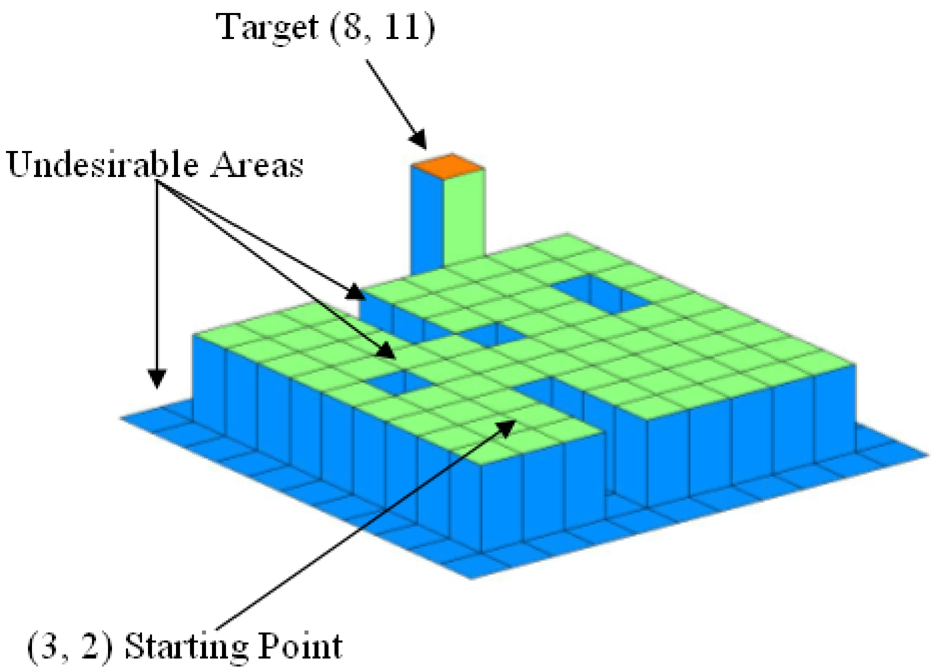


Fig. 5.1 An 11 × 11 two dimensional world

The world consists of three layers: (i) environmental cells - areas through which the robot should navigate on its way to the target, (ii) undesirable areas - reduced cells, and (iii) target - an elevated cell (Fig. 5.1). A robot can move from a cell to any one of its four adjacent neighbors with the restriction that it cannot move out of the world. The task of the learning agents is to learn to navigate the world by choosing optimal routes to a known target using what they learn from the environment and exploiting knowledge sharing.

The simulated system was evaluated using two experimental setups. In both setups one collaborative agent learns both from interaction with the environment and from one or more agents. The other agents learn only by interacting with the environment according to the standard $Q(\lambda)$-learning algorithm. Further, in both setups the number of learning episodes was set to 100, *i.e.*, the algorithm stops after an agent navigates from a starting point to the target 100 times. System performance was evaluated by choosing a starting point for each of the agents (coordinates (3, 2)). For this state, the optimal route length for traveling to the target at coordinates (8, 11) was found to be 14 steps (Fig. 5.1). An optimal solution consists of finding the shortest path to the target from the coordinates (3,2) while a convergence to a near optimal solution is measured by averaging a history of ten most recent trials and was set arbitrarily at twenty. The setups are described as follows:

1) Experimental setup I:

This experiment contains two robot agents where the first learns according to $CQ(\lambda)$ and the second learns according to the $Q(\lambda)$ algorithms. The following parameters were set: $\alpha_1 = \alpha_2 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = 0.99$, and $\lambda_1 = \lambda_2 = 0.5$.

2) Experimental setup II:

This experiment contains three agents, the first of which learns according to $CQ(\lambda)$ while the other two learn according to the $Q(\lambda)$ algorithm. The setup was set with the following parameters: $\alpha_1 = \alpha_2 = \alpha_3 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = \gamma_3 = 0.99$, and $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$ (Fig. 4.1 and Fig. 4.2).

In the above experimental setups, as in [Kaelbling *et al.*, 1996; Abramson and Wechsler, 2003], the learning rate for all agents was set to be considerately high (0.95) and was slowly decreased. In early stages of learning, a greater α is desired, this enables the agents to explore the environment. As the learning procedure proceeds, α is decreased to let the agents exploit the environment more often. As in [Clause, 1996], the discount rate (γ) was set to 0.99 for all agents. This makes the agents to give a higher significance to future rewards. As in [Abramson and Wechsler, 2003], the eligibility trace (λ) was set to 0.5. The 0.5 value was chosen to let a long sequence of values of state-action pairs to be updated while keeping the computational time to perform the algorithm reasonable.

5.4 Results and Discussion

Experiments based on fifty simulation runs were conducted for both setups. An example for state values of one of the runs is given in (Fig. 5.2). Additional examples are presented in Appendix XIII.

Fig. 5.3 illustrates fifty simulation runs[1] performed for the convergence of one $CQ(\lambda)$ collaborative agent and one $Q(\lambda)$ independent agent. The figure clearly shows that the learning curve of the collaborative agent is below the learning curve of the independent agent, *i.e.*, the collaborative agent's performance was superior to that of the independent agent. Additionally, learning curves are shown for the collaborative and independent agents, and learning rates[2] were calculated as 0.6 and 0.58 respectively.

[1] One simulation run contains 100 learning episodes. Each learning episode consists of placing the robot at a starting location in the environment. The robot explores the environment. A learning episode ends when the robot reaches the target. The number of steps a learning episode consists of is denoted "the path length."

[2] The learning rate parameter determines how significantly an agent improves. The reader should make a distinction between the learning rate evaluation performance measure which indicates the improvement of learning and α, the RL learning rate parameter.

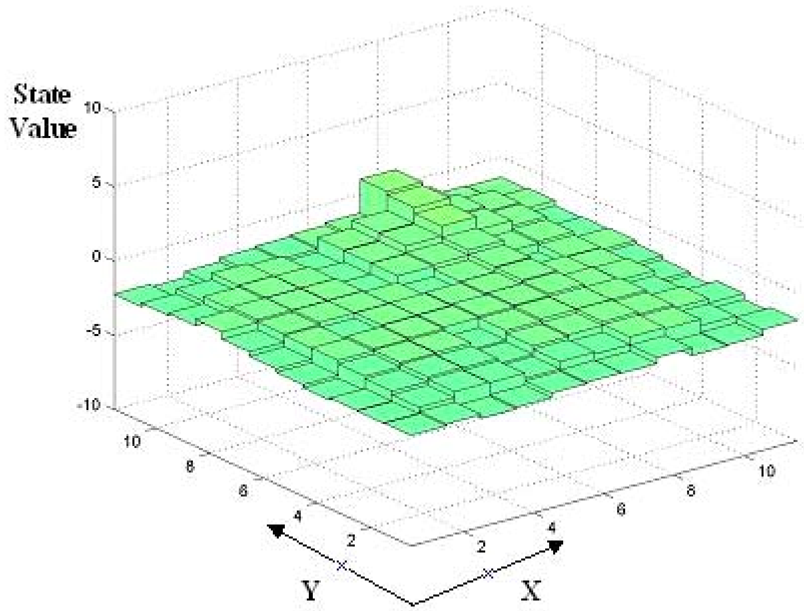


Fig. 5.2 An 11 × 11 world state-value map after 100 learning episodes

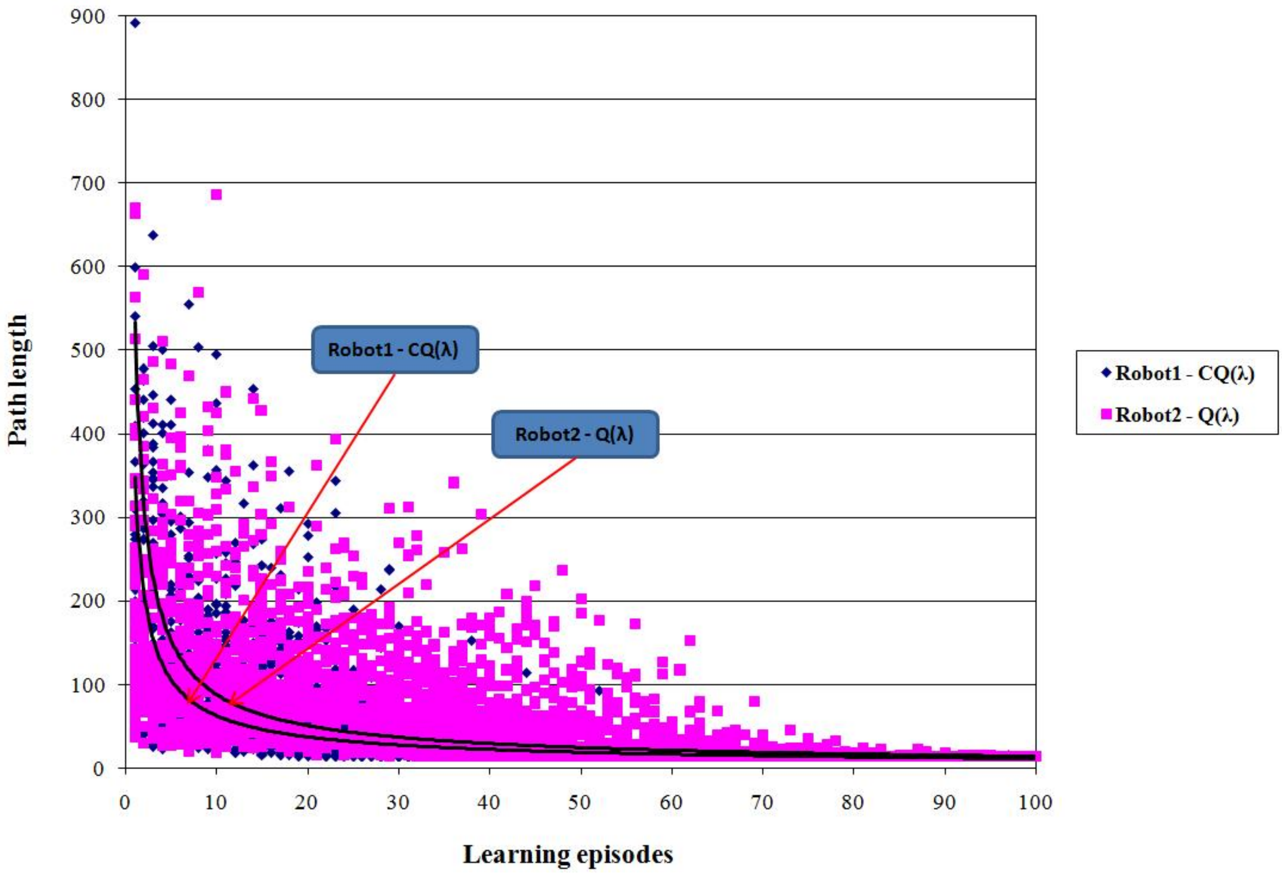


Fig. 5.3 Fifty simulation runs for convergence of two robots

To strengthen the statement that the collaborative agent is superior, Fig. 5.4 is presented. Fig. 5.4 compares the performance of the collaborative agent to the independent agent. It presents the percent of better or equal collaborative learning performance for each learning episode (one to 100). This is achieved by comparing individual learning episode performances of the fifty simulation runs. It is clearly seen here that only in a very few cases (at early stages of learning), the independent agent performed better.

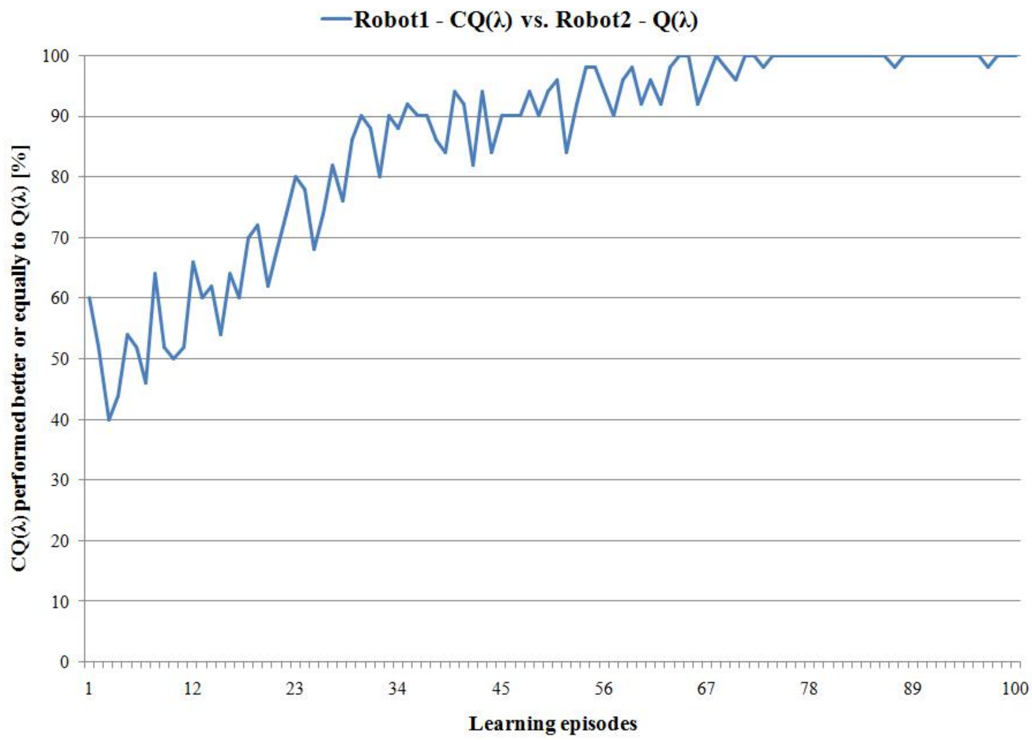


Fig. 5.4 [%] of simulations that *CQ(λ)* learner performed better or equally to *Q(λ)* learner

Similarly, fifty simulation runs for convergence of one $CQ(\lambda)$ collaborative agent and two $Q(\lambda)$ independent agents are shown in Fig. 5.5. The collaborative agent's superiority over the independent agents is again clearly shown. Learning curves are also shown for the collaborative and independent agents and learning rates were calculated as 0.6, 0.58, and 0.58 respectively. More detailed graphs are shown in Appendix XIV.

To strengthen the statement that the collaborative agent is superior to the two independent agents, Fig. 5.6 is presented. Fig. 5.6 compares the performance of the collaborative agent to the two independent agents. It presents the percent of better or equal collaborative learning performance for each learning episode (one to 100). This is achieved by comparing individual learning episode performances of the fifty simulation runs. It is clearly seen here that only in a very few cases (at early stages of learning), the independent agents performed better.

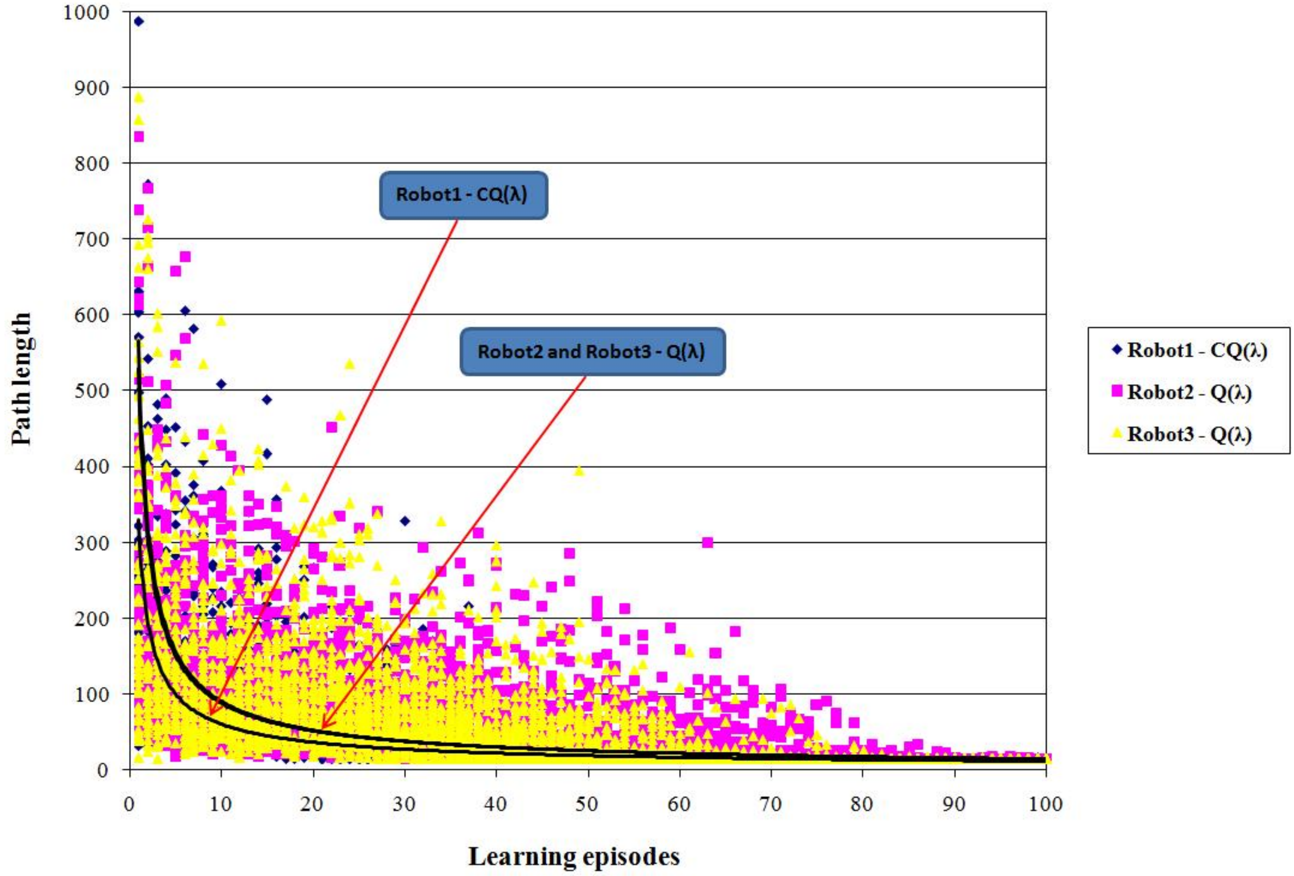


Fig. 5.5 Fifty simulation runs for convergence of three robots

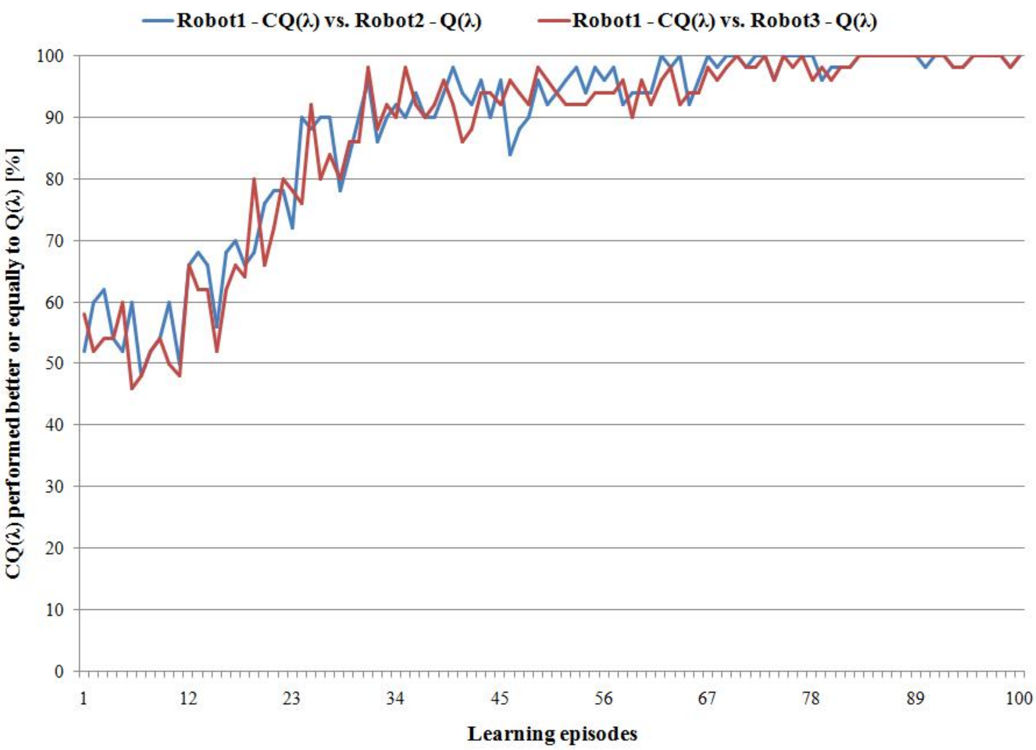


Fig. 5.6 [%] of simulations that *CQ(λ)* learner performed better or equally to *Q(λ)* learners

Based on the results shown in Table 5.1, a *T*-test was conducted. Ten hypotheses were evaluated to test the difference between the setups based on the mean number of steps required to converge to optimality and to near optimality (Section 3.4.1). The hypotheses are shown in Table 5.2.

Table 5.1 Summary of results for the multiple agents navigation task

	Experimental setup I		Experimental setup II		
	R_1*	R_2*	R_1*	R_2*	R_3*
Learning strategy	***CQ(λ)***	***Q(λ)***	***CQ(λ)***	***Q(λ)***	***Q(λ)***
Mean / standard deviation of steps to converge to optimality	**56.8 / 13.6**	68.4 / 14.0	**55.2 / 14.1**	68.9 / 13.9	69.4 / 14.4
Mean / standard deviation of steps to converge to near optimality	**37.7 / 7.2**	56.2 / 10.1	**36.8 / 6.5**	56.2 / 15.6	56.6 / 11.2

* R_1 - Collaborative agent, R_2 and R_3 are independent *Q(λ)* agents.

Table 5.2 *CQ(λ)* for multiple agents - performance evaluation hypotheses

Evaluation	Performance Measure*	Hypothesis**
Evaluation within experimental setup I learning agents	CO	H_{10}: There is no difference between R_1 and R_2. H_{11}: There is a difference between R_1 and R_2.
	CNO	H_{20}: There is no difference between R_1 and R_2. H_{21}: There is a difference between R_1 and R_2.
Evaluation within experimental setup II learning agents	CO	H_{30}: There is no difference between R_2 and R_3. H_{31}: There is a difference between R_2 and R_3.
	CO	H_{40}: There is no difference between R_1 and R_2. H_{41}: There is a difference between R_1 and R_2.
	CO	H_{50}: There is no difference between R_1 and R_3. H_{51}: There is a difference between R_1 and R_3.
	CNO	H_{60}: There is no difference between R_2 and R_3. H_{61}: There is a difference between R_2 and R_3.
	CNO	H_{70}: There is no difference between R_1 and R_2. H_{71}: There is a difference between R_1 and R_2.
	CNO	H_{80}: There is no difference between R_1 and R_3. H_{81}: There is a difference between R_1 and R_3.
Evaluation between experimental setups I and II	CO	H_{90}: There is no difference between the performance of one collaborative learner while learning from one or two robot agents. H_{91}: There is a difference between the performance of one collaborative learner while learning from one or two robot agents.
	CNO	H_{100}: There is no difference between the performance of one collaborative learner while learning from one or two robot agents. H_{101}: There is a difference between the performance of one collaborative learner while learning from one or two robot agents.

* CO - Convergence to optimality, CNO - Convergence to near optimality.
** R1 - Collaborative agent, R2 and R3 are independent *Q(λ)* agents.

Null hypotheses H_{10} and H_{20} were rejected with *P*-values of $2.86 \cdot 10^{-5}$ and $1.18 \cdot 10^{-17}$, respectively, which indicates that the mean number of steps to converge to optimality/near optimality of the learning agents are not equal. Null hypothesis H_{30} was not rejected (*P*-value of 0.43), which signifies that it can not be ruled out with sufficient confidence that the performance of the two learning agents is different. Null hypotheses H_{40} and H_{50} were rejected with P-values of $2.16 \cdot 10^{-6}$ and $1.41 \cdot 10^{-6}$, respectively, showing that there is a difference between the mean number of steps to converge to optimality/near optimality of the learning agents. The null hypothesis H_{60} was not rejected (P-value of 0.44), and is again indicative that it can not be ruled out with sufficient confidence that the performance of the two learning agents is different. Null hypotheses H_{70} and H_{80} were rejected with P-values of $9.9 \cdot 10^{-12}$ and $1.65 \cdot 10^{-17}$, respectively, showing that there is a difference between the means of number of steps to converge to optimality/near optimality of the learning agents. The mean number of learning episodes over fifty simulation runs for convergence of two and three robots is presented in Fig. 5.3 and Fig. 5.5 respectively. Based both on an evaluation of the hypotheses H_{10} through H_{80} and on Table 5.1, the robot using the $CQ(\lambda)$ algorithm has faster learning performance while converging either to near optimality or to optimality relative to robots that use the $Q(\lambda)$ algorithm. Also, looking at Fig. 5.3 and Fig. 5.5, note that initially, at the beginning of the experiment, the robots require a large number of steps to succeed at learning because at that time the agents lack sufficient knowledge about the world. As learning proceeds and the agents gain more knowledge, the number of steps drops dramatically. For evaluating whether adding a third learning agent improves learning, hypotheses nine and ten were tested (Table 5.2). Null hypotheses H_{90} and H_{100} were not rejected with P-values of 0.29 and 0.27, respectively. For both cases, it can not be ruled out with sufficient confidence that the performance of the two learning agents is different.

To test whether there is a significant difference between the learning rates[1] of each learning agent for both experimental setups, the following hypotheses were analyzed using a T-test (Table 5.3).

Table 5.3 *CQ(λ)* for multiple agents - learning rates evaluation hypotheses

Evaluation	Hypothesis*
Evaluation within experimental setup I learning agents	H_{110}: There is no difference between L_{R_1} and L_{R_2}. H_{111}: L_{R_1} is larger than L_{R_2}.
Evaluation within experimental setup II learning agents	H_{120}: There is no difference between L_{R_1} and L_{R_2}. H_{121}: L_{R_1} is larger than L_{R_2}.
	H_{130}: There is no difference between L_{R_1} and L_{R_3}. H_{131}: L_{R_1} is larger than L_{R_3}.
	H_{140}: There is no difference between L_{R_2} and L_{R_3}. H_{141}: There is a difference between L_{R_2} and L_{R_3}.

* L_{R_1} - Learning rate of the collaborative agent, L_{R_2} and L_{R_3} are learning rates of the independent *Q(λ)* agents

[1] The learning rate parameter determines how significantly an agent improves. The reader should make a distinction between the learning rate evaluation performance measure which indicates the improvement of learning and α, the RL learning rate parameter.

For experimental setup I, null hypothesis H_{110} was rejected with a P-value of 0.023, which indicates that the learning rates are not equal. For experimental setup II, null hypotheses H_{120} and H_{130} were rejected with P-values of 0.03 and $1.73 \cdot 10^{-3}$, respectively, which indicates that the two independent learning agents (L_{R_2} and L_{R_3}) has different learning rate values compared with that of the collaborative agent (L_{R_1}). An additional hypothesis (H_{140}) was tested to check whether the learning rates of the independent agents (L_{R_2} and L_{R_3}) are equal. With a P-value of 0.174, H_{140} was not rejected, indicating that it can not be ruled out with sufficient confidence that there is a difference between the independent agents' learning rates.

5.5 Summary

The simulation demonstrated the positive effect collaboration between several learning agents had on learning speed. The different experiments each comprised fifty simulation runs and a variety of robot combinations. Two robots compared according to their performance using the $Q(\lambda)$ algorithm showed an average improvement of 17.02% for the number of learning steps required to reach definite optimality and an average improvement of 32.98% for convergence to near optimality. Significant statistical differences were observed for both convergence to optimality and convergence to near optimality while comparing two robots, the first using $CQ(\lambda)$ and the second using $Q(\lambda)$. However, while using three robots, the first using the $CQ(\lambda)$ and the second and third using $Q(\lambda)$, no statistically significant differences were noted in either convergence to optimality or convergence to near optimality while comparing the $Q(\lambda)$-based robots' learning performance. But statistically significant differences were found in both convergence to optimality and convergence to near optimality while comparing the $CQ(\lambda)$-based robot to the other two. Additionally, no statistically significant differences were observed for either convergence to optimality or convergence to near optimality using a $CQ(\lambda)$-based robot learning in either two robot or three robot environments. In conclusion, the $CQ(\lambda)$ algorithm proved superior to the $Q(\lambda)$ algorithm for both setups. Furthermore, the performance of the collaborative agent did not improve when the number of independent learning agents was increased from one to two.

In terms of agent learning rate (the lower the learning rate, the greater the improvement in learning), the independent agents showed better improvements in learning than did the collaborative agent for both of the experimental sets described. Although the collaborative agent learns faster than the independent agents and reaches an optimal solution faster, the independent agents' improvement of learning is faster. This is, of course, reasonable since the independent agents learn less efficiently than the collaborative agent during the early stages of learning and because all agents (collaborative and independent) eventually converge to the same optimal solution (after many episodes); therefore, the independent agents must "catch up" with the collaborative agent.

In the multiple-agents systems described in this work, only instances of one $CQ(\lambda)$ learner per system were described while the others were independent $Q(\lambda)$ learners. Since it was shown that there is no advantage in using a system that contains more than two agents, the development of a system that contains two collaborative agents should be considered as an interesting future research topic. If such a system would follow the conditions described in Section 4.2, it would be expected to achieve a better solution than for the scenarios described in this work, resulting also an identical optimal policy for both $CQ(\lambda)$ learners.

6. Bag Shaking Experiment with a Fixed-Arm Robot

Chapter Overview

A robot can acquire a policy suggestion from human about how to empty the contents of a bag. In this chapter, the learning task described is to observe the position of a suspicious plastic bag located on a platform, grasp it with a robot manipulator, and shake out its contents. The $CQ(\lambda)$ algorithm is applied to this learning task.[1]

6.1 Introduction

The usual method for bomb squad personnel that encounter a suspicious bag is to blow up the bag and its contents, which may be explosives. However, if the bag contains chemical, biological, or radiological material, this method can have disastrous results. Furthermore, the "blow-up" method also destroys important clues such as fingerprints, type of explosive, detonators, and other useful evidence for subsequent forensic analysis. This section addresses an alternative to the conventional method, entailing extraction of the bag's contents using a robot, thus avoiding the problems outlined above [Kartoun, 2003; Edan *et al.*, 2004; Kartoun *et al.*, 2004].

For a robot to empty the contents of a bag, one side of its gripper must slide under the bag, but because of slippage, the robot's grasp may fail. Thus, it is important to know the type of bag and the location of the opening before the grasp point assessment is made. Moreover, the robot's grasp may also fail because of the soft, unknown surface texture of the object considered here. Once the bag type is known, a rule set specific to that type of bag is evoked to determine the best robot arm shake trajectories to discharge the contents of the bag for subsequent inspection. In a robotic bag inspection system, it is advantageous to automate bag classification, which is coupled to robotic tactics such as shaking out the bag's contents. Therefore, a multi-category bag classification for four bag classes was designed using support vector machines (SVMs). By finding a set of optimal features representing a bag, a classification rate of 96.25% was obtained for a polynomial kernel of degree nine (Appendix IV.) [Kartoun *et al.*, 2006 (c)]. A Motoman UP-6 fixed-arm-based robot learning system was developed for comparing the traditional $Q(\lambda)$ learning algorithm with the $CQ(\lambda)$ learning algorithm using the suspicious plastic bag type.

[1] Although other alternatives are available for solving the proposed security problem, such as cutting open the bag or sliding the objects to be inspected out of the bag, the application was selected to serve as a test-bed for the $CQ(\lambda)$-learning algorithm.

6.2 Task Definition

The learning task is to observe the position of a plastic bag located on an inspection surface, grasp it with a fixed-arm robot, learn how to shake out its contents in minimum time. This is done via both interaction with the environment and policies suggested by a human operator (HO).

6.3 Experimental Setup

6.3.1 Introduction

Three experimental setups were designed to compare the $Q(\lambda)$ and $CQ(\lambda)$-learning algorithms: (i) the rewards are calculated based on a linear function and measured manually, and the human is allowed to intervene and change the learning state-space by adjusting speeds and adjacent state distances of the robot's X, Y and Z axes; (ii) the rewards are calculated automatically using a digital scale and are based on a cumulative reward function, and similar to the first experimental setup, the human is allowed to intervene and change the learning state-space by adjusting speeds and adjacent state distances of the robot's X, Y and Z axes; and (iii) the rewards are calculated automatically using a digital scale, are based on an events-based reward function, and the human is allowed to intervene directly in the system Q table done by using an interface designed assume control of the different swing weights over the robot's X, Y and Z axes.

The system has no *a-priori* knowledge regarding the most efficient shaking policy for any given plastic bag, but it learns this information from interaction with the environment and from human guidance. For the three experimental setups, robot states denoted as $s_t \in S$ (Table 6.1) and pertain to its gripper location in a three-dimensional grid (Fig. 6.1). The performance of the task is a function of a set of actions, $a_t \in A$, for each physical state of the system.

Table 6.1 States description of the three-dimensional grid - the bag shaking task

State(s)	Description	Number of states	Number of possible actions from a state
$s(Center)_t$	State center	1	18
$s(X_{3-})_t$, $s(X_{2-})_t$, $s(X_{1-})_t$, $s(X_{1+})_t$, $s(X_{2+})_t$, $s(X_{3+})_t$	States where the robot can move over its X axis	6	2
$s(Y_{3-})_t$, $s(Y_{2-})_t$, $s(Y_{1-})_t$, $s(Y_{1+})_t$, $s(Y_{2+})_t$, $s(Y_{3+})_t$	States where the robot can move over its Y axis	6	2
$s(Z_{3-})_t$, $s(Z_{2-})_t$, $s(Z_{1-})_t$, $s(Z_{1+})_t$, $s(Z_{2+})_t$, $s(Z_{3+})_t$	States where the robot can move over its Z axis	6	2

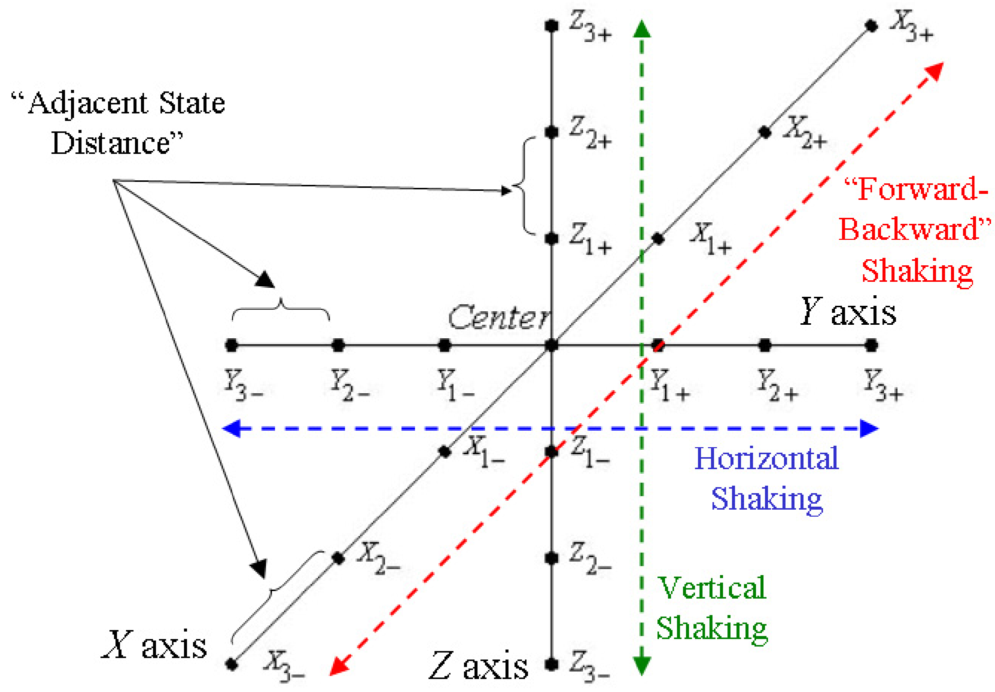


Fig. 6.1 Bag shaking task state-space

An action, a_t, consists of a robot movement from a point (X, Y, Z) along a single coordinate direction. The Y axis is defined as the horizontal shaking axis, *i.e.*, actions are performed in parallel to the horizon (left to right). The X axis is defined as the "In and Out" axis. Similarly to the Y axis, actions are performed in parallel to the horizon, but differently, the X axis is perpendicular to the Y axis. Over the Z axis, actions are performed vertically to the horizon (up to down). The robot starts a shaking policy from the $s_{(center)_t}$ state located above the inspection surface. From $s_{(center)_t}$ it can move in the direction of any of the three coordinates reaching any of the other 18 states. The distance (denoted "adjacent state distance") between any two close states is set *a-priori* to performing a shaking policy (*e.g.*, distances between $s_{(Z_{3-})_t}$ and $s_{(Z_{2-})_t}$ or between $s_{(center)_t}$ and $s_{(Z_{1+})_t}$ are 30 mm). From any robot state other than $s_{(center)_t}$, the robot is limited to either symmetrical actions or returning to the center position (*e.g.*, from state $s_{(X_{2-})_t}$ the robot can move only to $s_{(center)_t}$ or to $s_{(X_{2+})_t}$).[1]

Human-robot collaboration is unnecessary as long as the robot learns policies and adapts to new states without a serious deterioration in its performance. When the robot exhibits low learning performance (6.1), the HO is required to intervene and suggest alternative shaking speeds and adjacent state distances, which are then acquired by the robot's learning function:

---

[1] This limitation was set to keep the state-space size of the task in a reasonable size. Enabling actions such as moving diagonally from one axis to another would prevent the accomplishment of performing experiments within a reasonable time if convergence to optimal solutions is desired.

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) / N < \Lambda \qquad (6.1)$$

where n is the current learning episode, $i = n-N, n-N+1, n-N+2, ...n-1$. S_i, a scaler in the range [0, 1] calculated over the last N most recent learning episodes, indicates whether a policy was successful for the i^{th} episode or not. Based on empirical tests, the threshold for a successful episode was set to $\overline{R} = 25$ (6.2).

$$S_i = \begin{cases} 1 & if \quad R_i > \overline{R} \\ 0 & \quad else \end{cases} \qquad (6.2)$$

where R_i is the reward achieved for the i^{th} learning episode. Λ is defined as a minimum acceptable performance threshold, which is compared to the average performance, L_{ave}. If the robot performance fails below the threshold (4.7), the robot switches between fully autonomous operation and semi-autonomous operation and requests human intervention. The procedure is repeated M times where M (set *a-priori*) is the maximal number of learning episodes.

Two levels of collaboration are defined: (i) autonomous - the robot decides which actions to take, acting autonomously according to a $Q(\lambda)$ learning function, and (ii) semi-autonomous - the robot requests the HO to suggest a policy, thus using the suggestion to replace its own exploration process.

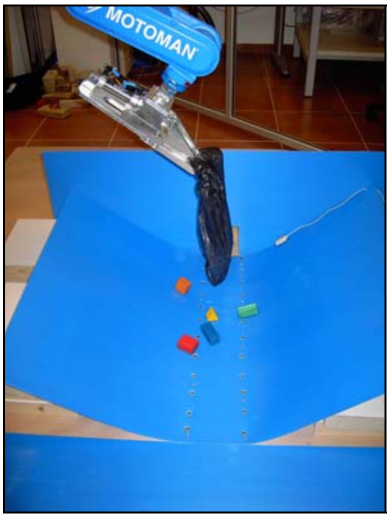
The three experimental setups described below utilize a Motoman UP-6 fixed-arm robot positioned over an inspection surface. A dedicated gripper was specifically designed for grasping a plastic bag. In all of the experiments, the first learning episode consists of a random shaking policy over the robot's X, Y or Z axes. The system earned rewards via its interaction with the environment, and initial system values were set at $\gamma = 0.9$, $\lambda = 0.5$, and $\alpha = 0.05$. To balance between exploration and exploitation (*e.g.*, [Guo *et al.*, 2004; Meng *et al.*, 2006]), an ε-greedy action selection with $\varepsilon = 0.1$ was used.

6.3.2 Experimental Setup I: Manually-Measured Reward Inspection System

In this experimental setup, rewards were measured manually with a standard timer using a linear-based reward function (6.3). When system learning performance was low, the human was asked to intervene and suggest various speeds and adjacent state distances over the X, Y and Z axes of the robot gripper. Robot learning experience was gained through direct experience with the environment according to rewards based on the number of items that fell from the bag after performing a shaking policy. Its value is linearly dependent on the number of falling items, *i.e.*, the robot gets a numerical value of 20 ($c = 20$ [1]) for every item that dropped. The reward is given to the robot when it completes the learning episode and is equal to the number of items fell during performing the shaking, multiplied by c. If no items fall out of the bag, the robot is "punished" by getting no reward.

$$R_n = c \cdot O \qquad (6.3)$$

where O is the number of items that fell from a bag during a shaking operation. At the beginning of each learning episode performed, the robot grasps and lifts a bag containing five wooden cubes (Fig. 6.2a and Fig. 6.2b) Then it performs a shaking policy.



(a) Robot and inspection surface

(b) Plastic bag and cubes

Fig. 6.2 Experimental setup - Motoman UP-6 fixed-arm robot system

The UP-6 robot has no *a-priori* knowledge in its initial learning stages; thus, in addition to interacting with the environment and getting rewards/punishments, policy adjustments are provided by the HO through an interface (Fig. 6.3).

[1] c is a positive constant to adjust the reward values achieved. Its value is determined based on empirical tests.

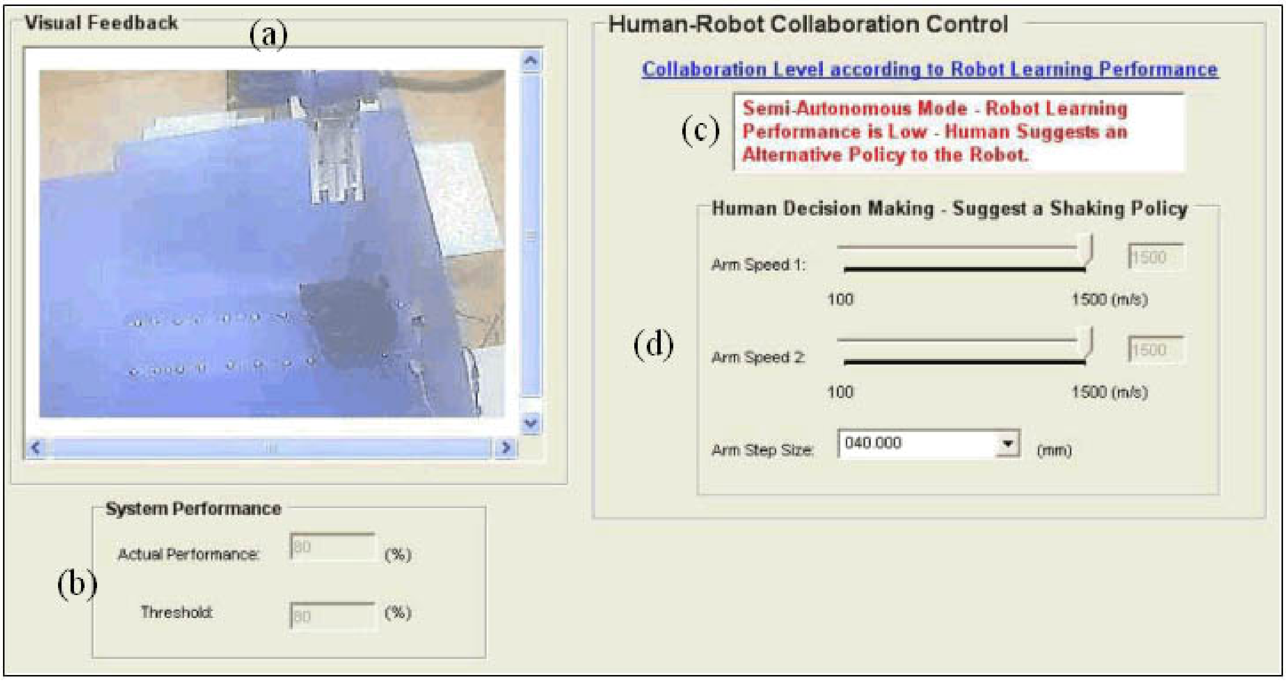


Fig. 6.3 Human interface - bag shaking task with a Motoman UP-6 fixed-arm robot system[1]

The interface views and controls consist of the following:

a) Real-time visual feedback captured from a web-camera located over the robotic scene.

b) System learning performance reporting - this includes the performance (in percents) of the last five episodes.

c) System mode reporting - autonomous or semi-autonomous.

d) Human decision making control - when asked to intervene, the HO can determine robot shaking adjacent state distances (10 - 50 mm) and speeds (100 - 1500 mm/s). The robot learning function acquires the suggested parameters and performs a new shaking policy.

Experiments

Two experiments were conducted, the first of which employed $CQ(\lambda)$-learning and comprised 75 learning episodes separated into three stages: (i) training - during the first ten runs the robot performs shaking policies autonomously. The initial shaking parameters were set at an adjacent state distance of 30 mm, and to speeds of 1000 and 1500 mm/s; (ii) collaboration - this stage consists of forty shaking policies, and human intervention is allowed based on the system learning performance. The human can adjust shaking policy parameters in the ranges of 10 to 50 mm for the adjacent state distance and 100 to 1500 mm/s for the speed, and (iii) testing - for measuring the efficiency of the

[1] In this preliminary experiment two speed settings are shown. This allowed to double the state-space size by defining two identical sets of states and action while for each one of them a different speed could be determined by the human operator. During further experiments described in this work this ability was eliminated because it is not intuitive.

human collaboration, the robot performed 25 policies using the original shaking parameters defined in the training stage. No human intervention was allowed in stages (i) and (iii).

To compare $CQ(\lambda)$ with the $Q(\lambda)$-learning algorithm, a second experiment was designed. The experiment consisted of 25 learning episodes in which the system learned according to the standard $Q(\lambda)$-learning algorithm with no human intervention.

6.3.3 Experimental Setup II: Automatically-Measured Cumulative-based Reward Inspection System

A digital scale was used for automatically measuring the rewards. This experiment (Fig. 6.4) was setup similar to that described in Section 6.3.2.

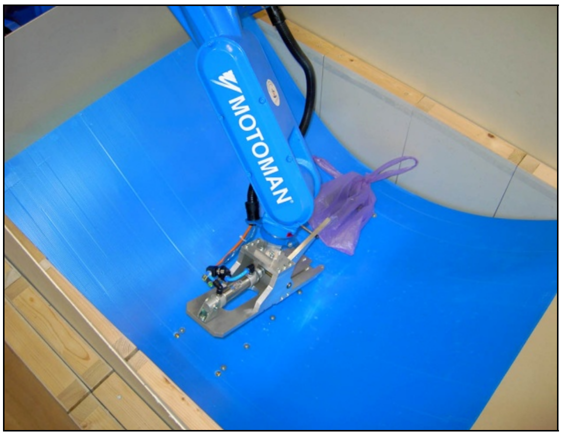


Fig. 6.4 Plastic bag placed on the inspection surface below the Motoman UP-6 fixed-arm robot

The digital scale was placed under the inspection surface (Fig. 6.5).

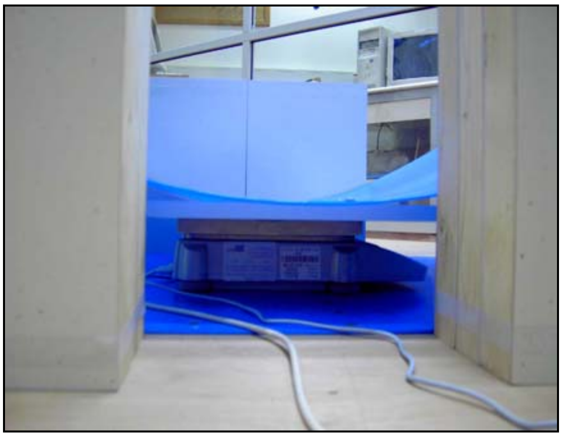
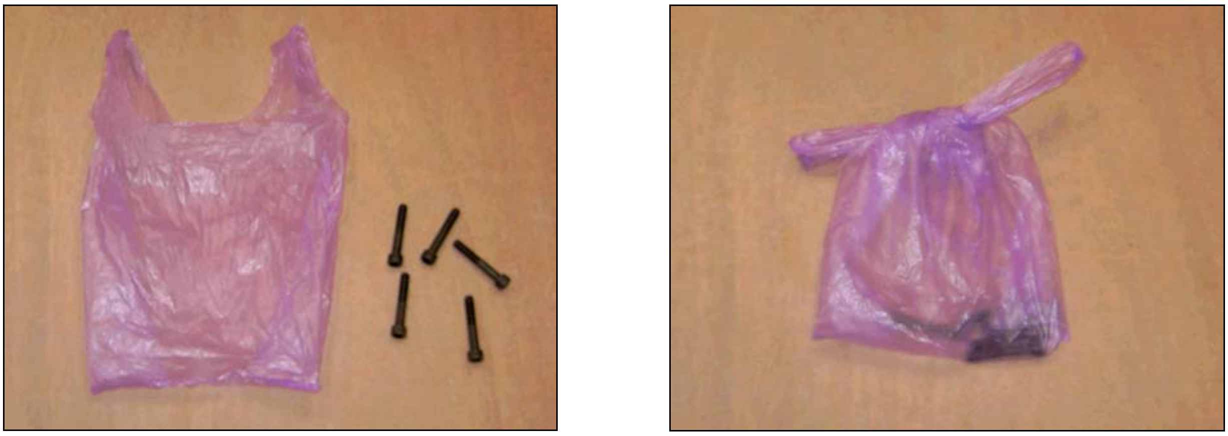


Fig. 6.5 Digital scale located under an inspection surface

In this experiment, five identical screws are inserted into the plastic bag while resting on the inspection surface (Fig. 6.6).



(a) Plastic bag and five screws **(b) Closed plastic bag with screws**

Fig. 6.6 Suspicious plastic bag

For evaluating the system performance, a cumulative-based reward function was used (6.4).

$$R_n = c \cdot \left(\sum_{i=0}^{T} \left(\frac{W_j}{t_j} \right) \right) \tag{6.4}$$

where W_j is the current weight measured by a digital scale at time t_j (increments of 0.25 second), *T=min{Fixed Horizon Time*[1]*, Amount of Time when all Objects Fell*[2]*}* is the time of shaking, and c is a positive constant to adjust the reward values achieved and R_n is the reward for learning episode n.

Similar to the first experimental setup, when system learning performance was low, the human was asked to intervene and suggest various speeds and adjacent state distances over the X, Y and Z axes of the robot gripper. Policy adjustments were provided by the HO through an interface (Fig. 6.7).

[1] *Fixed Horizon Time* is the time it takes the robot to perform a pre-defined number of state-action transitions (it was set to 100 state-action pairs).
[2] The amount of time when all objects fell is measured.

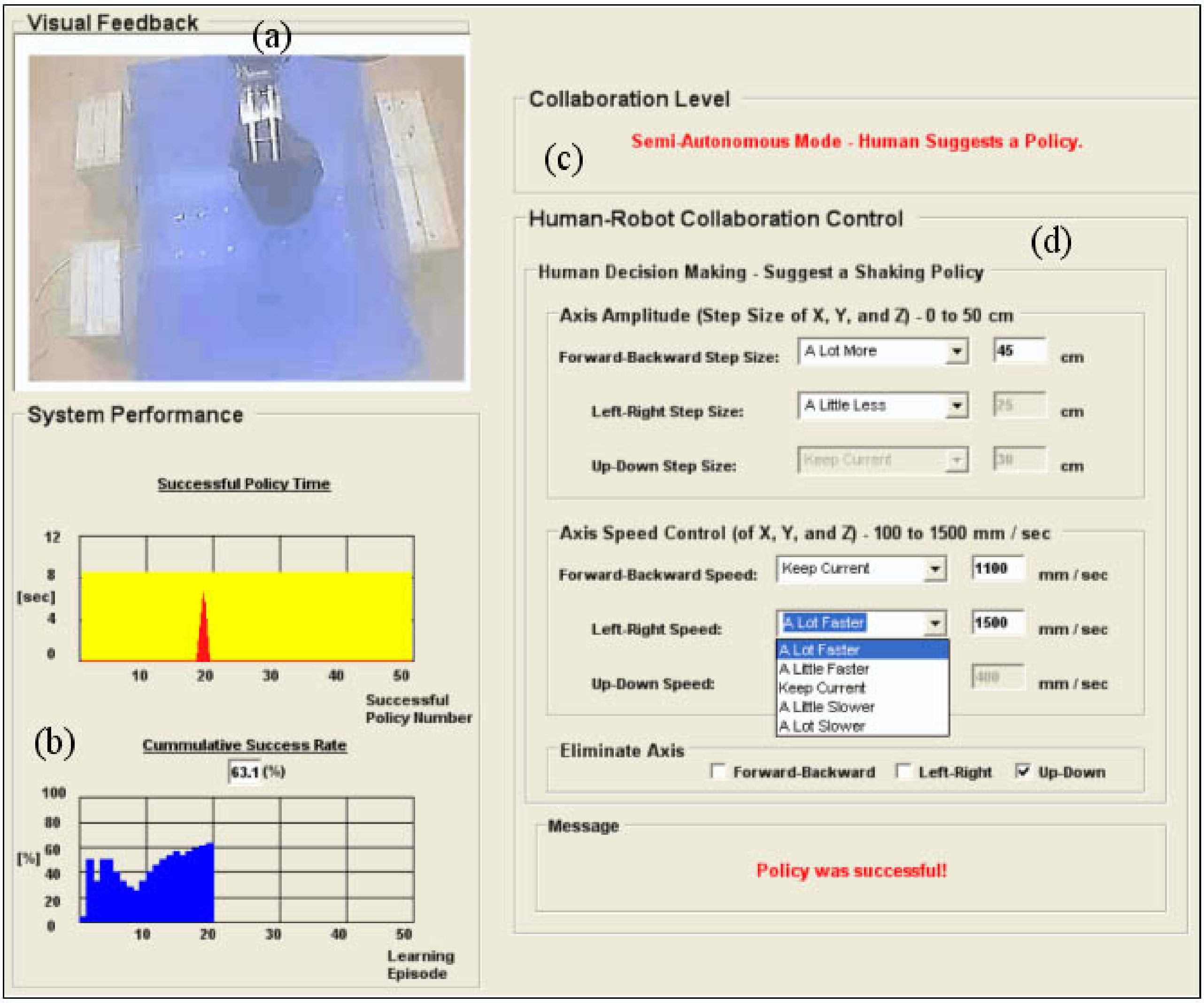


Fig. 6.7 Human interface - bag shaking task with a Motoman UP-6 fixed-arm robot system

The interface views and controls consist of the following:

a) Real-time visual feedback captured from a web-camera located over the robotic scene.

b) System learning performance reporting - this view includes two graphs:

   i. Successful performance time - a red arrow shows the time the robot needed to perform the last successful shaking compared with the average of all successful policies (the yellow surface).

   ii. Cumulative success rate - measures the percent of all learning episodes performed.

c) Collaboration level - autonomous or semi-autonomous.

d) Human-robot collaboration control - when asked to intervene, the HO can determine robot shaking adjacent state distance (0 - 50 mm) and speeds (100 - 1500 mm/s) for each axis separately. He can also eliminate robot actions over a specific axis. The interface allows the HO to guide the robot using linguistic terms (*e.g.*,“a lot more”, “a little faster”, etc.). The robot learning function acquires the suggested parameters and performs a new shaking policy.

Experiments

Robot learning is achieved through direct experience with the environment according to weight measurements achieved from the digital scale placed under the inspection surface. The weight on the scale depends on the number and occurrence of screws falling from the bag during a shaking learning episode. In this experimental setup, $Q(\lambda)$ is compared with $CQ(\lambda)$, running each of them for fifty learning episodes.

6.3.4 Experimental Setup III: Automatically-Measured Events-based Reward Inspection System

The experimental setup described here is identical to the setup described in Section 6.3.3. Similarly, a digital scale was used to automatically measure the rewards. The reward function used here, however, is based on events, *i.e.*, the occurrence of falling objects (6.5).

$$R_n = c \cdot \left(\sum_{i=0}^{T} \left(\frac{\Delta(t_j)\left(\frac{W_j - W_{j-1}}{w}\right)}{t_j} \right) \right), \qquad \Delta(t_j) = \begin{cases} 0, & \text{if no items fell} \\ 1, & \text{if an item(s) fell} \end{cases} \tag{6.5}$$

where R_n is the reward at learning episode n, W_j is the current weight measured by a digital scale located under the inspection surface at time t_j (increments of 0.25 second) when the j event occurred (an event is defined as the falling of one or more objects). Dividing the weight differences by t_j effectively increases the reward for items that fall early. w is the weight of one object (a constant value). W_{j-1} is the weight measured by the scale when the pervious (for the first event, $W_{j-1} = 0$). *T=min{Fixed Horizon Time, Amount of Time when all Objects Fell}* is the time of shaking. The value $\frac{W_j - W_{j-1}}{w}$ represents the number of objects that fell at time t_j and is rounded toward the closest integer value to eliminate scale inaccuracy. The positive constant c is used to adjust the reward values achieved.

Another difference from the experimental setups described in Sections 6.3.2 and 6.3.3 is the use of a different human interface. When system learning performance is low, the human is asked to intervene directly in the system Q table by enabling HO control over the Q state-action values of the robot's X , Y and Z axes. The HO uses the interface shown in Fig. 6.8 to guide the robot to prefer some actions over others without a demand to completely understand what Q value levels are.

The interface allows the HO to control the state-action Q table by controlling:

1) "Center Control" - by using the "Center Control" options, the HO allows the robot to prefer moving from the center position to a particular axis's set of states (all six possible states) for a specific axis. This means that the robot prefers to move to a specific axis rather than moving to the remaining two. If during shaking the robot passes through the center position while performing a shaking operation over a specific axis, the robot might switch its shaking actions to a different axis. This results a robot preference from which axis to start a new learning episode and from which axis to prefer if it passes through the center position while performing a learning episode.

2) "Swing Control" - by using the "Swing Control" options, the HO can let the robot prefer moving to a state's mirror position[1] than moving back to the center. Using this option allows the HO to let the robot perform longer sequence of actions over a specific axis by decreasing the likelihood of moving to the center position and switch to a different axis.

The HO does not know exactly at what state the robot is while performing a learning episode. The HO knows that the robot is about to grasp a bag and shake it for up to 100 movements (actions). He has to make a decision based on previous rewards the robot gained and from his intelligence/experience.

<u>"Center Control" (for each axis - *X*, *Y*, or *Z*) options include</u>:

1) "A Lot Higher" - increases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 50%.

2) "A Little Higher" - increases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 10%.

3) "Keep Current" - no change in the Q value table.

4) "A Little Lower" - decreases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 10%.

5) "A Lot Lower" - decreases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 50%.

[1] A mirror position of a state is the physical symmetrical state over the same axis. This state and its mirror are at the same distance from the center state.

“Swing Control” (for each axis - X, Y, or Z) options include:

1) “Much Higher Swings” - increases all six Q state-action value pairs of moving back to the center for the chosen axis by 50% while keeping current the state-action values of moving back to the center.

2) “Higher Swings” - increases all six Q state-action value pairs of moving back to the center for the chosen axis by 10% while keeping current the state-action values of moving back to the center.

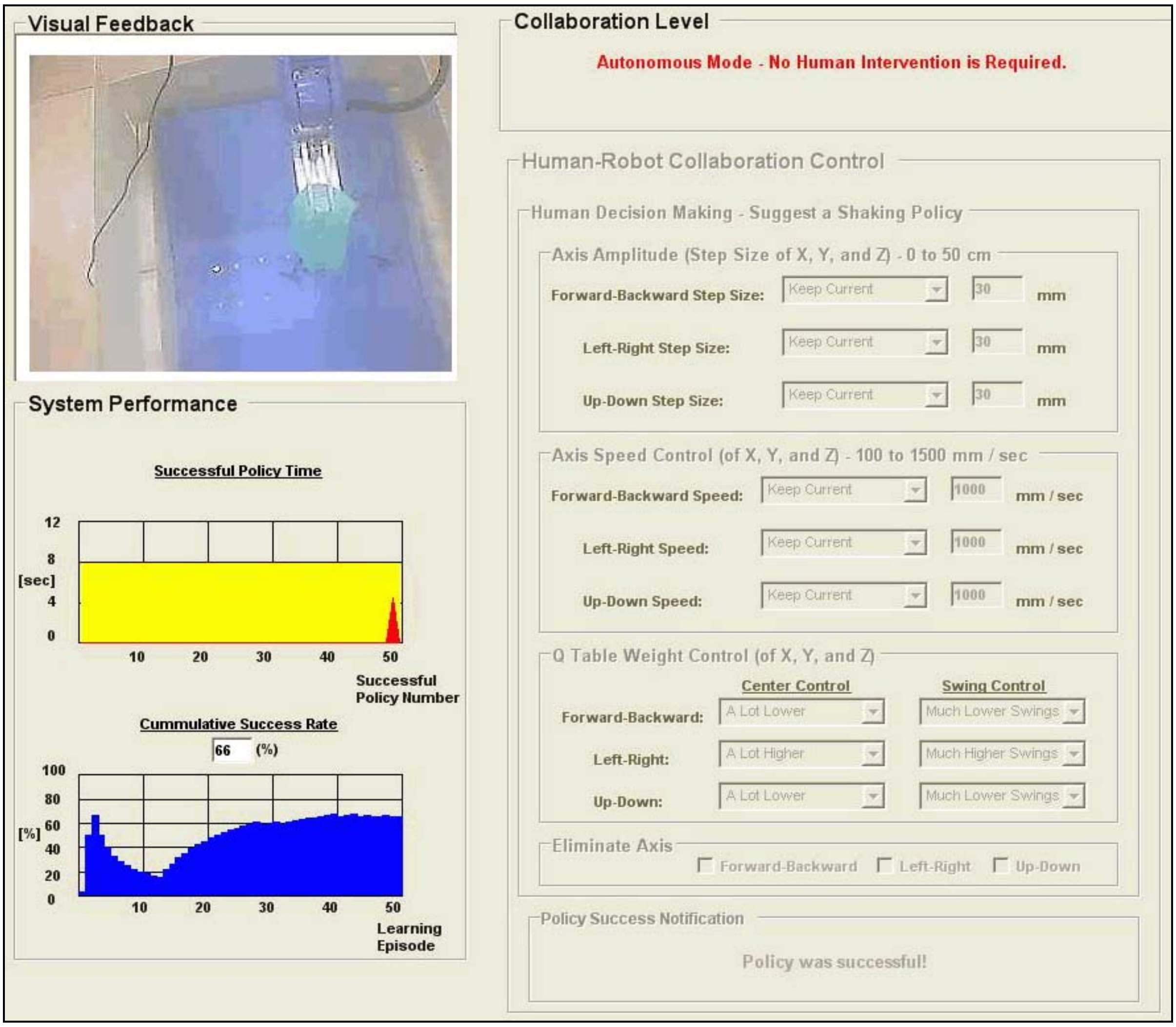
3) “No Change”- no change in the Q value table.

4) “Lower Swings” - decreases all six Q state-action value pairs of moving back to the center for the chosen axis by 10% while keeping current the state-action values of moving back to the center.

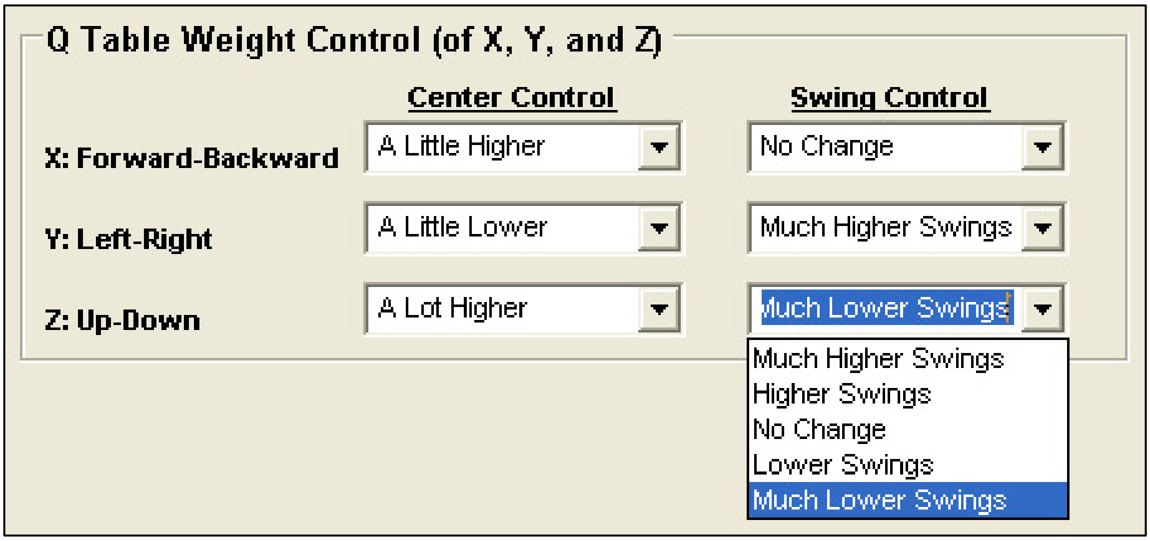
5) “Much Lower Swings” - decreases all six Q state-action value pairs of moving back to the center for the chosen axis by 50% while keeping current the state-action values of moving back to the center.

Experiments

Robot learning experience is gained through direct experience with the environment according to events-based measurements achieved from the digital scale placed under the inspection surface. The reward depends on the number and occurrence of screws falling from the bag during a shaking learning episode. In this experimental setup, $Q(\lambda)$ is compared with $CQ(\lambda)$ running each of them for fifty learning episodes.



(a) The entire interface



(b) Human-collaboration

Fig. 6.8 *Q*-value weight control human interface

6.4 Results and Discussion

For the "Manually-Measured Rewards Inspection System" described in Section 6.3.2, from a reward perspective, ranging at the scale of 0 to 100, in the training stage (first ten learning episodes), in which only four out of the ten policies revealed positive rewards, the average cumulative reward achieved was 32. In the collaboration stage (11-50 learning episodes), the average cumulative reward was 82. During the testing stage (51-75 learning episodes), when the shaking parameters were identical to those in the training stage, *i.e.*, adjacent state distance of 30 mm and speeds of 1000 and 1500 mm/s, the average cumulative reward achieved was 85.87. Fig. 6.9 shows the accumulated reward improvement during the three stages.

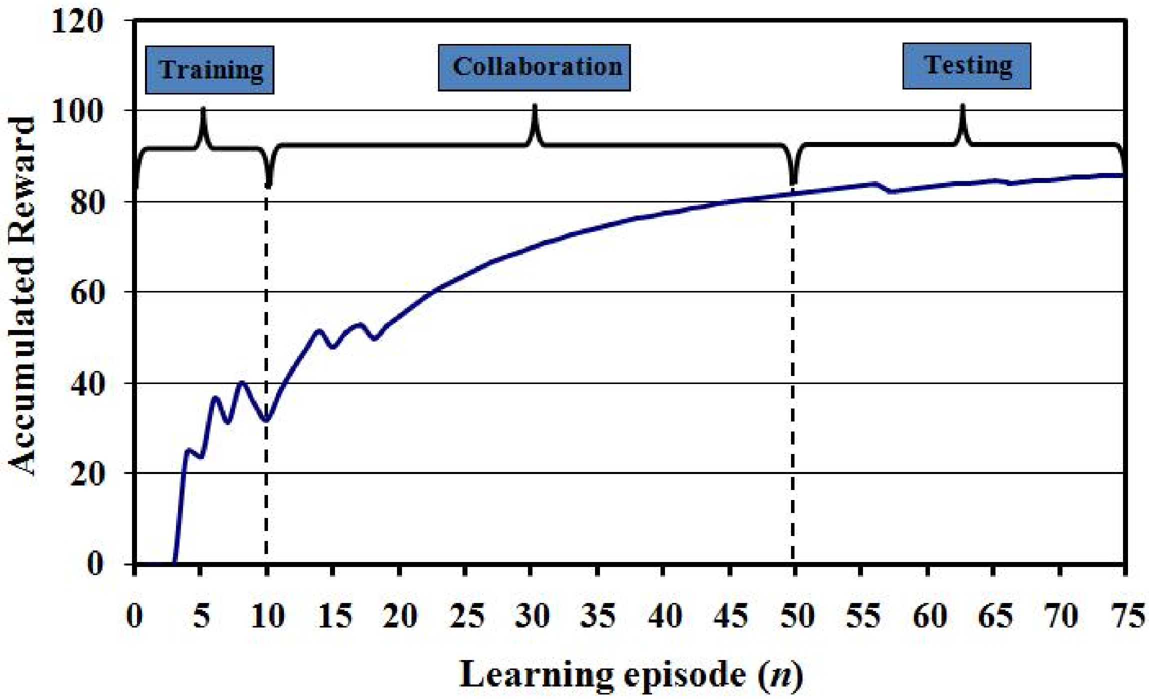


Fig. 6.9 Performance for linear-based rewards - *CQ(λ)* evaluation in three stages; training, collaboration and testing

While comparing the performance of the first experiment ($CQ(\lambda)$-learning) with the second experiment ($Q(\lambda)$-learning) (Section 6.3.2) over 25 learning trials[1], an improvement of 27% was observed in the average cumulative reward (64 vs. 50.4) while a human was asked to intervene though 11-23 learning episodes (Fig. 6.10). It is seen from Fig. 6.10 that $CQ(\lambda)$ superiority is achieved from the 12^{th} learning episodes. This superiority lasts till the end of the experiment (25^{th} learning episode).

[1] To compare between the algorithms, 25 learning episodes were taken into account for each one of them. For the *CQ(λ)* algorithm, the learning episodes consist of the ten training learning episodes and the first 15 learning episodes of the collaboration stage. This results a total of 25 learning episodes. For the *Q(λ)* algorithm an additional 25 learning episodes with no human intervention where considered.

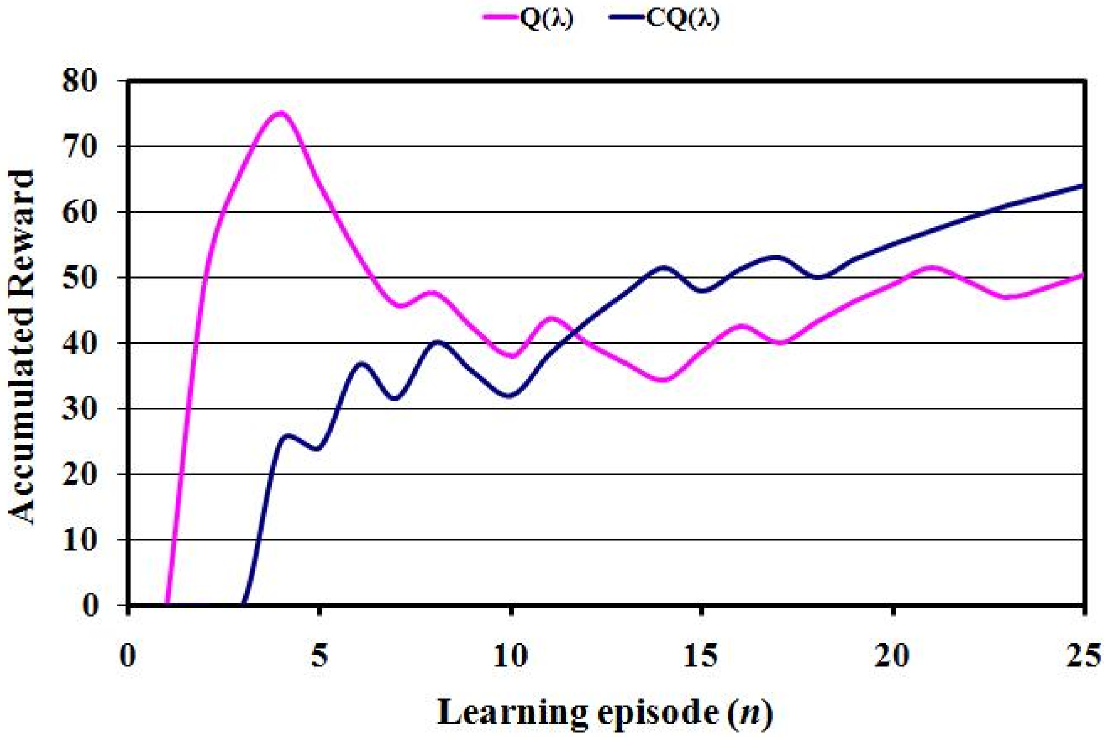


Fig. 6.10 Performance for linear-based rewards - a comparison between *Q(λ)* and *CQ(λ)*

For the "Automatically-Measured Cumulative-based Reward Inspection System" described in Section 6.3.3, in which $Q(\lambda)$-learning was compared with $CQ(\lambda)$, the average time to complete emptying the contents of a bag for the last forty[1] learning episodes was 12.43 s and 10.37 s respectively, *i.e.*, an improvement of 16.6%. In terms of rewards, the average cumulative reward achieved was 48.16 for $Q(\lambda)$ and 66.6 for $CQ(\lambda)$, *i.e.*, an improvement of 38.3%. The human intervention rate measured for the $CQ(\lambda)$-learning experiment was 30%, while system requests for collaboration occurred continuously during the first runs when human intervention was allowed (from the 11^{th} learning episode). A summary of the results is shown in Table 6.2. Additionally, learning curves for $Q(\lambda)$ and $CQ(\lambda)$ are shown in

[1] The first ten episodes were excluded from analysis of the results since during these episodes no human collaboration is allowed.

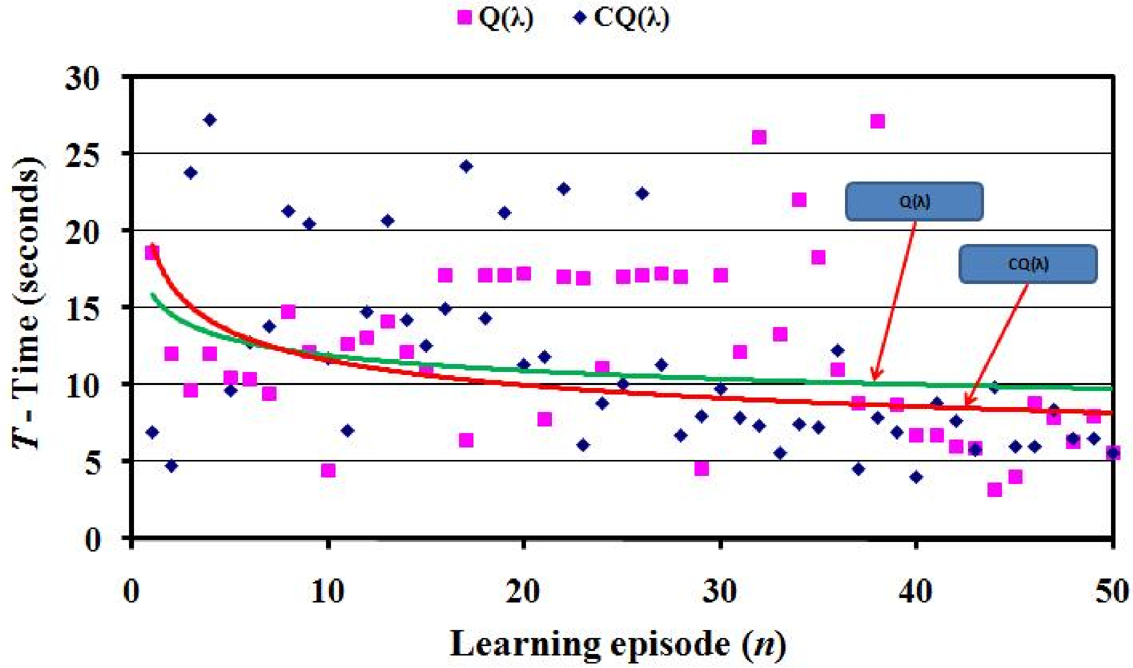


Fig. 6.11, and learning rates[1] were calculated as 0.87, and 0.82, respectively.

[1] The learning rate parameter determines how significantly an agent improves. The reader should make a distinction between the learning rate evaluation performance measure which indicates the improvement of learning and α, the RL learning rate parameter.

Table 6.2 Comparison between *Q(λ)* and *CQ(λ)* - cumulative-based rewards

		CQ(λ)	*Q(λ)*
Average time to complete emptying the contents of a bag	**Time (s)**	10.37	12.43
	Standard deviation	5.37	5.94
Human intervention rate	**[%]**	30	-

* Results are for the last forty learning episodes.

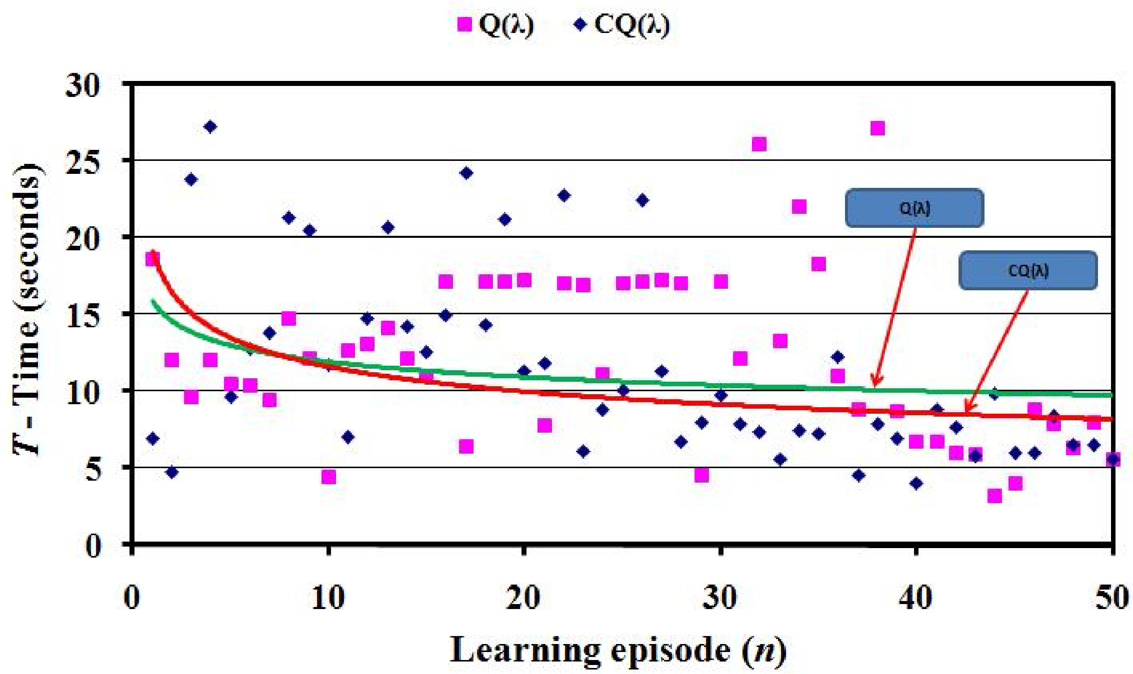


Fig. 6.11 Times for cumulative-based reward

Accumulated reward performance superiority of $CQ(\lambda)$ over $Q(\lambda)$ starting from the 4^{th} learning episode to the end of the experiment (50^{th} learning episode) is shown in Fig. 6.12.

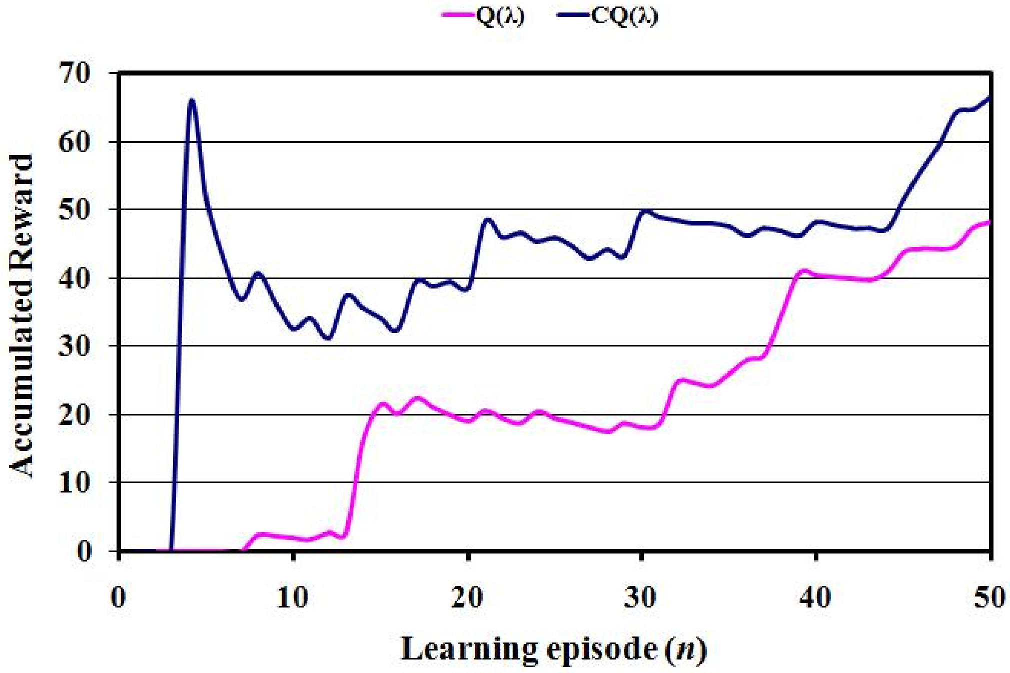


Fig. 6.12 Performance for cumulative-based rewards

Based on the results achieved, the following hypothesis was evaluated using a *T*-test (Table 6.3).

Table 6.3 ***Q(λ)*** **and** ***CQ(λ)*** **using a cumulative-based reward function evaluation**

Performance measure	Hypothesis
Time to complete emptying the contents of a bag	H_{10}: There is no difference between average times to complete emptying the contents of a bag while comparing *Q(λ)* with *CQ(λ)*-learning. H_{11}: There is a significant difference between the two learning methods.

The null hypothesis H_{10} was rejected with a *P*-value of 0.0375; in other words, the two learning methods exhibited unequal mean coefficients. Similar to the experiments described in Section 6.3.2, policies that concentrated with most shaking over the Y axis and with very few actions over the X axis were the most effective.

The robot starts experiencing the environment by performing a random shaking policy over the X, Y, and Z axes. The default speeds and adjacent state distance were set to 1000 mm/s and 30 mm, respectively for all axes. After the first ten episodes in which the robot is forced to learn the environment autonomously and thus no human collaboration is allowed, it reports low learning performance (6.1) and asks for human guidance. It was reasonable for him to increase the speeds on all axes to their maximum possible values (1500 mm/s). This decision was made because high speeds are more effective at causing the contents of the bag to drop out faster. For the same reason, he also decides to slightly increase the adjacent state distances to 40 mm on all axes. Human reasoning says that it is not necessarily worth increasing the adjacent state distance to the maximal possible value the system allows because extremely high adjacent state distances may create longer lasting shaking policies. On the one hand, the human objective is to make the system obtain the highest possible rewards, but on the other hand the human wants to help the robot find the shortest possible policies. The suggested policy (1500 mm/s and 40 mm) was tested on the system for several learning episodes but it was not successful, *i.e.*, resulted in low rewards, and therefore, the robot asked for human intervention consistently for each policy performed with these parameters. At this stage, the human suggested policies for the axes with lower speed combinations and higher adjacent state distance combinations. These policies occasionally produced positive rewards, but the results were not consistent and the robot tended to switch between autonomous and semi-autonomous modes. At approximately the 30^{th} learning episode, the human suggests a shaking policy with the maximal speeds over the three axes, a very high adjacent state distance over the Y axis (45 mm) and lower adjacent state distances in comparison with the high values defined at the beginning of the experiment (25 mm and 15 mm for the X and Z axes, respectively). This policy is preserved until the end of the experiment (the 50^{th} learning episode), *i.e.*, the policy is preferable.

For the “Automatically-Measured Events-based Reward Inspection System” described in Section 6.3.4 comparing $CQ(\lambda)$ with $Q(\lambda)$-learning, results of the average time to complete emptying the contents of a bag for the last forty[1] learning episodes was 7.5 s and 11.42 s, respectively, *i.e.*, an improvement of 34.3%. From a reward perspective, the cumulative average reward achieved was measured as 92.19 for $CQ(\lambda)$ and 70.68 for $Q(\lambda)$, *i.e.*, an improvement of 30.4%. The human intervention rate measured for the $CQ(\lambda)$-learning experiment was 20% while collaboration requests occurred continuously when collaboration was allowed (from the 11^{th} learning episode). A summary of these results is shown in Table 6.4. Additionally, learning curves for $Q(\lambda)$ and $CQ(\lambda)$ are shown in Fig. 6.13, and learning rates were calculated as 0.78 and 0.84, respectively.

[1] The first ten episodes were excluded from analysis of the results since during these episodes no human collaboration is allowed.

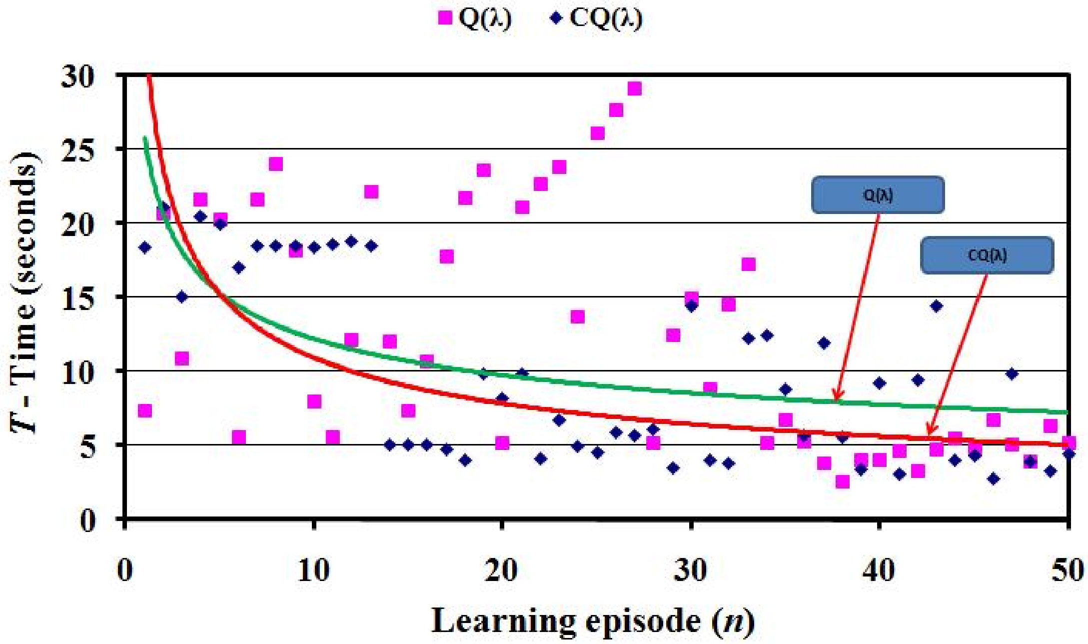


Fig. 6.13 Times for events-based rewards

Accumulated reward performance superiority of $CQ(\lambda)$ over $Q(\lambda)$ starting from the 24^{th} learning episode to the end of the experiment (50^{th} learning episode) is shown in Fig. 6.14.

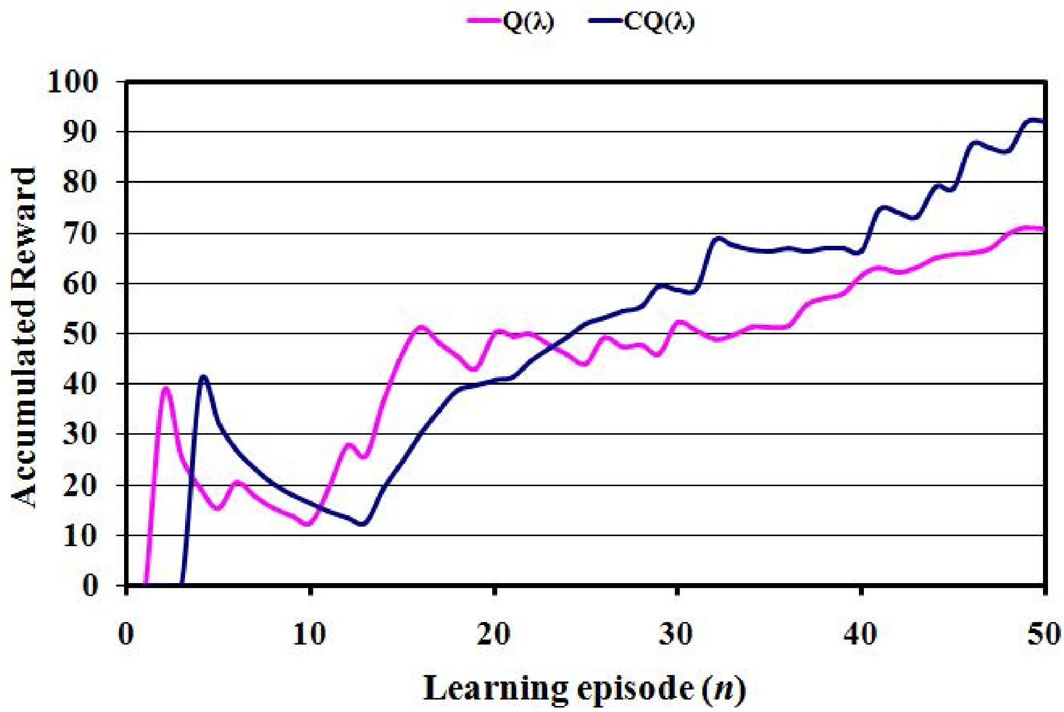


Fig. 6.14 Performance for events-based rewards

Table 6.4 Comparison between *Q(λ)* and *CQ(λ)* - events-based rewards

		CQ(λ)	***Q(λ)***
Average time to complete emptying the contents of a bag	**Time (s)**	7.5	11.42
	Standard deviation	4.53	8.07
Human intervention rate	**[%]**	20	-

* Results are for the last forty learning episodes.

Based on the results achieved, the following hypothesis was evaluated using a *T*-test (Table 6.5).

Table 6.5 *Q(λ)* and *CQ(λ)* using a events-based reward function evaluation

Performance measure	Hypothesis
Time to complete emptying the contents of a bag	H_{20}: There is no difference between average times to complete emptying the contents of a bag while comparing *Q(λ)* with *CQ(λ)*-learning. H_{21}: There is a significant difference between the two learning methods.

The null hypotheses H_{20} was rejected with a *P*-value of $5 \cdot 10^{-3}$; indicating that the learning agents' mean coefficients are not equal.

For the "Automatically-Measured Events-based Reward Inspection System", additional fifty learning episodes were performed to compare between the algorithms. Here, differently than the interface that was used in the experiments described above, the *Q*-value weight control human interface was used (Fig. 6.8). The average time to empty the contents of the bag using $CQ(\lambda)$ and $Q(\lambda)$-learning for the last forty[1] learning episodes was 8.28 and 9.82 seconds, respectively, *i.e.*, an improvement of 18.6%. From a reward perspective, the cumulative average reward achieved was measured as 28.81 for $CQ(\lambda)$ and 20.11 for $Q(\lambda)$, *i.e.*, an improvement of 43.3%. The human intervention rate measured for the $CQ(\lambda)$-learning experiment was 12%.

Table 6.6 Comparison between *Q(λ)* and *CQ(λ)* - events-based rewards (*Q*-value weight control interface)

		CQ(λ)	***Q(λ)***
Average time to complete emptying the contents of a bag	**Time (s)**	8.28	9.82
	Standard deviation	3.48	1.76
Human intervention rate	**[%]**	12	-

* Results are for the last forty learning episodes.

Based on the results presented in Table 6.7, the following hypothesis was evaluated using a *T*-test.

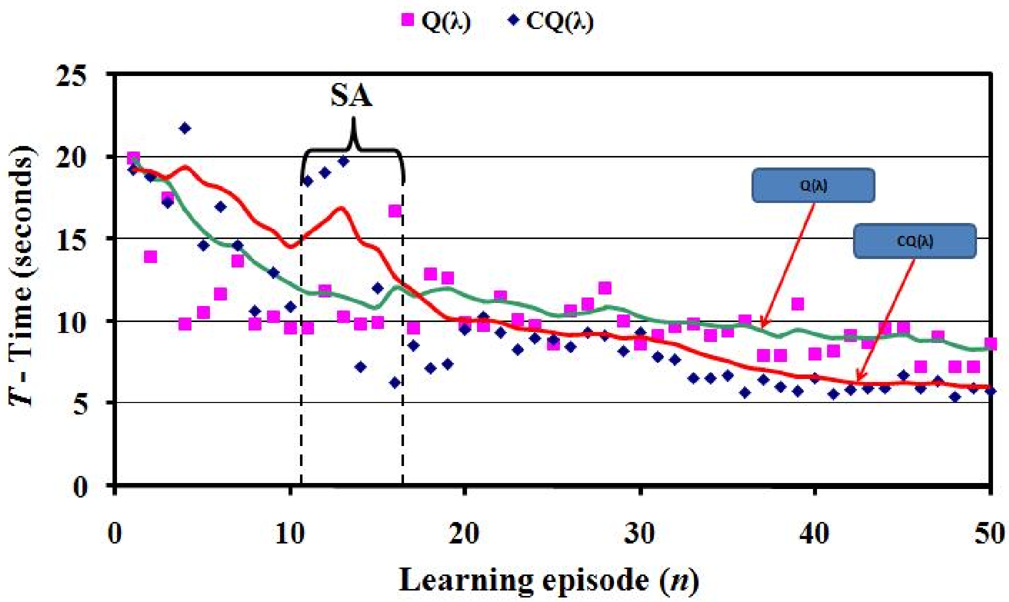
[1] The first ten episodes were excluded from analysis of the results since during these episodes no human collaboration is allowed.

Table 6.7 *Q(λ)* and *CQ(λ)* using a events-based reward function (last forty learning episodes)

Performance measure	Hypothesis
Average time to complete emptying the contents of a bag	H_{30}: There is no difference between average times to complete emptying the contents of a bag while comparing *Q(λ)* with *CQ(λ)*-learning for the last forty learning episodes. H_{31}: There is a significant difference between the two learning methods.

The null hypothesis H_{30} was rejected with a P-value of 0.0045. This means that the average times to complete emptying the contents of the bag are not equal while comparing $CQ(\lambda)$ with $Q(\lambda)$.

Performance times for each learning episode are shown in the scatter plot of Fig. 6.15 for both the $CQ(\lambda)$ and $Q(\lambda)$ runs. Exponential smoothing, using a damping factor of 0.8 for the measurements, are shown as green and red lines for $CQ(\lambda)$ and $Q(\lambda)$, respectively. It is clearly seen from smoothed data in Fig. 6.15 that $CQ(\lambda)$ is superior from the 16^{th} learning episode, having approached convergence at about episode 45. Cumulative reward performance superiority of $CQ(\lambda)$ over $Q(\lambda)$ is shown in Fig. 6.16. For $CQ(\lambda)$, Fig. 6.17 shows that the robot continued to request assistance from the HO immediately for six episodes after human intervention was first allowed right after episode 10. One sees that at the 11^{th}, 12^{th} and 13^{th} learning episodes the shaking performance times are significantly high. This can be explained due to the explorative strategies the HO tried over these episodes when first asked to intervene at this stage. The HO quickly recovered providing a good policy over the next three episodes. This was enough guidance so that the robot continued to operate autonomously and never request advice again.



SA - Semi-autonomous mode

Fig. 6.15 Performance times for events-based rewards

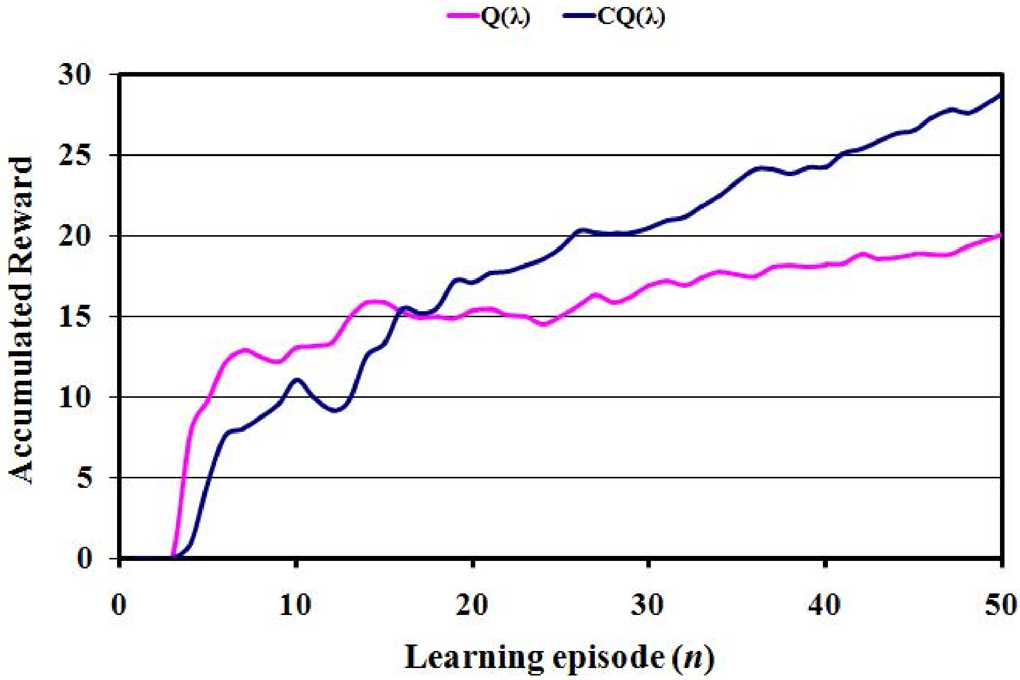
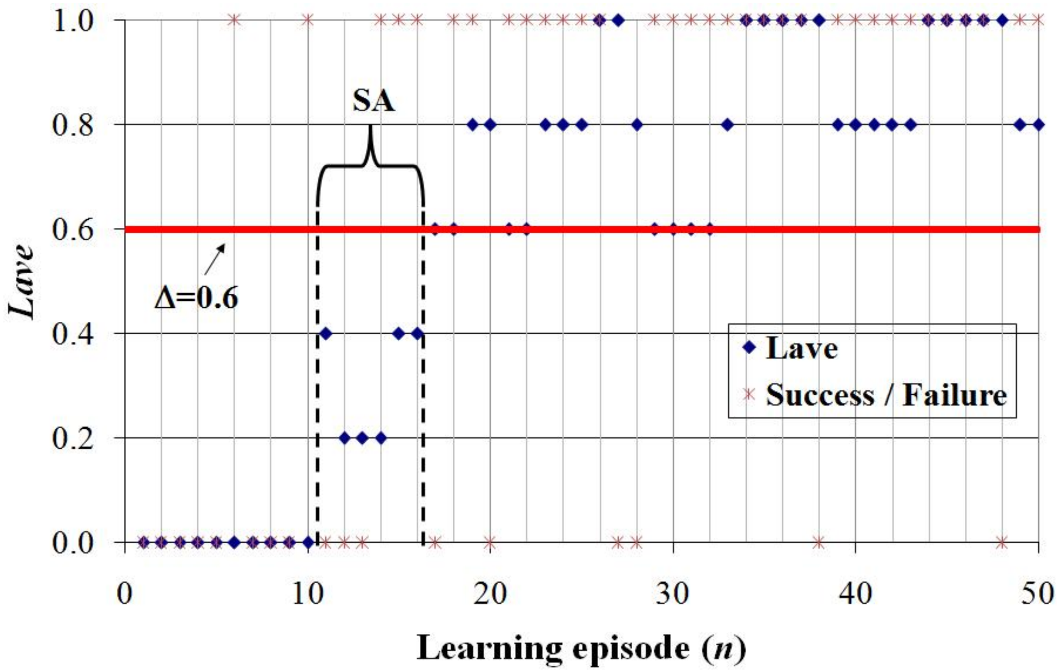


Fig. 6.16 Performance for events-based rewards using *Q*-value weight control human interface



SA - Semi-autonomous mode

Fig. 6.17 The moving average learning performance measure, L_{ave} during *CQ(λ)*-learning

The robot started experiencing the environment by performing a random shaking policy over the X, Y, and Z axes where the default speeds and adjacent state distances were 1000 mm/s and 30 mm respectively for all axes. After the first ten episodes in which the robot was forced to learn the environment autonomously and no human collaboration was allowed, it reported a low learning performance (6.1) and asked for human guidance. The human's superior intelligence and the experience he gathered while conducting the experiments described in the previous experimental sets taught the human that the robot should be guided to choose actions that will shake the bag mostly over the Y axis and with a small number of actions over the X axis. Therefore, when the HO was asked to intervene, he decided to prefer a strategy that will cause the robot to continuously perform actions that will mostly occur at locations as far as possible over the Y axis (Left-Right) and small amount of actions over the X (Forward-Backward) axis, while eliminating any action over the vertical axis (Up-Down). This knowledge was achieved due to previous experiments performed. Specifically, the chosen strategy is described as follows:

1) "Left-Right" - for "Center Control" keep choosing "A Little Lower" and for "Swing Control", keep choosing "Much Higher Swings". This strategy will make the robot almost to avoid stopping through the center position but still if sometimes the robot passes through the center position it will allow it to move to the forward-backward axis.

2) "Up-Down" - for "Center Control", keep choosing "A Lot Higher". That's because we want to prevent the robot from being shaked over the vertical axis. Moving back to the center rather than moving to one of the mirror vertical states will allow the robot to "escape" from the set of possible vertical states, go back to the center position and possibly to move to a horizontal or to a forward-backward shaking. That's why it is better also to keep choosing "much lower swings" which will contribute for moving back to the center.

3) "Forward-Backward" - for "Center Control" keep choosing "A Little Higher" and for "Swing Control", keep choosing "No Change". This strategy allows the robot to perform some shakings over the forward-backward axis, but causes it to "prefer" moving back to the center position.

6.5 Summary

Intuitively, vertical shaking should work best, but the experimental results showed that for both $CQ(\lambda)$ and $Q(\lambda)$ the best policies showed shaking most of the time over the Y axis and with very little activity over the X. One possible explanation for favoring the Y axis may be the type of knot holding the plastic bag closed; pulling it sideways loosens the knot faster. Furthermore, in the hypothetical situation of a human shaking the bag, he could have visually seen the servo feedback to determine the optimal time to pull it up in a horizontal strategy, an ability that the robot system used here does not have.

To interpret the results and to show that there were no subjective influences, a physical model of the opening of a plastic bag knot by a robot was developed (Appendix III). The model explains the results achieved for all three experimental setups. It showed that because acceleration developed over time, it was worthwhile to open the bag using a continuous shaking/motion from locations as far as possible over the Y axis. Ideally, the robot arm should be accelerated to match or closely match the gravitational acceleration downwards and should be oscillated over the Y axis to overcome most of the friction forces. To summarize, the results showed that learning was faster when the human operator was asked to intervene in the robot's activity.

7. Navigation of a Mobile Robot

Chapter Overview

This section describes an experimental setup using an Evolution Robotics ER-1 mobile robot.[1] In this system, the $CQ(\lambda)$ learning algorithm enables collaboration between the robot and a human.

7.1 Introduction

The test-bed selected is a robot task in which the robot must navigate toward a target location in a two-dimensional world. Based on its learning performance, an ER-1 mobile robot switches between fully autonomous operation and requesting human intervention. The $CQ(\lambda)$ algorithm was tested on a robot required to navigate toward a target location in a rectangular environment containing undesirable navigation areas.[2] During its initial learning stages, the robot has no knowledge as it has not yet acquired any from either its environment or collaboration; thus, in addition to interacting with the environment, human instructions are an effective acquisition technique toward autonomous behavior. The human is responsible for remotely monitoring the robot and suggests solutions when intervention is required. However, at a certain level of the human-robot system performance, it becomes unnecessary for the robot to follow human instructions. At that point, when the robot no longer needs instructions, it navigates autonomously.

7.2 Task Definition

The robot is remotely located relative to the human operator (HO), and uses environmental sensing capabilities. Learning is accomplished both by interaction with the environment and by acquiring suggestions from the HO. The environment contains undesirable areas that the robot learns to avoid. An optimal route is defined as the shortest route that the robot navigates most efficiently, *i.e.*, it moves toward the target and not away, while avoiding undesirable areas. The shortest route that the robot navigates most efficiently is in terms of path length remaining. It is calculated manually after accomplishing the experiments and is used for results evaluation and analysis.

Two levels of collaboration are defined: (i) autonomous - the robot decides which actions to take, acting autonomously according to a $Q(\lambda)$ learning function, and (ii) semi-autonomous - the robot requests the HO to suggest actions, thereby replacing its own exploration process. Human-robot collaboration is unnecessary as long as the robot learns policies and adapts to new states without a serious deterioration in its performance. When the two conditions below apply, therefore, the HO is

---

[1] The application and experiments were accomplished at the *Institute for Medical Informatics* at the *Washington Hospital Center*, Washington D.C. (currently, Microsoft).

[2] Undesirable navigation area - an area where a robot can physically pass through but it is not recommended.

required to intervene and suggest actions to the robot. The robots' learning function then incorporates the HO suggestions.

1) The robot's acceptable learning performance threshold, ω [1], is compared with T_{ave} , the average number of steps to reach a goal and get a positive reward over the last N most recent learning policies performed. The HO is required to intervene and suggest alternative actions if the robot's learning performance is low, *i.e.*, it reports a large average number of steps to reach the goal (7.1).

$$T_{ave} = \left(\sum_{i=t-N}^{t-1} (L_i) \right) / N > \omega \tag{7.1}$$

where t is the current learning episode, $i = t-N, t-N+1, t-N+2, ...t-1$, and L_i is the number of steps a learning agent performs at the i^{th} episode.

2) The remote robot is within a view of the human collaboration area described in Section 7.3.

The human sees only a small part of the environment (Fig. 7.2c) via visual feedback (Fig. 7.2d), and can suggest actions to the remotely located robot only in this viewable area (Section 7.3). The task was defined in such a way that human collaboration with the learning robot is not always enabled for all of the world states but only for a portion of them. For the experiments described in this section this portion of the world states was defined arbitrarily to be only a quarter of the world, the region around the target (nine states - see Fig. 7.2c). The reason why only a portion of the world was defined as a "human collaboration area" was to duplicate the experimental environment of the laboratory in a real environment where, due to technical circumstances, human intervention is not possible.[2]

Fig. 7.1 demonstrates an example for several learning episodes performed in which the robot switched its learning level from autonomous (self navigation) to semi-autonomous (acquiring human knowledge) based on its learning performance. The acceptable learning performance threshold for the robot was set to $\omega = 6$. The robot continuously measured its learning performance by averaging its last N most recent learning episodes and comparing this value, T_{ave}, to ω. In Fig. 7.1, if T_{ave} was

[1] The maximum acceptable performance threshold in terms of mean number of steps to reach a goal. Above this value, the human is called to intervene. This threshold value is determined based on empirical tests.
[2] An example for such a system might be a simulation of a mobile robot sent to explore a remote planet such as Mars for locating water resources. Since the planet is remote from earth and viewing abilities using a telescope are limited due to weather conditions or atmosphere masking, human assistance might not be possible constantly and the robot will have to explore and learn the environment autonomously even if its learning performance is low.

higher than ω then the robot learning performance was ascertained to be low and human intervention was allowed, *i.e.*, learning was performed semi-autonomously. Otherwise, no human intervention was required and the robot navigated autonomously.

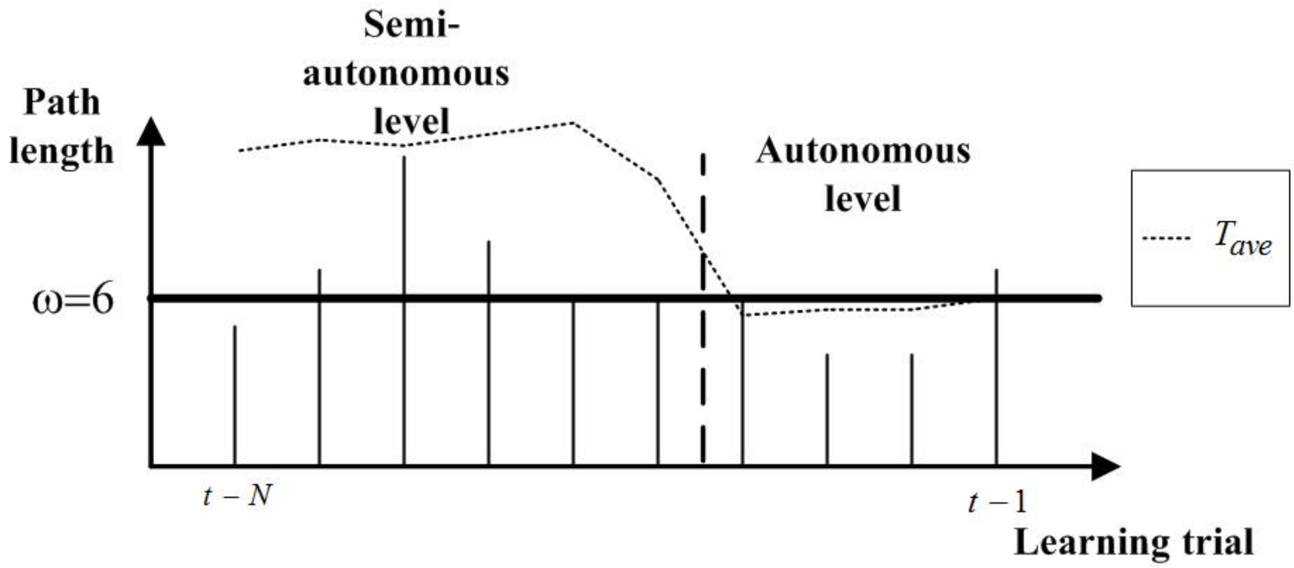


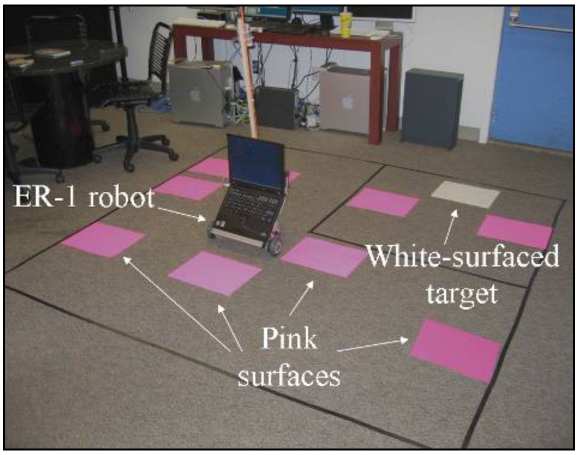
Fig. 7.1 Example of learning performances for an ER-1 mobile robot

7.3 Experimental Setup

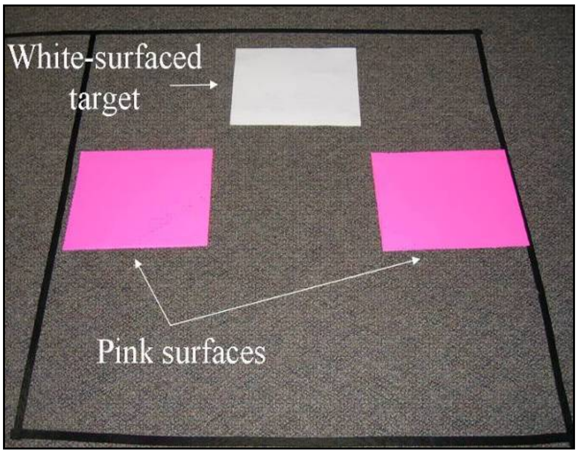
The robot is a three-wheeled ER-1 mobile platform equipped with an IBM ThinkPad laptop and an IREZ Kritter USB camera (Fig. 7.2a). Robot states include robot locations in a 6 × 6 two-dimensional world (Fig. 7.2b) where each cell is a 40 cm square floor structure. The world consists of three types of areas: (i) environmental cells (carpet) - areas through which the robot should navigate on its way to target; (ii) undesirable areas - pink surfaces, and (iii) the white-surfaced target cell. The robot senses the environment using a camera equipped with image processing and color thresholding capabilities to distinguish between the surfaces. The robot can move from one surface to any one of its four adjacent neighbors with the restriction that it cannot move out of its particular world.

The robot's state $s_t \in S$ is defined by: $s_t = (x_k, y_l)$ where $k \in (1, 2, ..., 6)$ and $l \in (1, 2, ..., 6)$. An action, $a_t \in A$, taken at each state is traveling north, west, south, or east. Rewards are defined as r_t where $r_t \in \{-1, 0, +1\}$. If the robot reaches the target (*i.e.*, when the camera recognizes a white surface), the reward is +1. If it passes through an undesirable area (the camera recognizes a pink surface) the reward is -1. Otherwise, the reward is zero. A learning episode comprises one session of reaching the target.

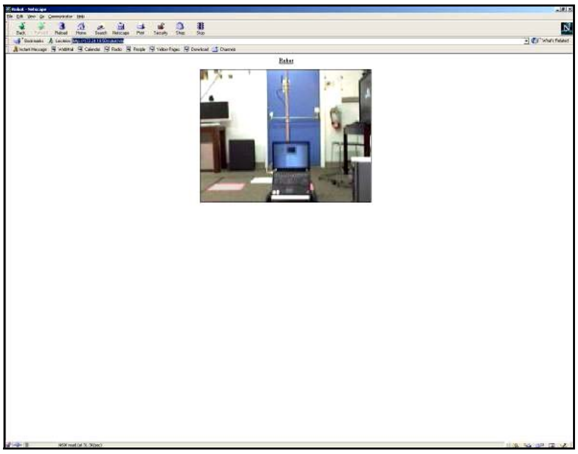
(a) Camera mounted on the ER-1 robot



(b) Robotic scene and the ER-1 robot



(c) Human control area



(d) Human view from a web-browser over part of the robotic scene

Fig. 7.2 Experimental setup - ER-1 mobile robot

Experiments using several robot autonomy modes were performed. The experiments covered more than 150 hours of robot motion in which different human interaction thresholds (ω) and RL parameters (γ and λ) were set to cover a wide range of acceptable values. The starting navigational states of the robot for each learning episode were distributed equally in the two-dimensional world. One learning episode consists of placing the robot at a random starting location in the environment. Then the robot explores the environment; a learning episode ends when the robot reaches the target. The experiments are as follows:

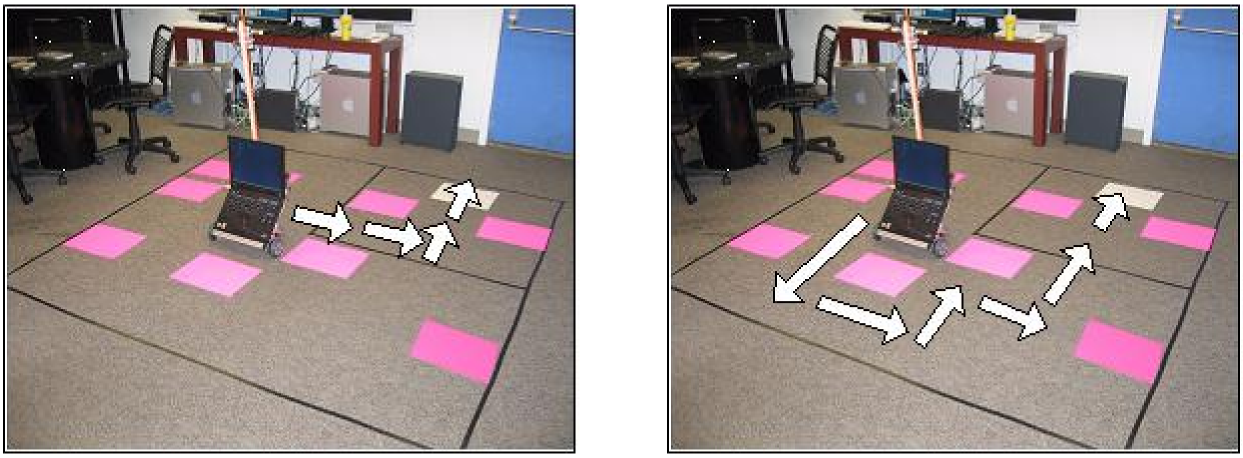
1) In the first experiment denoted as *EX1* in Table 7.1 and Table 7.2, the robot navigated autonomously without any human collaboration. In this experiment, to compare the $Q(\lambda)$ and the $CQ(\lambda)$ algorithms, the robot autonomously learned the environment with no human intervention (according to the standard $Q(\lambda)$ algorithm). The following parameters were set: $\alpha = 0.95$ (initial value), various γ (0.99, 0.95, and 0.9) and λ (0.75, 0.5, and 0.25) corresponding to each value of γ.

2) The second experiment consists of three sub experiments denoted as *EX2*, *EX3* and *EX4* (Table 7.1 and Table 7.2) - each sub experiment consists of a different acceptable learning performance threshold (ω) equal to 6, 8, and 10, respectively, *i.e.*, the robot switched its activity through semi-autonomous navigation to full autonomy based on the two conditions defined in Section 7.2. After every learning episode, the robot compared its learning performance with ω. High ω values indicate low learning performance. If the robot's learning performance was insufficient and it was within human view (Fig. 7.2c and Fig. 7.2d), it would decide to ask for human assistance. The following parameters were set: $\alpha = 0.95$ (initial value), various γ (0.99, 0.95, and 0.9) and λ (0.75, 0.5, and 0.25) corresponding to each value of γ.

Performance sensitivity of combinations of γ and λ values was conducted as shown in Table 7.1 and Table 7.2. Each combination of an autonomy level, Λ, γ, and λ consisted of fifty learning episodes; *i.e.*, in each experiment, the robot was placed randomly in one of 35 world states and then tried to navigate toward the target (the 36^{th} state).

System performance was evaluated using the following parameters:

- Mean number of steps to optimally reach the target - in each learning episode the robot starts at a random state and attempts to navigate toward the target. If the robot navigates without passing through an undesirable area or does not travel inefficiently (*e.g.*, movement away from the target), the route is defined as optimal (Fig. 7.3a). The shortest route that the robot navigates most efficiently is in terms of path length remaining. It is calculated manually after accomplishing the experiments and is used for results evaluation and analysis.
- Mean number steps to feasibly reach target - if the robot navigates without passing through an undesirable area but travels inefficiently (but still reaches the target), the route is defined as feasible (Fig. 7.3b).
- Percent of human interventions - measures how frequently a human collaborated with the robot. This value is calculated by measuring how many times out of all learning episodes perfomed the human was asked to intervene.



(a) Example for an optimal route **(b) Example for a feasible route**

Fig. 7.3 Examples for optimal and feasible routes for ER-1 navigation

7.4 Results and Discussion

Results of measuring the average number of steps; both feasibly and optimally, to reach the target (Table 7.1) and (Table 7.2) indicate that when setting $\gamma = 0.99$ with $\lambda = 0.75$ or $\lambda = 0.5$, using the $CQ(\lambda)$ algorithm while integrating human commands speeds up robot learning for all learning performance thresholds when compared with autonomic robot navigation using the standard $Q(\lambda)$ algorithm.

Table 7.1 Results for average number of steps to reach the taget feasibly

	Experiment Notation	Degree of human intervention	γ=0.99			γ=0.95			γ=0.9		
			λ=0.75	λ=0.5	λ=0.25	λ=0.75	λ=0.5	λ=0.25	λ=0.75	λ=0.5	λ=0.25
Q(λ) - robot only	**EX1**	**Autonomous**	5.85	5.86	7.2	5.07	5.72	6.07	6.95	7.7	6.05
CQ(λ) - human-robot collaboration	**EX2**	$\omega = 6$	5.50	5.78	6.98	5.07	5.61	5.41	6.22	6.44	7.41
	EX3	$\omega = 8$	**4.5**	5.08	7.98	5.11	5.43	6.14	6.39	7.09	7.62
	EX4	$\omega = 10$	5.76	5.81	6.10	5.47	5.04	5.80	5.63	6.48	5.98

Table 7.2 Results for average number of steps to reach the taget optimally

	Experiment Notation	Degree of human intervention	γ=0.99			γ=0.95			γ=0.9		
			λ=0.75	λ=0.5	λ=0.25	λ=0.75	λ=0.5	λ=0.25	λ=0.75	λ=0.5	λ=0.25
Q(λ) - robot only	**EX1**	**Autonomous**	4.85	4.74	4.52	4.6	4.39	3.58	4.84	5.06	4.23
CQ(λ) - human-robot collaboration	**EX2**	$\omega = 6$	4.21	4.39	4.81	4.89	4.51	4.14	4.83	5.06	4.01
	EX3	$\omega = 8$	**3.95**	4.56	4.50	4.30	4.42	4.05	5.30	4.92	4.01
	EX4	$\omega = 10$	4.72	4.49	4.47	4.53	4.83	3.48	4.85	5.28	3.95

Significant improvements in comparison with the $Q(\lambda)$ algorithm (learning with no human intervention) were achieved using the $CQ(\lambda)$ algorithm. In particular, for feasible and optimal solutions, improvements of 23.07% and 18.56% respectively were achieved for a collaboration threshold of $\omega = 8$ using $\gamma = 0.99$ and $\lambda = 0.75$ while the HO was asked to intervene in 30% of the robot navigational trials. To test whether there is a significant difference between the two collaborative modes for this case, the following hypotheses were analyzed using a *T*-test (Table 7.3).

Table 7.3 Hypotheses evaluation for mean number of steps to reach the target

Evaluation	Hypothesis
Mean number of steps to reach the target feasibly	H_{10}: There is no difference between the mean number of steps to reach the target feasibly between the autonomous and the collaborative modes. H_{11}: The mean number of steps to reach the target feasibly is higher for the autonomous mode than the collaborative mode.
Mean number of steps to reach the target optimally	H_{20}: There is no difference between the mean number of steps to reach the target optimally between the autonomous and the collaborative modes. H_{21}: The mean number of steps to reach the target optimally is higher for the autonomous mode than the collaborative mode.

Null hypotheses H_{10} and H_{20} were rejected with *P*-values of 0.018 and 0.019, respectively, which indicates that the mean number of steps to reach the target both feasibly and optimally are not equal. Results of percent of human interventions for various γ and λ combinations with different human-robot collaboration levels are described in Table 7.4. In all three of the autonomy experiments, when the robot learned the environment, human collaboration rate decreased, as expected, with an increase in ω.

Table 7.4 Percent of trials where human interventions occurred for various RL parameters

		γ=0.99			γ=0.95			γ=0.9		
Collaboration level threshold	**Experiment Notation**	**λ=0.75**	**λ=0.5**	**λ=0.25**	**λ=0.75**	**λ=0.5**	**λ=0.25**	**λ=0.75**	**λ=0.5**	**λ=0.25**
$\omega = 6$	**EX2**	36	62	58	24	80	64	50	68	70
$\omega = 8$	**EX3**	**30**	20	44	26	6	42	10	22	60
$\omega = 10$	**EX4**	18	12	4	6	4	4	8	10	44

For the best most significant improvement (EX3) using $\gamma = 0.99$ and $\lambda = 0.75$, the combination of high γ with high λ values that achieved the highest learning performance can be explained due to choosing values of λ large enough to allow longer sequences of values of state-action pairs to updated while restricting the computational solution to a reasonable time. In other experiments for various values of discount factors and eligibility traces no consistency was found in achieving a solution that fits all of human-robot threshold collaboration levels. This may be attributed to cases when human intervention may impaired the ability of the robot to explore the environment autonomously. Therefore, the robot's exploitation was enhanced on the account of less exploration by the human.

7.5 Summary

Evaluating robot performance in the navigational tasks revealed the superiority of the $CQ(\lambda)$ over the standard $Q(\lambda)$ algorithm for high values of discount factors and eligibility traces. In the application described, the mobile robot may fail to form the correct associations between the observed states and those actions that lead to higher rewards. Moreover, even for a well-defined and considerably small state-space, finding optimal policies is memory intensive. To surmount the robot's long interaction times with the environment and to speed up convergence toward satisfactory solutions, first, robot actions over the path were discretized, *i.e.*, state-action space size was limited to a certain number of state-action pairs to reach the target. Second, human intervention and the guidance capabilities that entails were applied in the system, thus decreasing the number of learning episodes and speeding up convergence to realize satisfactory solutions for a task. Human involvement brought superior intelligence to the robot's learning process, and thus affected the learning agent's behavior. Results show that on the one hand, human collaboration accelerated robot learning performance for different collaboration threshold values. On the other hand, the human intervention rate was not consistent with the extent of learning improvement exhibited by the robot.

8. Conclusions and Future Research

Chapter Overview

In this concluding section a comparison between the $CQ(\lambda)$ framework with the current best practice in the area in robot learning is presented. The section ends with a discussion of future work.

8.1 Conclusions

The main contribution of this work is in developing a new learning method. In this thesis, a new learning algorithm which is based on the $Q(\lambda)$-learning algorithm is presented. The proposed algorithm, denoted as the $CQ(\lambda)$ algorithm, enables the collaboration of learning of multiple agents in the environment. Collaboration can expedite the learning by exploiting human intelligence and expertise.

The $CQ(\lambda)$-learning algorithm was developed, tested and applied for two frameworks: (i) learning by multiple agents, and (ii) learning by human-robot systems. In the **first** framework, collaboration involves taking the maximum of state-action values, *i.e.*, the Q-value, across all learning agents at each update step. In the **second** framework, two levels of collaboration are defined for a human-robot learning system: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - a human operator (HO) guides the robot to take an action or a policy and the robot uses the suggestion to replace its own exploration process. The key idea here is to give the robot enough self awareness to adaptively switch its collaboration level from autonomous (self performing) to semi-autonomous (human intervention and guidance). This awareness was represented by a self test of its learning performance.

Theoretically, since a proof of convergence exists for the Q-learning algorithm [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994], the convergence for the $CQ(\lambda)$-learning algorithm for the case where $\lambda = 0$, *i.e.*, collaborative Q ($CQ(0)$) can be explained. For the $CQ(0)$ human-robot case, the only difference to standard Q-learning is that the exploration policy is changed to sometimes be determined by the human. The basic Q-learning convergence proof applies as long as the human does not systematically prevent the use of certain actions in particular states. In other words, as long as the autonomous operation still guarantees that every action is executed infinitely often in every state (a condition of the standard convergence proof is met), the convergence proof directly extends to the human-robot interaction case.

For the $CQ(0)$ case where human is asked to intervene (when system learning performance is low) and his suggestions/selections of actions are not necessarily optimal, $CQ(0)$ will converge also to an optimal solution. Convergence is achieved since the human activities whether or not optimal can be considered to be explorative (actions that have not been tried enough times and can bring to a better solution). The learning agent then uses these activities to exploit its environment. Of course if the human will choose intentionally and consistently the worst possible actions, the algorithm will converge as well, but slower. It is assumed that the human is an expert, and therefore will select beneficial actions. It is reasonable to consider the human for this $CQ(0)$ case as a greedy decision maker at times of human intervention, but differently from well described action-selection methods (*e.g.*, "softmax" or "ε-greedy") here intelligence of a human is considered. Since Q-learning has proven to converge regardless of the action-selection method, $CQ(0)$ will converge to an optimal solution if every state-action pair is visited infinitely often as well. Furthermore, $CQ(0)$ is a special case of Q-learning and therefore will also converge with probability one. For the multiple agents case where a system consists of one collaborative agent, Q_C and several independent Q-learners, it was shown mathematically that the learning function of the collaborative agent converges faster than those of the independent agents.

Extensive experimentation with different robotic systems in a variety of applications demonstrated the strengths and weaknesses of the $CQ(\lambda)$-learning algorithm. Specific applications developed to serve as a test-bed for the $CQ(\lambda)$-learning algorithm were demonstrated in the context of an intelligent environment using a mobile robot for navigation and a fixed-arm robot for the inspection of suspicious objects. Based on the accelerated learning performance of the robotic systems, the results revealed the superiority of the $CQ(\lambda)$ over the standard $Q(\lambda)$ algorithm.

Considering the **first** framework, but in contrast to [Matarić, 1997], where the learning algorithms of multiple robots consist of reward functions that combine individual conditions of a robot, the $CQ(\lambda)$ learning algorithm is based on a state-action value of an agent or learning process updated according to the best performing agent; collaboration is in taking the best state-action values, *i.e.*, the Q value, across all learners at each update step. Similar to the "leader-following Q-learning algorithm" in the joint policy approach described in [Gu and Hu, 2005], which allows cooperation between two agents, a leader and a follower, the $CQ(\lambda)$ learning algorithm enables collaboration of knowledge between many agents. In a multi-agent learning algorithm described in [Bowling and Veloso, 2003], the reward of an agent depends on the joint action of the agents whereas in $CQ(\lambda)$, the Q-value of a collaborative agent depends on the joint Q-value achieved through both by interaction with the environment as well as from the other independent agents existing in the learning system.

In the **second** framework, the proposed RL-based decision-making method is targeted for human-robot collaborative learning systems. The robot makes a decision whether to learn the task autonomously or to ask for human intervention. The goal is to integrate user instructions into an adaptive and flexible control framework and to adjust control policies on-line. To achieve this, user commands at different levels of abstraction are integrated into an autonomous learning system. Based on its learning performance, the robot switches between fully autonomous operation and the request for human intervention. Human suggestions are carried out by the robot, and it performs its learning functions accordingly. The $CQ(\lambda)$ learning algorithm accelerates robot learning using human interaction, thus overcoming the main criticism of RL, *i.e.*, long training periods. Unlike [Papudesi and Huber, 2003; Papudesi *et al.*, 2003], where the rewards are controlled by a human, or as described in [Wang *et al.*, 2003], where user commands are employed for modifying the robot's reward function, in $CQ(\lambda)$ the rewards are achieved by both interaction of a learning agent with the environment and by allowing the human to advise the robot to perform a specific action or to perform a policy. Another method for improving learning performance described in this dissertation was to directly control the Q table of a robot learning problem using a linguistic-based human interface. This method is similar to the approach described in [Papudesi and Huber, 2003; Papudesi *et al.*, 2003]. The difference is that [Papudesi and Huber, 2003; Papudesi *et al.*, 2003] describe a method where the human intervention illustrates the changes in rewards, and under $CQ(\lambda)$ the Q value table was controlled directly by the human. In the former work, the human intervention (besides guiding exploration) actually modifies the task by altering the reward function while in $CQ(\lambda)$-learning the human input is only used to determine which actions to take at this moment and thus to guide the exploration. The task (as defined by the reward function) stays the same.

In [Clause, 1996], a trainer was asked to intervene and suggest help when two extreme Q-values were sufficiently close. This was done by examining the minimum and maximum values and comparing the result to a pre-defined width parameter. In $CQ(\lambda)$, an intervention is based on examining the performance history of the agent over a pre-defined number of learning episodes and comparing it to a performance threshold, rather than examining specific Q-values as in [Clause, 1996].

[Blumberg *et al.*, 2002] describe an autonomous animated dog trained through RL involving human interaction. An example is described in the paper of a reward given to a dog by a human is supervisory signals (*e.g.*, getting a treat). In the case of dog training, a trivial job that technically anyone can do, such a reward can be given not only by an expert but also by any human since dog training can be considered a trivial operation that anyone can do. This is opposed to finding the appropriate reward given to a robot by a human since empting the contents of a bag is not considered as an everyday task. Similar to the approach described in [Thomaz and Breazeal, 2006], human RL signals applied in this approach depend not only on past actions but also on future rewards. When a human is triggered to intervene and to practically control the Q-table of a problem, it takes control over the rewards expected to be achieved in future learning episodes. This type of advising guides the robot and reflects the human desire to control what the robot will do next.

To summarize, a comparison of the $CQ(\lambda)$ framework with the current best practice in robot learning follows:

Robot learning

To become economically attractive, the robots of tomorrow will have to be constructed for a wide variety of tasks. As such, the robot must be able to learn new tasks under new working conditions from its new user in its new environment. Robot learning, therefore, is a very active research area. A major bottleneck, however, has yet to be addressed: traditionally, robot behaviors are often tailored to a specific task. This is not acceptable for a general-purpose robot learning system.

It is well established [*e.g.*, Ehrenmann *et al.*, 2001] that robot learning should make use of human intelligence in the learning process. Human interaction increases the learning capabilities of a robot in realistically complex situations involving many sensors. Human interaction and collaboration in the post-processing and editing of learned behaviors will further elevate robot intelligence. Human-robot collaboration tests using the $CQ(\lambda)$-learning algorithm were performed to accelerate learning [Kartoun *et al.*, 2005; Kartoun *et al.*, 2006 (a); Kartoun *et al.*, 2006 (b)].

Human-robot interaction

Remotely controlled robots are used when a task has to be performed in a hostile, unsafe, inaccessible, or remote environment [Bukchin *et al.*, 2002]. [Crandall *et al.*, 2005] suggest that a human-robot system has when the robot is remotely located: (i) autonomy mode - based on either artificial intelligence or computer control, it allows the robot to act, for a time, without human intervention, and (ii) human-robotic interfaces (HRI) - software installed at the human's location allow him to perceive the world and the robot states as well as to send instructions to the robot. One of the main issues in task-oriented HRI is achieving the right mixture of human and robot autonomy [Adams, 2002; Steinfeld *et al.*, 2006]. [Hirzinger *et al.*, 1993] indicate the importance of HRI in meeting operators' requirements: "an understanding of the human decision process should be incorporated into the design of human-robotic interfaces to support the process humans employ."

In this work several human-robotic interfaces were developed allowing for control over remote robots. The interfaces consist of: (i) real-time visual feedback, (ii) system learning performance reporting, (iii) system mode reporting - autonomous or semi-autonomous, and (iv) human decision making control - when asked to intervene, a human can suggest alternative strategies to the robot.

Analysis of changes between collaboration levels

Relatively few studies have dealt with the subject of dynamic changes in the levels of automation or of human-robot collaboration. Most of these studies were conducted in the context of adaptive automation. They indicated some of the possible advantages in changing collaboration levels (*e.g.*, maintaining acceptable levels of operator workload in a wide range of usage situations), but they also revealed some of the possible problems (*e.g.*, loss of situation and mode awareness). In this work, two levels of collaboration were defined: (i) autonomous - the robot decides which actions to take, acting autonomously according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - the HO suggests actions or policies and the robot replaces its own exploration process, *i.e.*, collaborative $Q(\lambda)$ ($CQ(\lambda)$-learning) is executed. Human-robot collaboration is unnecessary as long as the robot learns policies and adapts to new states without a serious deterioration in its performance. The HO is required to intervene and suggest alternative policies if the robot reports that its learning performance is low.

8.2 Future Research

Many research areas remain open for future expansion of this work:

Improving the learning algorithm

Preliminary results for $CQ(\lambda)$ learning look promising, but there are some issues that still require attention:

- Are there more efficient methods for improving learning performance?
- Optimization techniques should be used to find the learning algorithm parameters that lead to optimal learning performance.
- In the multiple-agents systems described in this work, only instances of one $CQ(\lambda)$ learner per system were described while the others were independent $Q(\lambda)$ learners. Since it was shown that there is no advantage in using a system that contains more than two agents, the development of a system that contains two collaborative agents should be considered as an interesting future research topic. If such a system would follow the conditions described in Section 4.2, it would be expected to achieve a better solution than for the scenarios described in this work, resulting also an identical optimal policy for both $CQ(\lambda)$ learners.

- In the suggested human-robot applications the robot learns according to the proposed $CQ(\lambda)$ learning algorithm, *i.e.*, a HO is triggered to intervene and collaborate with a robot when the robot's learning performance is below a predefined threshold. How can the robot be sure that all HO suggestions are beneficial? It may be the case that some human advise does not contribute to the robot learning process. One way to overcome this issue is to have the robot assess the quality of the human's advice and in the case of ineffective (unhelpful) learning episodes, the robot will reject the human suggestion and revert to an autonomous mode. Under such circumstances, the robot will perform a "backing up procedure," erase the inadequate human suggestions, and continue from there. Then it will compare its performance to that of the human and will choose the best actions, *i.e.*, the robot decides whether to use its information or the human's information.

Enhancing the learning algorithm

The learning algorithm can be enhanced by developing a framework based on the $CQ(\lambda)$ learning algorithm in which the HO learns how to be a good advisor to the robot, *i.e.*, how to be an effective collaborative advisor. This involves teaching the human to be a better advisor and enhancing the human's advisory role in assisting the robot, *i.e.*, the robot teaches the human to be a better advisor/expert in dispensing advice to the robot in a collaborative robot system. The human is given the state of the system, which are the current robot policy and performance. The human decides an action based on the state, and this advice is given to the robot. If the human performance is "bad,"[1] then the robot steps in and decides which suggestions to accept and which to reject. It notifies the human as to the worth of his suggestions (*e.g.*, "very bad", "bad", "average", "good", and "excellent") and the robot can either take control or allow the HO to improve himself by providing better suggestions (*e.g.*, during the next ten trials). Rules for human performance should be defined, such as, "when to ask for help from the human" and "how to evaluate human help and decide whether to accept or reject this help." In this framework the robot is given intelligence rules to not only ask for help, but also to evaluate the quality of the help and decide whether to accept it.

Larger state-space

The tested tasks described in this work consist of relatively small state-action spaces. In problems where Q tables are extremely large and it is infeasible for problems to be calculated in reasonable times, RL methods can be combined with function approximators to give good practical performance despite the lack of a theoretical guarantee of convergence to optimal policies. In applications where

[1] Human performance (*e.g.*, "good" or "bad") are predefined thresholds for a specific learning system.

the state-space of the problem is very large, $CQ(\lambda)$ table representation may not be feasible because of the huge amount of memory used and time required to complete the tables. The use of function approximators such as neural networks enables generalization, which, in turn, allows the use of only a representative sub-set of the entire state-space.

Convergence

Since there is no convergence proof for the $Q(\lambda)$-learning algorithm [Sutton, 1999; Glorennec, 2000], it is claimed here that a convergence proof for $CQ(\lambda)$ where $\lambda > 0$ is also unobtainable. Proving the convergence of both the $Q(\lambda)$ and $CQ(\lambda)$ algorithms is still an open issue in RL.

Additional experimental systems

Developing robot technologies that will be able to mesh further with an external environment using advanced physical interaction equipment (such as hand-gloves, head mounted displays, etc.). Possible virtual reality technologies applied as part of the user interface will extend human control capabilities over the robot, allow him to plan and suggest policies interactively and will increase his understanding of complex robot systems. Additional directions might include the design and implementation of an extended collaboration that includes human intervention using refined or rough intervention policies, robot autonomy, and pure human control. Additionally, robot autonomy will be expanded. One approach may include providing an external support resource such as an additional assisting robot.

9. References

[1] Abe S., Pattern classification: neuro-fuzzy methods and their comparison, *Springer-Verlag*, London, U.K., 2001.

[2] Abramson M. and Wechsler H., Tabu search exploration for on-policy reinforcement learning, *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, no. 20-24, pp. 2910-2915, 2003.

[3] Adams J. A., Critical considerations for human-robot interface development, *AAAI Fall Symposium: Human Robot Interaction*, Technical Report FS-02-03, pp. 1-8, 2002.

[4] Ambrym-Maillard O., Coulom R., and Preux P., Parallelization of the $TD(\lambda)$ learning algorithm, 7th European Workshop on Reinforcement Learning, Naples, 2005.

[5] Aminaiee H. and Ahmadabadi M. N., Learning individual skills and team behaviors for distributed object pushing, *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, pp. 1202-1209, 2006.

[6] Asadpour M., Ahmadabadi M. N., and Siegwart R., Heterogeneous and hierarchical cooperative learning via combining decision trees, *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2684-2690, 2006.

[7] Asoh H., Vlassis N. A., Motomura Y., Asano F., Hara I., Hayamizu S., Ito K., Kurita T., Matsui T., Bunschoten R. and Kröse Ben J. A., Jijo-2: an office robot that communicates and learns, *IEEE Intelligent Systems*, vol. 16. Num 5. pp. 46-55, 2001.

[8] Bagnell J. A., A robust architecture for multiple-agent reinforcement learning, M.Sc. Thesis, University of Florida, 1998.

[9] Bakker B., Zhumatiy V., Gruener G., and Schmidhuber, J., Quasi-online reinforcement learning for robots, *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 2997-3002, 2006.

[10] Baltus G., Fox D., Gemperle F., Goetz J., Hirsch T., Magaritis D., Montemerlo M., Pineau J., Roy N. Schulte J. and Thrun S., Towards personal service robots for the elderly, *Proceedings of the Workshop on Interactive Robotics and Entertainment (WIRE)*, Pittsburgh, 2000.

[11] Bellman R. and Kalaba R., Dynamic programming and modern control theory, *NY: Academic Press Inc.*, 1965.

[12] Bennett K. P. and Bredensteiner E. J., Multicategory classification by support vector machines, *Computational Optimizations and Applications*, vol. 12, pp. 53-79, 1999.

[13] Bertsekas D. P. and Tsitsiklis J. N., Parallel and distributed computation numerical methods, *Englewood Cliffs, N.J., Prentice Hall*, 1989.

[14] Bhanu B., Leang P., Cowden C., Lin Y., and Patterson M., Real-time robot learning, *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 491-498, 2001.

[15] Blumberg B., Downie M., Ivanov Y. A., Berlin M., Johnson M. P., and Tomlinson B., Integrated learning for interactive synthetic characters. *SIGGRAPH 2002*, pp. 417-426, 2002.

[16] Bowling M. and Veloso M., Simultaneous adversarial multi-robot learning, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 699-704, 2003.

[17] Breazeal C., Hoffman G., and Lockerd A., Teaching and working with robots as a collaboration, *AAMAS 2004*, pp. 1030-1037, 2004.

[18] Broadbent R. and Peterson T., Robot learning in partially observable, noisy, continuous worlds, *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4386- 4393, Barcelona, Spain, 2005.

[19] Bukchin J., Luquer R., and Shtub A., Learning in tele-operations, *IIE Transactions*, vol. 34, no. 3, pp. 245-252, 2002.

[20] Carreras M., Ridao P. Batlle J. and Nicosevici T., Efficient learning of reactive robot behaviors with a neural-Q learning approach, *IEEE International Conference on Automation, Quality and Testing*, Romania, 2002.

[21] Clouse J. A., An introspection approach to querying a trainer, *Technical Report: UM-CS-1996-013*, University of Massachusetts, Amherst, MA, 1996.

[22] Clouse J. A. and Utgoff P. E., A teaching method for reinforcement learning, *Machine Learning: Proceedings of the Ninth International Conference*, pp. 92-101, San Mateo, CA: Morgan Kaufmann, 1992.

[23] Coelho J. A., Piater J. H., and Grupen R. A., Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot , *Robotics and Autonomous Systems*, vol. 37, no. 2, pp. 195-218, 2001.

[24] Cortes C. and Vapnik V., Support vector networks, *Machine Learning*, vol. 20, pp. 1-25, 1995.

[25] Crandall J. W., Goodrich M. A., Olsen D. R., and Nielsen C. W., Validating human-robot interaction schemes in multi-tasking environments, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, Special Issue on Human-Robot Interaction*, vol. 35, no. 4, pp. 438-449, 2005.

[26] Cristianini N. and Shawe-Taylor J., Support vector machines and other kernel-based learning methods, *Cambridge University Press*, Cambridge, U.K., 2003.

[27] Dahmani Y. and A. Benyettou, Seek of an optimal way by Q-learning, *Journal of Computer Science*, vol. 1, no. 1, pp. 28-30, 2005.

[28] Djordjevic D. and Izquierdo E., 2007, An object and user driven system for semantic-based image annotation and retrieval, *IEEE Transactions on Circuits and Systems for Video Technology* (Accepted for future publication), 2007.

[29] Driessens K. and Džeroski S., Integrating guidance into relational reinforcement learning, *Machine Learning*, vol. 57, pp. 271-304, 2004.

[30] Dube S., El-Saden S., Cloughesy T. F., and Sinha U., Content based image retrieval for MR image studies of brain tumors, *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3337-3340, 2006.

[31] Dvoretzky A., On stochastic approximation, *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, University of California Press*, 1956.

[32] Edan Y., Kartoun U. and Stern H., Cooperative human-robot learning system using a virtual reality telerobotic interface, *Conference on Advances in Internet Technologies and Applications*, Purdue University, West Lafayette, Indiana, U.S.A., 2004.

[33] Ehrenmann, M., Rogalla, O., Zollner, R. and Dillmann, R., Teaching service robots complex tasks: programming by demonstration for workshop and household environments, *Proceedings of the 2001 International Conference on Field and Service Robots (FSR)*, vol. 1, pp. 397-402, Helsinki, Finland, 2001.

[34] Fong T., Thorpe C., Baur C., Collaboration, dialogue, and human-robot interaction, *10th International Symposium of Robotics Research*, Lorne , Victoria , Australia , November 2001.

[35] Freire da Silva V., Reali Costa A.H., and Lima, P., Inverse reinforcement learning with evaluation, *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 4246-4251, 2006.

[36] Heguy O., Rodriguez N., Luga H., Jessel J. P. and Duthen Y., Virtual environment for cooperative assistance in teleoperation, *The 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2001.

[37] Tesauro G. and Kephart J. O., Pricing in agent economies using multi-agent Q-learning, *Autonomous Agents and Multi-Agent Systems Journal*, Springer, vol. 5, no. 3, pp. 289-304, 2002.

[38] Glorennec P. Y., Reinforcement learning: an overview, *European Symposium on Intelligent Techniques*, Aachen, Germany, 2000.

[39] Goertzel B. and Venuto J., Accurate SVM text classification for highly skewed data using threshold tuning and query-expansion-based feature selection, *International Joint Conference on Neural Networks*, pp. 1220-1225, 2006.

[40] Grounds M. and Kudenko D., Parallel reinforcement learning by merging function approximations, *Sixth European Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS'06)*, 2006.

[41] Grounds M. and Kudenko D., Parallel reinforcement learning with linear function approximation, *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.

[42] Gu D. and Hu H., Fuzzy multi-agent cooperative Q-learning, *Proceedings of IEEE International Conference on Information Acquisition*, Hong Kong, China, pp. 193-197, 2005.

[43] Guo M., Liu Y., and Malec J., A new Q-learning algorithm based on the metropolis criterion, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 34, no. 5, pp. 2140-2143, 2004.

[44] Harvey N. R., Perkins S., Pope P. A., Theiler J., David N. A. and Porter R. B., Investigation of automated feature extraction using multiple data sources, *Proceedings of the AEROSENSE*. Orlando, 2003.

[45] Heisele B., Ho P., Wu J., and Poggio T., Face recognition: component-based versus global approaches, *Elsevier Science Inc.*, New York, U.S.A., vol. 91, pp. 6-21, 2003.

[46] Hellström T., Teaching a robot to behave like a cockroach, *Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield U.K.*, 2005.

[47] Hirzinger G., Brunner B., Dietrich J., and Heindl J., Sensor-based space robotics-ROTEX and its telerobotic features, *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 649-663, 1993.

[48] Honglak L., Yirong S., Chih-Han Y., Singh G., and Ng, A.Y. Quadruped robot obstacle negotiation via reinforcement learning, *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 3003-3010, 2006.

[49] Howard A., Probabilistic navigation: coping with uncertainty in robot navigation tasks, *Ph.D. Dissertation, Department of Computer Science and Software Engineering, University of Melbourne*, Australia, 1999.

[50] Isbell C. L., Shelton C. R., Kearns M., Singh S., and Stone P, A social reinforcement learning agent, *Proceedings of the 5th International Conference on Autonomous Agents*, pp. 377-384, Montreal, Canada, 2001.

[51] Jaakkola T., Jordan M. I., and Singh S. P., On the convergence of stochastic iterative dynamic programming algorithms, *Neural Computation*, vol. 6, no. 6, pp. 1185-1201, 1994.

[52] Jiuxian L., Siyu X., and Liangzheng X., Face detection based on self-skin segmentation and wavelet support vector machine, *International Conference on Computational Intelligence and Security*, vol. 1, pp. 755-758, 2006.

[53] Kaelbling L. P., Littman M. L., and Moore A. W., Reinforcement learning: a survey, *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[54] Kartoun, U., A human-robot collaborative learning system using a virtual reality telerobotic interface, *Ph.D. Thesis Proposal, Department of Industrial Engineering and Management at the Ben-Gurion University of the Negev*, Israel, 2003.

[55] Kartoun U., Stern H., and Edan Y., 2004, Virtual reality telerobotic system, *e-ENGDET 2004 4th International Conference on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University Yorkshire, U.K., 2004.

[56] Kartoun U., Stern H., Edan Y., Feied C., Handler J., Smith M. and Gillam M., Collaborative $Q(\lambda)$ reinforcement learning algorithm - a promising robot learning framework, *IASTED International Conference on Robotics and Applications (RA 2005)*, October 31 - November 2, Cambridge, U.S.A., 2005.

[57] Kartoun U.(a), Stern H., Edan Y., Human-robot collaborative learning of a bag shaking trajectory, *The Israel Conference on Robotics (ICR 2006)*, Tel Aviv University, Faculty of Engineering, June 29, 2006.

[58] Kartoun U.(b), Stern H., Edan Y., Human-robot collaborative learning system for inspection, *IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan. Finalist (5 papers) for the Best Student Paper Competition, Oct. 8 - Oct. 11, 2006.

[59] Kartoun U.(c), Stern H. and Edan Y., Bag classification using support vector machines, *Applied Soft Computing Technologies: The Challenge of Complexity Series: Advances in Soft Computing, Springer Berlin / Heidelberg*, ISBN: 978-3-540-31649-7, pp. 665-674, 2006.

[60] Kerr J. and Compton P., Toward generic model-based object recognition by knowledge acquisition and machine learning, *Workshop on Mixed-Initiative Intelligent Systems, International Joint Conference on Artificial Intelligence*, Mexico, pp. 80-86, 2003.

[61] Kollar T. and Roy N., Using reinforcement learning to improve exploration trajectories for error minimization, *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 3338-3343, 2006.

[62] Kretchmar R. M., Parallel reinforcement learning, *The 6th World Conference on Systemics, Cybernetics*, and Informatics, 2002.

[63] Kreuziger J., Application of machine learning to robotics - an analysis, *Proceedings of the Second International Conference on Automation, Robotics and Computer Vision*, Singapore, 1992.

[64] Kui-Hong P., Jun J., and Jong-Hwan K., Stabilization of biped robot based on two mode Q-learning, *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, pp. 446-451, New Zealand, 2004.

[65] Längle T., Lüth T. C., Stopp E. and Herzog G., Natural language access to intelligent robots: explaining automatic error recovery, A. M. Ramsay (ed.), *Artificial Intelligence: Methodology, Systems, Applications*. Amsterdam. pp. 259-267, 1996.

[66] Lee C. H., Yang H. C., Chen T. C., and Ma S. M., A comparative study on supervised and unsupervised learning approaches for multilingual text categorization, *First International Conference on Innovative Computing, Information and Control*, vol. 2, pp. 511-514, 2006.

[67] Lin X. D., Peng H., and Liu B., Support vector machines for text categorization in chinese question classification, *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 334-337, 2006.

[68] Lockerd A. and Breazeal C., Tutelage and socially guided robot learning, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[69] Ma J. and Ahalt S., OSU SVM Classifier Matlab Toolbox (ver. 3.00), 2003.

[70] MackWorth A. K., Poole D. and Goebel R. G., Computational intelligence: a logical approach, *Oxford University Press*, 1998.

[71] Martínez-Marín T. and Duckett T., Fast reinforcement learning for vision-guided mobile robots, *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[72] Matarić M. J., Reinforcement learning in the multi-robot domain, *Autonomous Robots, Kluwer Academic Publishers*, vol. 4, pp. 73-83, 1997.

[73] The MathWorks Inc., 1998, Matlab Software, Image Processing User's Guide.

[74] Menegatti E., Cicirelli G., Simionato C., Distante A., and Pagello E, Reinforcement learning based omnidirectional vision agent for mobile robot navigation, *Workshop Robotica del IX Convegno della Associazione Italiana Intelligenza Artificiale*, 2004.

[75] Meng X., Chen Y., Pi Y., and Yuan Q., A novel multiagent reinforcement learning algorithm combination with quantum computation, *The Sixth World Congress on Intelligent Control and Automation*, vol. 1, pp. 2613-2617, 2006.

[76] Mihalkova L. and Mooney R., Using active relocation to aid reinforcement, *Proceedings of the 19th International FLAIRS Conference (FLAIRS-2006)*, pp. 580-585, Melbourne Beach, Florida, 2006.

[77] Nakajima C., Pontil M., Heisele B., and Poggio T., Person recognition in image sequences: the MIT espresso machine system, *IEEE Transactions on Neural Networks*, 2000.

[78] Nason S. and Laird J. E., Soar-RL: integrating reinforcement learning with soar, *Proceedings of the International Conference on Cognitive Modeling*, pp. 51-59, 2004.

[79] Natarajan S. and Tadepalli P., Dynamic preferences in multi-criteria reinforcement learning, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, Bonn, Germany, 2005.

[80] Nehmzow U. and Walker K., Quantitative description of robot-environment interaction using chaos theory, *Robotics and Automation Systems*, vol. 53, pp. 177-193, 2005.

[81] Motamed M. and Yan J., A reinforcement learning approach to lift generation in flapping MAVs: simulation results, *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 2150-2155, 2006.

[82] Nourbakhsh I. R., Bobenage J., Grange S., Lutz R., Meyer R., and Soto A., An affective mobile robot educator with a full-time job, *Artificial Intelligence*, vol. 114, num. 1-2, pp. 95-124, 1999.

[83] Otsu N., 1979, A threshold selection method from gray level histograms, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62-66.

[84] Papudesi V. N. and Huber M., Learning from reinforcement and advice using composite reward functions, *Proceedings of the 16th International FLAIRS Conference*, pp. 361-365, St. Augustine, FL, 2003.

[85] Papudesi V. N., Wang Y., Huber M., and Cook D. J., Integrating user commands and autonomous task performance in a reinforcement learning framework, *AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*, Stanford University, CA., 2003.

[86] Peng J. and Williams R., Incremental multi-step Q-learning, *Machine Learning*, vol. 22, no. 1-3, pp. 283-290, 1996.

[87] Peters J. and Schaal S., Reinforcement learning for parameterized motor primitives, *International Joint Conference on Neural Networks*, pp. 73-80, 2006.

[88] Ribeiro C., Reinforcement learning agents, *Artificial Intelligence Review*, vol. 17, no. 3, pp. 223-250, 2002.

[89] Rosenstein M. T., Fagg A. H., Ou S., and Grupen R. A., User intentions funneled through a human-robot interface, *Proceedings of the 10th International Conference on Intelligent User Interfaces*, 257-259, 2005.

[90] Schölkopf B., Burges C., and Vapnik V., Extracting support data for a given task, In U. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, AAAI Press*, pp. 252-257, 1995.

[91] Scheffer T. and Joachims T., Expected error analysis for model selection, *Proceedings of the International Conference on Machine Learning*, 1999.

[92] Sheridan T. B., Supervisory control, *In: Handbook of Human Factors/Ergonomics*, G. Salvevely (Ed.), Wiley, ISBN 0471116, N.Y., 1987.

[93] Sheridan T. B., Telerobotics, automation, and human supervisory control, *M.I.T. Press*, 1992.

[94] Shigeo A., Support vector machines for pattern classification, *Springer*, 2001.

[95] Smart W. D., Making reinforcement learning work on real robots, *Ph.D. Dissertation, Brown University*, 2002.

[96] Smart W. D. and Kaelbling L., Practical reinforcement learning in continuous spaces, *Proceedings of the 17th International Conference on Machine Learning*, pp. 903-910, 2000.

[97] Sorid D. and Moore S. K., The virtual surgeon, *IEEE SPECTRUM*. pp. 26-39, 2000.

[98] Park M. S. and Choi J. Y., New reinforcement learning method using Multiple Q-tables, School of Electrical Engineering & Computer Science, Seoul National University, 2002.

[99] Steinfeld A. M., Fong T. W., Kaber D., Lewis M., Scholtz J., Schultz A., and Goodrich M., Common metrics for human-robot interaction, *Human-Robot Interaction Conference, ACM*, March, 2006.

[100] Sutton R. S., Learning to predict by the method of temporal differences, *Machine Learning*, vol. 3, pp. 9-44, 1988.

[101] Sutton R. S. and Barto A. G., Reinforcement learning: an introduction, *Cambridge, MA: M.I.T. Press*, 1998.

[102] Sutton R. S., Open theoretical questions in reinforcement learning, *Lecture Notes In Computer Science, Proceedings of the 4th European Conference on Computational Learning Theory*, vol. 1572, pp. 11-17, 1999.

[103] Theocharous G. and Mahadevan S., Approximate planning with hierarchical partially observable markov decision processes for robot navigation, *IEEE Conference on Robotics and Automation*. Washington D.C., 2002.

[104] Thomaz A. and Breazeal C. 2006 "Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance, *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[105] Touzet C. F., Q-learning for robots, the handbook of brain theory and neural networks, *Cambridge, MA: M. Arbib editor, M.I.T. Press*, 934-937, 2003.

[106] Touzet C. F., Distributed lazy Q-learning for cooperative mobile robots, *International Journal of Advanced Robotic Systems*, vol. 1, num. 1, pp. 5-13, 2004.

[107] Vapnik V., Statistical learning theory, *JohnWiley and Sons*, N.Y., 1998.

[108] Wang B., Li J. W., and Liu H., A heuristic reinforcement learning for robot approaching objects, *2006 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1-5, December 2006.

[109] Wang Y., Huber M., Papudesi V. N., and Cook D. J., User-guided reinforcement learning of robot assistive tasks for an intelligent environment, *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems*, vol. 1, pp. 424-429, 2003.

[110] Watkins C. J. C. H., Learning from Delayed Rewards, *Ph.D. Dissertation, Cambridge University*, 1989.

[111] Watkins C. J. C. H. and Dayan P., Q-learning, *Machine Learning*, vol. 8, pp. 279-292, 1992.

[112] Weber C., Wermter S., and Zochios A., Robot docking with neural vision and reinforcement, *Knowledge-based Systems*, vol. 17, num. 2-4, pp. 165-72, 2004.

[113] Yanco H. A., Baker M., Casey R., Chanler A., Desai M., Hestand D., Keyes B., and Thoren P., Improving human-robot interaction for remote robot operation, *Robot Competition and Exhibition Abstract, National Conference on Artificial Intelligence (AAAI-05)*, July 2005.

[114] Young S. and Downs T., CARVE - a constructive algorithm for real-valued examples, *IEEE Transactions on Neural Networks*, vol. 9., num. 6., pp. 1180-1190, 1998.

[115] Ziemke T., Adaptive behavior in autonomous agents presence - teleoperators and virtual environments, *Special issue on Autonomous Agents*, vol. 7, num. 6., pp. 564-587, 1998.

[116] Zimmerman T. and Kambhampati S., What next for learning in AI planning?, *International Conference on Artificial Intelligence*, Las Vegas. U.S.A., 2001.

[117] Zhu W. and Levinson S., Vision-based reinforcement learning for robot navigation, *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 1025-1030, Washington D.C., 2001.

Appendix I. Installation and Configuration - Motoman UP-6 Fixed-Arm Robot

The system consists of two separated applications exchanging parameters through the Windows registry: (i) digital scale, and (ii) learning system.

Pre-request Softwares

Install the following softwares:

1) XP Operating System + Internet Information Server (IIS) + FrontPage 2000 Server Extensions
2) Visual Studio .Net 2003
3) Matlab 7.1
4) Motocom32.DLL and plug

Communication Link between the Digital Scale and the Learning System

Efficient data exchange between the robot learning system and the digital scale is done by defining variables in the Windows registry (Fig. I.1).

Fig. I.1 Data exchange through the Windows registry

Initial variables configuration in the Windows registry (Fig. I.2 and Fig. I.3).

Fig. I.2 Registry editor operation

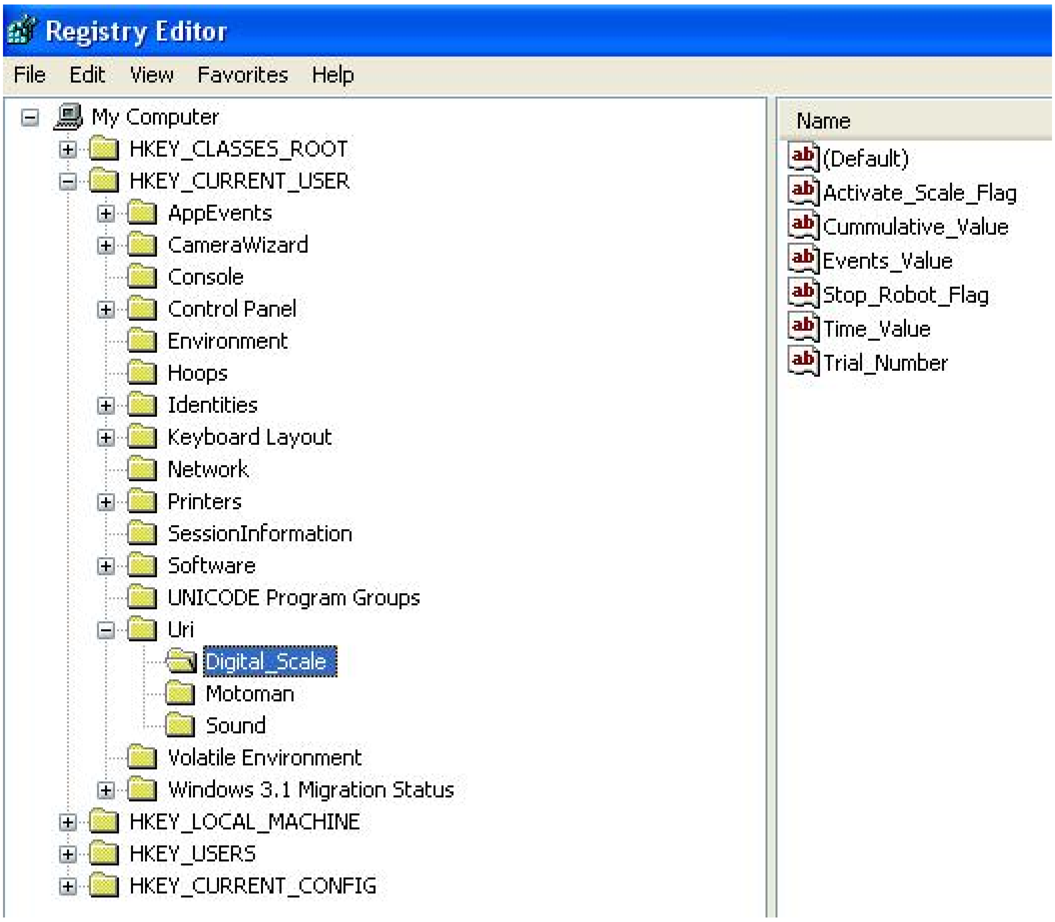


Fig. I.3 Registry variables

<u>The variables</u>:

1) Activate_Scale_Flag - whether to enable or disable the scale (0 or 1).
2) Cummulative_Value - The actual measured weight (grams).
3) Events_Value - Whether an object was dropped on the scale (0 or a positive value in grams).
4) Stop_Robot_Flag - Disable/enable robot (0 or 1).
5) Time_Value - raw time value measured (ms) - starts when a new learning episode is performed.
6) Trial_Number - what is the current number of the learning episode.

<u>The Digital Scale</u>

<u>Specifications</u>

The "Mettler Toledo" SB12001 digital scale (Fig. I.4) has an accuracy of ±0.1 grams and is equipped with RS232C allowing its measurements to be read by a PC program.

Fig. I.4 “Mettler Toledo” SB12001 digital scale

The following switch (Fig. I.5) allows restarting the scale remotely from the human operator computer. Restarting the scale recalibrates it.

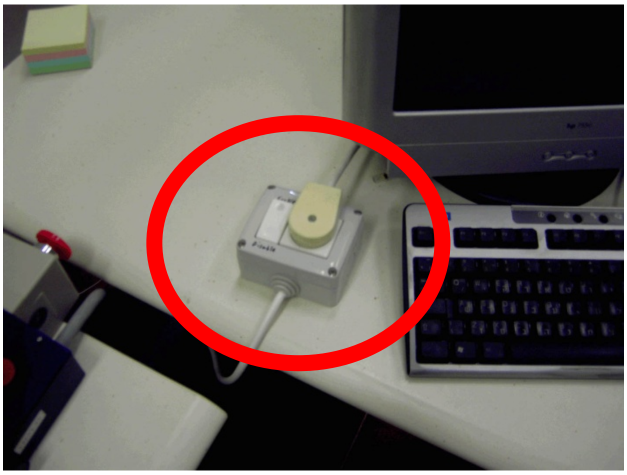


Fig. I.5 Digital scale remote switch

An inspection surface should be placed on top of the scale under the robot gripper (Fig. I.6).

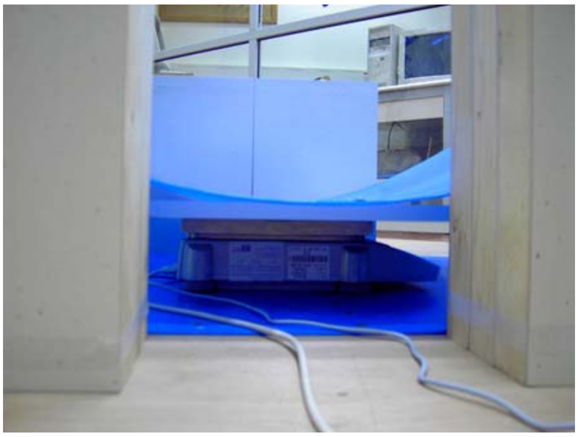


Fig. I.6 Digital scale and inspection surface located under the robot gripper

The digital scale is connected to a computer through RS-232 communication cable. In case the computer has only one serial port (two are required - for the controller and for the scale), a "USB to RS-232" converter should be used (Fig. I.7).

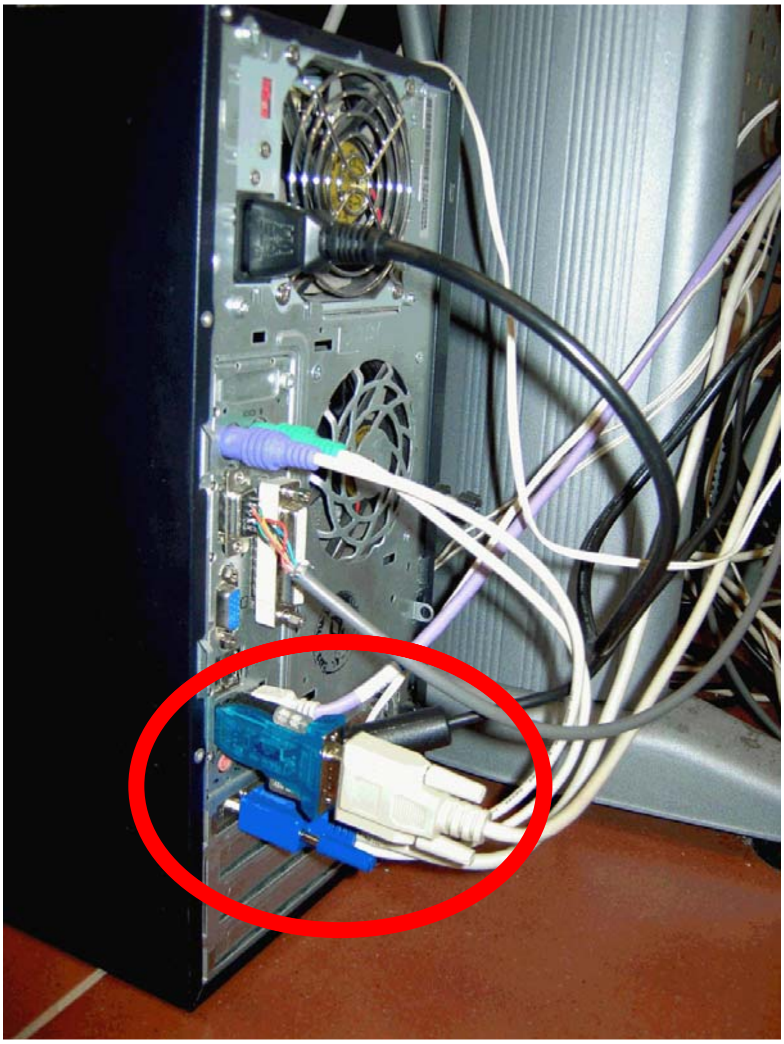


Fig. I.7 USB to serial converter connected to a PC

The Software

The software is written in VB .Net and running under Windows XP. The main VB project file name is: “Digital_Scale.vbproj”. The software reads values from a digital scale through serial communication and presents them as text fields and graphs (Fig. I.8).

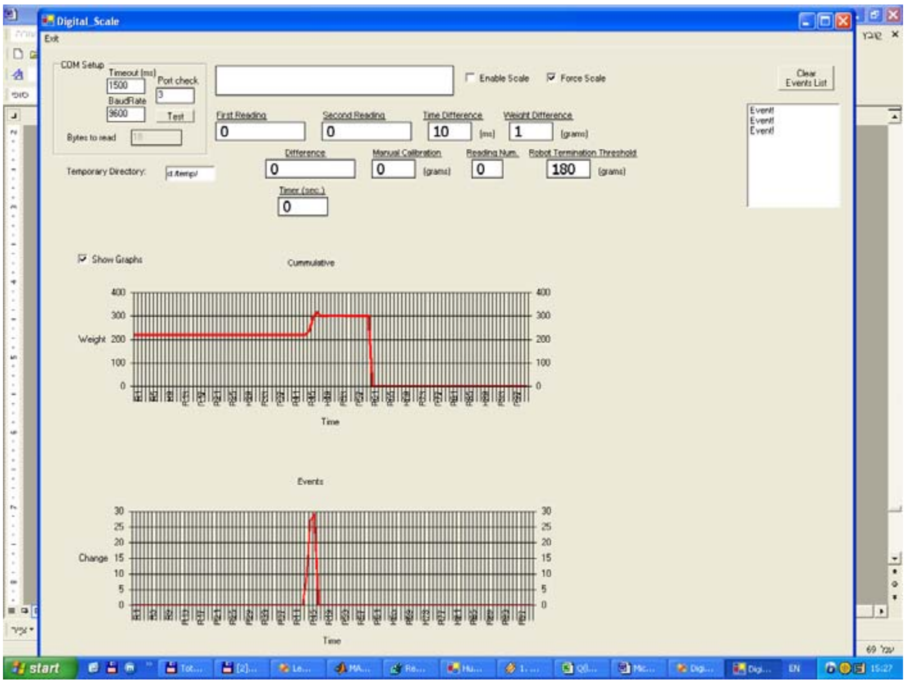


Fig. I.8 Digital scale software

Essential port settings (pre-programmed) are as follows (Fig. I.9).

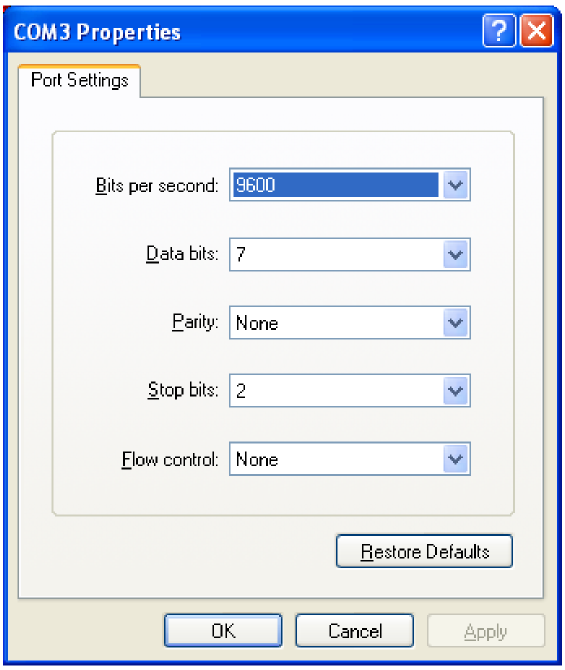


Fig. I.9 Digital scale serial communication configuration parameters

Parameters:

1) Communication (COM Setup): definition of serial communication parameters: baud rate: 9600, port number: 3, timeout (ms): 1500 (See also Fig. I.9).
2) Temporary Directory: a path for writing log files.
3) Force Scale Checkbox - enables scale operation regardless of the registry.
4) Cumulative Graph - the current weight (in grams) on the scale.
5) Events Graph - compares between two measurements ("First Reading" and "Second Reading" textboxes) to identify changing weight events. "Time Difference" is a configurable parameter for determining the time (ms) between the two measurements.
6) "Weight Difference" - sensitivity to events.
7) "Robot Termination Threshold" - if the weight on the scale is above this value the registry is updated and triggers the robot to halt.
8) Other textboxes - required for string manipulation of raw information arriving from the scale.

Remark: to close the program, use the "Exit" in the top left corner to avoid communication conflicts.

The Learning System

UP-6 Motoman Robot and XRC controller

The Motoman UP-6 robot located over an inspection surface (Fig. I.10) is controlled by a XRC controller (Fig. I.11).

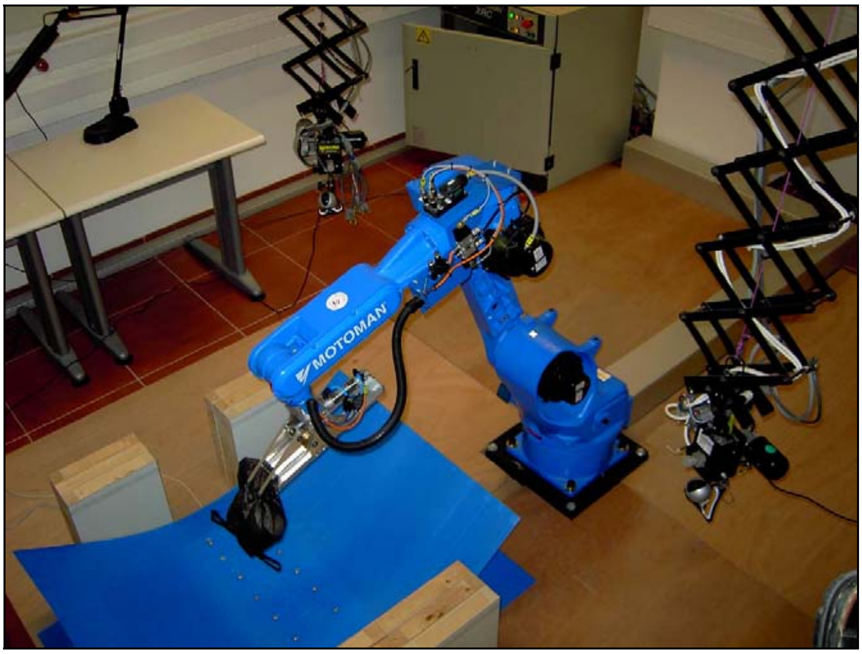


Fig. I.10 Motoman UP-6 robot

Fig. I.11 XRC controller

Creating programs that can be interrelated by the XRC controller and performed by the robot is made by using the teach pendant (Fig. I.12).

Fig. I.12 Teach pendant

Using the XRC remote control box (Fig. I.13) located near the human operator computer enables easier operation since the human does not have to approach the controller:

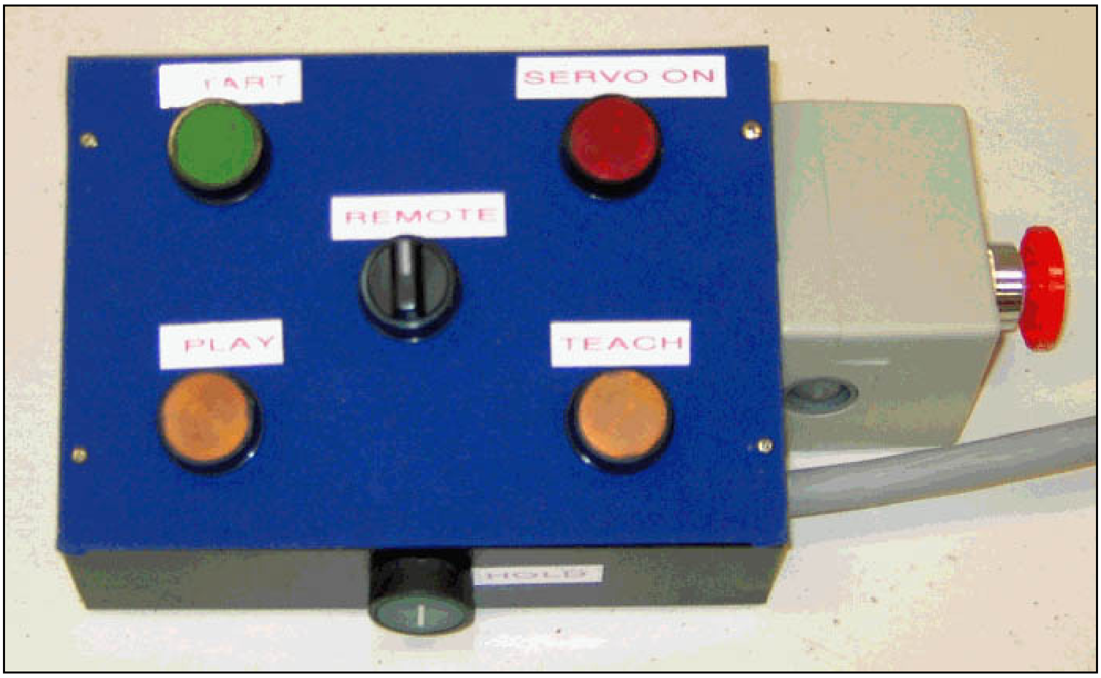


Fig. I.13 XRC controller remote control box

Communication between a PC computer and the XRC controller

The "Motocom32.dll" libraries and the additional security plug are required for setting a communication link between a PC and the Motoman UP-6 robot. A PC computer is connected to the XRC controller by RS-232 cable configured as follows:

Serial Cable Setting:

XRC Controller (9 pins connector)	PC (9 pins connector)
2	3
3	2
5	5
7	8
8	6 . 4

The following parameters should be configured in the XRC controller by using the teach pendant: Under "Management" security mode, choose "Parameter → RS" menu, make sure that the following parameters are set: RS000: 2, RS001: 0, RS002: 0, RS003: 5,RS004: 0, RS005: 0, RS030: 8 - Number of data bits, RS031: 0 - Number of stop bits, RS032: 0 - Parity check, RS033: 7 - 9600 baud, RS034: 30 - TimerA, RS035: 200 - TimerB, RS036: 10 - Retry 1, RS037: 3 - Retry 2, RS038: 0 - Block check method.

Under "Management" security mode, choose "In / Out $\rightarrow$ PSEUDO INPUT SIG" menu and make sure that the following parameters are set: #8214 INHIBIT IO: OFF, #8215 CMD REMOTE SEL: ON, #8216 INHIBIT PP/PANEL: OFF.

Under "Yaskawa" security mode, choose "Parameter" menu and make sure that the following parameter is set: FD003: 1 - Data transmission.

Physical connection of the serial cable to a PC and to the XRC controller is presented in Fig. I.14 and Fig. I.15.

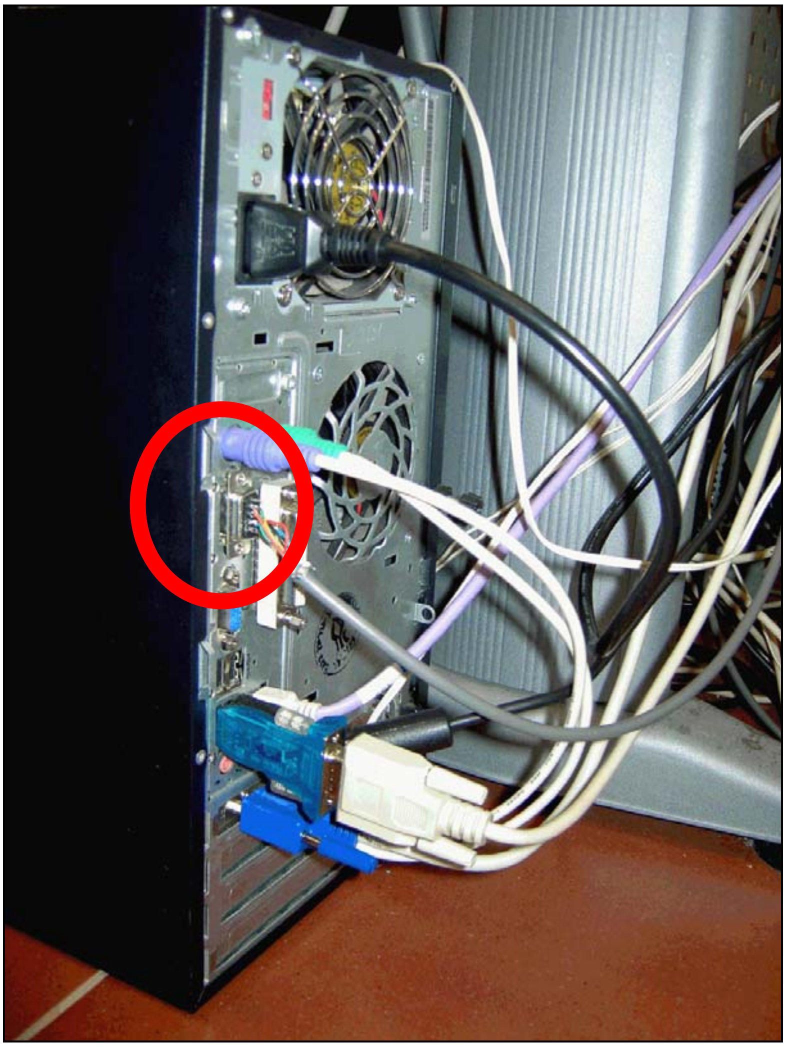


Fig. I.14 Serial cable connected to a PC

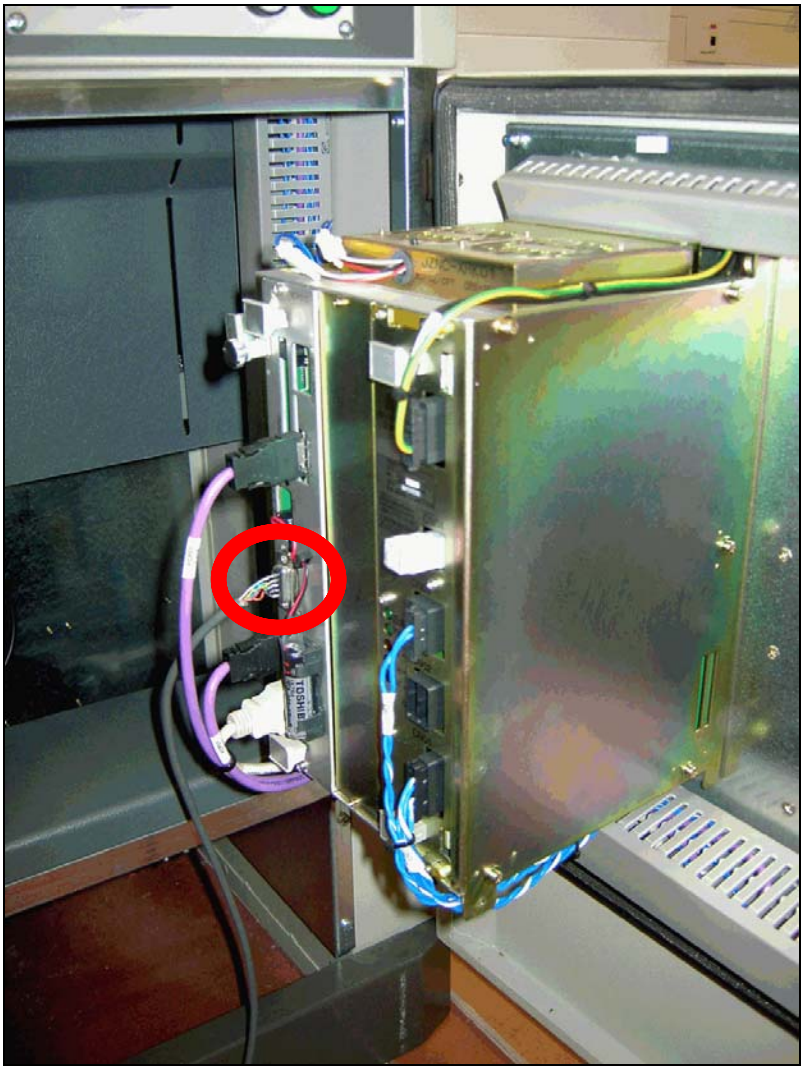


Fig. I.15 Serial cable connected to the XRC controller

<u>Remark</u>: make sure that the plug supplied with the Motocom32.DLL software is connected to the PC.

<u>Visual Feedback</u>

1) Connect a USB web-camera to a PC computer under Windows XP.
2) Install the “WebCam32” software and configure parameters as follows (Fig. I.16):

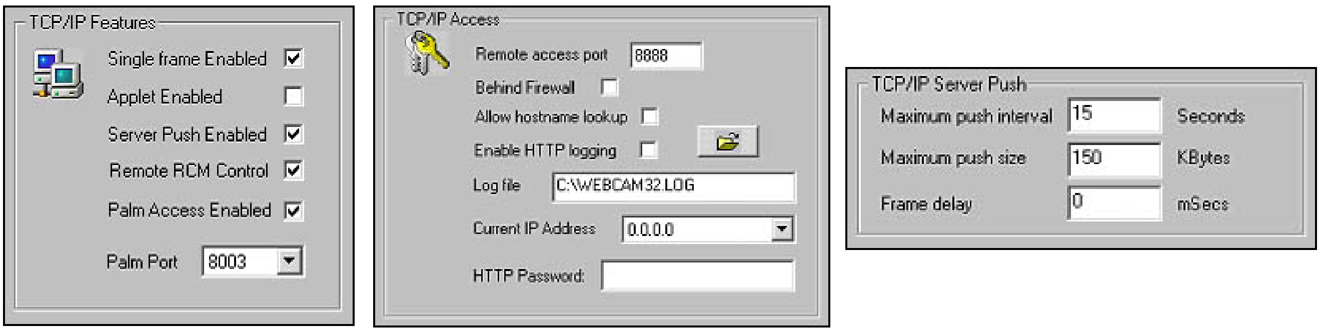


Fig. I.16 Communication configuration

By opening a web browser and typing the correct camera script[1] “htm” file a real-time visual feedback is shown (Fig. I.17):

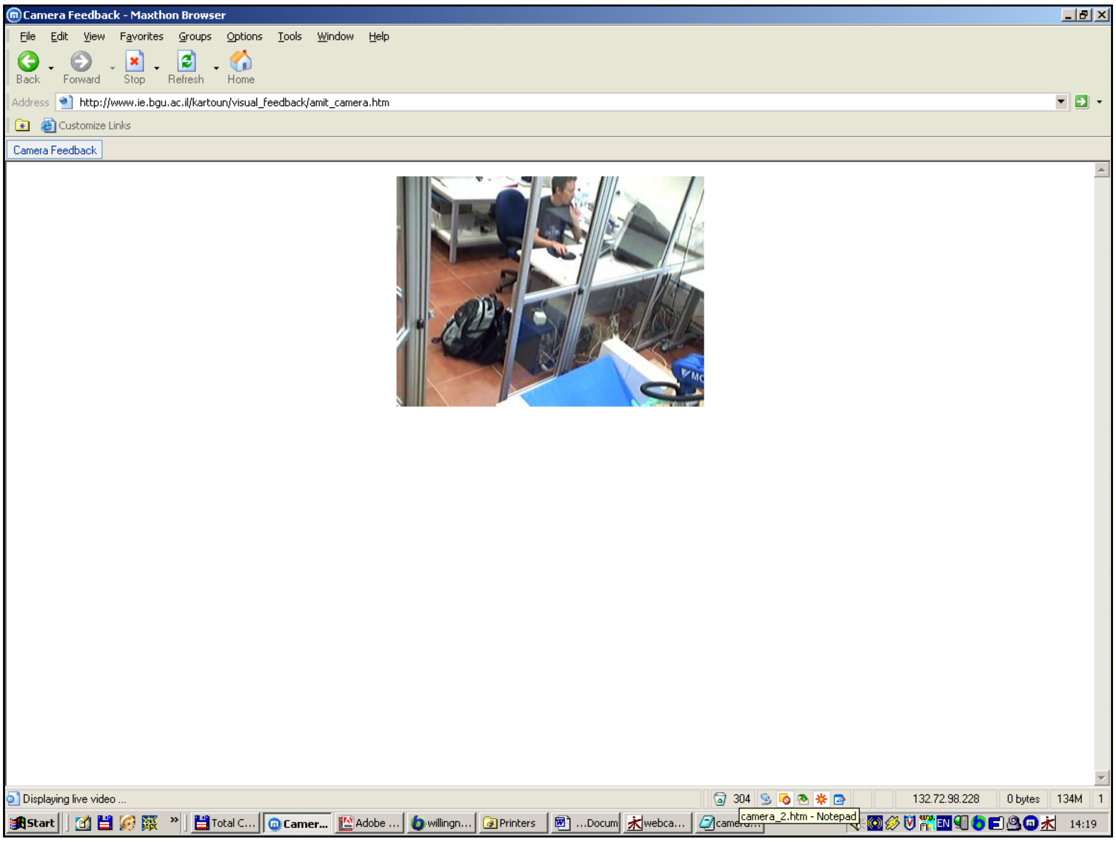


Fig. I.17 Visual feedback at a web browser

The Software - “Human-robot collaboration learning system”

The software is written in VB .Net and running under Windows XP. The main VB project file name is: “Learning_System.vbproj”.

The control tabs

The software consists of several tab views. Several of the tab views are not necessary for operating the learning system and are used for system additional development. The essential tabs are described as follows:

---

[1] The “htm” file should be placed on the department server, for example:
http://www.ie.bgu.ac.il/kartoun/visual_feedback/camera_1.htm
Additionally, update all IP numbers (*e.g.*, 132.72.98.228) at the “htm” file according to the IP of PC where the camera is connected to. If more than one camera is required, each camera should be connected to a different PC with a different IP address.

1) The "Human-Robot Collaboration" tab (Fig. I.18):

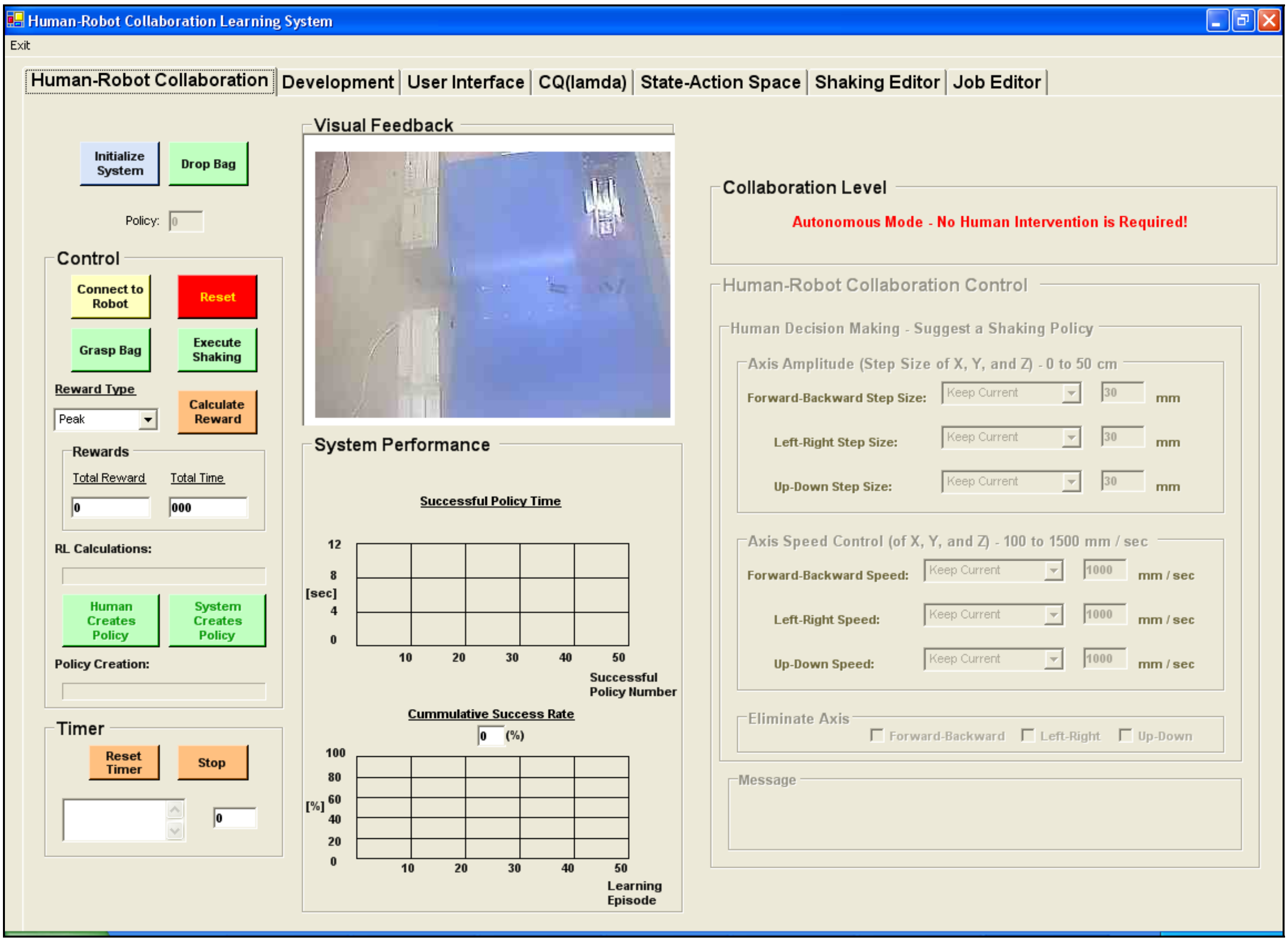


Fig. I.18 Human-robot collaboration tab

Visual feedback - displays a real-time feedback from the remote robotic environment.

System performance graphs - give an indication of how well the system learns: (i) comparison between the last successful policy and the mean of all successful policies, and (ii) cumulative success rate - the parentage of successful policies.

"Initialize System" button - builds a random starting policy and a preliminary Q table.

"Drop Bag" button - open the robot gripper.

"Connect to Robot" button - starts the robot controller and allows robot motion.

"Reset" button - disables robot operation.

"Grasp Bag" button - runs predefined programs denoted "GRASP1.JBI" for grasping a bag and "GRASPH.JBI" for lifting to a starting shaking location over an inspection surface.

"Execute Shaking" button - performs a shaking policy operation.

"Reward Type" - choosing whether the reward will be based on cumulative weight measured from a digital scale of whether it is based on events.

"Calculate Reward" button - after shaking was executed, a reward is being calculated based on the chosen reward type and updates the current policy (the progress is shown by the "RL calculations" track bar).

"Human Creates Policy" button - when human is asked to intervene he can adjust parameters in the "Human-Robot Collaboration Control" then by pressing the button a new policy is created. The data is written into a "csv" file which includes states, action, rewards, speeds and adjacent state distances for the policy and the current Q table. The "Policy creation" progress bar gives an indication for the new policy creation.

"System Creates Policy" button - writes the current policy and Q table to a "csv" file.

Timer - contains a textbox representing the shaking time in seconds. The "Reset Timer" and "Stop" buttons should be used only if there is a communication problem with the robot.

"Collaboration Level" - a message for the human operator to notify about the collaboration level, autonomous or semi-autonomous.

"Policy Success Notification" - notifies if the last policy was successful or failed.

"Human-Robot Collaboration Control" - when human is asked to intervene this panel is enabled. The controls at this panel allows the human to adjust the X, Y, and Z axis speeds and adjacent state distances. Additionally, it allows to eliminate movement for a specific axis.

2) The "Development" tab (Fig. I.19):

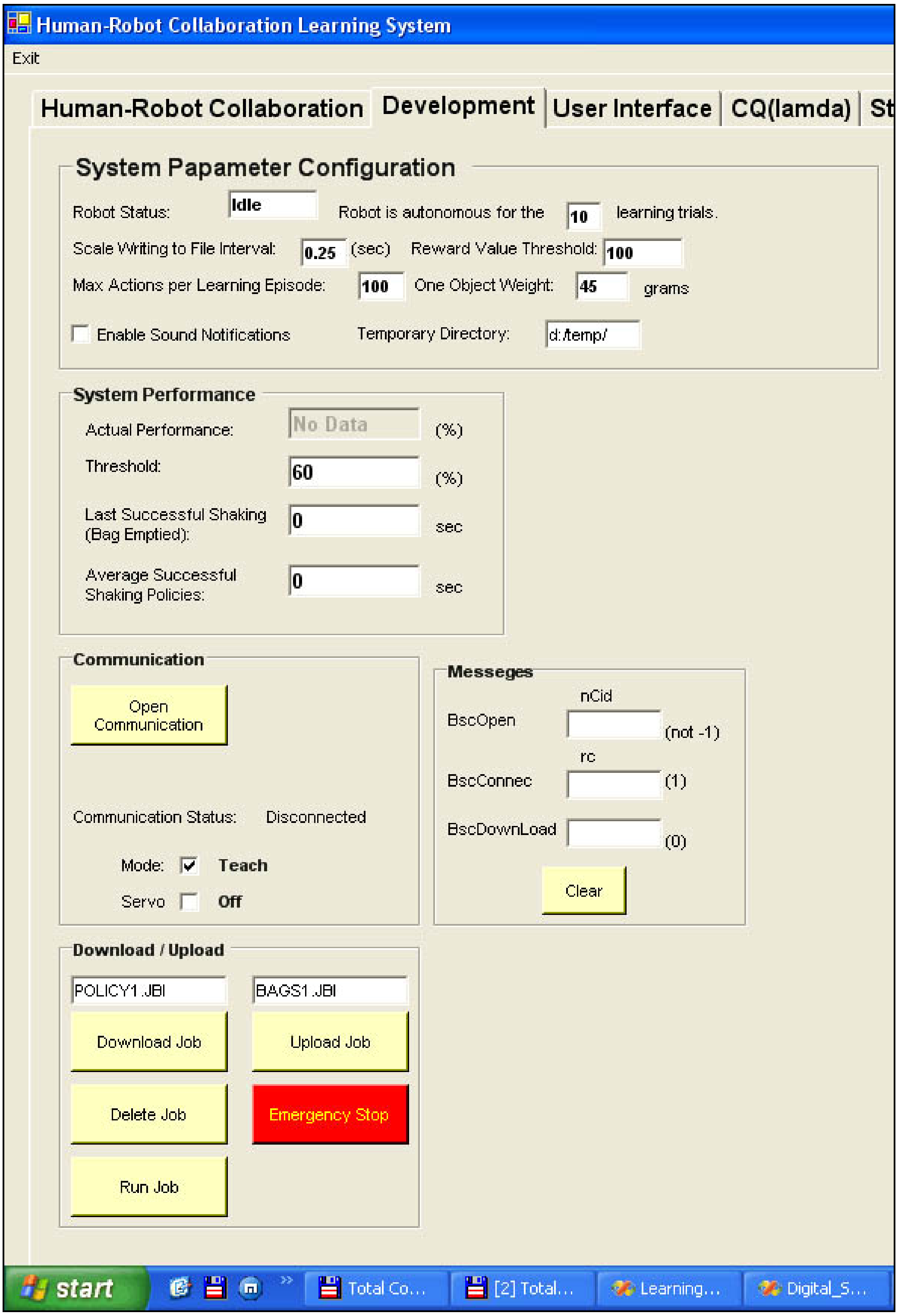


Fig. I.19 Development tab

<u>"System Parameter Configuration" panel</u>:

"Robot Status" - notifies whether the robot is operating or idle.

"Robot is autonomous for the … learning episodes" - determines for how many learning episodes the robot is enforced to learn autonomously.

"Scale Writing to File Interval" - sets the interval between reading values from the digital scale and writing the information to a log file.

"Max Actions per Learning Episode" - the maximal number of actions per learning episodes.

"Enable Sound Notifications" - enables/disables voice announcements.

"Reward Value Threshold" - threshold whether a policy was successful.

"One Object Weight" - the weight for one object.

"Temporary Directory" - a directory for writing log files.

"System Performance" panel:

"Actual Performance" - success rate of the system for the last five episodes.

"Threshold" - threshold for human intervention.

"Last Successful Shaking" - time to perform the last successful policy(s).

"Average Successful Shaking Policies" - average time of all successful policies(s).

<u>"Communication" panel</u> - controls XRC controller parameters and connects to robot. Presents the communication status.

<u>"Download/Upload" panel</u> - enables uploading, downloading, deleting and running robotic programs (JBI files).

<u>"Messages" panel</u> - parameter arrives from the XRC controller for notifying whether a communication operation was successful.

3) The “User Interface” tab (Fig. I.20):

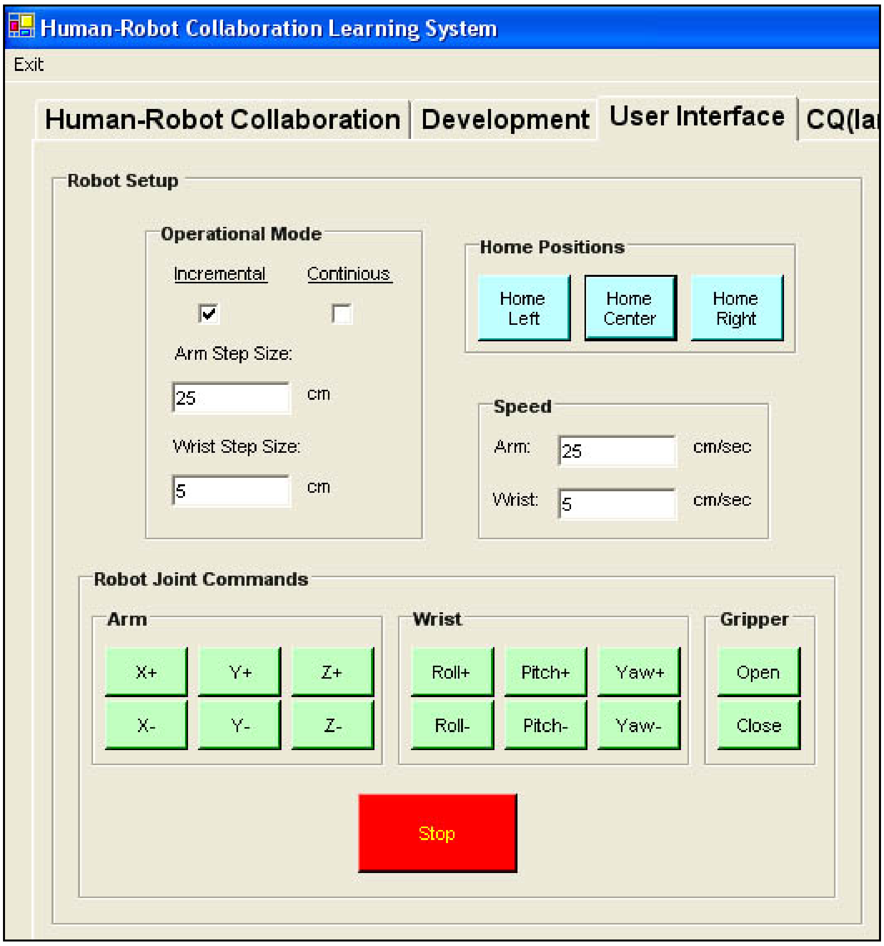


Fig. I.20 User interface tab

The “User Interface” tab allows supervisory control over the world coordinates robot and replaces some the teach pendant capabilities.

<u>“Operational Mode” panel</u> - enable to choose an incremental or continuous movement of the robot. If the incremental option is chosen then two parameters can be set: (i) the arm step movement size (cm), and the wrist step movement size (cm).

“<u>Speed” panel</u> - controlling the speeds of the arm and wrist (cm/s).

<u>“Home Positions” panel</u> - contains three predefined locations above the robotic environment, center (HL.JBI) , left (HL.JBI) and right (HR.JBI) positions.

<u>“Robot Joint Commands” panel</u> - enables the direct control of the robot using the parameters defined in the “Operational Mode” panel for the arm (*X*, *Y*, and *Z* axes), for the wrist (Roll, Pitch, and Yaw)

and for opening/closing the gripper. Pressing on one of the buttons creates a new "JBI" file which is downloaded to the XRC controller for execution. The "Stop" button disables any robot movements.

Additional tabs:

The tabs: "CQ(lambda)" and "State-Action Space" are not part of the user interface. The options presented at the tabs are used to view and analyze shaking policies the learning system creates or to create random shaking policies independently. The information presented at the tabs includes also the state-action space Q values and reward results. The "Shaking Editor" and "Job Editor" tabs might be used by other M.Sc./Ph.D. students if further system development is required.

Performing learning episodes

The learning system is preprogrammed to run fifty learning episodes (fifty trials of grasping, lifting and shaking a bag).[1]

For running a learning episode, the following procedure should be followed:

1) XRC controller should be turned on (Fig. I.21):

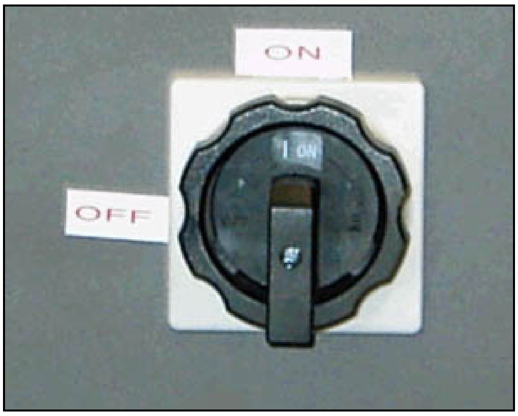


Fig. I.21 XRC controller on/off switch

2) Since the robot and its environment are surrounded by a security cage the doors should be closed (Fig. I.22). On both doors relay switches are installed (Fig. I.23). Opening one of the cage doors triggers a command sent to the controller to immediately disconnect the robot and stop any motion.[2]

[1] Note that the number of learning episodes required to determine the robot history learning performance was set to five. This value was determined programmatically and can not be changed through the interface.

[2] Before entering the robot's cage it is recommended to do as following: (i) press the "Reset" button", (ii) press the reset red button at the TP, (iii) turn the "Remote" switch left at the XRC controller remote control box, and (iv) disable the

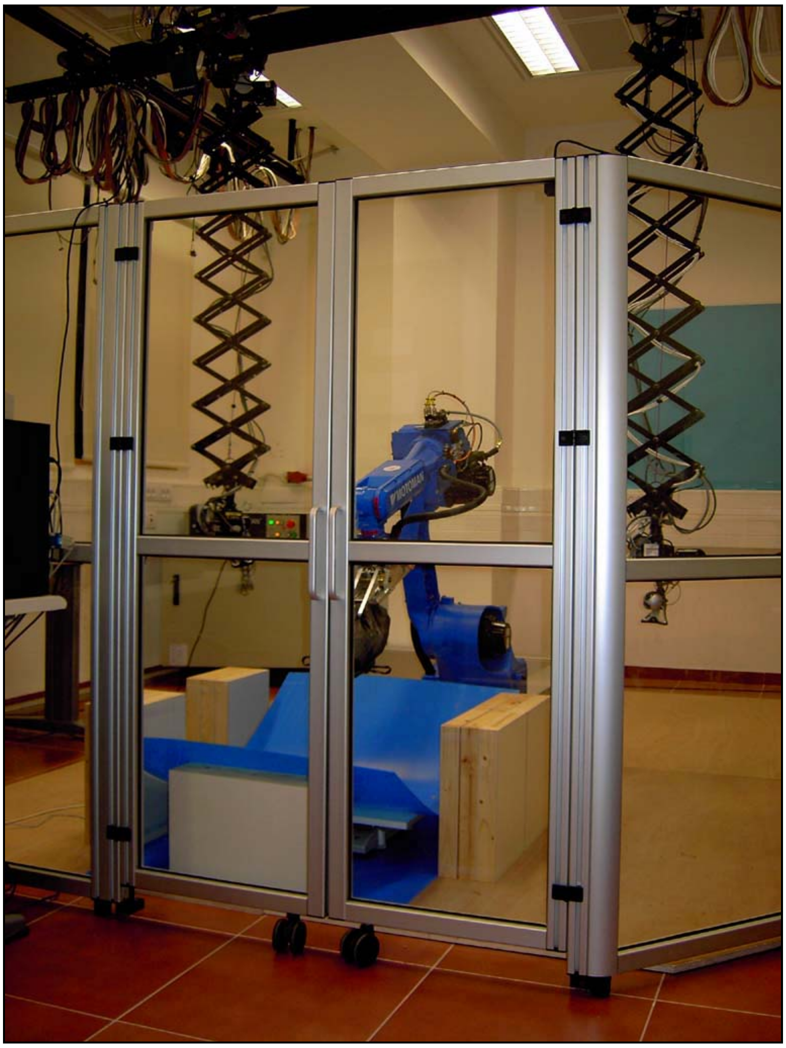
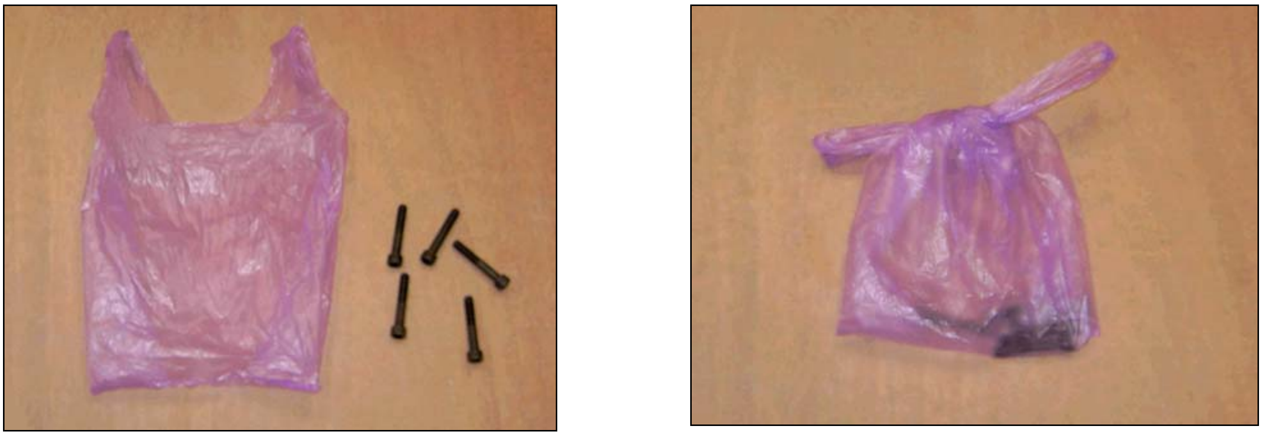


Fig. I.22 Robot surrounded by a security cage and doors

Fig. I.23 A relay installed on one of the cage doors

digital scale. When running a learning episode, release the resent button and press "Select" at the TP. Then turn the "Remote" button right and enable the digital scale.

3) Run the “Digital_Scale.vbproj” project and make sure that the screen in Fig. I.8 is shown.
4) Run the “Learning_System.vbproj” project and make sure that the screen in Fig. I.18 is shown.
5) Take a plastic bag and place in identical objects (for the experiments described in the work five screws each weights 45 grams were used) (Fig. I.24).



(a) Plastic bag and objects **(b) Plastic bag contains objects**

Fig. I.24 Suspicious plastic bag

6) Place the plastic bag with objects on the inspection surface (Fig. I.25).

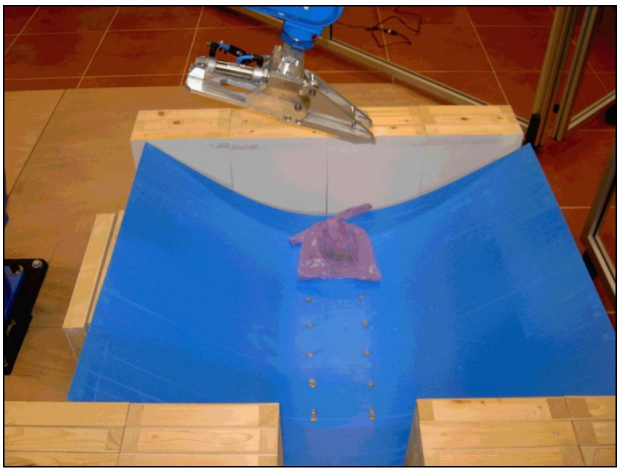


Fig. I.25 Plastic bag placed on the inspection surface

7) Close the cage’s doors.
8) Press the “SELECT” button on the teach pendant (Fig. I.26):

Fig. I.26 Teach pendant "SELECT" button

9) Turn the "REMOTE" switch left and then right (Fig. I.13).

10) In the "Development" tab (Fig. I.19) choose the parameters you desire.

11) In the "Human-Robot Collaboration" (Fig. I.18) tab:

   a. Press "Connect to Robot". Currently you should hear the sound of a relay triggered from the XRC controller. You will also be able to see that the "PLAY" and the "REMOTE" LEDs are turned on at the XRC controller. If the robot is not connected immediately then press the "Reset" and the "Connect to Robot" again.

   b. Make sure that at the temporary directory you chose (*e.g.*, d:/temp), the only file exists is "peakdetect.m". Please note that the "peakdetect.m" file is only required if events-based rewards are used. "peakdetect.m" subtracts the supplied signal vector from a one sample positively lagged version of the same signal and checks the resulting differenced signal for the sign changes where peaks occur.

   c. Press the "Initialize System" button. This will open a Matlab window and will build an initial and random shaking policy. The output will be written as a "0_Trial.csv" file to the temporary directory.

   d. Press the "Grasp a Bag" button. The robot will slide under the plastic bag, grasp it with it gripper and lift it to a starting shaking position above the inspection surface.

   e. Make sure that the inspection surface is free of bags/objects (Fig. I.27):

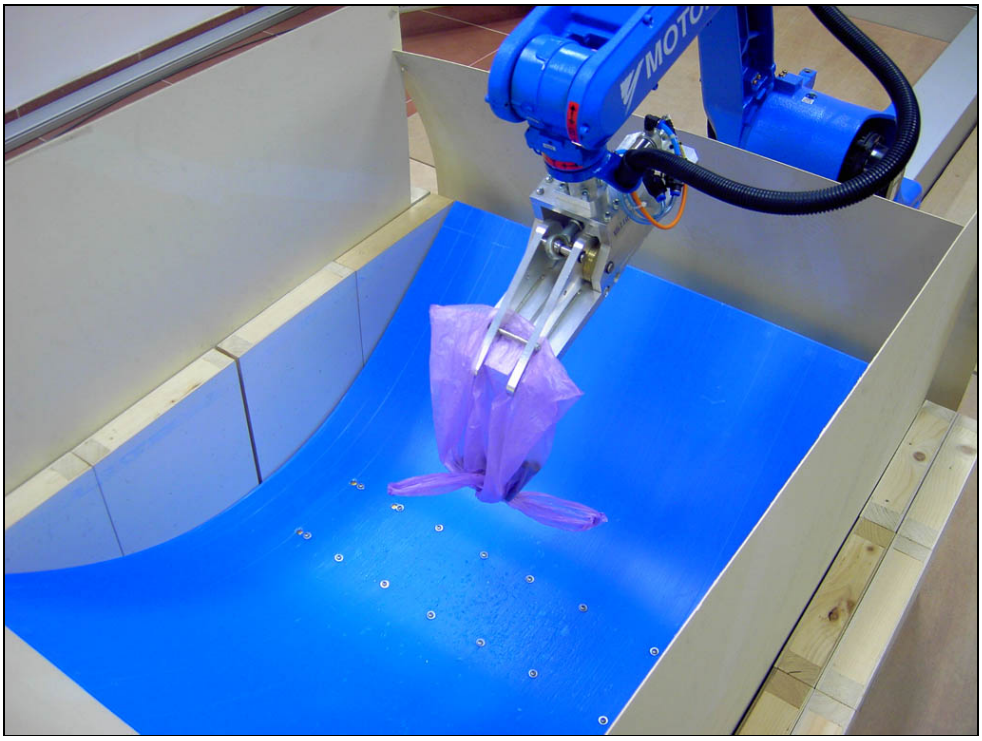


Fig. I.27 Robot center shaking position

12) Calibrate the digital scale:

a. Make sure that the inspection surface (Fig. I.27) is empty, then turn the digital scale remote switch to "Disable" and to "Enable" (Fig. I.28). Make sure that the scale measurement is around zero.

Fig. I.28 Digital scale remote switch

b. Press the "Execute Shaking" button. This will perform one robot shaking episode and in parallel will start the digital scale software for measuring the current weight on the scale and to notify on dropping events. The scale software will write the file "1_Trial_Scale_Output.csv" to the temporary directory (1 is for the first trial, 2 is for the second, and so on till the 50^{th} learning episode). The "1_Trial_Scale_Output.csv" contains three columns: (i) time (s); (ii) cumulative weight, and (iii) events data.

c. Press the "Calculate Reward" button. This will calculate the reward based on the chosen reward function (cumulative-based or events-based), calculate and update a new Q table.

d. From here, there are two possible options: (i) if the system is in an autonomous mode then press the "System Created Policy" button, and (ii) if the system in a semi-autonomous mode then the human operator can configure parameters at the "Human-Robot Collaboration Control" panel. Any of these two options create a new policy for the robot. Additionally, the file "1_Trial.csv" will be written to the temporary directory (1 is for the first trial, 2 is for the second, and so on till 50^{th} learning episode). The "1_Trial.csv" contains 10 columns (state, action, reward, time of shaking, three speeds and three adjacent state distances for the three axes). Note that initial speeds and adjacent state distances for the three axes were set to 1000 mm/s and 30 mm, respectively. Please also note that if a human has intervened then output files will be at the form of "n_Trial_Intervention.csv" while n is the current learning episode.

To perform more learning episodes make sure that the robot is connected (use the "Connect to Robot" button) and press the "Drop Bag" button. Then repeat all steps described above.

<u>**Remarks**</u>:

If from some reason the timer starts counting but the robot does not start a learning episode (that's probably a communication issue) then press the "Stop" and "Reset Timer" buttons one after the other (Fig. I.29). Then press the "Execute Shaking" button again. If this does not solve the problem then press "Reset" and "Connect to Robot".

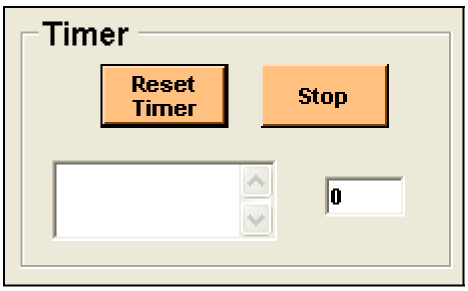


Fig. I.29 Learning episode timer

Appendix II. Convergence Proof for a Single Q-Learner

The proof is based on the observation that the Q-learning algorithm can be viewed as a stochastic process to which techniques of stochastic approximation are applicable. [Jaakkola *et al.*, 1994]'s proof for essential lemmas and theorems is presented below.

Lemma 1 A random process

$$\omega_{n+1}(x) = (1 - \alpha_n(x))\omega_n(x) + \beta_n(x) r_n(x)$$

converges to zero with probability one if the following conditions are satisfied:

1) $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, and $\sum_n \beta_n^2(x) < \infty$ uniformly over x with probability one.
2) $E\{r_n(x) \mid P_n, \beta_n\} = 0$ and $E\{r_n^2(x) \mid P_n, \beta_n\} \le C$ with probability one, where,

$$P_n = \{\omega_n, \omega_{n-1}, \ldots, r_{n-1}, r_{n-2}, \ldots, \alpha_{n-1}, \alpha_{n-2}, \ldots, \beta_{n-1}, \beta_{n-2}, \ldots\}$$

All the random variables are allowed to depend on the past P_n. $\alpha_n(x)$ and $\beta_n(x)$ are assumed to be non-negative and mutually independent given P_n.

Proof. Except for the appearance of $\beta_n(x)$ this is a standard result. It is stated in [Jaakkola *et al.*, 1994] "with the above definitions convergence follows directly from [Dvoretzky, 1956]'s extended theorem."

Lemma 2 Consider a stochastic iteration

$$X_{n+1}(x) = G_n(X_n, Y_n, x)$$

where G_n is a sequence of functions and Y_n is a random process. Let (Ω, F, P) be a probability space and assume that the process is scale invariant, that is, with probability one for all $\omega \in \Omega$.

$$G(\beta X_n, Y_n(\omega), x) = \beta G(X_n, Y_n(\omega), x)$$

Assume further that if we kept $\| X_n \|$ bounded by scaling, then X_n would converge to zero with probability one. These assumptions are sufficient to guarantee that the original process converges to zero with probability one.

Proof. Note that multiplying X_n by β corresponds to having initialized the process with βX_0. Now fix some constant C. If during the iteration, $\| X_n \|$ increases above C, then X_n is scaled so that $\| X_n \| = C$. By the second assumption then this process must converge with probability one. To show that the net effect of the corrections must stay finite with probability one we note that if $\| X_n \|$ converges then for any $\varepsilon > 0$ there exists M_ε such that $\| X_n \| < \varepsilon < C$ for all $n > M_\varepsilon$ with probability at least $1-\varepsilon$. But this implies that the iteration stays below C after M_ε and converges to zero without any further corrections.

Lemma 3. A stochastic process $X_{n+1}(x) = (1-\alpha_n(x))X_n(x) + \gamma\beta_n(x)\| X_n \|$ converges to zero with probability one provided

1) $x \in S$, where S is a finite set.
2) $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, and $E\{\beta_n(x) \mid P_n\} \le E\{\alpha_n(x) \mid P_n\}$ uniformly over x with probability one. Where

$$P_n = \{X_n, X_{n-1}, \ldots, \alpha_{n-1}, \alpha_{n-2}, \ldots \beta_{n-1}, \beta_{n-2}, \ldots\}$$

$\alpha_n(x)$ and $\beta_n(x)$ are assumed to be non-negative and mutually independent given P_n.

Proof. Essentially the proof is an application of Lemma 2. To this end, assume that we keep $\| X_n \| \le C_1$ by scaling which allows the iterative process to be bounded by

$$| X_{n+1}(x) | \le (1-\alpha_n(x)) | X_n(x) | + \gamma\beta_n(x) C_1$$

This is a linear in $| X_n(x) |$ and can be easily shown to converge with probability one to some $X^*(x)$, where $\| X^* \| \le \gamma C_1$. Hence, for small enough ε, there exits $M_1(\varepsilon)$ such that $\| X_n \| \le C_1/(1+\varepsilon)$ for all $n > M_1(\varepsilon)$ with probability at least $p_1(\varepsilon)$. With probability $p_1(\varepsilon)$ the procedure can be repeated for $C_2 = C_1/(1+\varepsilon)$. Continuing in the manner and choosing $p_k(\varepsilon)$ so that $\prod_k p_k(\varepsilon)$ goes to one as $\varepsilon \to 0$ we obtain with probability one convergence of the bounded iteration and Lemma 2 can be applied.

Theorem 1 A random iterative process $\Delta_{n+1}(x) = (1-\alpha_n(x))\Delta_n(x) + \beta_n(x)F_n(x)$ converges to zero with probability one under the following assumptions:

1) $x \in S$, where S is a finite set.
2) $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, and $E\{\beta_n(x) \mid P_n\} \le E\{\alpha_n(x) \mid P_n\}$ uniformly over x with probability one.
3) $\| E\{F_n(x) \mid P_n, \beta_n\} \|_W \le \gamma \| \Delta_n \|_W$, where $\gamma \in (0,1)$
4) $Var\{F_n(x) \mid P_n, \beta_n\} \le C(1 + \| \Delta_n \|_W)^2$, where C is some constant.

Here $P_n = \{X_n, X_{n-1}, \ldots, F_{n-1}, \ldots, \alpha_{n-1}, \ldots, \beta_{n-1}, \ldots\}$ stands for the past at step n. $F_n(x)$, $\alpha_n(x)$ and $\beta_n(x)$ are allowed to depend on the past. $\alpha_n(x)$ and $\beta_n(x)$ are assumed to be non-negative and mutually independent given P_n. The notation $\|\cdot\|_W$ refers to some weighted maximum norm.

Proof. By defining $r_n(x) = F_n(x) - E\{F_n(x) \mid P_n, \beta_n\}$ we can decompose the iterative process into two parallel processes given by

$$\delta_{n+1}(x) = (1-\alpha_n(x))\delta_n(x) + \beta_n(x)E\{F_n(x) \mid P_n, \beta_n\}$$
$$\varpi_{n+1}(x) = (1-\alpha_n(x))\varpi_n(x) + \beta_n(x)r_n(x)$$

where $\Delta_n(x) = \delta_n(x) + \varpi_n(x)$. Dividing the equations by $W(x)$ for each x and denoting $\delta'_n(x) = \delta_n(x)/W(x)$, $\varpi'_n(x) = \varpi_n(x)/W(x)$ and $r'_n(x) = r_n(x)/W(x)$ we can bound the $\delta'_n(x)$ process by assumption 3 and rewrite the equation pair as

$$\left|\delta'_{n+1}(x)\right| \le (1-\alpha_n(x))\left|\delta'_n(x)\right| + \gamma\beta_n(x)\left\| \left|\delta'\right| + \varpi'_n \right\|$$
$$\varpi'_{n+1}(x) = (1-\alpha_n(x))\varpi'_n(x) + \gamma\beta_n(x)r'_n(x)$$

Assume for a moment that the Δ_n process stays bounded. Then the variance of $r'_n(x)$ is bounded by some constant C and thereby ϖ'_n converges to zero with probability one according to Lemma 1. Hence, there exists M such that for all $n > M \left\| w'_n \right\| < \varepsilon$ with probability at least $1-\varepsilon$. This implies that the δ'_n process can be further bounded by

$$\left|\delta'_{n+1}(x)\right| \le 1-\alpha_n(x))\left|\delta'_n(x)\right| + \gamma\beta_n(x)\left\|\delta'_n(x) + \varepsilon\right\|$$

with probability $> 1-\varepsilon$. If we choose C such that $\gamma(C+1)/C < 1$ then for $\|\delta_n'\| > C\varepsilon$

$$\gamma\|\delta_n' + \varepsilon\| \le \gamma(C+1)/C\|\delta_n'\|$$

and the process defined by this upper bound converges to zero with probability one by Lemma 3. Thus $\|\delta_n'\|$ converges with probability one to some value bounded by $C\varepsilon$ which guarantees the with probability one convergence of the original process under the roundedness assumption.

By assumption (4) $r_n'(x)$ can be written as $(1+\|\delta_n + \varpi_n\|)s_n(x)$, where $E\{s_n^2(x) \mid P_n\} \le C$. Let us now decompose ϖ_n as $u_n + v_n$ with

$$u_{n+1}(x) = (1-\alpha_n(x))u_n(x) + \gamma\beta_n(x)\|\delta_n' + u_n + v_n\|s_n(x)$$

and v_n converges to zero with probability one by Lemma 1. Again by choosing C such that $\gamma(C+1)/C < 1$ we can bound the δ_n' and u_n processes for $\|\delta_n' + u_n\| > C\varepsilon$.

The pair (δ_n', u_n) is then a scale invariant process whose bounded version was proven earlier to converge to zero with probability one and therefore by Lemma 2 it too converges to zero with probability one. This proves with probability one convergence of the triple δ_n', u_n, and v_n bounding the original process $\gamma(C+1)/C < 1$.

Theorem 2 The Q-learning algorithm given by

$$Q_{t+1}(s_t,u_t) = (1-\alpha_t(s_t,u_t))Q_t(s_t,u_t) + \alpha_t(s_t,u_t)[c_{s_t}(u_t) + \gamma V_t(s_{t+1})]$$

converges to the optimal $Q^*(s,u)$ values if

1) The state and action spaces are finite.
2) $\sum_t \alpha_t(s,u) = \infty$ and $\sum_t \alpha_t^2(s,u) < \infty$ uniformly over s and u with probability one.
3) $Var\{c_s(u)\}$ is finite.
4) If $\gamma = 1$ all policies lead to a cost free terminal state with probability one.

Proof. By subtracting $Q^*(s,u)$ from both sides of the learning rule and by defining $\Delta_t(s,u) = Q_t(s,u) - Q^*(s,u)$ together with

$$F_t(s,u) = c_s(u) + \gamma V_t(s_{t+1}) - Q^*(s,u)$$

the Q-learning algorithm can be seen to have the form of the process in Theorem 1 with $\beta_t(s,u) = \alpha_t(s,u)$.

To verify that $F_t(s,u)$ has the required properties we begin by showing that it is a contraction mapping with respect to some maximum norm. This is done by relating F_t to the dynamic programming (DP) value iteration operator for the same Markov chain. More specifically,

$$\begin{aligned}
\max_u |E\{F_t(i,u)\}| &= \gamma \max_u |\sum_j p_{ij}(u)[V_t(j) - V^*(j)]| \\
&\le \gamma \max_u \sum_j p_{ij}(u) \max_v |Q_t(j,v) - Q^*(j,v)| \\
&= \gamma \max_u \sum_j p_{ij}(u) V^{\Delta}(j) = T\left(V^{\Delta}\right)(i)
\end{aligned}$$

where we have used the notation $V^{\Delta}(j) = \max_v \left|Q_t(j,v) - Q^*(j,v)\right|$ and T is the DP value iteration operator for the case where the costs associated with each state are zero:

$$V^{k+1}(i) = \min_{u \in U(i)} \{\overline{c_i}(u) + \gamma \sum_{j \in S} p_{ij}(u) V^{(k)}(j)\}$$

If $\gamma < 1$ the contraction property of $E\{F_t(i,u)\}$ can be seen from the fourth formula by bounding $\sum_j p_{ij}(u) V^{\Delta}(j)$ by $\max_j V^{\Delta}(j)$ and then including the γ factor. When the future costs are not discounted $(\gamma = 1)$ but the chain is absorbing and all policies lead to the terminal state with probability one there still exits a weighted maximum norm with respect to which T is contraction mapping [*e.g.*, Bertsekas and Tsitsikilis, 1989] thereby forcing the contraction of $E\{F_t(i,u)\}$. The variance of $F_t(s,u)$ given the past is within the bounds of Theorem 1 as it depends on $Q_t(s,u)$ as most linearly and the variance of $c_s(u)$ is bounded.

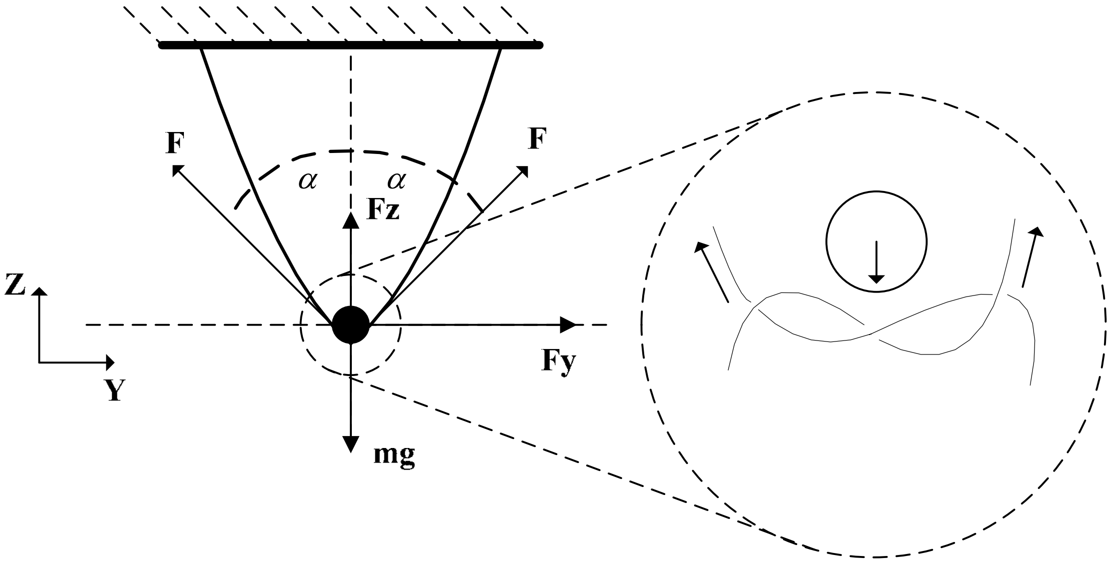
Appendix III. Physical Modeling of a Plastic Bag Knot

Objective

To find the directions of forces and moments required to open a plastic bag - which forces will contribute toward opening the knot and which forces will lock it further.

2D Analysis

Analysis includes the best movements for opening the knot while the bag contains only one rigid body (Fig. III.1).



F - Plastic bag tension

Fig. III.1 Two dimensional one object forces (applies for both for static and dynamic cases)

Static Case

For the static equilibrium case (Fig. III.1), it is assumed that the mass of the bag is negligible with respect to the weight of the object and to the friction forces in the knot. For opening the bag knot, the friction coefficient should be less that 0.5, as described below. The sum of the forces in the Z direction is:

$$2F\cos\alpha = mg \tag{III.1}$$

$$F = \frac{mg}{2\cos\alpha} \tag{III.2}$$

where F is the plastic bag tension. At the contact between the two bag handles F serves as the tangential friction force. Therefore for opening the bag knot through sliding the two handles on each other (III.3) shall hold.

$$F > \mu mg \tag{III.3}$$

or:

$$\frac{mg}{2\cos\alpha} > \mu mg \tag{III.4}$$

which yields to:

$$\mu < \frac{1}{2\cos\alpha} \leq 0.5 \tag{III.5}$$

for the limit on the friction coefficient. This means that when the bag is hanged down, a friction coefficient of at least 0.5 is required to keep it closed and a higher friction coefficient to keep it closed as the ends are pulled apart further.

Dynamic Case

For the dynamic case, when the robot shakes a bag with a knot aligned with the Y axis (Fig. III.1), (III.6) expresses Newton's law of the object along the Y direction:

$$\sum F_y = F_y = F\sin\alpha + ma_y \tag{III.6}$$

where F_y is a force activated on the bag by the object and a_y is the object acceleration. Now we write Newton's law in the Z direction:

$$\sum F_z = \max\left[mg - ma_z, 0\right] \tag{III.7}$$

where 0 means detachment of the object from the bag.

A condition that the knot will be open is (III.8):

$$\sum F_y > \mu \sum F_z \tag{III.8}$$

Based on (III.6) (III.7) and (III.8), (III.9) is organized:

$$F \sin\alpha + ma_y > \mu \max\left[mg - ma_z, 0\right] \tag{III.9}$$

where both a_y and a_z are controllable by commanding different bag accelerations.

For opening the bag knot, it is desired that the expression $F \sin\alpha + ma_y$ will be as large as possible and the expression $\mu \max\left[ma_z - mg, 0\right]$ will be as small as possible. Thereby, it is desired to increase both a_y and a_z. a_z is desired to be increased to $a_z = -g$, then the right hand side in (III.9) will be equal to zero. The reason for doing that is due to the vertical force that contributes toward increasing the normal load and thus increasing the friction against opening the bag. On the other hand, accelerating along the Y axis will increase the left hand side of (III.9) and therefore will act to open the bag. However, a conflict arises in case where a force is activated over the Y axis for a certain amount of time since the value $\sin\alpha$ is decreased over time. This explains why it is desired to change the side of activating forces over the Y axis of the bag, *i.e.*, to shake it. Another reason for shaking might be that maintaining acceleration in a constant direction results in ever increasing speeds which are not maintainable. In conclusion, it is desirable to shake the bag a little up and down over the Z axis to compensate over the gravitational forces that lock the bag (this decreases the friction between the bag and object) and shake it a lot over the Y axis to slide the bag handles and open the knot.

3D Analysis

Now we move to the case where the knot is no longer aligned with the Y axis, and it is not flat anymore, i.e., each handle a wide, triangular shaped, stripe stretched away from the knot center. The two dotted areas shown in Fig. III.2 are three dimensional sectors. Each sector is spanned by two forces, F_u and F_v derived from the robot shaking activity. F_L and F_R are the left and right tension vectors over the bag knot, respectively.

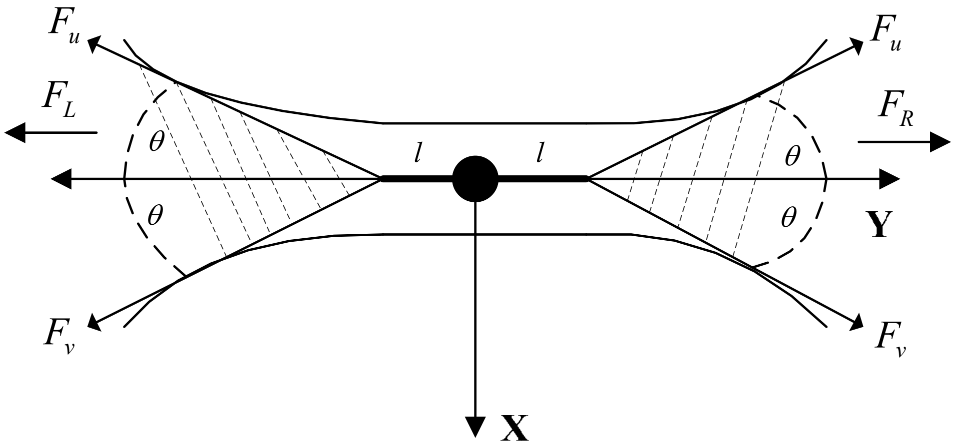


Fig. III.2 Top view of a 3D one object and part of the plastic bag bottom forces

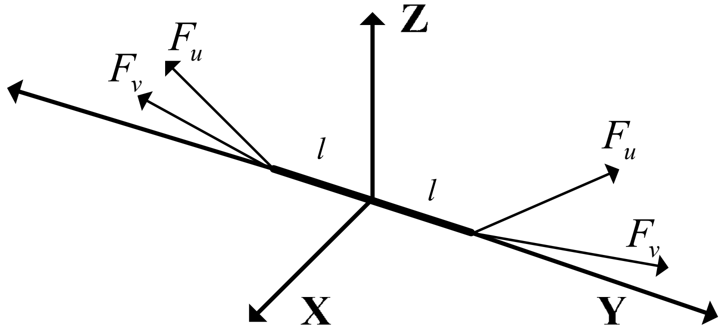


Fig. III.3 Three dimensional plastic bag axes

Based on Fig. III.2 and Fig. III.3, equations III.10 and III.11 are expressed:

$$\mathbb{F}_L = \left\{ F_L = \gamma_L F_{U_L} + \delta_L F_{V_L} : \quad \gamma_L, \delta_L \geq 0 \right\} \tag{III.10}$$

$$\mathbb{F}_R = \left\{ F_R = \gamma_R F_{U_R} + \delta_R F_{V_R} : \quad \gamma_R, \delta_R \geq 0 \right\} \tag{III.11}$$

where $\mathbb{F}_L$ and $\mathbb{F}_R$ are groups of all forces activated over to the left and to the right of the bag knot, respectively. $F_L \in \mathbb{F}_L$ and $F_R \in \mathbb{F}_R$ respectively, are the left and right possible force values activating in the spanning dotted area (Fig. III.2).

From (III.10) and (III.11), equation (III.12) is organized:

$$\vec{F}_L = F_{L_x}\hat{x} + F_{L_y}\hat{y} + F_{L_z}\hat{z} \quad and \ \vec{F}_L \in \mathbb{F}_L \tag{III.12}$$

where $\hat{x}$, $\hat{y}$, and $\hat{z}$ are unit vectors in the x, y, and z directions, respectively.

Similarly, for $\vec{F}_R$, equation (III.13) is expressed:

$$\vec{F}_R = F_{R_x}\hat{x} + F_{R_y}\hat{y} + F_{R_z}\hat{z} \quad and\ \vec{F}_R \in \mathbb{F}_R \tag{III.13}$$

Based on Newton's second law, (III.14) is written:

$$F_R + F_L + mg = ma \tag{III.14}$$

which yields to (III.15):

$$\begin{aligned} x: F_{L_x} + F_{R_x} &= ma_x \\ y: F_{L_y} + F_{R_y} &= ma_y \\ z: F_{L_z} + F_{R_z} - mg &= ma_z \end{aligned} \tag{III.15}$$

where a_x, a_y, and a_z are the accelerations at the X, Y, and Z axes, g is the gravitational acceleration and mg is the force that the object activates on the bag.[1]

Under the constraint for opening the bag:

$$\max\left[\sqrt{F_{R_x}^2 + F_{R_y}^2}, \sqrt{F_{L_x}^2 + F_{L_y}^2}\right] > \mu(mg + ma_z) \tag{III.16}$$

where $\sqrt{F_{R_x}^2 + F_{R_y}^2}$ and $\sqrt{F_{L_x}^2 + F_{L_y}^2}$ are tangential forces, *i.e.*, components parallel to the bag surface and $\mu(mg + ma_z)$ is the normal force, *i.e.*, a component perpendicular to the surface of the bag.

Rearranging (III.16), yields:

$$\max\left[F_{R_x}^2 + F_{R_y}^2, F_{L_x}^2 + F_{L_y}^2\right] > \mu^2 m^2 (g + a_z)^2 \tag{III.17}$$

[1] It should be noted that Newton's second law as expressed in (III.14) and (III.15) applies on the object inside the bag, however based on Newton's third law, equal forces (but with opposite reaction) apply on the bag itself.

Using (III.15) and (III.17), yields:

$$\max\left[F_{R_x}^2 + F_{R_y}^2, (ma_x - F_{R_x})^2 + (ma_y - F_{R_y})^2\right] > \mu^2 m^2 (g + a_z)^2 \tag{III.18}$$

It is required to accelerate the bag at a direction that is opposite to the direction of the force in order to maximize the left hand side of (III.18). Further, the right hand side of (III.18) is required to be minimal as possible and this can be achieved by accelerating the bag downwards. However, accelerating the bag downwards at a high value might cause the object in the bag to collide with the robot's gripper, thereby the value of this acceleration should be bounded by $g-\varepsilon$ where $\varepsilon \to 0$. Further, due to the unique structure of the plastic bag, l is not a negligible length and θ is small (Fig. III.2). If the length of l was close to 0 then there was no preference to activate any force exists in the sector that is bounded by F_u and F_v. Since l is significantly larger than 0 then at the section ($+l$, $-l$) around the bag knot center, it is preferable to activate forces over the Y axis.

<u>Conclusion</u>

Ideally, it is desirable to accelerate the robot arm at $a_z = g - \varepsilon$ downwards and to oscillate it over the Y axis; this to overcome of most friction forces. Since acceleration is developed over time, it is worthwhile to open the bag by activating forces continuously while holding locations as far as possible from the center of the horizontal axis. By applying $CQ(\lambda)$-learning the robot converged to the same policy that was derived from the model. This result suggests that both the model and the learning process are valid since they independently converged to the same optimal solution.

Appendix IV. Bag Classification using Support Vector Machines

This section describes the design of multi-category support vector machines (SVMs) for classification of bags. Although the focus of this thesis assumes one bag class, *i.e.* a plastic bag has been detected, a multi-category support vector classification of bags will be needed in a pre-selection phase for future research. The classification approach, image processing operations, kernel optimization and optimal feature selection experiments are described.

Introduction

SVMs belong to the class of maximum margin classifiers [Heisele *et al.*, 2003]. The goal of maximum margin classification in binary SVM classification [Vapnik, 1998] is to separate the two classes by a hyperplane such that the distance to the support vectors is maximized. SVMs perform pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed support vectors. The procedure starts with a training set of points $x_i \in \Re_n$, $1, 2, ..., N$ where each point x_i belongs to one of two classes identified by the label $y_i \in \{-1, 1\}$. This hyperplane is called the optimal separating hyperplane (OSH). The OSH has the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + b. \tag{IV.1}$$

The coefficients α_i and b in (IV.1) are the solutions of a quadratic programming problem [Vapnik, 1998]. A data point x is classified by the sign of the right side of (IV.1). For the dual category SVM classification, (IV.2) is used.

$$d(x) = \frac{\sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + b}{\| \sum_{i=1}^{N} \alpha_i y_i x_i \|}. \tag{IV.2}$$

The sign of d is the classification result for x, and $|d|$ is the distance from x to the hyperplane. Intuitively, the farther away a point is from the decision surface, *i.e.*, the larger $|d|$, the more reliable the classification result. The entire construction can be extended to the case of nonlinear separating surfaces. Each point x in the input space is mapped to a point $z = \Phi(x)$ of a higher dimensional space, called the feature space, where the data are separated by a hyperplane. The key property in this construction is that the mapping $\Phi(\cdot)$ is subject to the condition that the dot product of two points in the feature space $\Phi(x) \cdot \Phi(y)$ can be rewritten as a kernel function $K(x, y)$. The decision surface (IV.3) is presented.

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b, \tag{IV.3}$$

where, the coefficients α_i and b are solutions of a quadratic programming problem. Note, that $f(x)$ does not depend on the dimensionality of the feature space. Kernel functions commonly used for pattern recognition problems are polynomial of degree d (IV.4) and gaussian radial basis (IV.5).

$$K(x, y) = (1 + x \cdot y)^d, \tag{IV.4}$$

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \tag{IV.5}$$

The dual category SVM classification was extended to multi-category classification by [Bennett and Bredensteiner, 1999]. There are two basic strategies for solving q-class problems with SVMs. In the one-vs-all approach, q SVMs are trained. Each of the SVMs separates a single class from all remaining classes [Schölkopf *et al.*, 1995; Cortes and Vapnik, 1995]. In the pairwise approach (used in this work), $\frac{q(q-1)}{2}$ machines are trained. Each SVM separates a pair of classes. The pairwise classifiers are arranged in trees, where each tree node represents a SVM. The run-time complexity of the two strategies is similar: the one-vs-all and the pairwise approaches require the evaluation of q and $q-1$ SVMs, respectively. Results on person recognition indicate similar classification performance for the two strategies [Nakajima *et al.*, 2000]. The input to the SVMs is a set of features obtained from the bag image.

Task Definition

To perform the task of bag classification the SVMs method was used. To train and test the SVMs a collection of 120 images of different types of bags were used (backpacks, small shoulder bags, plastic flexible bags, and small briefcases). Tests were conducted to establish the best polynomial and Gaussian RBF (radial basis function) kernels. The task was to design a multi-category support vector machines (SVMs) for classification of bags. The heart of the support vector machine design is the kernel selection. Kernels project the data into a high dimensional feature space and thereby increase the computational power of the linear learning machines [Vapnik, 1998]. The kernels chosen for the bag classification problem are the most common ones used in support sector machines, *i.e.*, the polynomial and the Gaussian RBF kernels [Cristianini and Shawe-Taylor, 2003]. Since it is a multi-class classification problem the SVM model used is the pairwise approach. The kernels were tested using the K-fold cross validation procedure to assure reliability. The procedure was performed using three subsets on a training set that contained 80 bag images (20 for each of the four classes). For testing, forty bag images (ten for each class) were used. The SVMs procedure was implemented in the "OSU SVM MATLAB Toolbox" [Ma and Ahalt, 2003]. As it is well known that SVMs are sensitive to the number of features in pattern classification applications, the performance of the SVMs as a function of the number and type of features was also studied. The goal here, in feature selection is to obtain a smaller set of features that accurately represent the original set.

Image Processing and Object Features

Image processing starts with a 24-bit color image of the robotic scene that contains a bag located on a platform. Four different bag types are considered: backpacks, small shoulder bags, plastic flexible bags, and small briefcases. Image processing operations used are [Kartoun, 2003]: conversion to gray-scale, thresholding using Otsu's method [Otsu, 1979], removal of noise from the image due to optical lens distortion, adaptation to ambient and external lighting conditions, segmentation of the bag from the background, spatial erosion and dilation for closing holes at the bag binary images, etc. (Fig. IV.1). Using the MATLAB's Image Processing Toolbox [The MathWorks Inc., 1998], nine popular object shape features were extracted from each segmented bag image. The features were: A - Area, B - Bounding Box Ratio, C - Major Axis Length, D - Minor Axis Length, E - Eccentricity, F - Equivalent Diameter, G - Extent, H - Roundness, and I - Convex Perimeter.

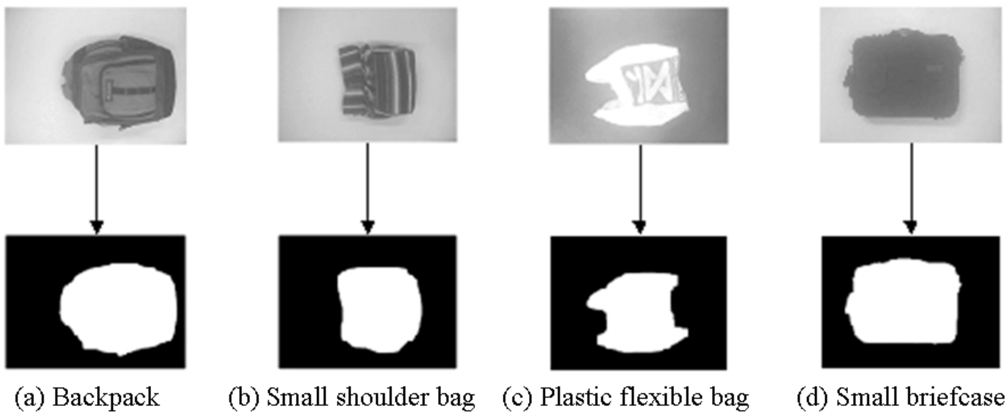


Fig. IV.1 Image processing operations for four bag classes

Experiments

Two experiments were performed. In the first one, a kernel optimization procedure was conducted, using nine bag features, for finding the optimal polynomial degrees and RBF sigmas. In the second experiment, an optimal feature selection was conducted for finding the subset of features and kernel optimization parameters (polynomial and RBF sigma) that results in the highest classification rate.

Kernel Optimization Experiment for Bag Classification

Bag classification was performed for both kernels by finding the optimal polynomial degrees (1-100) and RBF sigmas (1-100) using all nine features.

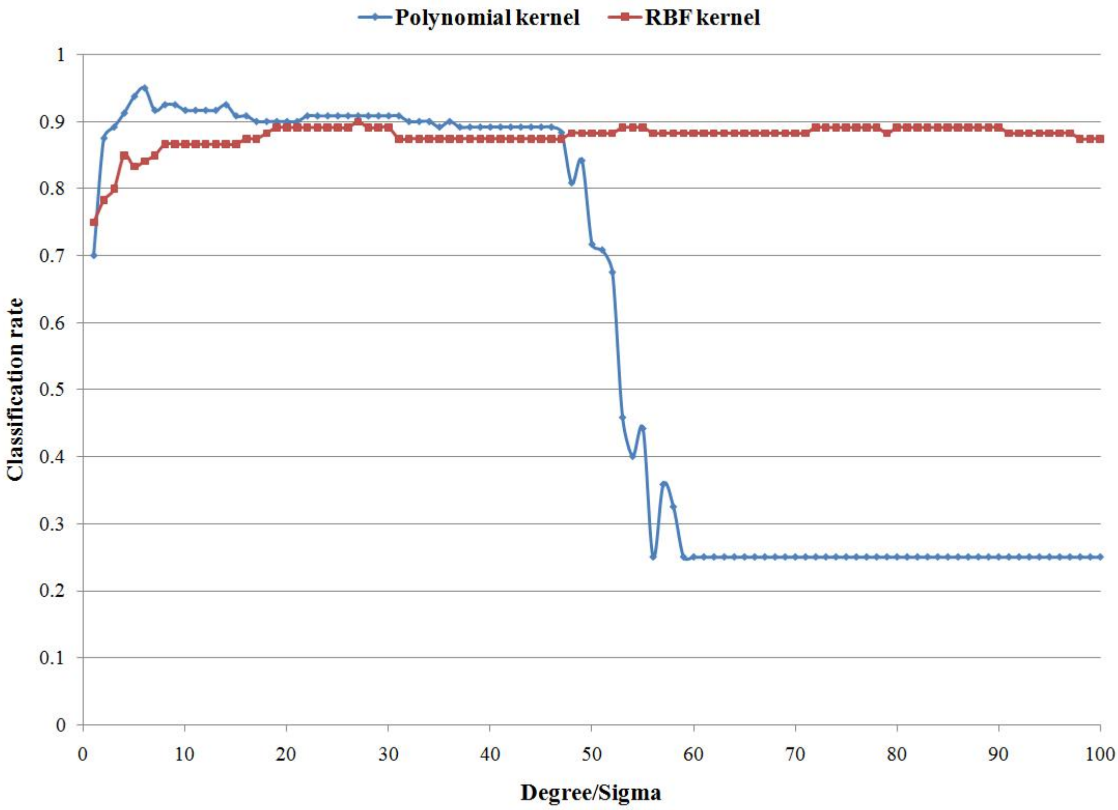


Fig. IV.2 Classification rate vs. degree and sigma

As can be seen in Fig. IV.2 for the nine bag features case, the highest average classification rate between the three subsets achieved was 95% using a polynomial kernel with six degree and 90% using a RBF kernel with 27 sigma. The confusion matrices that summarize the results are shown in Table IV.1, where the upper and lower diagonal values in each cell correspond to percent of correct classifications for the polynomial and RBF kernels, respectively.

Table IV.1 Confusion matrices for polynomial/RBF kernels

predicted class	true class classification rates [%]			
	backpack	**small shoulder bag**	**plastic flexible bag**	**small briefcase**
backpack	96.7% / 93.3%	0% / 0%	0% / 0%	3.3% / 6.7%
small shoulder bag	3.3% / 0%	93.4%/ 93.3%	0% / 0%	3.3% / 6.7%
plastic flexible bag	0% / 3.3%	0% / 3.3%	96.7% / 93.4%	3.3% / 0%
small briefcase	3.3% / 20%	3.3% / 0%	0% / 0%	93.4% / 80%

Optimal Feature Selection

A full enumeration feature selection procedure was performed to choose a set of optimal features to improve the results. Since there are up to nine features to be used in the classification procedure, there are $2^9 - 1 = 511$ combinations for selection. The classification process was performed for the range of 1-100 degrees/sigmas applying both kernels to find the optimal set of bag image features corresponding to the optimal polynomial degrees and RBF sigmas giving the highest classification results. As can be seen from Fig. IV.3, the largest average classification rate can be obtained by using only four features for both the polynomial and RBF kernels.

Details of the feature selections and the average classification rates appear in Table IV.2. The polynomial kernel's average classification rate of 96.25% was superior to that of the RBF (91.6%). It is noted that for the polynomial kernel, the minimum set of features (among the ties) consists of four features: bounding box ratio, major axis length, extent and roundness. Also, from Table IV.2 these features correspond to the optimal polynomial kernel of degree nine.

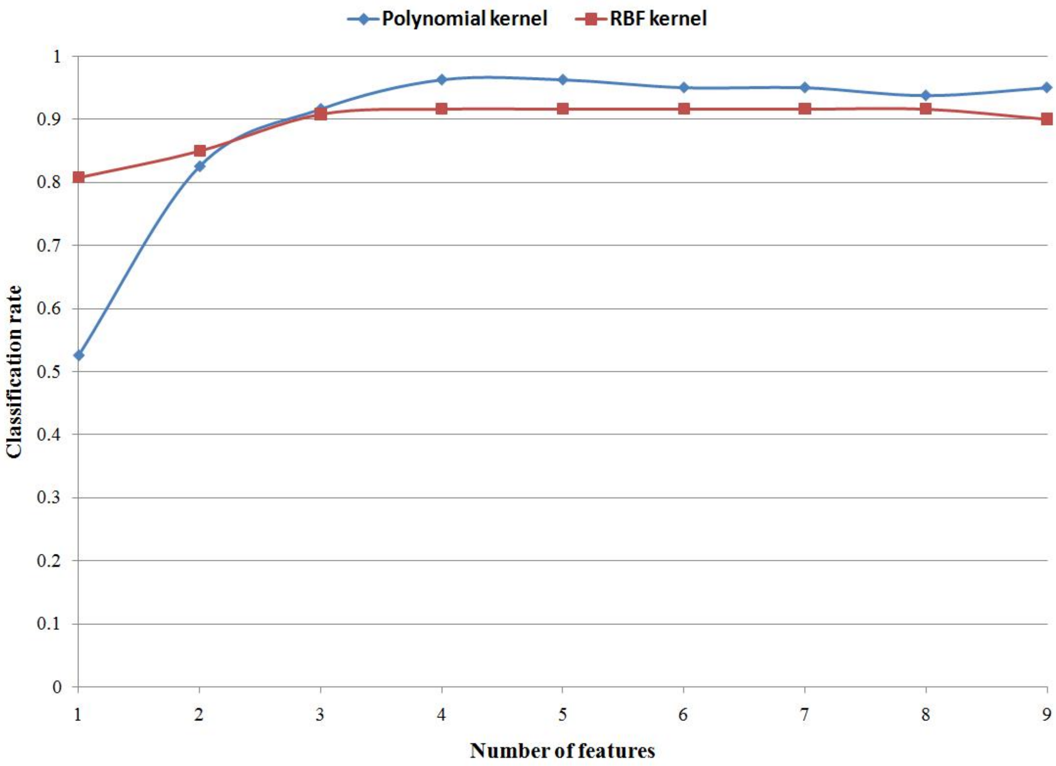


Fig. IV.3 Classification rate vs. number of features using the optimal degrees/sigmas

Table IV.2 Feature selection results for the highest classification rates achieved

kernel	number of features	1	2	3	4	5	6	7	8	9
polynomial	*features*	G	G, I	D, F, G	B, C, G, H	B, E, F, G, H	B, C, D, E, G, H	B, C, D, E, G, H, I	A, B, C, D, E, F, H, I	A, B, C, D, E, F, G, H, I
	degree	6	6	8	9	6	6	6	6	6
	classification rate (%)	52.5	82.5	91.6	96.25	96.25	95	95	93.75	95
RBF	*features*	F	A, G	D, G, H	B, D, E, G	B, D, E, G, H	B, C, D, E, G, H	B, C, D, E, G, H, I	A, C, D, E, F, G, H, I	A, B, C, D, E, F, G, H, I
	sigma	53	9	33	48	39	60	56	63	27
	classification rate (%)	80.8	85	90.8	91.6	91.6	91.6	91.6	91.6	90

* A Area, B Bounding Box Ratio, C Major Axis Length, D Minor Axis Length, E Eccentricity, F Equivalent Diameter, G Extent, H Roundness, I Convex Perimeter.

Another view of the results is shown in Fig. IV.4 using the optimal set of features (bounding box ratio, major axis length, extent and roundness). The curve shows the average classification rate peaks for an optimal polynomial kernel degree equals to nine.

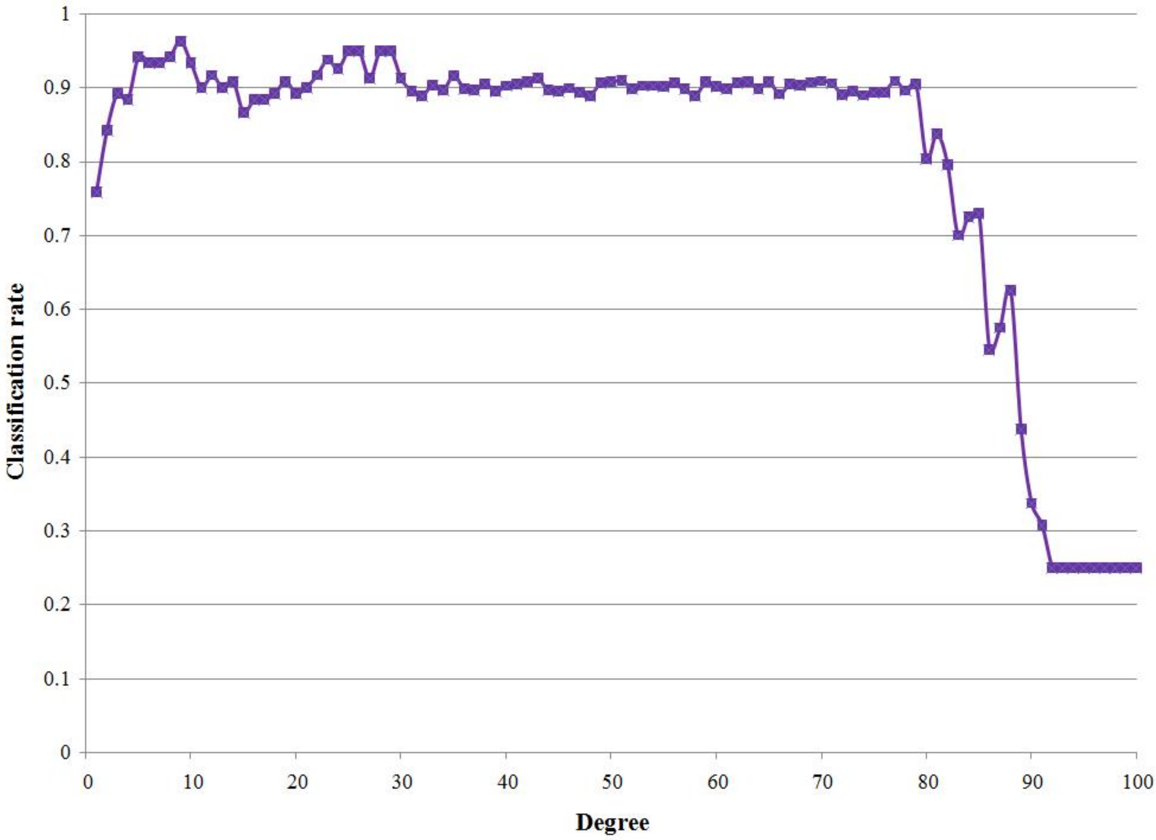


Fig. IV.4 Classification rate vs. degree for four-bag classification using optimal features

Results and Discussion

The polynomial kernel's classification rate of 96.25% was superior to that of the RBF (91.6%) using only four out of the nine original features. The small number of features deemed to be optimal was hypothesized to be due to correlation between the features. This is verified by examining the correlation matrix (Table IV.3), which exhibited correlation coefficients as high as 0.99.

Table IV.3 Correlation matrix

features	area	bounding box ratio	major axis length	minor axis length	eccentricity	equivalent diameter	extent	roundness	convex perimeter
area	1	0.08	0.96	0.92	0.28	0.99	-0.27	-0.26	0.85
bounding box ratio	0.08	1	0.28	-0.18	0.69	0.1	0.01	-0.74	0.48
major axis length	0.96	0.28	1	0.81	0.49	0.96	-0.3	-0.49	0.96
minor axis length	0.92	-0.18	0.81	1	-0.08	0.93	-0.31	0.11	0.61
eccentricity	0.28	0.69	0.49	-0.08	1	0.26	0	-0.99	0.69
equivalent diameter	0.99	0.1	0.96	0.93	0.26	1	-0.24	-0.25	0.86
extent	-0.27	0.01	-0.3	-0.31	0	-0.24	1	0.02	-0.2
roundness	-0.26	-0.74	-0.49	0.11	-0.99	-0.25	0.02	1	-0.7
convex perimeter	0.85	0.48	0.96	0.61	0.69	0.86	-0.2	-0.7	1

To further confirm the collinearity between the features a principle component analysis (PCA) was performed, where a reduction to four components accounted for 99.3% of the variation for each of the bag types. Eigenvalues of the covariance matrix and the cumulative percentage of the total variance in the observations explained by the eigenvectors are shown in Table IV.4.

Table IV.4 Eigenvalues and variance explained matrix

Variance explained [%]	62.65	93.46	96.9	99.31	99.79	99.93	99.97	100	100
Eigenvalue	0.0948	0.056	0.009	0.003	0.0007	0.0003	0.0001	0	0

It is noted, that the reduction from nine to four features was done by a complete enumeration feature selection method. It is a coincidence that a PCA allowed a reduction to four "components", but the PCA was not used to represent the feature reduction through a linear combination of the original features, as in the usual case.

Summary

Multi-category bag classification for four-bag classes was performed using SVMs. The SVMs procedure was tested using polynomial and RBF kernels. A K-fold cross validation procedure with three subsets was used. In a kernel optimization experiment using nine popular object shape features, classification rates of 95% and 90% were achieved using a polynomial kernel of degree six and a RBF kernel with 27 sigma, respectively. The confusion matrices indicate that decreased classification rates may be due to the inability to discriminate between the backpacks and the small briefcases bag images. To improve these results, a full enumeration feature selection procedure for choosing a set of optimal features for describing a bag image was performed. The set of optimal features found was: bounding box ratio, major axis length, extent and roundness. Using these features a classification rate of 96.25% was obtained for a polynomial kernel of degree nine. It was also found that using more than four features resulted in no improvement. The resulting optimal reduction of features from nine to four was hypothesized to be due to correlation between the features.

Appendix V. Bag Shaking Experiment: State-Action Space

ID	**State**	**Action**																	
Id	State	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	Center	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	X_Plus_1	1	1	**4**	1	1	**0**	1	1	1	1	1	1	1	1	1	1	1	1
2	X_Plus_2	2	2	2	**5**	2	**0**	2	2	2	2	2	2	2	2	2	2	2	2
3	X_Plus_3	3	3	3	3	**6**	**0**	3	3	3	3	3	3	3	3	3	3	3	3
4	X_Minus_1	**1**	4	4	4	4	**0**	4	4	4	4	4	4	4	4	4	4	4	4
5	X_Minus_2	5	**2**	5	5	5	**0**	5	5	5	5	5	5	5	5	5	5	5	5
6	X_Minus_3	6	6	**3**	6	6	**0**	6	6	6	6	6	6	6	6	6	6	6	6
7	Y_Plus_1	7	7	**10**	7	7	**0**	7	7	7	7	7	7	7	7	7	7	7	7
8	Y_Plus_2	8	8	8	**11**	8	**0**	8	8	8	8	8	8	8	8	8	8	8	8
9	Y_Plus_3	9	9	9	9	**12**	**0**	9	9	9	9	9	9	9	9	9	9	9	9
10	Y_Minus_1	10	10	**7**	10	10	**0**	10	10	10	10	10	10	10	10	10	10	10	10
11	Y_Minus_2	11	11	11	**8**	11	**0**	11	11	11	11	11	11	11	11	11	11	11	11
12	Y_Minus_3	12	12	12	12	**9**	**0**	12	12	12	12	12	12	12	12	12	12	12	12
13	Z_Plus_1	13	13	**16**	13	13	**0**	13	13	13	13	13	13	13	13	13	13	13	13
14	Z_Plus_2	14	14	14	**17**	14	**0**	14	14	14	14	14	14	14	14	14	14	14	14
15	Z_Plus_3	15	15	15	15	**18**	**0**	15	15	15	15	15	15	15	15	15	15	15	15
16	Z_Minus_1	16	16	**13**	16	16	**0**	16	16	16	16	16	16	16	16	16	16	16	16
17	Z_Minus_2	17	17	17	**14**	17	**0**	17	17	17	17	17	17	17	17	17	17	17	17
18	Z_Minus_3	18	18	18	18	**15**	**0**	18	18	18	18	18	18	18	18	18	18	18	18

Fig. V.1 Bag shaking experiment - state-action space

Appendix VI. Rewards Calculating Examples

Examples for reward calculations are shown below. Table VI.1 presents one learning episode that lasted 7.75 s where all five objects were extracted.

Table VI.1 A rewards numerical example I

	Time (s)	Cumulative Weight (grams)	Weight Change
	0.25	0	0
	0.5	0	0
	0.75	0	0
	1	0	0
	1.25	2.1	0
	1.5	2.1	4.3
	1.75	15.4	35.2
An event:**	**2**	**114**	**49.6**
	2.25	115	18
	2.5	114.5	9.2
	2.75	114	4.2
	3	113	0
	3.25	115	0
	3.5	114	0
	3.75	112	0
	4	115	0
	4.25	112	2
	4.5	110	11
An event*:	**4.75**	**165**	**20**
	5	165	10
	5.25	164	4
	5.5	160	0
	5.75	159	0
	6	162	4
	6.25	165	12
An event:**	**6.5**	**250**	**50.2**
	6.75	250	16
	7	250	6
	7.25	250	0
	7.5	250	0
	7.75	250	0

*** One object fell**
**** Two objects fell**

Fig. VI.1 and Fig. VI.2 present the cumulative weight and weight change, respectively.

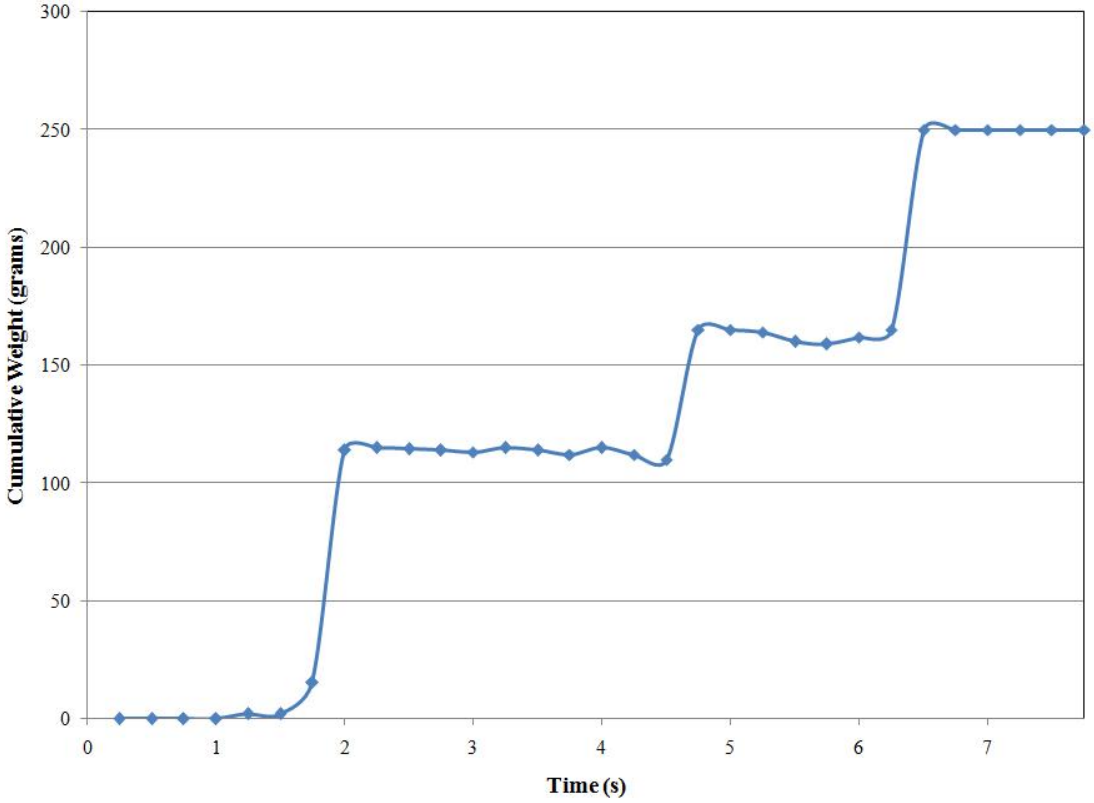


Fig. VI.1 Reward cumulative weight change example

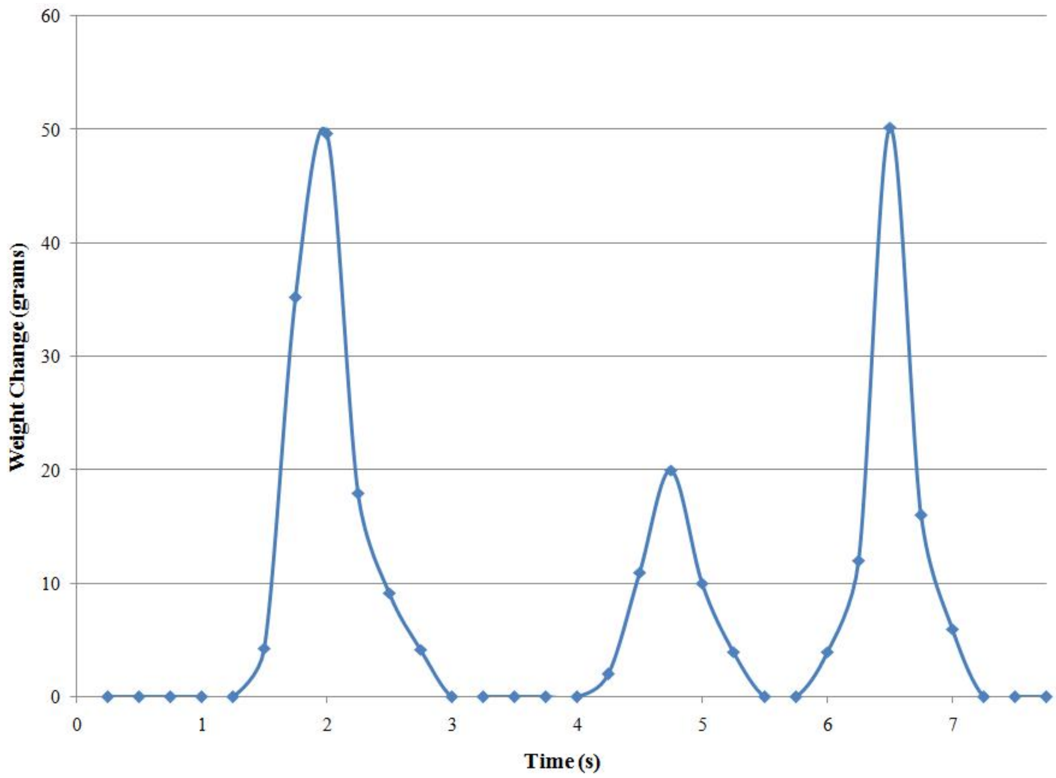


Fig. VI.2 Reward events weight change example

Analyzing raw scale data (Table VI.1), events-based and cumulative-based rewards are calculated as shown in (VI.1) and (VI.2), respectively (Section 3.4.2):[1]

$$R = \left(\frac{\frac{114}{48.5}}{2} \right) + \left(\frac{\frac{165-114}{48.5}}{4.75} \right) + \left(\frac{\frac{250-165}{48.5}}{6.5} \right) = 1.52 \tag{VI.1}$$

$$R = \left(\frac{0}{0.25} \right) + \left(\frac{0}{0.5} \right) + \left(\frac{0}{0.75} \right) + \left(\frac{0}{1} \right) + \left(\frac{2.1}{1.25} \right) + \left(\frac{2.1}{1.5} \right) + \left(\frac{15.4}{1.75} \right) + \ldots + \left(\frac{250}{7.25} \right) + \left(\frac{250}{7.5} \right) + \left(\frac{250}{7.75} \right) = 842.7 \tag{VI.2}$$

While the events-based reward function represents well the bag handling task, the cumulative function might pose a problem with the intended learning task. The data presented in Table VI.2 contains one learning episode that lasted 10 s where the first 7.75 s are identical to Table VI.1. Here, only three out of the five objects were extracted and the robot performed all 100 actions (the maximal number of actions for one learning episode).

[1] c=1, w = 48.5

Table VI.2 A rewards numerical example II

	Time (s)	Cumulative Weight (grams)	Weight Change
	0.25	0	0
	0.5	0	0
	0.75	0	0
	1	0	0
	1.25	2.1	0
	1.5	2.1	4.3
	1.75	15.4	35.2
An event:**	**2**	**114**	**49.6**
	2.25	115	18
	2.5	114.5	9.2
	2.75	114	4.2
	3	113	0
	3.25	115	0
	3.5	114	0
	3.75	112	0
	4	115	0
	4.25	112	2
	4.5	110	11
An event*:	**4.75**	**165**	**20**
	5	165	10
	5.25	164	4
	5.5	160	0
	5.75	159	0
	6	162	0
	6.25	165	0
	6.5	165	0
	6.75	165	0
	7	165	0
	7.25	165	0
	7.5	165	0
	7.75	165	0
	8	165	0
	8.25	165	0
	8.5	165	0
	8.75	165	0
	9	165	0
	9.25	165	0
	9.5	165	0
	9.75	165	0
	10	165	0

*** One object fell**
**** Two objects fell**

Here (VI.3), although the system did not succeed to extract all five objects, the reward is higher than as in (VI.2).

$$R = \left(\frac{0}{0.25}\right) + \left(\frac{0}{0.5}\right) + \left(\frac{0}{0.75}\right) + \left(\frac{0}{1}\right) + \left(\frac{2.1}{1.25}\right) + \left(\frac{2.1}{1.5}\right) + \left(\frac{15.4}{1.75}\right) + \ldots + \left(\frac{165}{9.5}\right) + \left(\frac{165}{9.75}\right) + \left(\frac{165}{10}\right) = 936.7 \quad \text{(VI.3)}$$

This limitation however, did not affect the performance of the bag handling task; in most of the learning episodes performed, all five objects were extracted at the same time, approximately. Only in rare incedences objects were extracted seperatly.

Appendix VII. Motoman UP-6 Manual Programming

1) Turn on the controller and close the cage.
2) There are two modes of operation: (i) teach - using the teach pendant (TP) of defining spatial locations and writing programs (jobs), (ii) play - running a job.
3) “Servo On Ready” led blinking - means that the operator will choose the play or the teach modes for controlling the robot’s engines, otherwise it is disabled.
4) For writing a job, choose “TEACH LOCK” at the TP (the green led at this button is constant).
5) By pressing the control hand button (CHB) (on the rear left of the TP) constantly power is supplied to the robot and the “Servo On Ready” led will be constant.
6) “COORD” button - the way of moving the robot (joint-based or world coordinates).
7) “TOP MENU” button - shows the main menu.
8) Button with three connected circles (to the right of the teach lock) - additional menus such as “CYCLE” and “SECURITY” that determines robot programming permissions (“Operation” - enables only to run jobs but not to write new ones, “Editing” - allows writing and running jobs, “Management” (password: 99999999) and “Yaskawa” (password: 32.12.02) - enable configuring parameters for the controller.
9) Controlling the robot manually - press constantly the CHB. Choose the coordinates method (“COORD”). Adjust the speed by using “FST” and “SLW”.
10) Writing a job - choose the “JOB” menu (use “SELECT”). Choose “CREATE NEW JOB”. Select “NEW JOB CREATE”. Choose a job name and press “ENTER”. Choose “EXEC”. Use the TP for moving the robot to desired spatial locations. Choose “INSERT” and press “ENTER” for each location. Adjust speeds (VJ) for each location. For opening/closing the gripper it is required to call either the “OPEN” or the “CLOSE” jobs. For calling a job, choose “INFORM LIST”, choose “CONTROL”, choose “CALL”, press “SELECT”, press “ENTER”.
11) For running a job, choose “MASTER JOB” from the “JOB” menu choose “SETTING MASTER JOB”. From the “Controller Remote Control Box” choose “PLAY” (Make sure that you are in a “Remote” mode - the led is on. Press “SERVO ON” (activates the robot’s engines). Press “START”. To stop - choose “TEACH”.

Appendix VIII. “Softmax” Action Selection Example

Given a state and four possible actions (Fig. VIII.1), actions are chosen with probabilities P_1, P_2, P_3, P_4 which are calculated according to (VIII.1).

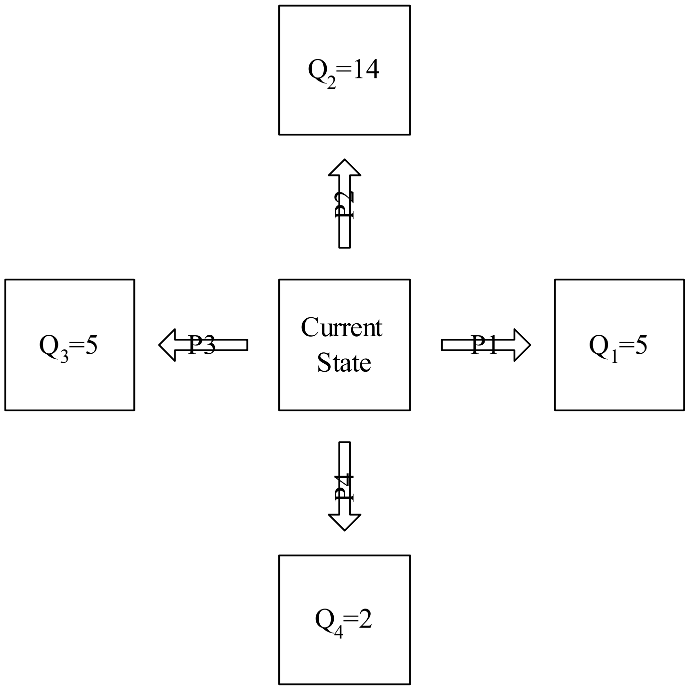


Fig. VIII.1 “Softmax” action selection example

$$P(a_t \mid s_t) = \frac{e^{Q(s_t, a_t)/T}}{\sum\limits_{a_{t+1} \in A} e^{Q(s_t, a_{t+1})/T}} \tag{VIII.1}$$

Probability calculations are presented in Table VIII.1.

Table VIII.1 “Softmax”-based probability calculations

Q	P	T = 10	T = 5	T = 0.1
5	P1	0.19	0.12	0
14	P2	0.47	0.70	1
5	P3	0.19	0.12	0
2	P4	0.14	0.06	0
	Sum:	**1**	**1**	**1**

It can be seen from Table VIII.1 that lower values of T gives higher probabilities to high Q values.

Appendix IX. Multiple Mobile Robot Navigation - Source Code

gwi.m

```
function out = gwi( arg, arg2)
% gwi: a grid world interface
close all
home
% World #1
global alpha beta gamma delta lambda
global gt ge gs bt be bs at ae as stop
global fx fy fm ff

if( nargin < 1)
        gwi( ’new’);
        return;
end

% call backs
switch( arg)
case ’new’
gw( ’new’);

% World #1
ff = figure(3); % focus
f3 = get(3,’Position’);
set(ff,’Position’,[600, 300,400,200],’Name’,’Command’);
clf;

% buttons
uicontrol(ff,’Style’,’pushbutton’,’Position’,[120,140,90,40],...
’Callback’,’gw try’,’String’,’Try’,’FontSize’,18);
uicontrol(ff,’Style’,’pushbutton’,’Position’,[220,140,90,40],...
’Callback’,’gw 10 try’,’String’,’Repeat’,’FontSize’,18);
end
```

gwf.m

```
function gwf( arg, arg2)
% World #1
global Rew_1 Val_1 Elig_1
global state_1 action_1 reward_1 value_1 delta_1
global nx_1 ny_1 na_1
global fx_1 fy_1 fm_1 ff_1
global ax_1 ay_1 az_1 asurf_1

% World #2
global Rew_2 Val_2 Elig_2
global state_2 action_2 reward_2 value_2 delta_2
global nx_2 ny_2 na_2
global fx_2 fy_2 fm_2 ff_2
global ax_2 ay_2 az_2 asurf_2

% World #3
global Rew_3 Val_3 Elig_3
global state_3 action_3 reward_3 value_3 delta_3
global nx_3 ny_3 na_3
global fx_3 fy_3 fm_3 ff_3
global ax_3 ay_3 az_3 asurf_3

switch( arg)
case ’init’
ss = get(0,’ScreenSize’);
fm_1 = [ 10, 20, 42, 3]; % figure margins
if nargin < 2
```

```
fx_1 = min( ss(3)-4, 1024)/2 - (fm_1(1)+fm_1(2));
fy_1 = (ss(4)-24)/2 - (fm_1(3)+fm_1(4));
else
fx_1 = arg2(1);
fy_1 = arg2(2);
end

ff_1 = 1; % figure to be focused: 4 for gwi
figure(1); set(1,'Position',[fm_1(1),ss(4)-fm_1(3)-fy_1,fx_1,fy_1],'Name','Reward'); clf;
figure(2); set(2,'Position',[fm_1(1),ss(4)-fm_1(3)*2-fm_1(4)-fy_1*2,fx_1,fy_1],'Name','Value'); clf;

% Robot
[ax_1,ay_1,az_1] = sphere(20);
ax_1 = 0.5*ax_1(11:end,:);
ay_1 = 0.5*ay_1(11:end,:);
az_1 = az_1(11:end,:)+0.01;

case 'reward'
        figure(1); clf;
        step( Rew_1'); caxis([-2,2]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -2 2]);
        axis( 'off');
case 'value'
        figure(2); clf;
        step( Val_1'); caxis([-50,50]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -10 10]);
        % title( sprintf('Value'));
case 'elig'
        figure(2); clf;
        step( Elig_1'); caxis([-50,50]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -10 10]);
        % title( sprintf('Eligibility'));
case 'agent'
        figure(1);
        if ishandle(asurf_1)
                set( asurf_1, 'xdata',arg2(1)+ax_1, 'ydata',arg2(2)+ay_1, 'zdata',arg2(3)+az_1);
        else
                hold( 'on');
                asurf_1 = surf( arg2(1)+ax_1, arg2(2)+ay_1, arg(3)+az_1, az_1*2);
        end
case 'traj'
        figure(1);
        line( arg2(:,1), arg2(:,2), 'Color', 'w');
end

function s = step( x, y, z)

% step: 3D plot of a step function

if nargin < 3
        z = x;
        [nx_1,ny_1] = size(z);
        x = 1:nx_1;
        y = 1:ny_1;
else
        [nx_1,ny_1] = size(z);

end

% double the data
x = [ x(:)-0.5, x(:)+0.5]'; x = x(:);
y = [ y(:)-0.5, y(:)+0.5]'; y = y(:);
z = reshape( [ z(:), z(:)]', 2*nx_1, ny_1);
```

```
z = reshape( [z;z], 2*nx_1, 2*ny_1);
s = surf( x, y, z);
```

gw.m

```
function out = gw( arg, arg2)

% World #1
global nx_1 ny_1 ns_1 na_1 move_1 loss_1 addaptive_alpha_1 visiting_counter_1
global start_1 goal_1 Rew_1 Val_1 Elig_1
global alpha_1 beta_1 gamma_1 delta_1 lambda_1
global state_1 action_1 value_1 reward_1 t_1 tmax_1 stop_1
global ff_1 Mov_1
global last_steps_1 trial_counter_1
global counter1_1;
global mean_Val_1 temp1_1 temp2_1

% World #2
global nx_2 ny_2 ns_2 na_2 move_2 loss_2 addaptive_alpha_2 visiting_counter_2
global start_2 goal_2 Rew_2 Val_2 Elig_2
global alpha_2 beta_2 gamma_2 delta_2 lambda_2
global state_2 action_2 value_2 reward_2 t_2 tmax_2 stop_2
global ff_2 Mov_2
global last_steps_2 trial_counter_2
global counter1_2;
global mean_Val_2 temp1_2 temp2_2

% World #3
global nx_3 ny_3 ns_3 na_3 move_3 loss_3 addaptive_alpha_3 visiting_counter_3
global start_3 goal_3 Rew_3 Val_3 Elig_3
global alpha_3 beta_3 gamma_3 delta_3 lambda_3
global state_3 action_3 value_3 reward_3 t_3 tmax_3 stop_3
global ff_3 Mov_3
global last_steps_3 trial_counter_3
global counter1_3;
global mean_Val_3 temp1_3 temp2_3

global enable_graphics
enable_graphics=0;

set(0,'RecursionLimit',20000);

global st_1 st_2 st_3 choosing_Val

global iterations stop_condition_threshold

iterations = 500;
stop_condition_threshold=70;
stop_condition_1=100;
stop_condition_2=100;
stop_condition_3=100;

st_3 = [1, 1];

switch( arg)

case 'new' % Setup
gw( 'world'); gw( 'agent'); gw( 'init');
case 'world' % A new world

% World #1
trial_counter_1=0;

nx_1 = 11; ny_1 = 11; ns_1 = nx_1*ny_1;
```

```
% actions
na_1 = 4; % R,U,L,D
move_1 = [1,0; 0,1; -1,0; 0,-1];
loss_1 = -0.1*[ 1; 1; 1; 1];

% reward field
Rew_1 = zeros( nx_1, ny_1);

Rew_1(5,1) = -1; Rew_1(5,2) = -1; Rew_1(5,3) = -1; Rew_1(5,4) = -1; Rew_1(5,11) = -1; Rew_1(5,10) = -1;
Rew_1(5,9) = -1; Rew_1(5,8) = -1; Rew_1(1,1) = -1; Rew_1(1,2) = -1; Rew_1(1,3) = -1; Rew_1(1,4) = -1; Rew_1(1,5) =
-1; Rew_1(1,6) = -1; Rew_1(1,7) = -1; Rew_1(1,8) = -1; Rew_1(1,9) = -1; Rew_1(1,10) = -1; Rew_1(1,11) = -1;
Rew_1(11,1) = -1; Rew_1(11,2) = -1; Rew_1(11,3) = -1; Rew_1(11,4) = -1; Rew_1(11,5) = -1; Rew_1(11,6) = -1;
Rew_1(11,7) = -1; Rew_1(11,8) = -1; Rew_1(11,9) = -1; Rew_1(11,10) = -1; Rew_1(11,11) = -1; Rew_1(1,1) = -1;
Rew_1(2,1) = -1; Rew_1(3,1) = -1; Rew_1(4,1) = -1; Rew_1(5,1) = -1; Rew_1(6,1) = -1; Rew_1(7,1) = -1; Rew_1(8,1) =
-1; Rew_1(9,1) = -1; Rew_1(10,1) = -1; Rew_1(11,1) = -1; Rew_1(1,11) = -1; Rew_1(2,11) = -1; Rew_1(3,11) = -1;
Rew_1(4,11) = -1; Rew_1(5,11) = -1; Rew_1(6,11) = -1; Rew_1(7,11) = -1; Rew_1(8,11) = -1; Rew_1(9,11) = -1;
Rew_1(10,11) = -1; Rew_1(11,11) = -1;  Rew_1(8,11) = 1; Rew_1(3,6) = -1; Rew_1(6,7) = -1; Rew_1(9,7) = -1;
Rew_1(9,8) = -1;

counter1_1=0;
last_steps_1=[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000];

temp1_1=round(rand*nx_1);    temp2_1=round(rand*ny_1);
if temp1_1==0 temp1_1=1; end
if temp2_1==0 temp2_1=1; end

start_1 = [temp1_1,temp2_1]; goal_1 = [8,11];
start_1 = [3,2];

% World #2
trial_counter_2=0;

nx_2 = 11; ny_2 = 11; ns_2 = nx_2*ny_2;

% actions
na_2 = 4; % R,U,L,D
move_2 = [1,0; 0,1; -1,0; 0,-1];
loss_2 = -0.1*[ 1; 1; 1; 1];
% reward field
Rew_2 = zeros( nx_2, ny_2);

Rew_2(5,1) = -1; Rew_2(5,2) = -1; Rew_2(5,3) = -1; Rew_2(5,4) = -1; Rew_2(5,11) = -1; Rew_2(5,10) = -1;
Rew_2(5,9) = -1; Rew_2(5,8) = -1; Rew_2(1,1) = -1; Rew_2(1,2) = -1; Rew_2(1,3) = -1; Rew_2(1,4) = -1; Rew_2(1,5) =
-1; Rew_2(1,6) = -1; Rew_2(1,7) = -1; Rew_2(1,8) = -1; Rew_2(1,9) = -1; Rew_2(1,10) = -1; Rew_2(1,11) = -1;
Rew_2(11,1) = -1; Rew_2(11,2) = -1; Rew_2(11,3) = -1; Rew_2(11,4) = -1; Rew_2(11,5) = -1; Rew_2(11,6) = -1;
Rew_2(11,7) = -1; Rew_2(11,8) = -1; Rew_2(11,9) = -1; Rew_2(11,10) = -1; Rew_2(11,11) = -1; Rew_2(1,1) = -1;
Rew_2(2,1) = -1; Rew_2(3,1) = -1; Rew_2(4,1) = -1; Rew_2(5,1) = -1; Rew_2(6,1) = -1; Rew_2(7,1) = -1; Rew_2(8,1) =
-1; Rew_2(9,1) = -1; Rew_2(10,1) = -1; Rew_2(11,1) = -1; Rew_2(1,11) = -1; Rew_2(2,11) = -1; Rew_2(3,11) = -1;
Rew_2(4,11) = -1; Rew_2(5,11) = -1; Rew_2(6,11) = -1; Rew_2(7,11) = -1; Rew_2(8,11) = -1; Rew_2(9,11) = -1;
Rew_2(10,11) = -1; Rew_2(11,11) = -1; Rew_2(8,11) = 1; Rew_2(3,6) = -1; Rew_2(6,7) = -1; Rew_2(9,7) = -1;
Rew_2(9,8) = -1;

counter1_2=0;
last_steps_2=[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000];

temp1_2=round(rand*nx_2); temp2_2=round(rand*ny_2);
if temp1_2==0 temp1_2=1; end
if temp2_2==0 temp2_2=1; end

start_2 = [temp1_2,temp2_2]; goal_2 = [8,11];
start_2 = [3,2];

% World #3
```

```
trial_counter_3=0;
nx_3 = 11; ny_3 = 11; ns_3 = nx_3*ny_3;

% actions
na_3 = 4; % R,U,L,D
move_3 = [1,0; 0,1; -1,0; 0,-1];
loss_3 = -0.1*[ 1; 1; 1; 1];

% reward field
Rew_3 = zeros( nx_3, ny_3);

Rew_3(5,1) = -1; Rew_3(5,2) = -1; Rew_3(5,3) = -1; Rew_3(5,4) = -1; Rew_3(5,11) = -1; Rew_3(5,10) = -1;
Rew_3(5,9) = -1; Rew_3(5,8) = -1; Rew_3(1,1) = -1; Rew_3(1,2) = -1; Rew_3(1,3) = -1; Rew_3(1,4) = -1; Rew_3(1,5) =
-1; Rew_3(1,6) = -1; Rew_3(1,7) = -1; Rew_3(1,8) = -1; Rew_3(1,9) = -1; Rew_3(1,10) = -1; Rew_3(1,11) = -1;
Rew_3(11,1) = -1; Rew_3(11,2) = -1; Rew_3(11,3) = -1; Rew_3(11,4) = -1; Rew_3(11,5) = -1; Rew_3(11,6) = -1;
Rew_3(11,7) = -1; Rew_3(11,8) = -1; Rew_3(11,9) = -1; Rew_3(11,10) = -1; Rew_3(11,11) = -1; Rew_3(1,1) = -1;
Rew_3(2,1) = -1; Rew_3(3,1) = -1; Rew_3(4,1) = -1; Rew_3(5,1) = -1; Rew_3(6,1) = -1; Rew_3(7,1) = -1; Rew_3(8,1) =
-1; Rew_3(9,1) = -1; Rew_3(10,1) = -1; Rew_3(11,1) = -1; Rew_3(1,11) = -1; Rew_3(2,11) = -1; Rew_3(3,11) = -1;
Rew_3(4,11) = -1; Rew_3(5,11) = -1; Rew_3(6,11) = -1; Rew_3(7,11) = -1; Rew_3(8,11) = -1; Rew_3(9,11) = -1;
Rew_3(10,11) = -1; Rew_3(11,11) = -1; Rew_3(8,11) = 1; Rew_3(3,6) = -1; Rew_3(6,7) = -1; Rew_3(9,7) = -1;
Rew_3(9,8) = -1;

counter1_3=0;
last_steps_3=[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000];
temp1_3=round(rand*nx_3); temp2_3=round(rand*ny_3);
if temp1_3==0 temp1_3=1; end
if temp2_3==0 temp2_3=1; end

start_3 = [temp1_3,temp2_3]; goal_3 = [8,11];
start_3 = [3,2];
if enable_graphics==1 gwf( 'init'); end
if enable_graphics==1 gwf( 'reward'); end
case 'agent'        % A new agent

% World #1
Val_1 = zeros(nx_1,ny_1); alpha_1 = 0.95; gamma_1 = 0.99; lambda_1 = 0.5; beta_1 = 2;

visiting_counter_1=zeros(11,11);
   for i=1:11
      for j=1:11
         addaptive_alpha_1(i,j)=alpha_1;
      end
   end

% World #2
Val_2 = zeros(nx_2,ny_2); alpha_2 = 0.95; gamma_2 = 0.99; lambda_2 = 0.5; beta_2 = 2;
visiting_counter_2=zeros(11,11);
   for i=1:11
      for j=1:11
         addaptive_alpha_2(i,j)=alpha_2;
      end
   end

% World #3
Val_3 = zeros(nx_3,ny_3); alpha_3 = 0.95; gamma_3 = 0.99; lambda_3 = 0.5; beta_3 = 2;
visiting_counter_3=zeros(11,11);
   for i=1:11
      for j=1:11
         addaptive_alpha_3(i,j)=alpha_3;
      end
   end

case 'init' % A new trial
```

```
% World #1
tmax_1 = 10000; state_1 = zeros(tmax_1,2); action_1 = zeros(tmax_1,1); reward_1 = zeros(tmax_1,1);
value_1 = zeros(tmax_1,1); delta_1 = zeros(tmax_1,1);
Elig_1 = zeros(nx_1,ny_1);

% World #2
tmax_2 = 10000; state_2 = zeros(tmax_2,2); action_2 = zeros(tmax_2,1); reward_2 = zeros(tmax_2,1);
value_2 = zeros(tmax_2,1); delta_2 = zeros(tmax_2,1);
Elig_2 = zeros(nx_2,ny_2);

% World #3
tmax_3 = 10000; state_3 = zeros(tmax_3,2); action_3 = zeros(tmax_3,1); reward_3 = zeros(tmax_3,1);        value_3 =
zeros(tmax_3,1); delta_3 = zeros(tmax_3,1);
Elig_3 = zeros(nx_3,ny_3);

case 'run'
stop_1 = 0;

% World #1
temp1_1=round(rand*nx_1); temp2_1=round(rand*ny_1); if temp1_1==0 temp1_1=1; end
if temp2_1==0 temp2_1=1; end
   start_1 = [temp1_1,temp2_1];
   st_1 = start_1;
   st_1=[3,2];
          for t_1=1:tmax_1
                 if stop_1==1, break; end
                 % new state
                 state_1(t_1,:) = st_1;

if enable_graphics==1       gwf( 'agent', [st_1,Rew_1(st_1(1),st_1(2))]); figure(ff_1); drawnow; end

% value
value_1(t_1) = Val_1(st_1(1),st_1(2));

if t_1 > 1
        visiting_counter_1(st_1(1),st_1(2)) = visiting_counter_1(st_1(1),st_1(2))+1;
        addaptive_alpha_1(st_1(1),st_1(2)) = alpha_1/(visiting_counter_1(st_1(1),st_1(2))+1);

% TD error
delta_1(t_1-1) = reward_1(t_1-1) + gamma_1*value_1(t_1) - value_1(t_1-1);

% update value
for i=1:11
   for j=1:11
      cell(i,j) = Val_1(i,j)-Val_2(i,j);
      if (cell(i,j)<=0)
         max_Val_1_Val_2(i,j) = Val_1(i,j);
      else
         max_Val_1_Val_2(i,j) = Val_2(i,j);
      end
   end
end

for i=1:11
   for j=1:11
      cell(i,j) = max_Val_1_Val_2(i,j)-Val_3(i,j);
      if (cell(i,j)<=0)
         max_Val_1_Val_2_Val_3(i,j) = max_Val_1_Val_2(i,j);
      else
         max_Val_1_Val_2_Val_3(i,j) = Val_3(i,j);
      end
   end
```

```
end
Val_1=max_Val_1_Val_2_Val_3;
Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1-1)*Elig_1;

end

% update eligibility trace
Elig_1 = gamma_1*lambda_1*Elig_1;
Elig_1(st_1(1),st_1(2)) = 1;

% final step
if state_1(t_1,:)==goal_1
reward_1(t_1) = Rew_1(st_1(1),st_1(2));
delta_1(t_1) = reward_1(t_1) - value_1(t_1);

Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1)*Elig_1;

trial_counter_1 = trial_counter_1+1;
break;
end

% predict next states: each row for an action
pstate_1 = repmat( st_1, na_1, 1) + move_1;
pstate_1 = min( max( pstate_1,1), repmat([nx_1,ny_1],na_1,1));
% linear index
istate_1 = sub2ind( [nx_1,ny_1], pstate_1(:,1), pstate_1(:,2));
% take an action by softmax
pq_1 = loss_1 + gamma_1*Val_1(istate_1); % each row for an action
prob_1 = exp( beta_1*pq_1);
prob_1 = prob_1./(sum(prob_1));    % selection probablity
act_1 = find( cumsum(prob_1) > rand(1));
action_1(t_1) = act_1(1);   % index of selected action
% reward: from state and action
reward_1(t_1) = Rew_1(st_1(1),st_1(2)) + loss_1(action_1(t_1));

next_action_1 = action_1(t_1,1);
% next state
st_1 = st_1 + move_1(action_1(t_1),:); st_1 = min( max( st_1, 1), [nx_1,ny_1]);

end

beta_1_f = fopen('beta_1.txt','a');    fprintf(beta_1_f,'%g\r\n', beta_1);   fclose(beta_1_f);
rewards_1 = fopen('rewards_1.txt','a');    fprintf(rewards_1,'%g\r\n',mean( reward_1(1:t_1)));   fclose(rewards_1);
steps_1 = fopen('steps_1.txt','a');    fprintf(steps_1,'%g\r\n',t_1);    fclose(steps_1);

counter1_1=counter1_1+1;
last_steps_1(counter1_1)=t_1(max(size(t_1)));
stop_condition_1=mean(last_steps_1);

%stop_condition_f_1 = fopen('stop_condition_1.txt','a');    fprintf(stop_condition_f_1,'%g\r\n',stop_condition_1);
fclose(stop_condition_f_1);

if counter1_1>=10 counter1_1=0; end

if (stop_condition_1>=max(stop_condition_2,stop_condition_3))
   beta_1 = 2;
else
   beta_1 = beta_1 + 1;
end

if (beta_1<=2)
   beta_1=2;
end
```

```
% World #2
temp1_2=round(rand*nx_2);    temp2_2=round(rand*ny_2);    if temp1_2==0 temp1_2=1;    end
if temp2_2==0 temp2_2=1;    end
start_2 = [temp1_2,temp2_2];
st_2 = start_2;
st_2=[3,2];
temp_d=0;

for t_2=1:tmax_2
% new state
state_2(t_2,:) = st_2;

% value
value_2(t_2) = Val_2(st_2(1),st_2(2));
if t_2 > 1
visiting_counter_2(st_2(1),st_2(2)) = visiting_counter_2(st_2(1),st_2(2))+1;
addaptive_alpha_2(st_2(1),st_2(2)) = alpha_2/(visiting_counter_2(st_2(1),st_2(2))+1);

% TD error
delta_2(t_2-1) = reward_2(t_2-1) + gamma_2*value_2(t_2) - value_2(t_2-1);

temp_d=temp_d + delta_2(t_2-1) ;

% update value
Val_2 = Val_2 + addaptive_alpha_2(st_2(1),st_2(2))*delta_2(t_2-1)*Elig_2;
end

% update eligibility trace
Elig_2 = gamma_2*lambda_2*Elig_2;
Elig_2(st_2(1),st_2(2)) = 1;

% final step
if state_2(t_2,:)==goal_2
reward_2(t_2) = Rew_2(st_2(1),st_2(2));
delta_2(t_2) = reward_2(t_2) - value_2(t_2);
Val_2 = Val_2 + addaptive_alpha_2(st_2(1),st_2(2))*delta_2(t_2)*Elig_2;
trial_counter_2 = trial_counter_2+1;
temp_d=temp_d/t_2
break;
end

% predict next states: each row for an action
pstate_2 = repmat( st_2, na_2, 1) + move_2;
pstate_2 = min( max( pstate_2,1), repmat([nx_2,ny_2],na_2,1));
% linear index
istate_2 = sub2ind( [nx_2,ny_2], pstate_2(:,1), pstate_2(:,2));
% take an action by softmax
pq_2 = loss_2 + gamma_2*Val_2(istate_2); % each row for an action
prob_2 = exp( beta_2*pq_2);
prob_2 = prob_2./(sum(prob_2));    % selection probablity
act_2 = find( cumsum(prob_2) > rand(1));
action_2(t_2) = act_2(1);               % index of selected action
% reward: from state and action
reward_2(t_2) = Rew_2(st_2(1),st_2(2)) + loss_2(action_2(t_2));

next_action_2 = action_2(t_2,1);

% next state
st_2 = st_2 + move_2(action_2(t_2),:); st_2 = min( max( st_2, 1), [nx_2,ny_2]);
end

beta_2_f = fopen('beta_2.txt','a'); fprintf(beta_2_f,'%g\r\n', beta_2);   fclose(beta_2_f);
```

```
rewards_2 = fopen('rewards_2.txt','a'); fprintf(rewards_2,'%g\r\n',mean( reward_2(1:t_2)));   fclose(rewards_2);
steps_2 = fopen('steps_2.txt','a'); fprintf(steps_2,'%g\r\n',t_2); fclose(steps_2);

counter1_2=counter1_2+1;
last_steps_2(counter1_2)=t_2(max(size(t_2)));
stop_condition_2=mean(last_steps_2);

fclose(stop_condition_f_2);

if counter1_2>=10 counter1_2=0; end

if (stop_condition_2>=stop_condition_threshold)
   beta_2 = 2;
else
   beta_2 = beta_2 + 1;
end

if (beta_2<=2)
   beta_2=2;
end

% World #3
temp1_3=round(rand*nx_3); temp2_3=round(rand*ny_3); if temp1_3==0 temp1_3=1; end
if temp2_3==0 temp2_3=1; end
start_3 = [temp1_3,temp2_3];
st_3 = start_3;
st_3=[3,2];
for t_3=1:tmax_3

% new state
state_3(t_3,:) = st_3;
% value
value_3(t_3) = Val_3(st_3(1),st_3(2));

if t_3 > 1
visiting_counter_3(st_3(1),st_3(2)) = visiting_counter_3(st_3(1),st_3(2))+1;
addaptive_alpha_3(st_3(1),st_3(2)) = alpha_3/(visiting_counter_3(st_3(1),st_3(2))+1);

% TD error
delta_3(t_3-1) = reward_3(t_3-1) + gamma_3*value_3(t_3) - value_3(t_3-1);
% update value
Val_3 = Val_3 + addaptive_alpha_3(st_3(1),st_3(2))*delta_3(t_3-1)*Elig_3;

end
% update eligibility trace
Elig_3 = gamma_3*lambda_3*Elig_3;
Elig_3(st_3(1),st_3(2)) = 1;

% final step
if state_3(t_3,:)==goal_3
reward_3(t_3) = Rew_3(st_3(1),st_3(2));
delta_3(t_3) = reward_3(t_3) - value_3(t_3);
Val_3 = Val_3 + addaptive_alpha_3(st_3(1),st_3(2))*delta_3(t_3)*Elig_3;
trial_counter_3 = trial_counter_3+1;
break;
end

% predict next states: each row for an action
pstate_3 = repmat( st_3, na_3, 1) + move_3;
pstate_3 = min( max( pstate_3,1), repmat([nx_3,ny_3],na_3,1));
% linear index
istate_3 = sub2ind( [nx_3,ny_3], pstate_3(:,1), pstate_3(:,2));
% take an action by softmax
```

```
pq_3 = loss_3 + gamma_3*Val_3(istate_3); % each row for an action
prob_3 = exp( beta_3*pq_3);
prob_3 = prob_3./(sum(prob_3));    % selection probability
act_3 = find( cumsum(prob_3) > rand(1));
action_3(t_3) = act_3(1);                 % index of selected action
% reward: from state and action
reward_3(t_3) = Rew_3(st_3(1),st_3(2)) + loss_3(action_3(t_3));

next_action_3 = action_3(t_3,1);

% next state
st_3 = st_3 + move_3(action_3(t_3),:);   st_3 = min( max( st_3, 1), [nx_3,ny_3]);

end

beta_3_f = fopen('beta_3.txt','a'); fprintf(beta_3_f,'%g\r\n', beta_3);   fclose(beta_3_f);
rewards_3 = fopen('rewards_3.txt','a'); fprintf(rewards_3,'%g\r\n',mean( reward_3(1:t_3)));   fclose(rewards_3);
steps_3 = fopen('steps_3.txt','a'); fprintf(steps_3,'%g\r\n',t_3); fclose(steps_3);

counter1_3=counter1_3+1;
last_steps_3(counter1_3)=t_3(max(size(t_3)));
stop_condition_3=mean(last_steps_3);

fclose(stop_condition_f_3);

if counter1_3>=10 counter1_3=0; end

if (stop_condition_3>=stop_condition_threshold)
   beta_3 = 2;
else
   beta_3 = beta_3 + 1;
end

if (beta_3<=2)
   beta_3=2;
end

case 'try'
gw( 'init');
gw( 'run');
if enable_graphics==1       gwf( 'value'); end

if (trial_counter_1==iterations)
index=[1:5];
size(index)
[step_1_y_axis] = textread('steps_1.txt','%d');
size(step_1_y_axis)
[step_2_y_axis] = textread('steps_2.txt','%d');
[step_3_y_axis] = textread('steps_3.txt','%d');
figure(11)
plot( index, step_1_y_axis, 'r', index, step_2_y_axis, 'g', index, step_3_y_axis, 'b', 'LineWidth',1);
[rewards_1_y_axis] = textread('rewards_1.txt','%f');
[rewards_2_y_axis] = textread('rewards_2.txt','%f');
[rewards_3_y_axis] = textread('rewards_3.txt','%f');
figure(12)
plot( index, rewards_1_y_axis, 'r', index, rewards_2_y_axis, 'g', index, rewards_3_y_axis, 'b', 'LineWidth',1);
[beta_1_y_axis] = textread('beta_1.txt','%d');
[beta_2_y_axis] = textread('beta_2.txt','%d');
[beta_3_y_axis] = textread('beta_3.txt','%d');
figure(13)
plot( index, beta_1_y_axis, 'r', index, beta_2_y_axis, 'g', index, beta_3_y_axis, 'b', 'LineWidth',1);
end
 while (trial_counter_1<=iterations-1)
```

```
 gw( 'try');
end
end
```

Appendix X. Navigation of a Mobile Robot - Source Code

vfm.m

```
function varargout = vfm(varargin)
% VFM  Perform frame grabbing from any Video for Windows source
% The function wraps a number of sub-functions. These are parameterised
% in the first paramter. Invocation of a sub-function, say 'grab',
% is of the form:
%  vfw('grab', ...parameters...)
%
% VFM('grab'?,   framecount?)
%  Grabs framecount frames. framecount defaults to 1
%  Returns M x N x 3 x framecount array of uint8, where M and N are the
%  height and width respectively. Images are in RGB format.
%
% VFM('preview'?, bPreview?)
%  Switches preview mode on or off, according to the boolean value bPreview.
%  bPreview defaults to 1.
%
% VFM('show'?, bShow?)
%  Shows or hides the capture window according to the value of the boolean, bShow.
%  The window is displayed if and only if bShow is 1. bShow defaults to 1.
%
% VFM('configsource')
%  Displays the source configuration dialog, if available for the current driver.
%
% VFM('configformat')
%  Displays the format configuration dialog, if available for the current driver.
%
% VFM('configdisplay')
%  Displays the display configuration dialog, if available for the current driver.
%
% VFM('drivername', index)
%  Returns the name of a system driver for the given index. index must be in the
%  the range 1-10. If a driver exists for that index, a string, representing the
%  name of the driver is returned.
%
% VFM('setdriver', index)
%  Sets the driver according to the index. index must be in the range 1-10. If
%  a driver does not exist for the given index, a warning is issued.
%
%   Farzad Pezeshkpour,
%   School of Information Systems,
%   University of East Anglia
%   Revision: 0.1   Date: 1998/12/16
error('Missing MEX-file VFM.DLL');
```

capture.m

```
%close all
%clear all
home

VFM('show',0);

captured_image = VFM('grab');

threshold1 = 0.35;
threshold2 = 0.1;

%captured_image = imread('yellow.jpg');

captured_image_red = (captured_image(:,:,1));
```

```
captured_image_green = (captured_image(:,:,2));
captured_image_blue = (captured_image(:,:,3));

average_captured_image_red = sum((captured_image_red),2);
average_captured_image_red = sum((average_captured_image_red),1)/240/320/256

average_captured_image_green = sum((captured_image_green),2);
average_captured_image_green = sum((average_captured_image_green),1)/240/320/256

average_captured_image_blue = sum((captured_image_blue),2);
average_captured_image_blue = sum((average_captured_image_blue),1)/240/320/256

if
    (average_captured_image_red<=threshold2)&&(average_captured_image_green<=threshold2)&&(average_captured
    _image_blue<=threshold1)
   'Black'
elseif
    (average_captured_image_red>=threshold1)&&(average_captured_image_green>=threshold1)&&(average_captured
    _image_blue<=threshold2)
   'Yellow'
elseif
    (average_captured_image_red>=threshold1)&&(average_captured_image_green>=threshold1)&&(average_captured
    _image_blue>=threshold1)
   'White'
else
   %'Ask human...'
   'Black'
end

imshow(captured_image);
```

gwi.m

```
function out = gwi( arg, arg2)
    % gwi: a grid world interface

    delete beta_1.csv
    delete beta_2.csv
    delete steps_1.csv
    delete steps_2.csv
    delete rewards_1.csv
    delete rewards_2.csv
    delete summary.csv
    %summary_f = fopen('summary.csv','w'); fclose(summary_f);
    delete stop_condition_1.csv
    delete average_val.csv

    close all
    %clear all

    home
    % World #1
    global alpha beta gamma delta lambda
    global gt ge gs bt be bs at ae as stop
    global fx fy fm ff

    if( nargin < 1)
            gwi( 'new');
            return;
    end

    % call backs
    switch( arg)
    case 'new'
```

```
    gw( 'new');

    % World #1

    gw( 'try');
    end
```

gwf.m

```
function gwf( arg, arg2)

    % World #1
    global Rew_1 Val_1 Elig_1
    global state_1 action_1 reward_1 value_1 delta_1
    global nx_1 ny_1 na_1
    global fx_1 fy_1 fm_1 ff_1
    global ax_1 ay_1 az_1 asurf_1

    % World #2
    global Rew_2 Val_2 Elig_2
    global state_2 action_2 reward_2 value_2 delta_2
    global nx_2 ny_2 na_2
    global fx_2 fy_2 fm_2 ff_2
    global ax_2 ay_2 az_2 asurf_2

    switch( arg)
    case 'init'
        ss = get(0,'ScreenSize');
        fm_1 = [ 10, 20, 42, 3];     % figure margins
        if nargin < 2
            fx_1 = min( ss(3)-4, 1024)/2 - (fm_1(1)+fm_1(2));
            fy_1 = (ss(4)-24)/2 - (fm_1(3)+fm_1(4));
        else
            fx_1 = arg2(1);
            fy_1 = arg2(2);
        end
        ff_1 = 1;            % figure to be focused: 4 for gwi
        figure(1); set(1,'Position',[fm_1(1),ss(4)-fm_1(3)-fy_1,fx_1,fy_1],'Name','Reward'); %%%clf;
        figure(2); set(2,'Position',[fm_1(1),ss(4)-fm_1(3)*2-fm_1(4)-fy_1*2,fx_1,fy_1],'Name','Value'); %%%clf;
        % Robot
        [ax_1,ay_1,az_1] = sphere(20);
        ax_1 = 0.5*ax_1(11:end,:);
        ay_1 = 0.5*ay_1(11:end,:);
        az_1 = az_1(11:end,:)+0.01;
    case 'reward'
        figure(1); clf;
        step( Rew_1'); caxis([-2,2]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -2 2]);
        axis( 'off');
    case 'value'
        figure(2); clf;
        step( Val_1'); caxis([-50,50]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -10 10]);
        % title( sprintf('Value'));
    case 'elig'
        figure(2); clf;
        step( Elig_1'); caxis([-50,50]);
        axis([0.5 nx_1+0.5 0.5 ny_1+0.5 -10 10]);
        % title( sprintf('Eligibility'));
    case 'agent'
        figure(1);
        %hold( 'on');
        if ishandle(asurf_1)
            set( asurf_1, 'xdata',arg2(1)+ax_1, 'ydata',arg2(2)+ay_1, 'zdata',arg2(3)+az_1);
```

```
        else
                hold( 'on');
                asurf_1 = surf( arg2(1)+ax_1, arg2(2)+ay_1, arg(3)+az_1, az_1*2);
        end
        %hold( 'off');
        %line( arg2(1), arg2(2), 'LineStyle', 'none', 'Marker', 'o',...
        %       'MarkerSize', 10, 'MarkerEdgeColor', 'k');
    case 'traj'
        figure(1);
        line( arg2(:,1), arg2(:,2), 'Color', 'w');
    end
    %drawnow;

    %%%%
    function s = step( x, y, z)
    % step: 3D plot of a step function

    if nargin < 3
        z = x;
        [nx_1,ny_1] = size(z);
        x = 1:nx_1;
        y = 1:ny_1;
    else
        [nx_1,ny_1] = size(z);

    end
    % double the data
    x = [ x(:)-0.5, x(:)+0.5]'; x = x(:);
    y = [ y(:)-0.5, y(:)+0.5]'; y = y(:);
    z = reshape( [ z(:), z(:)]', 2*nx_1, ny_1);
    z = reshape( [z;z], 2*nx_1, 2*ny_1);
    s = surf( x, y, z);
```

gw.m

```
function out = gw( arg, arg2)

%diary on
global enable_graphics
global enable_robot_movement
global enable_human_collaboration
global time_between_movements
global robot_orientation
global steps_counter
global dont_move_robot_flag

steps_counter=0;

robot_orientation=2;
time_between_movements=3;
enable_graphics=1;
enable_human_collaboration=0;
enable_robot_movement=0;
enable_greedy=1; % 0 - adaptive softmax, 1 - greedy
dont_move_robot_flag=1

% World #1
global nx_1 ny_1 ns_1 na_1 move_1 loss_1 addaptive_alpha_1 visiting_counter_1
global start_1 goal_1 Rew_1 Val_1 Elig_1
global alpha_1 beta_1 gamma_1 delta_1 lambda_1
global state_1 action_1 value_1 reward_1 t_1 tmax_1 stop_1
global ff_1 Mov_1
global last_steps_1 trial_counter_1
```

```
global counter1_1;
global mean_Val_1 temp1_1 temp2_1

% World #2
global nx_2 ny_2 ns_2 na_2 move_2 loss_2 addaptive_alpha_2 visiting_counter_2
global start_2 goal_2 Rew_2 Val_2 Elig_2
global alpha_2 gamma_2 delta_2 lambda_2
global state_2 action_2 value_2 reward_2 t_2 tmax_2 stop_2
global ff_2 Mov_2
global last_steps_2 trial_counter_2
global counter1_2;
global mean_Val_2 temp1_2 temp2_2

set(0,'RecursionLimit',20000);

global st_1 st_2 choosing_Val

global iterations stop_condition_threshold

%iterations = 100;
if (enable_human_collaboration==1)
iterations = 10;
else
iterations = 10;
end
stop_condition_threshold=500;
stop_condition_1=100;
stop_condition_2=100;

switch( arg)

case 'new'          % Setup
gw( 'world'); gw( 'agent'); gw( 'init');
case 'world'        % A new world


% World #1
trial_counter_1=0;

nx_1 = 6; ny_1 = 6; ns_1 = nx_1*ny_1;

% actions
na_1 = 4; % R,U,L,D
move_1 = [1,0; 0,1; -1,0; 0,-1];
loss_1 = -0.1*[ 1; 1; 1; 1];
% reward field
Rew_1 = zeros( nx_1, ny_1);

counter1_1=0;
temp1_1=round(rand*nx_1); temp2_1=round(rand*ny_1);
if temp1_1==0 temp1_1=1; end
if temp2_1==0 temp2_1=1; end

if ((temp1_1==5)&&(temp2_1==6))
temp1_1=1;
temp2_1=1;
end

start_1 = [temp1_1,temp2_1]; goal_1 = [4,6];
start_1 = [temp1_1, temp2_1];

% World #2
trial_counter_2=0;
```

```
nx_2 = 6; ny_2 = 6; ns_2 = nx_2*ny_2;
% actions
na_2 = 4; % R,U,L,D
move_2 = [1,0; 0,1; -1,0; 0,-1];
loss_2 = -0.1*[ 1; 1; 1; 1];
% reward field
Rew_2 = zeros( nx_2, ny_2);

counter1_2=0;

goal_2 = [5,6];
start_2 = [temp1_1, temp2_1];

if enable_graphics==1 gwf( 'init'); end
if enable_graphics==1 gwf( 'reward'); end
case 'agent'          % A new agent

% World #1
Val_1 = zeros(nx_1,ny_1); alpha_1 = 0.95; gamma_1 = 0.99; lambda_1 = 0.5; %beta_1 = 10;

visiting_counter_1=zeros(6,6);
for i=1:6
   for j=1:6
      addaptive_alpha_1(i,j)=alpha_1;
   end
end

if (enable_greedy==1) beta_1=10; else beta_1=10; end

% World #2
Val_2 = zeros(nx_2,ny_2); alpha_2 = 0.95; gamma_2 = 0.99; lambda_2 = 0.5;
visiting_counter_2=zeros(6,6);
for i=1:6
   for j=1:6
      addaptive_alpha_2(i,j)=alpha_2;
   end
end

case 'init' % A new trial
% World #1
tmax_1 = 500; state_1 = zeros(tmax_1,2); action_1 = zeros(tmax_1,1);  reward_1 = zeros(tmax_1,1); value_1 =
zeros(tmax_1,1);
delta_1 = zeros(tmax_1,1);
Elig_1 = zeros(nx_1,ny_1);

% World #2
tmax_2 = 500; state_2 = zeros(tmax_2,2); action_2 = zeros(tmax_2,1);  reward_2 = zeros(tmax_2,1); value_2 =
zeros(tmax_2,1);
delta_2 = zeros(tmax_2,1);
Elig_2 = zeros(nx_2,ny_2);

case 'run'
stop_1 = 0;

% World #1
if (enable_greedy==0)
if (t_1-1>=15)   %stop_condition_1 % stop_condition_threshold
   beta_1 = beta_1+1;
   beta_1 = 10;
else
   beta_1 = 2;
   beta_1 = 10;
```

```
end

if (beta_1<=2)
   beta_1=2;
end
end

beta_1_f = fopen('beta_1.csv','a'); fprintf(beta_1_f,'%g\r\n', beta_1); fclose(beta_1_f);

temp1_1=round(rand*nx_1);    temp2_1=round(rand*ny_1);
   if temp1_1==0 temp1_1=1; end
   if temp2_1==0 temp2_1=1; end

   if ((temp1_1==5)&&(temp2_1==6))
      temp1_1=1;
      temp2_1=1;
   end

start_1 = [temp1_1,temp2_1];
st_1 = start_1;
st_1=[temp1_1,temp2_1];
st_1 = [4,2];
for t_1=1:tmax_1

state_1(t_1,:) = st_1;

if enable_graphics==1 gwf( 'agent', [st_1,Rew_1(st_1(1),st_1(2))]); figure(ff_1); drawnow; end


if ((t_1-1)>6000)
enable_human_collaboration=1;
else
enable_human_collaboration=0;
end

if((enable_human_collaboration==1)&(((st_1(1)==4)&&(st_1(2)==4))||((st_1(1)==5)&&(st_1(2)==4))||((st_1(1)==6)&&
(st_1(2)==4))||((st_1(1)==4)&&(st_1(2)==5))||((st_1(1)==5)&&(st_1(2)==5))||((st_1(1)==6)&&(st_1(2)==5))||((st_1(1)=
=5)&&(st_1(2)==6))||((st_1(1)==6)&&(st_1(2)==6))))
%if ((st_1(1)==4)&&(st_1(2)==6))break; end

reply = input('Human Collaboration Area: Where to go? [L, R, U, D]?','s');
         if (reply=='l')
            'Left';
            next_action_1 = 3;
         end
         if (reply=='r')
            'Right';
            next_action_1 = 1;
         end
         if (reply=='u')
            'Up';
            next_action_1 = 2;
         end
         if (reply=='d')
            'Down';
            next_action_1 = 4;
         end

% value
value_1(t_1) = Val_1(st_1(1),st_1(2));

if t_1 > 1
    visiting_counter_1(st_1(1),st_1(2)) = visiting_counter_1(st_1(1),st_1(2))+1;
```

```
    addaptive_alpha_1(st_1(1),st_1(2)) = alpha_1/(visiting_counter_1(st_1(1),st_1(2))+1);
           % TD error
           % delta_1(t_1-1) = min(reward_1(t_1-1), reward_2(t_1-1)) + gamma_1*value_1(t_1) - value_1(t_1-1);

delta_1(t_1-1) = reward_1(t_1-1) + gamma_1*value_1(t_1) - value_1(t_1-1);

% update value
Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1-1)*Elig_1;

end

% update eligibility trace
Elig_1 = gamma_1*lambda_1*Elig_1;
Elig_1(st_1(1),st_1(2)) = Elig_1(st_1(1),st_1(2)) + 1;

% final step
state_1(t_1,:)

if state_1(t_1,:)==goal_1
reward_1(t_1) = Rew_1(st_1(1),st_1(2));
delta_1(t_1) = reward_1(t_1) - value_1(t_1);

Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1)*Elig_1;

trial_counter_1 = trial_counter_1+1;

Rew_1(st_1(1),st_1(2)); %current reward
                                    break;
end

if (((st_1(1)==6)&&(st_1(2)==2)&&(next_action_1==1))||   ((st_1(1)==6)&&(st_1(2)==3)&&(next_action_1==1))||
((st_1(1)==6)&&(st_1(2)==4)&&(next_action_1==1))||   ((st_1(1)==6)&&(st_1(2)==5)&&(next_action_1==1))||
((st_1(1)==1)&&(st_1(2)==2)&&(next_action_1==3))||  ((st_1(1)==1)&&(st_1(2)==3)&&(next_action_1==3))||
((st_1(1)==1)&&(st_1(2)==4)&&(next_action_1==3))||   ((st_1(1)==1)&&(st_1(2)==5)&&(next_action_1==3))||
((st_1(1)==2)&&(st_1(2)==6)&&(next_action_1==2))||   ((st_1(1)==3)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==4)&&(st_1(2)==6)&&(next_action_1==2))||   ((st_1(1)==5)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==2)&&(st_1(2)==1)&&(next_action_1==4))||  ((st_1(1)==3)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==4)&&(st_1(2)==1)&&(next_action_1==4))||   ((st_1(1)==5)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==6)&&(st_1(2)==1)&&(next_action_1==1))||   ((st_1(1)==6)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==1)&&(st_1(2)==6)&&(next_action_1==3))||   ((st_1(1)==1)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==1)&&(st_1(2)==1)&&(next_action_1==4))||   ((st_1(1)==1)&&(st_1(2)==1)&&(next_action_1==3))||
((st_1(1)==6)&&(st_1(2)==6)&&(next_action_1==2))||   ((st_1(1)==6)&&(st_1(2)==6)&&(next_action_1==1)))
   dont_move_robot_flag=1;
else
   dont_move_robot_flag=1;
end

dont_move_robot_flag

% next state
st_1 = st_1 + move_1(next_action_1,:); st_1 = min( max( st_1, 1), [nx_1,ny_1]);

if (dont_move_robot_flag==0)

if ((robot_orientation==2)&&(next_action_1==2))
   winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
   pause(time_between_movements);
   robot_orientation=2;
elseif ((robot_orientation==2)&&(next_action_1==4))
   winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
   pause(time_between_movements);
   robot_orientation=2;
elseif ((robot_orientation==2)&&(next_action_1==1))
```

```
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==2)&&(next_action_1==3))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;

elseif ((robot_orientation==4)&&(next_action_1==2))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==4)&&(next_action_1==4))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==4)&&(next_action_1==1))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==4)&&(next_action_1==3))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;

elseif ((robot_orientation==1)&&(next_action_1==2))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==1)&&(next_action_1==4))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==1)&&(next_action_1==1))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==1)&&(next_action_1==3))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;

elseif ((robot_orientation==3)&&(next_action_1==2))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==3)&&(next_action_1==4))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==3)&&(next_action_1==1))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;
elseif ((robot_orientation==3)&&(next_action_1==3))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;
end

%%%%%
pause(8);
```

```
%%% start capturing
figure(20)

VFM('show',0);

captured_image = VFM('grab');

threshold1 = 0.4;
threshold2 = 0.1;

%captured_image = imread('yellow.jpg');

captured_image_red = (captured_image(:,:,1));
captured_image_green = (captured_image(:,:,2));
captured_image_blue = (captured_image(:,:,3));

average_captured_image_red = sum((captured_image_red),2);
average_captured_image_red = sum((average_captured_image_red),1)/240/320/256

average_captured_image_green = sum((captured_image_green),2);
average_captured_image_green = sum((average_captured_image_green),1)/240/320/256

average_captured_image_blue = sum((captured_image_blue),2);
average_captured_image_blue = sum((average_captured_image_blue),1)/240/320/256

if(average_captured_image_red<=threshold1)&&(average_captured_image_green<=threshold1)&&(average_captured_i
mage_blue<=threshold1)
   'Black'
   Rew_1(st_1(1),st_1(2))=0
elseif (average_captured_image_red>=0.4)&&(average_captured_image_red<=0.65)
   'Pink'
   Rew_1(st_1(1),st_1(2))=-1
elseif
(average_captured_image_red>=threshold1)&&(average_captured_image_green>=threshold1)&&(average_captured_im
age_blue>=threshold1)
   'White'
   Rew_1(st_1(1),st_1(2))=1.5
   'Put robot at starting point'
   pause
   %quit
end

imshow(captured_image);

if ((st_1(1)==4)&&(st_1(2)==6)) Rew_1(st_1(1),st_1(2))=1.5; end

%%% end capture
end
st_1
%pause
else
% value
value_1(t_1) = Val_1(st_1(1),st_1(2));

if t_1 > 1
visiting_counter_1(st_1(1),st_1(2)) = visiting_counter_1(st_1(1),st_1(2))+1;
addaptive_alpha_1(st_1(1),st_1(2)) = alpha_1/(visiting_counter_1(st_1(1),st_1(2))+1);

% TD error
% delta_1(t_1-1) = min(reward_1(t_1-1), reward_2(t_1-1)) + gamma_1*value_1(t_1) - value_1(t_1-1);

delta_1(t_1-1) = reward_1(t_1-1) + gamma_1*value_1(t_1) - value_1(t_1-1);
```

```
% update value
Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1-1)*Elig_1;

% update eligibility trace
Elig_1 = gamma_1*lambda_1*Elig_1;
Elig_1(st_1(1),st_1(2)) = Elig_1(st_1(1),st_1(2)) + 1;

% final step
if state_1(t_1,:)==goal_1
        reward_1(t_1) = Rew_1(st_1(1),st_1(2));
        delta_1(t_1) = reward_1(t_1) - value_1(t_1);

Val_1 = Val_1 + addaptive_alpha_1(st_1(1),st_1(2))*delta_1(t_1)*Elig_1;

trial_counter_1 = trial_counter_1+1;
%st_1
Rew_1(st_1(1),st_1(2)); %current reward

break;
end

% predict next states: each row for an action
pstate_1 = repmat( st_1, na_1, 1) + move_1;
pstate_1 = min( max( pstate_1,1), repmat([nx_1,ny_1],na_1,1)); % set of possible states to move to
% linear index
istate_1 = sub2ind( [nx_1,ny_1], pstate_1(:,1), pstate_1(:,2));

% take an action by softmax
pq_1 = loss_1 + gamma_1*Val_1(istate_1); % each row for an action
prob_1 = exp( beta_1*pq_1);
prob_1 = prob_1./(sum(prob_1));    % selection probablity
act_1 = find( cumsum(prob_1) > rand(1));
action_1(t_1) = act_1(1);               % index of selected action
% reward: from state and action
reward_1(t_1) = Rew_1(st_1(1),st_1(2)) + loss_1(action_1(t_1));

Rew_1(st_1(1),st_1(2)) %current reward

 % st_1 % current robot state

next_action_1 = action_1(t_1,1)

if (((st_1(1)==6)&&(st_1(2)==2)&&(next_action_1==1))||  ((st_1(1)==6)&&(st_1(2)==3)&&(next_action_1==1))||
((st_1(1)==6)&&(st_1(2)==4)&&(next_action_1==1))||  ((st_1(1)==6)&&(st_1(2)==5)&&(next_action_1==1))||
((st_1(1)==1)&&(st_1(2)==2)&&(next_action_1==3))||  ((st_1(1)==1)&&(st_1(2)==3)&&(next_action_1==3))||
((st_1(1)==1)&&(st_1(2)==4)&&(next_action_1==3))||  ((st_1(1)==1)&&(st_1(2)==5)&&(next_action_1==3))||
((st_1(1)==2)&&(st_1(2)==6)&&(next_action_1==2))||  ((st_1(1)==3)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==4)&&(st_1(2)==6)&&(next_action_1==2))||  ((st_1(1)==5)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==2)&&(st_1(2)==1)&&(next_action_1==4))||  ((st_1(1)==3)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==4)&&(st_1(2)==1)&&(next_action_1==4))||  ((st_1(1)==5)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==6)&&(st_1(2)==1)&&(next_action_1==1))||  ((st_1(1)==6)&&(st_1(2)==1)&&(next_action_1==4))||
((st_1(1)==1)&&(st_1(2)==6)&&(next_action_1==3))||  ((st_1(1)==1)&&(st_1(2)==6)&&(next_action_1==2))||
((st_1(1)==1)&&(st_1(2)==1)&&(next_action_1==4))||  ((st_1(1)==1)&&(st_1(2)==1)&&(next_action_1==3))||
((st_1(1)==6)&&(st_1(2)==6)&&(next_action_1==2))||  ((st_1(1)==6)&&(st_1(2)==6)&&(next_action_1==1)))
   dont_move_robot_flag=1;
else
   dont_move_robot_flag=0;
end

st_1 = st_1 + move_1(action_1(t_1),:);  st_1 = min( max( st_1, 1), [nx_1,ny_1]);

if (dont_move_robot_flag==0)
```

```
if ((robot_orientation==2)&&(next_action_1==2))
   winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==2)&&(next_action_1==4))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==2)&&(next_action_1==1))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==2)&&(next_action_1==3))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;
elseif ((robot_orientation==4)&&(next_action_1==2))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==4)&&(next_action_1==4))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==4)&&(next_action_1==1))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==4)&&(next_action_1==3))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;
elseif ((robot_orientation==1)&&(next_action_1==2))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==1)&&(next_action_1==4))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==1)&&(next_action_1==1))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==1)&&(next_action_1==3))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=1;
elseif ((robot_orientation==3)&&(next_action_1==2))
    winopen('c:\tmp\rl\turn_robot_90_right_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=2;
elseif ((robot_orientation==3)&&(next_action_1==4))
    winopen('c:\tmp\rl\turn_robot_90_left_and_move_forward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=4;
elseif ((robot_orientation==3)&&(next_action_1==1))
    winopen('c:\tmp\rl\move_robot_backward_15_inches.exe')
    pause(time_between_movements);
    robot_orientation=3;
elseif ((robot_orientation==3)&&(next_action_1==3))
    winopen('c:\tmp\rl\move_robot_forward_15_inches.exe')
    pause(time_between_movements);
```

```
    robot_orientation=3;
end

%%%%%
pause(8);
%%% start capturing
figure(20)
VFM('show',0);

captured_image = VFM('grab');

threshold1 = 0.4;
threshold2 = 0.1;

%captured_image = imread('yellow.jpg');

captured_image_red = (captured_image(:,:,1));
captured_image_green = (captured_image(:,:,2));
captured_image_blue = (captured_image(:,:,3));

average_captured_image_red = sum((captured_image_red),2);
average_captured_image_red = sum((average_captured_image_red),1)/240/320/256

average_captured_image_green = sum((captured_image_green),2);
average_captured_image_green = sum((average_captured_image_green),1)/240/320/256

average_captured_image_blue = sum((captured_image_blue),2);
average_captured_image_blue = sum((average_captured_image_blue),1)/240/320/256

if
(average_captured_image_red<=threshold1)&&(average_captured_image_green<=threshold1)&&(average_captured_im
age_blue<=threshold1)
    'Black'
    Rew_1(st_1(1),st_1(2))=0
elseif (average_captured_image_red>=0.4)&&(average_captured_image_red<=0.65)
    'Pink'
    Rew_1(st_1(1),st_1(2))=-1
elseif
(average_captured_image_red>=threshold1)&&(average_captured_image_green>=threshold1)&&(average_captured_im
age_blue>=threshold1)
    'White'
    Rew_1(st_1(1),st_1(2))=1.5
    'Put robot at starting point'
    pause
    %quit
end

imshow(captured_image);

%%% end capturing
end
end
end
end
if (trial_counter_1<=10)
    stop_condition_1 = t_1-1;
else
counter1_1=counter1_1+1;
last_steps_1(counter1_1)=t_1(max(size(t_1-1)));
stop_condition_1=mean(last_steps_1)-1;
if counter1_1>=10 counter1_1=0; end
end
```

```
rewards_1 = fopen('rewards_1.csv','a');    fprintf(rewards_1,'%g\r\n',mean( reward_1(1:t_1-1)));   fclose(rewards_1);
steps_1 = fopen('steps_1.csv','a');    fprintf(steps_1,'%g\r\n',t_1-1);    fclose(steps_1);
stop_condition_1_f = fopen('stop_condition_1.csv','a');    fprintf(stop_condition_1_f,'%g\r\n', stop_condition_1);
fclose(stop_condition_1_f);
average_val_f = fopen('average_val.csv','a');    fprintf(average_val_f,'%g\r\n', Val_1(4,6));    fclose(average_val_f);

[step_1_y_axis] = textread('steps_1.csv','%d');
stop_condition_threshold = mean(step_1_y_axis)

summary_f = fopen('summary.csv','a'); fprintf(summary_f, '%g, %g, %g, %g, %g \n',  t_1-1, beta_1, mean(
reward_1(1:t_1-1)), stop_condition_1, Val_1(4,6)); fclose(summary_f);

case 'try'

gw( 'init');
gw( 'run');

if enable_graphics==1 gwf( 'value'); end

trial_counter_1;

if (trial_counter_1==iterations)

index=[1:iterations];

[step_1_y_axis] = textread('steps_1.csv','%d');
figure(11)
plot( index, step_1_y_axis, 'r',  'LineWidth',1);

steps_counter = sum(steps_counter + step_1_y_axis)

[rewards_1_y_axis] = textread('rewards_1.csv','%f');
figure(12)
plot( index, rewards_1_y_axis, 'r',  'LineWidth',1);

[beta_1_y_axis] = textread('beta_1.csv','%d');
figure(13)
plot( index, beta_1_y_axis, 'r', 'LineWidth',1);

[stop_condition_1_y_axis] = textread('stop_condition_1.csv','%f');
figure(14)
plot( index, stop_condition_1_y_axis, 'r', 'LineWidth',1);

[average_val_1_y_axis] = textread('average_val.csv','%f');
figure(15)
plot( index, average_val_1_y_axis, 'r', 'LineWidth',1);

stop_condition_threshold = mean(step_1_y_axis)

end

%winopen('c:\tmp\rl\Simple_Grid_World\summary.csv');
while (trial_counter_1<=iterations-1)
gw( 'try');

end
end
```

Appendix XI. Bag Shaking Experiment with a Fixed-Arm Robot - Source Code

Digital Scale

CRs232.vb

```
Imports System.Runtime.InteropServices
Imports System.Text
Imports System.Threading
Imports System.ComponentModel
Imports System.IO

#Region "RS232"
Public Class Rs232 : Implements IDisposable
   '// Class Members
Private mhRS As IntPtr = New IntPtr(0)        '// Handle to Com Port
Private miPort As Integer = 3   '//  Default is COM1
Private miTimeout As Int32 = 70   '// Timeout in ms
Private miBaudRate As Int32 = 9600
Private meParity As DataParity = 0
Private meStopBit As DataStopBit = 0
Private miDataBit As Int32 = 8
Private miBufferSize As Int32 = 512   '// Buffers size default to 512 bytes
Private mabtRxBuf As Byte()   '//  Receive buffer
Private meMode As Mode  '//  Class working mode
Private moThreadTx As Thread
Private moThreadRx As Thread
Private moEvents As Thread
Private miTmpBytes2Read As Int32
Private meMask As EventMasks
Private mbDisposed As Boolean
Private mbUseXonXoff As Boolean
Private mbEnableEvents As Boolean
Private miBufThreshold As Int32 = 1
Private muOvlE As OVERLAPPED
Private muOvlW As OVERLAPPED
Private muOvlR As OVERLAPPED
Private mHE As GCHandle
Private mHR As GCHandle
Private mHW As GCHandle
'-----------------------------------------------------------------------------------------

#Region "Enums"
        '// Parity Data
        Public Enum DataParity
                Parity_None = 0
    Parity_Odd
                Parity_Even
                Parity_Mark
        End Enum
        '// StopBit Data
        Public Enum DataStopBit
                StopBit_1 = 1
                StopBit_2
        End Enum
        <Flags()> Public Enum PurgeBuffers
                RXAbort = &H2
                RXClear = &H8
                TxAbort = &H1
                TxClear = &H4
        End Enum
        Private Enum Lines
                SetRts = 3
                ClearRts = 4
```

```
            SetDtr = 5
            ClearDtr = 6
            ResetDev = 7            '       // Reset device if possible
            SetBreak = 8            '       // Set the device break line.
            ClearBreak = 9          '       // Clear the device break line.
        End Enum
        '// Modem Status
        <Flags()> Public Enum ModemStatusBits
            ClearToSendOn = &H10
            DataSetReadyOn = &H20
            RingIndicatorOn = &H40
            CarrierDetect = &H80
        End Enum
        '// Working mode
        Public Enum Mode
            NonOverlapped
            Overlapped
        End Enum
        '// Comm Masks
        <Flags()> Public Enum EventMasks
            RxChar = &H1
            RXFlag = &H2
            TxBufferEmpty = &H4
            ClearToSend = &H8
            DataSetReady = &H10
            CarrierDetect = &H20
            Break = &H40
            StatusError = &H80
            Ring = &H100
        End Enum

#End Region
#Region "Structures"
        <StructLayout(LayoutKind.Sequential, Pack:=1)> Private Structure DCB
            Public DCBlength As Int32
            Public BaudRate As Int32
            Public Bits1 As Int32
            Public wReserved As Int16
            Public XonLim As Int16
            Public XoffLim As Int16
            Public ByteSize As Byte
            Public Parity As Byte
            Public StopBits As Byte
            Public XonChar As Char
            Public XoffChar As Char
            Public ErrorChar As Char
            Public EofChar As Char
            Public EvtChar As Char
            Public wReserved2 As Int16
        End Structure
        <StructLayout(LayoutKind.Sequential, Pack:=1)> Private Structure COMMTIMEOUTS
            Public ReadIntervalTimeout As Int32
            Public ReadTotalTimeoutMultiplier As Int32
            Public ReadTotalTimeoutConstant As Int32
            Public WriteTotalTimeoutMultiplier As Int32
            Public WriteTotalTimeoutConstant As Int32
        End Structure
    <StructLayout(LayoutKind.Sequential, Pack:=8)> Private Structure COMMCONFIG
      Public dwSize As Int32
      Public wVersion As Int16
      Public wReserved As Int16
      Public dcbx As DCB
      Public dwProviderSubType As Int32
```

```
      Public dwProviderOffset As Int32
      Public dwProviderSize As Int32
      Public wcProviderData As Int16
   End Structure
   <StructLayout(LayoutKind.Sequential, Pack:=1)> Public Structure OVERLAPPED
      Public Internal As Int32
      Public InternalHigh As Int32
      Public Offset As Int32
      Public OffsetHigh As Int32
      Public hEvent As IntPtr
   End Structure
   <StructLayout(LayoutKind.Sequential, Pack:=1)> Private Structure COMSTAT
   Dim fBitFields As Int32
   Dim cbInQue As Int32
   Dim cbOutQue As Int32
   End Structure

#End Region
#Region "Constants"
         Private Const PURGE_RXABORT As Integer = &H2
         Private Const PURGE_RXCLEAR As Integer = &H8
         Private Const PURGE_TXABORT As Integer = &H1
         Private Const PURGE_TXCLEAR As Integer = &H4
         Private Const GENERIC_READ As Integer = &H80000000
         Private Const GENERIC_WRITE As Integer = &H40000000
         Private Const OPEN_EXISTING As Integer = 3
         Private Const INVALID_HANDLE_VALUE As Integer = -1
         Private Const IO_BUFFER_SIZE As Integer = 1024
         Private Const FILE_FLAG_OVERLAPPED As Int32 = &H40000000
         Private Const ERROR_IO_PENDING As Int32 = 997
         Private Const WAIT_OBJECT_0 As Int32 = 0
         Private Const ERROR_IO_INCOMPLETE As Int32 = 996
         Private Const WAIT_TIMEOUT As Int32 = &H102&
         Private Const INFINITE As Int32 = &HFFFFFFFF

#End Region
#Region "Win32API"
   '// Win32 API
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetCommState(ByVal hCommDev As
IntPtr, ByRef lpDCB As DCB) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function GetCommState(ByVal hCommDev As
IntPtr, ByRef lpDCB As DCB) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True, CharSet:=CharSet.Auto)> Private Shared Function
BuildCommDCB(ByVal lpDef As String, ByRef lpDCB As DCB) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetupComm(ByVal hFile As IntPtr, ByVal
dwInQueue As Int32, ByVal dwOutQueue As Int32) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetCommTimeouts(ByVal hFile As IntPtr,
ByRef lpCommTimeouts As COMMTIMEOUTS) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function GetCommTimeouts(ByVal hFile As IntPtr,
ByRef lpCommTimeouts As COMMTIMEOUTS) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function ClearCommError(ByVal hFile As IntPtr,
ByRef lpErrors As Int32, ByRef lpComStat As COMSTAT) As Int32
   End Function
   <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function PurgeComm(ByVal hFile As IntPtr, ByVal
dwFlags As Int32) As Int32
   End Function
```

```
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function EscapeCommFunction(ByVal hFile As
IntPtr, ByVal ifunc As Int32) As Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function WaitCommEvent(ByVal hFile As IntPtr,
ByRef Mask As EventMasks, ByRef lpOverlap As OVERLAPPED) As Int32
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function WriteFile(ByVal hFile As IntPtr, ByVal
Buffer As Byte(), ByVal nNumberOfBytesToWrite As Integer, ByRef lpNumberOfBytesWritten As Integer, ByRef
lpOverlapped As OVERLAPPED) As Integer
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function ReadFile(ByVal hFile As IntPtr, <Out()>
ByVal Buffer As Byte(), ByVal nNumberOfBytesToRead As Integer, ByRef lpNumberOfBytesRead As Integer, ByRef
lpOverlapped As OVERLAPPED) As Integer
    End Function
    <DllImport("kernel32.dll", SetlastError:=True, CharSet:=CharSet.Auto)> Private Shared Function CreateFile(ByVal
lpFileName As String, ByVal dwDesiredAccess As Integer, ByVal dwShareMode As Integer, ByVal
lpSecurityAttributes As Integer, ByVal dwCreationDisposition As Integer, ByVal dwFlagsAndAttributes As Integer,
ByVal hTemplateFile As Integer) As IntPtr
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function CloseHandle(ByVal hObject As IntPtr) As
Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Public Shared Function GetCommModemStatus(ByVal hFile As
IntPtr, ByRef lpModemStatus As Int32) As Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetEvent(ByVal hEvent As IntPtr) As
Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True, CharSet:=CharSet.Auto)> Private Shared Function CreateEvent(ByVal
lpEventAttributes As IntPtr, ByVal bManualReset As Int32, ByVal bInitialState As Int32, ByVal lpName As String) As
IntPtr
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function WaitForSingleObject(ByVal hHandle As
IntPtr, ByVal dwMilliseconds As Int32) As Int32
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function GetOverlappedResult(ByVal hFile As IntPtr,
ByRef lpOverlapped As OVERLAPPED, ByRef lpNumberOfBytesTransferred As Int32, ByVal bWait As Int32) As
Int32
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetCommMask(ByVal hFile As IntPtr,
ByVal lpEvtMask As Int32) As Int32
    End Function
    <DllImport("kernel32.dll", SetlastError:=True, CharSet:=CharSet.Auto)> Private Shared Function
GetDefaultCommConfig(ByVal lpszName As String, ByRef lpCC As COMMCONFIG, ByRef lpdwSize As Integer) As
Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function SetCommBreak(ByVal hFile As IntPtr) As
Boolean
    End Function
    <DllImport("kernel32.dll", SetlastError:=True)> Private Shared Function ClearCommBreak(ByVal hFile As IntPtr) As
Boolean
    End Function


#End Region
#Region "Events"
    Public Event CommEvent As CommEventHandler
#End Region
#Region "Delegates"
    Public Delegate Sub CommEventHandler(ByVal source As Rs232, ByVal Mask As EventMasks)
#End Region

    Public Property Port() As Integer
```

```
    Get
      Return miPort
    End Get
    Set(ByVal Value As Integer)
      miPort = Value
    End Set
  End Property
  Public Sub PurgeBuffer(ByVal Mode As PurgeBuffers)
    If (mhRS.ToInt32 > 0) Then PurgeComm(mhRS, Mode)
  End Sub
  Public Overridable Property Timeout() As Integer
    Get
      Return miTimeout
    End Get
    Set(ByVal Value As Integer)
      miTimeout = CInt(IIf(Value = 0, 500, Value))
      '// If Port is open updates it on the fly
      pSetTimeout()
    End Set
  End Property
  Public Property Parity() As DataParity
    Get
      Return meParity
    End Get
    Set(ByVal Value As DataParity)
      meParity = Value
    End Set
  End Property
  Public Property StopBit() As DataStopBit
    Get
      Return meStopBit
    End Get
    Set(ByVal Value As DataStopBit)
      meStopBit = Value
    End Set
  End Property
  Public Property BaudRate() As Integer
    Get
      Return miBaudRate
    End Get
    Set(ByVal Value As Integer)
      miBaudRate = Value
    End Set
  End Property
  Public Property DataBit() As Integer
    Get
      Return miDataBit
    End Get
    Set(ByVal Value As Integer)
      miDataBit = Value
    End Set
  End Property
  Public Property BufferSize() As Integer
    Get
      Return miBufferSize
    End Get
    Set(ByVal Value As Integer)
      miBufferSize = Value
    End Set
  End Property
  Public Overloads Sub Open()
    '// Get Dcb block,Update with current data
  Dim uDcb As DCB, iRc As Int32
```

```
    '// Set working mode
    meMode = Mode.Overlapped
  Dim iMode As Int32 = Convert.ToInt32(IIf(meMode = Mode.Overlapped, FILE_FLAG_OVERLAPPED, 0))
    '// Initializes Com Port
    If miPort > 0 Then
      Try
        '// Creates a COM Port stream handle
        mhRS = CreateFile("\\.\COM" & miPort.ToString, GENERIC_READ Or GENERIC_WRITE, 0, 0,
OPEN_EXISTING, iMode, 0)
        If (mhRS.ToInt32 > 0) Then
          '// Clear all comunication errors
        Dim lpErrCode As Int32
          iRc = ClearCommError(mhRS, lpErrCode, New COMSTAT)
          '// Clears I/O buffers
          iRc = PurgeComm(mhRS, PurgeBuffers.RXClear Or PurgeBuffers.TxClear)
          '// Gets COM Settings
          iRc = GetCommState(mhRS, uDcb)
          '// Updates COM Settings
        Dim sParity As String = "NOEM"
          sParity = sParity.Substring(meParity, 1)
          '// Set DCB State
        Dim sDCBState As String = String.Format("baud={0} parity={1} data={2} stop={3}", miBaudRate, sParity,
miDataBit, CInt(meStopBit))
          iRc = BuildCommDCB(sDCBState, uDcb)
          uDcb.Parity = CByte(meParity)
          '// Set Xon/Xoff State
          If mbUseXonXoff Then
            uDcb.Bits1 = 768
          Else
            uDcb.Bits1 = 0
          End If
          iRc = SetCommState(mhRS, uDcb)
          If iRc = 0 Then
          Dim sErrTxt As String = New Win32Exception().Message
            'Throw New CIOChannelException("Unable to set COM state " & sErrTxt)
          End If
          '// Setup Buffers (Rx,Tx)
          iRc = SetupComm(mhRS, miBufferSize, miBufferSize)
          '// Set Timeouts
          pSetTimeout()
          '//Enables events if required
          If mbEnableEvents ThenMe.EnableEvents()
        Else
          '// Raise Initialization problems
        Dim sErrTxt As String = New Win32Exception().Message
          'Throw New CIOChannelException("Unable to open COM" + miPort.ToString + ControlChars.CrLf +
sErrTxt)
        End If
      Catch Ex As Exception
        '// Generica error
        Throw New CIOChannelException(Ex.Message, Ex)
      End Try
    Else
      '// Port not defined, cannot open
      Throw New ApplicationException("COM Port not defined,use Port property to set it before invoking InitPort")
    End If
  End Sub
  Public Overloads Sub Open(ByVal Port As Integer, ByVal BaudRate As Integer, ByVal DataBit As Integer, ByVal
Parity As DataParity, ByVal StopBit As DataStopBit, ByVal BufferSize As Integer)
    Me.Port = Port
    Me.BaudRate = BaudRate
    Me.DataBit = DataBit
    Me.Parity = Parity
```

```
        Me.StopBit = StopBit
        Me.BufferSize = BufferSize
        Open()
    End Sub
    Public Sub Close()
        If mhRS.ToInt32 > 0 Then
            If mbEnableEvents = True Then
                Me.DisableEvents()
            End If
        Dim ret As Boolean = CloseHandle(mhRS)
            If Not ret Then Throw New Win32Exception
            mhRS = New IntPtr(0)
        End If
    End Sub
    ReadOnly Property IsOpen() As Boolean
        Get
            Return CBool(mhRS.ToInt32 > 0)
        End Get
    End Property
    Public Overloads Sub Write(ByVal Buffer As Byte())
    Dim iRc, iBytesWritten As Integer, hOvl As GCHandle
        '-----------------------------------------------------------------
        muOvlW = New Overlapped
        If mhRS.ToInt32 <= 0 Then
            Throw New ApplicationException("Please initialize and open port before using this method")
        Else
            '// Creates Event
            Try
                hOvl = GCHandle.Alloc(muOvlW, GCHandleType.Pinned)
                muOvlW.hEvent = CreateEvent(Nothing, 1, 0, Nothing)
                If muOvlW.hEvent.ToInt32 = 0 Then Throw New ApplicationException("Error creating event for overlapped
writing")
                '// Clears IO buffers and sends data
                iRc = WriteFile(mhRS, Buffer, Buffer.Length, 0, muOvlW)
                If iRc = 0 Then
                    If Marshal.GetLastWin32Error <> ERROR_IO_PENDING Then
                        Throw New ApplicationException("Write command error")
                    Else
                        '// Check Tx results
                        If GetOverlappedResult(mhRS, muOvlW, iBytesWritten, 1) = 0 Then
                            Throw New ApplicationException("Write pending error")
                        Else
                            '// All bytes sent?
                            If iBytesWritten <> Buffer.Length Then Throw New ApplicationException("Write Error - Bytes
Written " & iBytesWritten.ToString & " of " & Buffer.Length.ToString)
                        End If
                    End If
                End If
            Finally
                '//Closes handle
                CloseHandle(muOvlW.hEvent)
                If (hOvl.IsAllocated = True) Then hOvl.Free()
            End Try
        End If
    End Sub
    Public Overloads Sub Write(ByVal Buffer As String)
    Dim oEncoder As New System.Text.ASCIIEncoding
    Dim oEnc As Encoding = oEncoder.GetEncoding(1252)
        '------------------------------------------------------------
    Dim aByte() As Byte = oEnc.GetBytes(Buffer)
        Me.Write(aByte)
    End Sub
    Public Function Read(ByVal Bytes2Read As Integer) As Integer
```

```
Dim iReadChars, iRc As Integer, bReading As Boolean, hOvl As GCHandle
    '-------------------------------------------------------------
    '// If Bytes2Read not specified uses Buffersize
    If Bytes2Read = 0 Then Bytes2Read = miBufferSize
    muOvlR = New Overlapped
    If mhRS.ToInt32 <= 0 Then
        Throw New ApplicationException("Please initialize and open port before using this method")
    Else
        '// Get bytes from port
        Try
            hOvl = GCHandle.Alloc(muOvlR, GCHandleType.Pinned)
            muOvlR.hEvent = CreateEvent(Nothing, 1, 0, Nothing)
            If muOvlR.hEvent.ToInt32 = 0 Then Throw New ApplicationException("Error creating event for overlapped
reading")
            '// Clears IO buffers and reads data
            ReDim mabtRxBuf(Bytes2Read - 1)
            iRc = ReadFile(mhRS, mabtRxBuf, Bytes2Read, iReadChars, muOvlR)
            If iRc = 0 Then
                If Marshal.GetLastWin32Error() <> ERROR_IO_PENDING Then
                    Throw New ApplicationException("Read pending error")
                Else
                    '// Wait for characters
                    iRc = WaitForSingleObject(muOvlR.hEvent, miTimeout)
                    Select Case iRc
                        Case WAIT_OBJECT_0
                            '// Some data received...
                            If GetOverlappedResult(mhRS, muOvlR, iReadChars, 0) = 0 Then
                                Throw New ApplicationException("Read pending error.")
                            Else
                                Return iReadChars
                            End If
                        Case WAIT_TIMEOUT
                            Throw New IOTimeoutException("Read Timeout.")
                        Case Else
                            Throw New ApplicationException("General read error.")
                    End Select
                End If
            Else
                Return (iReadChars)
            End If
        Finally
            '//Closes handle
            CloseHandle(muOvlR.hEvent)
            If (hOvl.IsAllocated) Then hOvl.Free()
        End Try
    End If
End Function
Overridable ReadOnly Property InputStream() As Byte()
    Get
        Return mabtRxBuf
    End Get
End Property
Overridable ReadOnly Property InputStreamString() As String
    Get
    Dim oEncoder As New System.Text.ASCIIEncoding
    Dim oEnc As Encoding = oEncoder.GetEncoding(1252)
        '-------------------------------------------------------------
        If NotMe.InputStream Is Nothing Then Return oEnc.GetString(Me.InputStream)
    End Get
End Property
Public Sub ClearInputBuffer()
    If mhRS.ToInt32 > 0 Then
        PurgeComm(mhRS, PURGE_RXCLEAR)
```

```
        End If
    End Sub
    Public WriteOnly Property Rts() As Boolean
        Set(ByVal Value As Boolean)
            If mhRS.ToInt32 > 0 Then
                If Value Then
                    EscapeCommFunction(mhRS, Lines.SetRts)
                Else
                    EscapeCommFunction(mhRS, Lines.ClearRts)
                End If
            End If
        End Set
    End Property
    Public WriteOnly Property Dtr() As Boolean
        Set(ByVal Value As Boolean)
            If mhRS.ToInt32 > 0 Then
                If Value Then
                    EscapeCommFunction(mhRS, Lines.SetDtr)
                Else
                    EscapeCommFunction(mhRS, Lines.ClearDtr)
                End If
            End If
        End Set
    End Property
    Public ReadOnly Property ModemStatus() As ModemStatusBits
        Get
            If mhRS.ToInt32 <= 0 Then
                Throw New ApplicationException("Please initialize and open port before using this method")
            Else
                '// Retrieve modem status
            Dim lpModemStatus As Int32
                If Not GetCommModemStatus(mhRS, lpModemStatus) Then
                    Throw New ApplicationException("Unable to get modem status")
                Else
                    Return CType(lpModemStatus, ModemStatusBits)
                End If
            End If
        End Get
    End Property
    Public Function CheckLineStatus(ByVal Line As ModemStatusBits) As Boolean
        Return Convert.ToBoolean(ModemStatus And Line)
    End Function
    Public Property UseXonXoff() As Boolean
        Get
            Return mbUseXonXoff
        End Get
        Set(ByVal Value As Boolean)
            mbUseXonXoff = Value
        End Set
    End Property
    Public Sub EnableEvents()
        If mhRS.ToInt32 <= 0 Then
            Throw New ApplicationException("Please initialize and open port before using this method")
        Else
            If moEvents Is Nothing Then
                mbEnableEvents = True
                moEvents = New Thread(AddressOf pEventsWatcher)
                moEvents.IsBackground = True
                moEvents.Start()
            End If
        End If
    End Sub
    Public Sub DisableEvents()
```

```
        If mbEnableEvents = True Then
            SyncLock Me
                mbEnableEvents = False     '// This should kill the thread
            End SyncLock
            '// Let WaitCommEvent exit...
            If muOvlE.hEvent.ToInt32 <> 0 Then SetEvent(muOvlE.hEvent)
            moEvents = Nothing
        End If
    End Sub
    Public Property RxBufferThreshold() As Int32
        Get
            Return miBufThreshold
        End Get
        Set(ByVal Value As Int32)
            miBufThreshold = Value
        End Set
    End Property
    Public Shared Function IsPortAvailable(ByVal portNumber As Int32) As Boolean
        If portNumber <= 0 Then
            Return False
        Else
        Dim cfg As COMMCONFIG
        Dim cfgsize As Int32 = Marshal.SizeOf(cfg)
            cfg.dwSize = cfgsize
        Dim ret As Boolean = GetDefaultCommConfig("COM" + portNumber.ToString, cfg, cfgsize)
            Return ret
        End If
    End Function
    Public Sub SetBreak()
        If mhRS.ToInt32 > 0 Then
            If SetCommBreak(mhRS) = False Then Throw New Win32Exception
        End If
    End Sub
    Public Sub ClearBreak()
        If mhRS.ToInt32 > 0 Then
            If ClearCommBreak(mhRS) = False Then Throw New Win32Exception
        End If

    End Sub
    Public ReadOnly Property InBufferCount() As Int32
        Get
        Dim comStat As COMSTAT
        Dim lpErrCode As Int32
        Dim iRc As Int32
            comStat.cbInQue = 0
            If mhRS.ToInt32 > 0 Then
                iRc = ClearCommError(mhRS, lpErrCode, comStat)
                Return comStat.cbInQue
            End If
            Return 0
        End Get
    End Property


#Region "Finalize"
    Protected Overrides Sub Finalize()
        Try
            If Not mbDisposed Then
                If mbEnableEvents ThenMe.DisableEvents()
                Close()
            End If
        Finally
            MyBase.Finalize()
```

```
      End Try
   End Sub
#End Region

#Region "Private Routines"
   Private Sub pSetTimeout()
   Dim uCtm As COMMTIMEOUTS
      '// Set ComTimeout
      If mhRS.ToInt32 <= 0 Then
         Exit Sub
      Else
         '// Changes setup on the fly
         With uCtm
            .ReadIntervalTimeout = 0
            .ReadTotalTimeoutMultiplier = 0
            .ReadTotalTimeoutConstant = miTimeout
            .WriteTotalTimeoutMultiplier = 10
            .WriteTotalTimeoutConstant = 100
         End With
         SetCommTimeouts(mhRS, uCtm)
      End If
   End Sub
   Private Sub pDispose() Implements IDisposable.Dispose
      If (Not mbDisposed AndAlso (mhRS.ToInt32 > 0)) Then
         '// Closes Com Port releasing resources
         Try
           Me.Close()
         Finally
            mbDisposed = True
            '// Suppress unnecessary Finalize overhead
            GC.SuppressFinalize(Me)
         End Try
      End If


   End Sub
   Private Sub pEventsWatcher()
      '// Events to watch
   Dim lMask As EventMasks = EventMasks.Break Or EventMasks.CarrierDetect Or EventMasks.ClearToSend Or _
      EventMasks.DataSetReady Or EventMasks.Ring Or EventMasks.RxChar Or EventMasks.RXFlag Or _
      EventMasks.StatusError
   Dim lRetMask As EventMasks, iBytesRead, iTotBytes, iErrMask As Int32, iRc As Int32, aBuf As New ArrayList
   Dim uComStat As COMSTAT
      '-----------------------------------
      '// Creates Event
      muOvlE = New Overlapped
   Dim hOvlE As GCHandle = GCHandle.Alloc(muOvlE, GCHandleType.Pinned)
      muOvlE.hEvent = CreateEvent(Nothing, 1, 0, Nothing)
      If muOvlE.hEvent.ToInt32 = 0 Then Throw New ApplicationException("Error creating event for overlapped
reading")
      '// Set mask
      SetCommMask(mhRS, lMask)
      '// Looks for RxChar
      While mbEnableEvents = True
         WaitCommEvent(mhRS, lMask, muOvlE)
         Select Case WaitForSingleObject(muOvlE.hEvent, INFINITE)
            Case WAIT_OBJECT_0
               '// Event (or abort) detected
               If mbEnableEvents = False Then Exit While
               If (lMask And EventMasks.RxChar) > 0 Then
                  '// Read incoming data
                  ClearCommError(mhRS, iErrMask, uComStat)
                  If iErrMask = 0 Then
```

```
                Dim ovl As New Overlapped
                Dim hOvl As GCHandle = GCHandle.Alloc(ovl, GCHandleType.Pinned)
                   ReDim mabtRxBuf(uComStat.cbInQue - 1)
                   If ReadFile(mhRS, mabtRxBuf, uComStat.cbInQue, iBytesRead, ovl) > 0 Then
                      If iBytesRead > 0 Then
                         '// Some bytes read, fills temporary buffer
                         If iTotBytes < miBufThreshold Then
                            aBuf.AddRange(mabtRxBuf)
                            iTotBytes += iBytesRead
                         End If
                         '// Threshold reached?, raises event
                         If iTotBytes >= miBufThreshold Then
                            '//Copies temp buffer into Rx buffer
                            ReDim mabtRxBuf(iTotBytes - 1)
                            aBuf.CopyTo(mabtRxBuf)
                            '// Raises event
                            Try
                              Me.OnCommEventReceived(Me, lMask)
                            Finally
                               iTotBytes = 0
                               aBuf.Clear()
                            End Try
                         End If
                      End If
                   End If
                   If (hOvl.IsAllocated) Then hOvl.Free()
                End If
             Else
                '// Simply raises OnCommEventHandler event
               Me.OnCommEventReceived(Me, lMask)
             End If
          Case Else
          Dim sErr As String = New Win32Exception().Message
             Throw New ApplicationException(sErr)
       End Select
    End While
    '// Release Event Handle
    CloseHandle(muOvlE.hEvent)
    muOvlE.hEvent = IntPtr.Zero
    If (hOvlE.IsAllocated) Then hOvlE.Free()
    muOvlE = Nothing
 End Sub

#End Region

#Region "Protected Routines"
 Protected Sub OnCommEventReceived(ByVal source As Rs232, ByVal mask As EventMasks)
 Dim del As CommEventHandler =Me.CommEventEvent
    If (Not del Is Nothing) Then
    Dim SafeInvoker As ISynchronizeInvoke
       Try
          SafeInvoker = DirectCast(del.Target, ISynchronizeInvoke)
       Catch
       End Try
       If (Not SafeInvoker Is Nothing) Then
          SafeInvoker.Invoke(del, New Object() {source, mask})
       Else
          del.Invoke(source, mask)
       End If
    End If
 End Sub
#End Region
```

```
End Class
#End Region

#Region "Exceptions"
Public Class CIOChannelException : Inherits ApplicationException
    Sub New(ByVal Message As String)
        MyBase.New(Message)
    End Sub
    Sub New(ByVal Message As String, ByVal InnerException As Exception)
        MyBase.New(Message, InnerException)
    End Sub
End Class
Public Class IOTimeoutException : Inherits CIOChannelException

        Sub New(ByVal Message As String)
                MyBase.New(Message)
        End Sub
        Sub New(ByVal Message As String, ByVal InnerException As Exception)
                MyBase.New(Message, InnerException)
        End Sub
End Class

#End Region
```

Form1.vb

```
Imports Microsoft.Win32
Imports System.IO
Imports System.Security.Permissions
Imports System.Math

Public Class Form1
    Inherits System.Windows.Forms.Form
    ' Global Declarations
Dim column%, Row%
Dim Cummulative_Value As Double
Dim Events_Value As Double
Dim Exit_Flag_1 As Integer

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents GroupBox32 As System.Windows.Forms.GroupBox
    Friend WithEvents Label102 As System.Windows.Forms.Label
    Friend WithEvents btnTest As System.Windows.Forms.Button
    Friend WithEvents txtPortNum As System.Windows.Forms.TextBox
    Friend WithEvents txtTimeout As System.Windows.Forms.TextBox
    Friend WithEvents txtBaudrate As System.Windows.Forms.TextBox
    Friend WithEvents Label104 As System.Windows.Forms.Label
    Friend WithEvents Label105 As System.Windows.Forms.Label
    Friend WithEvents txtBytes2Read As System.Windows.Forms.TextBox
    Friend WithEvents txtRx As System.Windows.Forms.TextBox
    Friend WithEvents CheckBox7 As System.Windows.Forms.CheckBox
    Friend WithEvents Button58 As System.Windows.Forms.Button
    Friend WithEvents ListBox11 As System.Windows.Forms.ListBox
    Friend WithEvents Label111 As System.Windows.Forms.Label
    Friend WithEvents Label112 As System.Windows.Forms.Label
    Friend WithEvents TextBox150 As System.Windows.Forms.TextBox
    Friend WithEvents Label109 As System.Windows.Forms.Label
    Friend WithEvents Label108 As System.Windows.Forms.Label
    Friend WithEvents Label107 As System.Windows.Forms.Label
    Friend WithEvents Label106 As System.Windows.Forms.Label
    Friend WithEvents TextBox148 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox144 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox143 As System.Windows.Forms.TextBox
    Friend WithEvents Label110 As System.Windows.Forms.Label
    Friend WithEvents TextBox149 As System.Windows.Forms.TextBox
    Friend WithEvents Label113 As System.Windows.Forms.Label
    Friend WithEvents Label114 As System.Windows.Forms.Label
    Friend WithEvents TextBox153 As System.Windows.Forms.TextBox
    Friend WithEvents AxMSChart1 As AxMSChart20Lib.AxMSChart
    Friend WithEvents AxMSChart2 As AxMSChart20Lib.AxMSChart
    Friend WithEvents Scale_Timer_1 As System.Windows.Forms.Timer

    Private miComPort As Integer
    Private WithEvents moRS232 As Rs232
    Private mlTicks As Long
    Private Delegate Sub CommEventUpdate(ByVal source As Rs232, ByVal mask As Rs232.EventMasks)
    Friend WithEvents ToolTip1 As System.Windows.Forms.ToolTip
    Friend WithEvents ToolTip3 As System.Windows.Forms.ToolTip
    Friend WithEvents Scale_Timer_2 As System.Windows.Forms.Timer
    Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
    Friend WithEvents CheckBox1 As System.Windows.Forms.CheckBox
    Public WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
    Public WithEvents Label103 As System.Windows.Forms.Label
    Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents TextBox4 As System.Windows.Forms.TextBox
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents CheckBox2 As System.Windows.Forms.CheckBox

    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
      Me.components = New System.ComponentModel.Container
    Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form1))
      Me.GroupBox32 = New System.Windows.Forms.GroupBox
      Me.Label102 = New System.Windows.Forms.Label
```

```
Me.btnTest = New System.Windows.Forms.Button
Me.txtPortNum = New System.Windows.Forms.TextBox
Me.txtTimeout = New System.Windows.Forms.TextBox
Me.Label103 = New System.Windows.Forms.Label
Me.txtBaudrate = New System.Windows.Forms.TextBox
Me.Label104 = New System.Windows.Forms.Label
Me.Label105 = New System.Windows.Forms.Label
Me.txtBytes2Read = New System.Windows.Forms.TextBox
Me.txtRx = New System.Windows.Forms.TextBox
Me.CheckBox7 = New System.Windows.Forms.CheckBox
Me.Button58 = New System.Windows.Forms.Button
Me.ListBox11 = New System.Windows.Forms.ListBox
Me.Label111 = New System.Windows.Forms.Label
Me.Label112 = New System.Windows.Forms.Label
Me.TextBox150 = New System.Windows.Forms.TextBox
Me.Label109 = New System.Windows.Forms.Label
Me.Label108 = New System.Windows.Forms.Label
Me.Label107 = New System.Windows.Forms.Label
Me.Label106 = New System.Windows.Forms.Label
Me.TextBox148 = New System.Windows.Forms.TextBox
Me.TextBox144 = New System.Windows.Forms.TextBox
Me.TextBox143 = New System.Windows.Forms.TextBox
Me.Label110 = New System.Windows.Forms.Label
Me.TextBox149 = New System.Windows.Forms.TextBox
Me.Label113 = New System.Windows.Forms.Label
Me.Label114 = New System.Windows.Forms.Label
Me.TextBox153 = New System.Windows.Forms.TextBox
Me.AxMSChart1 = New AxMSChart20Lib.AxMSChart
Me.AxMSChart2 = New AxMSChart20Lib.AxMSChart
Me.Scale_Timer_1 = New System.Windows.Forms.Timer(Me.components)
Me.ToolTip1 = New System.Windows.Forms.ToolTip(Me.components)
Me.ToolTip3 = New System.Windows.Forms.ToolTip(Me.components)
Me.Scale_Timer_2 = New System.Windows.Forms.Timer(Me.components)
Me.MainMenu1 = New System.Windows.Forms.MainMenu
Me.MenuItem1 = New System.Windows.Forms.MenuItem
Me.CheckBox1 = New System.Windows.Forms.CheckBox
Me.Label1 = New System.Windows.Forms.Label
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.TextBox2 = New System.Windows.Forms.TextBox
Me.Label3 = New System.Windows.Forms.Label
Me.Label2 = New System.Windows.Forms.Label
Me.TextBox3 = New System.Windows.Forms.TextBox
Me.Label4 = New System.Windows.Forms.Label
Me.TextBox4 = New System.Windows.Forms.TextBox
Me.Label5 = New System.Windows.Forms.Label
Me.CheckBox2 = New System.Windows.Forms.CheckBox
Me.GroupBox32.SuspendLayout()
 CType(Me.AxMSChart1, System.ComponentModel.ISupportInitialize).BeginInit()
 CType(Me.AxMSChart2, System.ComponentModel.ISupportInitialize).BeginInit()
Me.SuspendLayout()
 '
 'GroupBox32
 '
Me.GroupBox32.Controls.Add(Me.Label102)
Me.GroupBox32.Controls.Add(Me.btnTest)
Me.GroupBox32.Controls.Add(Me.txtPortNum)
Me.GroupBox32.Controls.Add(Me.txtTimeout)
Me.GroupBox32.Controls.Add(Me.Label103)
Me.GroupBox32.Controls.Add(Me.txtBaudrate)
Me.GroupBox32.Controls.Add(Me.Label104)
Me.GroupBox32.Controls.Add(Me.Label105)
Me.GroupBox32.Controls.Add(Me.txtBytes2Read)
Me.GroupBox32.Location = New System.Drawing.Point(16, 16)
```

```
Me.GroupBox32.Name = "GroupBox32"
Me.GroupBox32.Size = New System.Drawing.Size(198, 123)
Me.GroupBox32.TabIndex = 34
Me.GroupBox32.TabStop = False
Me.GroupBox32.Text = "COM Setup"
'
'Label102
'
Me.Label102.Location = New System.Drawing.Point(132, 20)
Me.Label102.Name = "Label102"
Me.Label102.Size = New System.Drawing.Size(58, 14)
Me.Label102.TabIndex = 8
Me.Label102.Text = "Port check"
'
'btnTest
'
Me.btnTest.Location = New System.Drawing.Point(132, 64)
Me.btnTest.Name = "btnTest"
Me.btnTest.Size = New System.Drawing.Size(49, 17)
Me.btnTest.TabIndex = 7
Me.btnTest.Text = "Test"
Me.ToolTip1.SetToolTip(Me.btnTest, "Test port availability")
'
'txtPortNum
'
Me.txtPortNum.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
Me.txtPortNum.Location = New System.Drawing.Point(132, 36)
Me.txtPortNum.Name = "txtPortNum"
Me.txtPortNum.Size = New System.Drawing.Size(49, 20)
Me.txtPortNum.TabIndex = 6
Me.txtPortNum.Text = "3"
Me.ToolTip1.SetToolTip(Me.txtPortNum, "Enter port number")
'
'txtTimeout
'
Me.txtTimeout.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
Me.txtTimeout.Location = New System.Drawing.Point(69, 24)
Me.txtTimeout.Name = "txtTimeout"
Me.txtTimeout.Size = New System.Drawing.Size(49, 20)
Me.txtTimeout.TabIndex = 3
Me.txtTimeout.Text = "1500"
Me.ToolTip3.SetToolTip(Me.txtTimeout, "COM Port timeout in ms")
'
'Label103
'
Me.Label103.Location = New System.Drawing.Point(69, 46)
Me.Label103.Name = "Label103"
Me.Label103.Size = New System.Drawing.Size(82, 14)
Me.Label103.TabIndex = 4
Me.Label103.Text = "BaudRate"
'
'txtBaudrate
'
Me.txtBaudrate.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
Me.txtBaudrate.Location = New System.Drawing.Point(69, 60)
Me.txtBaudrate.Name = "txtBaudrate"
Me.txtBaudrate.Size = New System.Drawing.Size(49, 20)
Me.txtBaudrate.TabIndex = 5
Me.txtBaudrate.Text = "9600"
Me.ToolTip1.SetToolTip(Me.txtBaudrate, "COM Port Baudrate")
'
'Label104
'
```

```
        Me.Label104.Location = New System.Drawing.Point(69, 10)
        Me.Label104.Name = "Label104"
        Me.Label104.Size = New System.Drawing.Size(82, 14)
        Me.Label104.TabIndex = 2
        Me.Label104.Text = "Timeout (ms)"
        '
        'Label105
        '
        Me.Label105.Location = New System.Drawing.Point(16, 93)
        Me.Label105.Name = "Label105"
        Me.Label105.Size = New System.Drawing.Size(82, 14)
        Me.Label105.TabIndex = 11
        Me.Label105.Text = "Bytes to read"
        '
        'txtBytes2Read
        '
        Me.txtBytes2Read.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.txtBytes2Read.Enabled = False
        Me.txtBytes2Read.Location = New System.Drawing.Point(100, 90)
        Me.txtBytes2Read.Name = "txtBytes2Read"
        Me.txtBytes2Read.Size = New System.Drawing.Size(65, 20)
        Me.txtBytes2Read.TabIndex = 12
        Me.txtBytes2Read.Text = "18"
        Me.ToolTip1.SetToolTip(Me.txtBytes2Read, "Bytes to read from COM buffer (this number effects also
CommEvent)")
        '
        'txtRx
        '
        Me.txtRx.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.txtRx.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.txtRx.Location = New System.Drawing.Point(224, 24)
        Me.txtRx.Multiline = True
        Me.txtRx.Name = "txtRx"
        Me.txtRx.Size = New System.Drawing.Size(305, 38)
        Me.txtRx.TabIndex = 35
        Me.txtRx.Text = ""
        '
        'CheckBox7
        '
        Me.CheckBox7.Location = New System.Drawing.Point(544, 32)
        Me.CheckBox7.Name = "CheckBox7"
        Me.CheckBox7.Size = New System.Drawing.Size(104, 16)
        Me.CheckBox7.TabIndex = 52
        Me.CheckBox7.Text = "Enable Scale"
        '
        'Button58
        '
        Me.Button58.Location = New System.Drawing.Point(944, 24)
        Me.Button58.Name = "Button58"
        Me.Button58.Size = New System.Drawing.Size(72, 32)
        Me.Button58.TabIndex = 51
        Me.Button58.Text = "Clear Events List"
        Me.ToolTip1.SetToolTip(Me.Button58, "Test port availability")
        '
        'ListBox11
        '
        Me.ListBox11.Location = New System.Drawing.Point(904, 72)
        Me.ListBox11.Name = "ListBox11"
        Me.ListBox11.Size = New System.Drawing.Size(120, 134)
        Me.ListBox11.TabIndex = 50
        '
        'Label111
```

```
        '
        Me.Label111.Location = New System.Drawing.Point(664, 104)
        Me.Label111.Name = "Label111"
        Me.Label111.Size = New System.Drawing.Size(48, 14)
        Me.Label111.TabIndex = 62
        Me.Label111.Text = "(grams)"
        '
        'Label112
        '
        Me.Label112.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label112.Location = New System.Drawing.Point(592, 80)
        Me.Label112.Name = "Label112"
        Me.Label112.Size = New System.Drawing.Size(104, 14)
        Me.Label112.TabIndex = 61
        Me.Label112.Text = "Weight Difference"
        '
        'TextBox150
        '
        Me.TextBox150.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox150.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox150.Location = New System.Drawing.Point(600, 96)
        Me.TextBox150.Multiline = True
        Me.TextBox150.Name = "TextBox150"
        Me.TextBox150.Size = New System.Drawing.Size(56, 24)
        Me.TextBox150.TabIndex = 60
        Me.TextBox150.Text = "1"
        '
        'Label109
        '
        Me.Label109.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label109.Location = New System.Drawing.Point(360, 80)
        Me.Label109.Name = "Label109"
        Me.Label109.Size = New System.Drawing.Size(104, 14)
        Me.Label109.TabIndex = 59
        Me.Label109.Text = "Second Reading"
        '
        'Label108
        '
        Me.Label108.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label108.Location = New System.Drawing.Point(224, 80)
        Me.Label108.Name = "Label108"
        Me.Label108.Size = New System.Drawing.Size(104, 14)
        Me.Label108.TabIndex = 58
        Me.Label108.Text = "First Reading"
        '
        'Label107
        '
        Me.Label107.Location = New System.Drawing.Point(560, 104)
        Me.Label107.Name = "Label107"
        Me.Label107.Size = New System.Drawing.Size(33, 14)
        Me.Label107.TabIndex = 57
        Me.Label107.Text = "(ms)"
        '
        'Label106
        '
        Me.Label106.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label106.Location = New System.Drawing.Point(488, 80)
        Me.Label106.Name = "Label106"
```

```
        Me.Label106.Size = New System.Drawing.Size(104, 14)
        Me.Label106.TabIndex = 56
        Me.Label106.Text = "Time Difference"
        '
        'TextBox148
        '
        Me.TextBox148.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox148.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox148.Location = New System.Drawing.Point(496, 96)
        Me.TextBox148.Multiline = True
        Me.TextBox148.Name = "TextBox148"
        Me.TextBox148.Size = New System.Drawing.Size(56, 24)
        Me.TextBox148.TabIndex = 55
        Me.TextBox148.Text = "30"
        '
        'TextBox144
        '
        Me.TextBox144.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox144.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox144.Location = New System.Drawing.Point(360, 96)
        Me.TextBox144.Multiline = True
        Me.TextBox144.Name = "TextBox144"
        Me.TextBox144.Size = New System.Drawing.Size(115, 24)
        Me.TextBox144.TabIndex = 54
        Me.TextBox144.Text = "0"
        '
        'TextBox143
        '
        Me.TextBox143.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox143.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox143.Location = New System.Drawing.Point(224, 96)
        Me.TextBox143.Multiline = True
        Me.TextBox143.Name = "TextBox143"
        Me.TextBox143.Size = New System.Drawing.Size(115, 24)
        Me.TextBox143.TabIndex = 53
        Me.TextBox143.Text = "0"
        '
        'Label110
        '
        Me.Label110.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label110.Location = New System.Drawing.Point(312, 128)
        Me.Label110.Name = "Label110"
        Me.Label110.Size = New System.Drawing.Size(104, 14)
        Me.Label110.TabIndex = 67
        Me.Label110.Text = "Difference"
        '
        'TextBox149
        '
        Me.TextBox149.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox149.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox149.Location = New System.Drawing.Point(288, 144)
        Me.TextBox149.Multiline = True
        Me.TextBox149.Name = "TextBox149"
        Me.TextBox149.Size = New System.Drawing.Size(115, 24)
        Me.TextBox149.TabIndex = 66
        Me.TextBox149.Text = "0"
        '
        'Label113
```

```
        '
        Me.Label113.Location = New System.Drawing.Point(488, 152)
        Me.Label113.Name = "Label113"
        Me.Label113.Size = New System.Drawing.Size(48, 14)
        Me.Label113.TabIndex = 70
        Me.Label113.Text = "(grams)"
        '
        'Label114
        '
        Me.Label114.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label114.Location = New System.Drawing.Point(424, 128)
        Me.Label114.Name = "Label114"
        Me.Label114.Size = New System.Drawing.Size(104, 14)
        Me.Label114.TabIndex = 69
        Me.Label114.Text = "Manual Calibration"
        '
        'TextBox153
        '
        Me.TextBox153.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox153.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox153.Location = New System.Drawing.Point(424, 144)
        Me.TextBox153.Multiline = True
        Me.TextBox153.Name = "TextBox153"
        Me.TextBox153.Size = New System.Drawing.Size(56, 24)
        Me.TextBox153.TabIndex = 68
        Me.TextBox153.Text = "0"
        '
        'AxMSChart1
        '
        Me.AxMSChart1.DataSource = Nothing
        Me.AxMSChart1.Location = New System.Drawing.Point(24, 248)
        Me.AxMSChart1.Name = "AxMSChart1"
        Me.AxMSChart1.OcxState = CType(resources.GetObject("AxMSChart1.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.AxMSChart1.Size = New System.Drawing.Size(648, 264)
        Me.AxMSChart1.TabIndex = 71
        '
        'AxMSChart2
        '
        Me.AxMSChart2.DataSource = Nothing
        Me.AxMSChart2.Location = New System.Drawing.Point(24, 528)
        Me.AxMSChart2.Name = "AxMSChart2"
        Me.AxMSChart2.OcxState = CType(resources.GetObject("AxMSChart2.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.AxMSChart2.Size = New System.Drawing.Size(648, 264)
        Me.AxMSChart2.TabIndex = 72
        '
        'Scale_Timer_1
        '
        Me.Scale_Timer_1.Interval = 1
        '
        'Scale_Timer_2
        '
        Me.Scale_Timer_2.Enabled = True
        Me.Scale_Timer_2.Interval = 1
        '
        'MainMenu1
        '
        Me.MainMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem1})
        '
        'MenuItem1
```

```
'
Me.MenuItem1.Index = 0
Me.MenuItem1.Text = "Exit"
'
'CheckBox1
'
Me.CheckBox1.Location = New System.Drawing.Point(48, 264)
Me.CheckBox1.Name = "CheckBox1"
Me.CheckBox1.Size = New System.Drawing.Size(104, 16)
Me.CheckBox1.TabIndex = 73
Me.CheckBox1.Text = "Show Graphs"
'
'Label1
'
Me.Label1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(32, 152)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(120, 16)
Me.Label1.TabIndex = 75
Me.Label1.Text = "Temporary Directory:"
'
'TextBox1
'
Me.TextBox1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox1.Location = New System.Drawing.Point(160, 152)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(64, 20)
Me.TextBox1.TabIndex = 74
Me.TextBox1.Text = "d:/temp/"
'
'TextBox2
'
Me.TextBox2.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
Me.TextBox2.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox2.Location = New System.Drawing.Point(552, 144)
Me.TextBox2.Multiline = True
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.Size = New System.Drawing.Size(40, 24)
Me.TextBox2.TabIndex = 76
Me.TextBox2.Text = "0"
'
'Label3
'
Me.Label3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label3.Location = New System.Drawing.Point(544, 128)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(80, 14)
Me.Label3.TabIndex = 78
Me.Label3.Text = "Reading Num."
'
'Label2
'
Me.Label2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label2.Location = New System.Drawing.Point(624, 128)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(152, 14)
Me.Label2.TabIndex = 80
Me.Label2.Text = "Robot Termination Threshold"
```

```
        '
        'TextBox3
        '
        Me.TextBox3.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox3.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox3.Location = New System.Drawing.Point(648, 144)
        Me.TextBox3.Multiline = True
        Me.TextBox3.Name = "TextBox3"
        Me.TextBox3.Size = New System.Drawing.Size(56, 24)
        Me.TextBox3.TabIndex = 79
        Me.TextBox3.Text = "160"
        '
        'Label4
        '
        Me.Label4.Location = New System.Drawing.Point(712, 152)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(48, 14)
        Me.Label4.TabIndex = 81
        Me.Label4.Text = "(grams)"
        '
        'TextBox4
        '
        Me.TextBox4.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.TextBox4.Font = New System.Drawing.Font("Tahoma", 14.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.TextBox4.Location = New System.Drawing.Point(304, 192)
        Me.TextBox4.Multiline = True
        Me.TextBox4.Name = "TextBox4"
        Me.TextBox4.Size = New System.Drawing.Size(64, 24)
        Me.TextBox4.TabIndex = 82
        Me.TextBox4.Text = "0"
        '
        'Label5
        '
        Me.Label5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label5.Location = New System.Drawing.Point(304, 176)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(72, 14)
        Me.Label5.TabIndex = 83
        Me.Label5.Text = "Timer (sec.)"
        '
        'CheckBox2
        '
        Me.CheckBox2.Location = New System.Drawing.Point(648, 32)
        Me.CheckBox2.Name = "CheckBox2"
        Me.CheckBox2.Size = New System.Drawing.Size(104, 16)
        Me.CheckBox2.TabIndex = 84
        Me.CheckBox2.Text = "Force Scale"
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(800, 793)
        Me.Controls.Add(Me.CheckBox2)
        Me.Controls.Add(Me.TextBox4)
        Me.Controls.Add(Me.TextBox3)
        Me.Controls.Add(Me.TextBox2)
        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.TextBox153)
        Me.Controls.Add(Me.TextBox149)
        Me.Controls.Add(Me.TextBox150)
```

```
        Me.Controls.Add(Me.TextBox148)
        Me.Controls.Add(Me.TextBox144)
        Me.Controls.Add(Me.TextBox143)
        Me.Controls.Add(Me.txtRx)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.CheckBox1)
        Me.Controls.Add(Me.AxMSChart2)
        Me.Controls.Add(Me.AxMSChart1)
        Me.Controls.Add(Me.Label113)
        Me.Controls.Add(Me.Label114)
        Me.Controls.Add(Me.Label110)
        Me.Controls.Add(Me.Label111)
        Me.Controls.Add(Me.Label112)
        Me.Controls.Add(Me.Label109)
        Me.Controls.Add(Me.Label108)
        Me.Controls.Add(Me.Label107)
        Me.Controls.Add(Me.Label106)
        Me.Controls.Add(Me.CheckBox7)
        Me.Controls.Add(Me.Button58)
        Me.Controls.Add(Me.ListBox11)
        Me.Controls.Add(Me.GroupBox32)
        Me.Menu =Me.MainMenu1
        Me.Name = "Form1"
        Me.Text = "Digital_Scale"
        Me.GroupBox32.ResumeLayout(False)
         CType(Me.AxMSChart1, System.ComponentModel.ISupportInitialize).EndInit()
         CType(Me.AxMSChart2, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub btnTest_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnTest.Click
        Try
            If Rs232.IsPortAvailable(Int32.Parse(txtPortNum.Text)) Then
                MessageBox.Show("Port available", "Port test", MessageBoxButtons.OK, MessageBoxIcon.Information)
            Else
                MessageBox.Show("Port NOT available", "Port test", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        Catch ex As Exception
            MessageBox.Show("Port test failed", "Port test", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    Private Sub Button58_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button58.Click
        ListBox11.Items.Clear()
    End Sub

    Private Sub Form1_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs)
Handles MyBase.Closing
        If Not moRS232 Is Nothing Then
            '// Disables Events if active
            moRS232.DisableEvents()
            If moRS232.IsOpen Then moRS232.Close()
        End If
    End Sub
```

```
    Private Sub Scale_Timer_1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Scale_Timer_1.Tick

    Dim String_1 As String
    Dim Measure_1 As Double

        Try
            moRS232.Read(Int32.Parse(txtBytes2Read.Text))
            txtRx.Text = moRS232.InputStreamString

        Dim aBytes As Byte() = moRS232.InputStream
        Dim iPnt As Int32

        Catch Ex As Exception

            txtRx.Text = "Error occurred " & Ex.Message & "  data fetched: " & moRS232.InputStreamString
        End Try
        String_1 = Mid(txtRx.Text, 8, 18)

        TextBox143.Text = Val(String_1).ToString

        System.Threading.Thread.Sleep(Val(TextBox148.Text))

        Try
            moRS232.Read(Int32.Parse(txtBytes2Read.Text))
            txtRx.Text = moRS232.InputStreamString

        Dim aBytes As Byte() = moRS232.InputStream
        Dim iPnt As Int32

        Catch Ex As Exception

            txtRx.Text = "Error occurred " & Ex.Message & "  data fetched: " & moRS232.InputStreamString
        End Try
        String_1 = Mid(txtRx.Text, 8, 18)

        TextBox144.Text = Val(String_1).ToString
        TextBox149.Text = (Val(TextBox144.Text) - Val(TextBox143.Text)).ToString
        TextBox143.Text = (Val(TextBox143.Text) + Val(TextBox153.Text)).ToString

        If Val(TextBox149.Text) < Val(TextBox150.Text) Then
            TextBox149.Text = "0"
        End If

        If Val(TextBox143.Text) < Val(TextBox150.Text) Then
            TextBox143.Text = "0"
        End If

        If Val(TextBox144.Text) < Val(TextBox150.Text) Then
            TextBox144.Text = "0"
        End If

        If CheckBox1.Checked = True Then
            Plot_Graph_Events()
            Plot_Graph_Cummulative()
        End If

        If Val(TextBox149.Text) > Val(TextBox150.Text) Then
            ListBox11.Items.Add("Event!")
        End If

        If Val(TextBox143.Text) > Val(TextBox3.Text) Then
        Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
```

```
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Stop_Robot_Flag", "1")
    End If

End Sub

Function Plot_Graph_Events()

    With AxMSChart2

        .Data = Val(TextBox149.Text)

        .chartType = AxMSChart2.chartType.VtChChartType2dLine
        .ColumnCount = 1
        .RowCount = 100

        If .Row > 99 Then
            .Row = 1
        End If

        If .Row <> 1 And Row <> 100 Then
            .Row = .Row + 1
            .Data = Val(TextBox149.Text)

        Else
            Initial_Plot_Graph_Events()
            .Row = 1
            .Data = Val(TextBox149.Text)
            .Row = .Row + 1
        End If

    End With

Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
    pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
    pRegKey_Events.SetValue("Events_Value", (Round(Val(TextBox149.Text), 2)).ToString)

End Function

Function Plot_Graph_Cummulative()
    With AxMSChart1
        .Data = Val(TextBox143.Text) ' - Val(TextBox153.Text)
        .chartType = AxMSChart1.chartType.VtChChartType2dLine
        .ColumnCount = 1
        .RowCount = 100
        If .Row > 99 Then
            .Row = 1
        End If
        If .Row <> 1 And Row <> 100 Then
            .Row = .Row + 1
            If Val(TextBox143.Text) >= 0 Then
                .Data = Val(TextBox143.Text) '- Val(TextBox153.Text)
            Else
                .Data = 0
            End If
        Else
            Initial_Plot_Graph_Cummulative()
            .Row = 1
            If Val(TextBox143.Text) >= 0 Then
                .Data = Val(TextBox143.Text) '- Val(TextBox153.Text)
                .Row = .Row + 1
            Else
                .Data = 0
```

```
            End If
        End If
        TextBox2.Text = (.Row).ToString
    End With

Dim pRegKey_Cummulative As RegistryKey = Registry.CurrentUser
    pRegKey_Cummulative = pRegKey_Cummulative.OpenSubKey("Uri\Digital_Scale", True)

    If Val(TextBox143.Text) >= 5 Then
        pRegKey_Cummulative.SetValue("Cummulative_Value", (Round(Val(TextBox143.Text), 2)).ToString)
    Else
        pRegKey_Cummulative.SetValue("Cummulative_Value", ("0").ToString)
    End If


End Function

Function Initial_Plot_Graph_Events()
Dim i As Integer
    With AxMSChart2
        .chartType = AxMSChart2.chartType.VtChChartType2dLine
        .ColumnCount = 1
        .RowCount = 100

        For i = 1 To 100
            .Row = i
            .Data = 0
        Next
        .Repaint = True
    End With
End Function

Function Initial_Plot_Graph_Cummulative()
Dim i As Integer
    With AxMSChart1
        .chartType = AxMSChart1.chartType.VtChChartType2dLine
        .ColumnCount = 1
        .RowCount = 100
        For i = 1 To 100
            .Row = i
            .Data = 0
        Next
        .Repaint = True
    End With
End Function

Private Sub Write2File(ByVal msg As String, ByVal filePath As String)
Dim fs As FileStream = New FileStream(filePath, FileMode.Append, FileAccess.Write)
Dim sw As StreamWriter = New StreamWriter(fs)
    sw.WriteLine(msg)
    sw.Flush()
    sw.Close()
    fs.Close()
End Sub

Function Open_Scale_RS232_Communication()
    miComPort = 3
    moRS232 = New Rs232
    Try
        With moRS232
            .Port = miComPort
            .BaudRate = Int32.Parse(txtBaudrate.Text)
            .DataBit = 7
```

```
            .StopBit = Rs232.DataStopBit.StopBit_1

            .Parity = Rs232.DataParity.Parity_Odd
            .Timeout = Int32.Parse(txtTimeout.Text)
        End With
        moRS232.Open()
    Catch Ex As Exception
        MessageBox.Show(Ex.Message, "Connection Error", MessageBoxButtons.OK)
    Finally
    End Try
End Function

Function Close_Scale_RS232_Communication()
    If Not moRS232 Is Nothing Then
        '// Disables Events if active
        moRS232.DisableEvents()
        If moRS232.IsOpen Then moRS232.Close()
    End If
End Function

Private Sub CheckBox7_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox7.CheckedChanged
    If CheckBox7.Checked = True Then
        Open_Scale_RS232_Communication()
        Scale_Timer_1.Enabled = True
    Else
        Scale_Timer_1.Enabled = False
        moRS232.DisableEvents()
        Close_Scale_RS232_Communication()
        Initial_Plot_Graph_Cummulative()
        Initial_Plot_Graph_Events()
    End If
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    TextBox4.Text = "0"
    txtRx.Text = ""
    CheckBox1.Checked = True
    Exit_Flag_1 = 0
    Close_Scale_RS232_Communication()
    Initial_Plot_Graph_Events()
    Initial_Plot_Graph_Cummulative()

Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
    pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
    pRegKey_Events.SetValue("Activate_Scale_Flag", "0")
    pRegKey_Events.SetValue("Events_Value", "0")
    pRegKey_Events.SetValue("Cummulative_Value", "0")
    pRegKey_Events.SetValue("Stop_Robot_Flag", "0")

End Sub

Private Sub Scale_Timer_2_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Scale_Timer_2.Tick

Dim pRegKey As RegistryKey = Registry.CurrentUser
    pRegKey = pRegKey.OpenSubKey("Uri\Digital_Scale")
Dim val1 As Object = pRegKey.GetValue("Activate_Scale_Flag")
Dim Time_Value As Object = pRegKey.GetValue("Time_Value")
Dim Trial_Number As Object = pRegKey.GetValue("Trial_Number")

    If Val(val1) = 0 Then
```

```
            'System.Threading.Thread.Sleep(2000)
            TextBox143.Text = "0"
            TextBox144.Text = "0"
            TextBox153.Text = "0"
            TextBox149.Text = "0"
            TextBox2.Text = "0"
            txtRx.Text = ""
            'System.Threading.Thread.Sleep(50)
            CheckBox7.Checked = False
        Else
            'Write2File((Val(Time_Value)).ToString + ", " + TextBox143.Text + ", " + TextBox149.Text, TextBox1.Text +
(Val(Trial_Number) - 1).ToString + "_Trial_" + "Scale_Output.csv")
            CheckBox7.Checked = True
        End If

        If Exit_Flag_1 = 1 Then
            Scale_Timer_1.Enabled = False
            Close_Scale_RS232_Communication()
            Initial_Plot_Graph_Cummulative()
            Initial_Plot_Graph_Events()
            Scale_Timer_2.Enabled = False
            CheckBox7.Checked = False
        Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
            pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
            pRegKey_Events.SetValue("Events_Value", "0")
            pRegKey_Events.SetValue("Cummulative_Value", "0")
            pRegKey_Events.SetValue("Time_Value", "0")
            pRegKey_Events.SetValue("Activate_Scale_Flag", "0")
            Close()
        End If
    End Sub

    Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MenuItem1.Click
        Exit_Flag_1 = 1
    End Sub

    Private Sub CheckBox2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox2.CheckedChanged
    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        If CheckBox2.Checked = True Then
            Open_Scale_RS232_Communication()
            Scale_Timer_1.Enabled = True
        Else
            Scale_Timer_1.Enabled = False
            moRS232.DisableEvents()
            Close_Scale_RS232_Communication()
            Initial_Plot_Graph_Cummulative()
            Initial_Plot_Graph_Events()
        End If
    End Sub
End Class
```

peakdetect.m

```
function [pospeakind,negpeakind]=peakdetect(signal)
%  PEAKDETECT peak detection
%
%  [pospeakind,negpeakind]=peakdetect(signal)
%
%  The positive and negative polarity (concave down and up) peak index vectors are
%  generated from the signal vector and graphically displayed.  Positive and negative
```

```
%  polarity peaks occur at points of positive to negative and negative to positive
%  slope adjacency, respectively.  The typically rare contingencies of peaks
%  occurring at the lagging edges of constant intervals are supported.  Complex
%  signals are modified to the modulus of the elements.  If unspecified, the signal
%  vector is entered after the prompt from the keyboard.
%  Implemented using MATLAB 6.0.0
%
%  Examples:
%
%  » [p,n]=peakdetect([-1 -1 0 1 0 1 0 -1 -1])
%
%  p =
%
%       4     6
%
%  n =
%
%       1     5     8
%
%  » [p,n]=peakdetect(cos(2*pi*(0:999999)/500000))
%
%  p =
%
%            1      500001     1000000
%
%  n =
%
%        250001      750001
%
%  Copyright (c) 2001
%  Tom McMurray
%  mcmurray@teamcmi.com
%  if signal is not input, enter signal or return for empty outputs

if ~nargin
   signal=input('enter signal vector or return for empty outputs\n');
   if isempty(signal)
      pospeakind=[];
      negpeakind=[];
      return
   end
end
sizsig=size(signal);

%  while signal is unsupported, enter supported signal or return for empty outputs

while isempty(signal)|~isnumeric(signal)|~all(all(isfinite(signal)))...
      |length(sizsig)>2|min(sizsig)~=1
   signal=input(['signal is empty, nonnumeric, nonfinite, or nonvector:\nenter '...
         'finite vector or return for empty outputs\n']);
   if isempty(signal)
      pospeakind=[];
      negpeakind=[];
      return
   end
   sizsig=size(signal);
end

%  if signal is complex, modify to modulus of the elements

if ~isreal(signal)
   signal=abs(signal);
end
```

```
% if signal is constant, return empty outputs

if ~any(signal-signal(1))
   pospeakind=[];
   negpeakind=[];
   disp('constant signal graph suppressed')
   return
end
sizsig1=sizsig(1);
lensig=sizsig1;

% if signal is a row vector, modify to a column vector

if lensig==1
   signal=signal(:);
   lensig=sizsig(2);
end
lensig1=lensig-1;
lensig2=lensig1-1;

% if signal length is 2, return max/min as positive/negative polarity peaks

if ~lensig2
   [sig,pospeakind]=max(signal);
   [sig,negpeakind]=min(signal);
   disp('2 element signal graph suppressed')
   return
end

% generate difference signal

difsig=diff(signal);

% generate vectors corresponding to positive slope indices

dsgt0=difsig>0;
dsgt00=dsgt0(1:lensig2);
dsgt01=dsgt0(2:lensig1);

% generate vectors corresponding to negative slope indices

dslt0=difsig<0;
dslt00=dslt0(1:lensig2);
dslt01=dslt0(2:lensig1);

% generate vectors corresponding to constant intervals

dseq0=difsig==0;
dseq01=dseq0(2:lensig1);
clear difsig

% positive to negative slope adjacencies define positive polarity peaks

pospeakind=find(dsgt00&dslt01)+1;

% negative to positive slope adjacencies define negative polarity peaks

negpeakind=find(dsgt01&dslt00)+1;

% positive slope to constant interval adjacencies initiate positive polarity peaks

peakind=find(dsgt00&dseq01)+1;
```

```
lenpeakind=length(peakind);

%  determine positive polarity peak terminations
for k=1:lenpeakind
  peakindk=peakind(k);
  l=peakindk+1;

%  if end constant interval occurs, positive polarity peak exists

  if l==lensig
    pospeakind=[pospeakind;peakindk];

%  else l<lensig, determine next nonzero slope index

  else
    dseq0l=dseq0(l);
    while dseq0l&l<lensig1
      l=l+1;
      dseq0l=dseq0(l);
    end

%  if negative slope or end constant interval occurs, positive polarity peaks exist

    if dslt0(l)|dseq0l;
      pospeakind=[pospeakind;peakindk];
    end
  end
end

%  negative slope to constant interval adjacencies initiate negative polarity peaks

peakind=find(dslt00&dseq01)+1;
lenpeakind=length(peakind);
clear dseq01

%  determine negative polarity peak terminations

for k=1:lenpeakind
  peakindk=peakind(k);
  l=peakindk+1;

%  if end constant interval occurs, negative polarity peak exists

  if l==lensig
    negpeakind=[negpeakind;peakindk];

%  else l<lensig, determine next nonzero slope index

  else
    dseq0l=dseq0(l);
    while dseq0l&l<lensig1
      l=l+1;
      dseq0l=dseq0(l);
    end

%  if positive slope or end constant interval occurs, negative polarity peaks exist

    if dsgt0(l)|dseq0l;
      negpeakind=[negpeakind;peakindk];
    end
  end
end
clear dsgt0 peakind
```

```
% if initial negative slope occurs, initial positive polarity peak exists

if dslt00(1)
  pospeakind=[1;pospeakind];

% elseif initial positive slope occurs, initial negative polarity peak exists

elseif dsgt00(1)
  negpeakind=[1;negpeakind];

% else initial constant interval occurs, determine next nonzero slope index

else
  k=2;
  dseq0k=dseq0(2);
  while dseq0k
    k=k+1;
    dseq0k=dseq0(k);
  end

% if negative slope occurs, initial positive polarity peak exists

  if dslt0(k)
    pospeakind=[1;pospeakind];

% else positive slope occurs, initial negative polarity peak exists

  else
    negpeakind=[1;negpeakind];
  end
end
clear dsgt00 dslt0 dslt00 dseq0

% if final positive slope occurs, final positive polarity peak exists

if dsgt01(lensig2)
  pospeakind=[pospeakind;lensig];

% elseif final negative slope occurs, final negative polarity peak exists

elseif dslt01(lensig2)
  negpeakind=[negpeakind;lensig];
end
clear dsgt01 dslt01

% if peak indices are not ascending, order peak indices

if any(diff(pospeakind)<0)
  pospeakind=sort(pospeakind);
end
if any(diff(negpeakind)<0)
  negpeakind=sort(negpeakind);
end

% if signal is a row vector, modify peak indices to row vectors

if sizsig1==1
  pospeakind=pospeakind.';
  negpeakind=negpeakind.';
end

% plot signal peaks
```

```
plot(0:lensig1,signal,pospeakind-1,signal(pospeakind),'b^',negpeakind-1,...
  signal(negpeakind),'bv')
xlabel('Sample')
ylabel('Signal')
grid
```

Learning System

cMatLib.vb

```
Option Strict Off
Option Explicit On

Imports System.Math

Public Class MatLib
   Private Shared Sub Find_R_C(ByVal Mat(,) As Double, ByRef Row As Integer, ByRef Col As Integer)
      Row = Mat.GetUpperBound(0) 'D:\Ph.D\Source_Codes\MotoCom\Communicate_with_XRC.vb
      Col = Mat.GetUpperBound(1)
   End Sub

#Region "Add Matrices"
   Public Shared Function Add(ByVal Mat1(,) As Double, ByVal Mat2(,) As Double) As Double(,)
   Dim sol(,) As Double
   Dim i, j As Integer
   Dim Rows1, Cols1 As Integer
   Dim Rows2, Cols2 As Integer

      On Error GoTo Error_Handler

      Find_R_C(Mat1, Rows1, Cols1)
      Find_R_C(Mat2, Rows2, Cols2)

      If Rows1 <> Rows2 Or Cols1 <> Cols2 Then
         GoTo Error_Dimension
      End If

      ReDim sol(Rows1, Cols1)
      For i = 0 To Rows1
         For j = 0 To Cols1
            sol(i, j) = Mat1(i, j) + Mat2(i, j)
         Next j
      Next i

      Return sol

Error_Dimension:
      Err.Raise("5005", , "Dimensions of the two matrices do not match !")

Error_Handler:
      If Err.Number = 5005 Then
         Err.Raise("5005", , "Dimensions of the two matrices do not match !")
      Else
         Err.Raise("5022", , "One or both of the matrices are null, this operation cannot be done !!")
      End If

   End Function
#End Region

#Region "Subtract Matrices"
   Public Shared Function Subtract(ByVal Mat1(,) As Double, ByVal Mat2(,) As Double) As Double(,)
   Dim i, j As Integer
```

```
    Dim sol(,) As Double
    Dim Rows1, Cols1 As Integer
    Dim Rows2, Cols2 As Integer

        On Error GoTo Error_Handler

        Find_R_C(Mat1, Rows1, Cols1)
        Find_R_C(Mat2, Rows2, Cols2)

        If Rows1 <> Rows2 Or Cols1 <> Cols2 Then
            GoTo Error_Dimension
        End If

        ReDim sol(Rows1, Cols1)

        For i = 0 To Rows1
            For j = 0 To Cols1
                sol(i, j) = Mat1(i, j) - Mat2(i, j)
            Next j
        Next i

        Return sol

Error_Dimension:
        Err.Raise("5007", , "Dimensions of the two matrices do not match !")

Error_Handler:
        If Err.Number = 5007 Then
            Err.Raise("5007", , "Dimensions of the two matrices do not match !")
        Else
            Err.Raise("5022", , "One or both of the matrices are null, this operation cannot be done !!")
        End If

    End Function

#End Region

#Region "Multiply Matrices"
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Multiply two matrices, their dimensions should be compatible!
    ' Function returns the solution or errors due to
    ' dimensions incompatibility
    ' Example:
    '  Check Main Form !!
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function Multiply(ByVal Mat1(,) As Double, ByVal Mat2(,) As Double) As Double(,)
    Dim l, i, j As Integer
    Dim OptiString As String
    Dim sol(,) As Double, MulAdd As Double
    Dim Rows1, Cols1 As Integer
    Dim Rows2, Cols2 As Integer

        On Error GoTo Error_Handler

        MulAdd = 0

        Find_R_C(Mat1, Rows1, Cols1)
        Find_R_C(Mat2, Rows2, Cols2)

        If Cols1 <> Rows2 Then
            GoTo Error_Dimension
        End If
```

```
        ReDim sol(Rows1, Cols2)

        For i = 0 To Rows1
            For j = 0 To Cols2
                For l = 0 To Cols1
                    MulAdd = MulAdd + Mat1(i, l) * Mat2(l, j)
                Next l
                sol(i, j) = MulAdd
                MulAdd = 0
            Next j
        Next i

        Return sol

Error_Dimension:
        Err.Raise("5009", , "Dimensions of the two matrices not suitable for multiplication !")

Error_Handler:
        If Err.Number = 5009 Then
            Err.Raise("5009", , "Dimensions of the two matrices not suitable for multiplication !")
        Else
            Err.Raise("5022", , "One or both of the matrices are null, this operation cannot be done !!")
        End If

    End Function

#End Region

#Region "Determinant of a Matrix"
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Determinant of a matrix should be (nxn)
    ' Function returns the solution or errors due to
    ' dimensions incompatibility
    ' Example:
    '  Check Main Form !!
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function Det(ByVal Mat(,) As Double) As Double
    Dim DArray(,) As Double, S As Integer
    Dim k, k1, i, j As Integer
    Dim save, ArrayK As Double
    Dim M1 As String
    Dim Rows, Cols As Integer

        On Error GoTo Error_Handler

        Find_R_C(Mat, Rows, Cols)

        If Rows <> Cols Then GoTo Error_Dimension

        S = Rows
        Det = 1
        DArray = Mat.Clone()

        For k = 0 To S
            If DArray(k, k) = 0 Then
                j = k
                Do While ((j < S) And (DArray(k, j) = 0))
                    j = j + 1
                Loop
                If DArray(k, j) = 0 Then
                    Det = 0
                    Exit Function
                Else
```

```
                For i = k To S
                    save = DArray(i, j)
                    DArray(i, j) = DArray(i, k)
                    DArray(i, k) = save
                Next i
            End If

            Det = -Det
        End If
        ArrayK = DArray(k, k)
        Det = Det * ArrayK
        If k < S Then
            k1 = k + 1
            For i = k1 To S
                For j = k1 To S
                    DArray(i, j) = DArray(i, j) - DArray(i, k) * (DArray(k, j) / ArrayK)
                Next j
            Next i
        End If
    Next

    Exit Function

Error_Dimension:
    Err.Raise("5011", , "Matrix should be a square matrix !")

Error_Handler:
    If Err.Number = 5011 Then
        Err.Raise("5011", , "Matrix should be a square matrix !")
    Else
        Err.Raise("5022", , "In order to do this operation values must be assigned to the matrix !!")
    End If
  End Function

#End Region

#Region "Inverse of a Matrix"
  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  ' Inverse of a matrix, should be (nxn) and det(Mat)<>0
  ' Function returns the solution or errors due to
  ' dimensions incompatibility
  ' Example:
  '  Check Main Form !!
  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  Public Shared Function Inv(ByVal Mat(,) As Double) As Double(,)
  Dim AI(,) As Double, AIN As Double, AF As Double, _
        Mat1(,) As Double
  Dim LL As Integer, LLM As Integer, L1 As Integer, _
        L2 As Integer, LC As Integer, LCA As Integer, _
        LCB As Integer, i As Integer, j As Integer
  Dim Rows, Cols As Integer

    On Error GoTo Error_Handler

    Find_R_C(Mat, Rows, Cols)
    If Rows <> Cols Then GoTo Error_Dimension

    If Det(Mat) = 0 Then GoTo Error_Zero

    LL = Rows
    LLM = Cols
    Mat1 = Mat.Clone()
    ReDim AI(LL, LL)
```

```
        For L2 = 0 To LL
            For L1 = 0 To LL
                AI(L1, L2) = 0
            Next
            AI(L2, L2) = 1
        Next

        For LC = 0 To LL
            If Abs(Mat1(LC, LC)) < 0.0000000001 Then
                For LCA = LC + 1 To LL
                    If LCA = LC Then GoTo 1090
                    If Abs(Mat1(LC, LCA)) > 0.0000000001 Then
                        For LCB = 0 To LL
                            Mat1(LCB, LC) = Mat1(LCB, LC) + Mat1(LCB, LCA)
                            AI(LCB, LC) = AI(LCB, LC) + AI(LCB, LCA)
                        Next
                        GoTo 1100
                    End If
1090:           Next
            End If

1100:
            AIN = 1 / Mat1(LC, LC)
            For LCA = 0 To LL
                Mat1(LCA, LC) = AIN * Mat1(LCA, LC)
                AI(LCA, LC) = AIN * AI(LCA, LC)
            Next

            For LCA = 0 To LL
                If LCA = LC Then GoTo 1150
                AF = Mat1(LC, LCA)
                For LCB = 0 To LL
                    Mat1(LCB, LCA) = Mat1(LCB, LCA) - AF * Mat1(LCB, LC)
                    AI(LCB, LCA) = AI(LCB, LCA) - AF * AI(LCB, LC)
                Next
1150:       Next

        Next

        Return AI

Error_Zero:
        Err.Raise("5012", , "Determinent equals zero, inverse can't be found !")

Error_Dimension:
        Err.Raise("5014", , "Matrix should be a square matrix !")

Error_Handler:
        If Err.Number = 5012 Then
            Err.Raise("5012", , "Determinent equals zero, inverse can't be found !")
        ElseIf Err.Number = 5014 Then
            Err.Raise("5014", , "Matrix should be a square matrix !")
        End If

    End Function

#End Region

#Region "Multiply Vectors"
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Multiply two vectors, dimensions should be (3x1)
    ' Function returns the solution or errors due to
```

```
    ' dimensions incompatibility
    ' Example:
    '  Check Main Form !!
    '''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function MultiplyVectors(ByVal Mat1(,) As Double, ByVal Mat2(,) As Double) As Double(,)
    Dim i, j, k As Double
    Dim sol(2, 0) As Double
    Dim Rows1, Cols1 As Integer
    Dim Rows2, Cols2 As Integer

        On Error GoTo Error_Handler

        Find_R_C(Mat1, Rows1, Cols1)
        Find_R_C(Mat2, Rows2, Cols2)

        If Rows1 <> 2 Or Cols1 <> 0 Then
            GoTo Error_Dimension
        End If

        If Rows2 <> 2 Or Cols2 <> 0 Then
            GoTo Error_Dimension
        End If

        i = Mat1(1, 0) * Mat2(2, 0) - Mat1(2, 0) * Mat2(1, 0)
        j = Mat1(2, 0) * Mat2(0, 0) - Mat1(0, 0) * Mat2(2, 0)
        k = Mat1(0, 0) * Mat2(1, 0) - Mat1(1, 0) * Mat2(0, 0)

        sol(0, 0) = i : sol(1, 0) = j : sol(2, 0) = k

        Return sol

Error_Dimension:
        Err.Raise("5016", , "Dimension should be (2 x 0) for both matrices in order to do cross multiplication !")

Error_Handler:

        If Err.Number = 5016 Then
            Err.Raise("5016", , "Dimension should be (2 x 0) for both matrices in order to do cross multiplication !")
        Else
            Err.Raise("5022", , "One or both of the matrices are null, this operation cannot be done !!")
        End If

    End Function

#End Region

#Region "Magnitude of a Vector"

    '''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Magnitude of a Vector, vector should be (3x1)
    ' Function returns the solution or errors due to
    ' dimensions incompatibility
    ' Example:
    '  Check Main Form !!
    '''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function VectorMagnitude(ByVal Mat(,) As Double) As Double

    Dim Rows, Cols As Integer

        On Error GoTo Error_Handler

        Find_R_C(Mat, Rows, Cols)
```

```
        If Rows <> 2 Or Cols <> 0 Then
            GoTo Error_Dimension
        End If

        Return Sqrt(Mat(0, 0) * Mat(0, 0) + Mat(1, 0) * Mat(1, 0) + Mat(2, 0) * Mat(2, 0))

Error_Dimension:
        Err.Raise("5018", , "Dimension of the matrix should be (2 x 0) in order to find the vector's norm !")

Error_Handler:
        If Err.Number = 5018 Then
            Err.Raise("5018", , "Dimension of the matrix should be (2 x 0) in order to find the vector's magnitude !")
        Else
            Err.Raise("5022", , "In order to do this operation values must be assigned to the matrix !!")
        End If

    End Function
#End Region

#Region "Transpose of a Matrix"
    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Transpose of a matrix
    ' Function returns the solution or errors
    ' Example:
    '  Check Main Form !!
    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function Transpose(ByVal Mat(,) As Double) As Double(,)
    Dim Tr_Mat(,) As Double
    Dim i, j, Rows, Cols As Integer

        On Error GoTo Error_Handler

        Find_R_C(Mat, Rows, Cols)

        ReDim Tr_Mat(Cols, Rows)

        For i = 0 To Cols
            For j = 0 To Rows
                Tr_Mat(j, i) = Mat(i, j)
            Next j
        Next i

        Return Tr_Mat

Error_Handler:
        Err.Raise("5028", , "In order to do this operation values must be assigned to the matrix !!")

    End Function
#End Region

#Region "Multiply a matrix or a vector with a scalar quantity"

    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Multiply a matrix or a vector with a scalar quantity
    ' Function returns the solution or errors
    ' Example:
    '  Check Main Form !!
    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function ScalarMultiply(ByVal Value As Double, ByVal Mat(,) As Double) As Double(,)
    Dim i, j, Rows, Cols As Integer
    Dim sol(,) As Double

        On Error GoTo Error_Handler
```

```
        Find_R_C(Mat, Rows, Cols)
        ReDim sol(Rows, Cols)

        For i = 0 To Rows
            For j = 0 To Cols
                sol(i, j) = Mat(i, j) * Value
            Next j
        Next i

        Return (sol)

Error_Handler:
        Err.Raise("5022", , "Matrix was not assigned")
    End Function

#End Region

#Region "Divide a matrix or a vector with a scalar quantity"
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Divide matrix elements or a vector by a scalar quantity
    ' Function returns the solution or errors
    ' Example:
    '  Check Main Form !!
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function ScalarDivide(ByVal Value As Double, ByVal Mat(,) As Double) As Double(,)
    Dim i, j, Rows, Cols As Integer
    Dim sol(,) As Double

        On Error GoTo Error_Handler

        Find_R_C(Mat, Rows, Cols)
        ReDim sol(Rows, Cols)

        For i = 0 To Rows
            For j = 0 To Cols
                sol(i, j) = Mat(i, j) / Value
            Next j
        Next i

        Return sol

        Exit Function

Error_Handler:
        Err.Raise("5022", , "Matrix was not assigned")
    End Function

#End Region


#Region "Print Matrix"

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Print a matrix to multitext text box
    ' Function returns the solution or errors
    ' Example:
    '  Check Main Form !!
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public Shared Function PrintMat(ByVal Mat(,) As Double) As String
    Dim N_Rows As Integer, N_Columns, k As Integer, _
         i As Integer, j As Integer, m As Integer
    Dim StrElem As String, StrLen As Long, _
```

```
        Greatest() As Integer, LarString As String
    Dim OptiString As String, sol As String

      Find_R_C(Mat, N_Rows, N_Columns)

      sol = ""
      OptiString = ""

      ReDim Greatest(N_Columns)

      For i = 0 To N_Rows
        For j = 0 To N_Columns
          If i = 0 Then
            Greatest(j) = 0
            For m = 0 To N_Rows
              StrElem = Format$(Mat(m, j), "0.00")
              StrLen = Len(StrElem)
              If Greatest(j) < StrLen Then
                Greatest(j) = StrLen
                LarString = StrElem
              End If
            Next m
            If Mid$(LarString, 1, 1) = "-" Then Greatest(j) = Greatest(j) + 1
          End If
          StrElem = Format$(Mat(i, j), "0.00")
          If Mid$(StrElem, 1, 1) = "-" Then
            StrLen = Len(StrElem)
            If Greatest(j) >= StrLen Then
              For k = 1 To (Greatest(j) - StrLen)
                OptiString = OptiString & "  "
              Next k
              OptiString = OptiString & " "
            End If
          Else
            StrLen = Len(StrElem)
            If Greatest(j) > StrLen Then
              For k = 1 To (Greatest(j) - StrLen)
                OptiString = OptiString & "  "
              Next k
            End If
          End If
          OptiString = OptiString & "  " & Format$(Mat(i, j), "0.00")
        Next j
        If i <> N_Rows Then
          sol = sol & OptiString & vbCrLf
          OptiString = ""
        End If
        sol = sol & OptiString
        OptiString = ""
      Next i

      PrintMat = sol

      Exit Function
    End Function
#End Region
End Class
```

sqlConn.vb

```
Public Class sqlConn
#Region "Class Members"
    Friend WithEvents OLEConn As New System.Data.OleDb.OleDbConnection
```

```
    Friend WithEvents OLEComm As New System.Data.OleDb.OleDbCommand

    Private sqlString As String
    Private err As System.Exception

    Public Shared dataReturned As New ArrayList
#End Region

#Region "class properties"

    Public Property db() As String
        Get
            db = "Shaking_Policies_1.mdb"
        End Get
        Set(ByVal Value As String)
            Value = db
        End Set
    End Property

    Public Property xOLE() As String
        Get
            xOLE = "Provider=Microsoft.Jet.OLEDB.4.0;Data source="
        End Get
        Set(ByVal Value As String)
            Value = xOLE
        End Set
    End Property

#End Region

#Region "class methods"

    Sub New()
    End Sub

    Function connectMe(ByVal sqlString) As Boolean
        Try
            OLEConn.ConnectionString = xOLE & db
            OLEConn.Open()
            OLEComm.CommandText = sqlString
            Return True
        Catch err As System.Exception
            MsgBox(err.Message)
            Return False
        End Try
    End Function

    Function getData(ByVal column1 As String) As ArrayList
        Try

            OLEComm.Connection = OLEConn

            getData = New ArrayList

        Dim d As OleDb.OleDbDataReader = OLEComm.ExecuteReader()
            Do While d.Read
                getData.Add(d(column1.ToString))
            Loop

            'Returns array collection
            dataReturned = getData

            Try
```

```
            OLEConn.Close()
         Catch err As System.Exception
            MsgBox(err.Message)
         End Try
      Catch err As System.Exception
         MsgBox(err.Message)
      End Try
   End Function
#End Region
End Class
```

Learning_System.vb

```
Option Strict Off
Option Explicit On
Imports Microsoft.Win32
Imports System.IO
Imports System.Security.Permissions
Imports System.Math
Imports System.Data.SqlClient
Imports System.Data.OleDb

Public Class Form1
   Inherits System.Windows.Forms.Form
Dim nCid As Integer

#Region "Windows Form Designer generated code "
   Public Sub New()
      MyBase.New()
      If m_vb6FormDefInstance Is Nothing Then
         If m_InitializingDefInstance Then
            m_vb6FormDefInstance = Me
         Else
            Try
               'For the start-up form, the first instance created is the default instance.
               If System.Reflection.Assembly.GetExecutingAssembly.EntryPoint.DeclaringType IsMe.GetType Then
                  m_vb6FormDefInstance = Me
               End If
            Catch
            End Try
         End If
      End If
      'This call is required by the Windows Form Designer.
      InitializeComponent()
   End Sub
   'Form overrides dispose to clean up the component list.
   Protected Overloads Overrides Sub Dispose(ByVal Disposing As Boolean)
      If Disposing Then
         If Not components Is Nothing Then
            components.Dispose()
         End If
      End If
      MyBase.Dispose(Disposing)
   End Sub

 'Required by the Windows Form Designer
 Private components As System.ComponentModel.IContainer
 Public WithEvents CmdDownLoad As System.Windows.Forms.Button
 'NOTE: The following procedure is required by the Windows Form Designer
 'It can be modified using the Windows Form Designer.
 'Do not modify it using the code editor.
 Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
 Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
```

```
Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
Public WithEvents Button5 As System.Windows.Forms.Button
Public WithEvents Button6 As System.Windows.Forms.Button
Friend WithEvents CheckBox1 As System.Windows.Forms.CheckBox
Friend WithEvents CheckBox2 As System.Windows.Forms.CheckBox
Public WithEvents Button7 As System.Windows.Forms.Button
Friend WithEvents TextBox4 As System.Windows.Forms.TextBox
Public WithEvents Button9 As System.Windows.Forms.Button
Friend WithEvents Label13 As System.Windows.Forms.Label
Friend WithEvents Label14 As System.Windows.Forms.Label
Friend WithEvents Label15 As System.Windows.Forms.Label
Public WithEvents Button10 As System.Windows.Forms.Button
Public WithEvents Button11 As System.Windows.Forms.Button
Public WithEvents Button12 As System.Windows.Forms.Button
Public WithEvents Button13 As System.Windows.Forms.Button
Public WithEvents Button14 As System.Windows.Forms.Button
Public WithEvents Button15 As System.Windows.Forms.Button
Public WithEvents Button16 As System.Windows.Forms.Button
Public WithEvents Button17 As System.Windows.Forms.Button
Public WithEvents Button18 As System.Windows.Forms.Button
Public WithEvents Button19 As System.Windows.Forms.Button
Public WithEvents Button20 As System.Windows.Forms.Button
Public WithEvents Button8 As System.Windows.Forms.Button
Public WithEvents Button21 As System.Windows.Forms.Button
Public WithEvents Button22 As System.Windows.Forms.Button
Public WithEvents Button23 As System.Windows.Forms.Button
Public WithEvents Button24 As System.Windows.Forms.Button
Public WithEvents Button25 As System.Windows.Forms.Button
Friend WithEvents TextBox6 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox6 As System.Windows.Forms.GroupBox
Friend WithEvents Label19 As System.Windows.Forms.Label
Friend WithEvents TextBox8 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox7 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox8 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox9 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox10 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox11 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox12 As System.Windows.Forms.GroupBox
Friend WithEvents Label21 As System.Windows.Forms.Label
Friend WithEvents Label20 As System.Windows.Forms.Label
Friend WithEvents Label22 As System.Windows.Forms.Label
Friend WithEvents TextBox9 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox13 As System.Windows.Forms.GroupBox
Friend WithEvents CheckBox4 As System.Windows.Forms.CheckBox
Friend WithEvents Label17 As System.Windows.Forms.Label
Friend WithEvents CheckBox5 As System.Windows.Forms.CheckBox
Friend WithEvents Label23 As System.Windows.Forms.Label
Friend WithEvents Label24 As System.Windows.Forms.Label
Friend WithEvents TextBox10 As System.Windows.Forms.TextBox
Friend WithEvents TabControl1 As System.Windows.Forms.TabControl
Friend WithEvents TabPage2 As System.Windows.Forms.TabPage
Friend WithEvents TabPage1 As System.Windows.Forms.TabPage
Friend WithEvents GroupBox5 As System.Windows.Forms.GroupBox
Friend WithEvents AxWebBrowser1 As AxSHDocVw.AxWebBrowser
Friend WithEvents AxWebBrowser2 As AxSHDocVw.AxWebBrowser
Friend WithEvents TabPage3 As System.Windows.Forms.TabPage
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu
Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
Friend WithEvents TextBox12 As System.Windows.Forms.TextBox
```

```
Friend WithEvents TextBox13 As System.Windows.Forms.TextBox
Friend WithEvents TextBox14 As System.Windows.Forms.TextBox
Friend WithEvents TextBox15 As System.Windows.Forms.TextBox
Friend WithEvents TextBox16 As System.Windows.Forms.TextBox
Friend WithEvents TextBox17 As System.Windows.Forms.TextBox
Friend WithEvents TextBox18 As System.Windows.Forms.TextBox
Friend WithEvents TextBox19 As System.Windows.Forms.TextBox
Friend WithEvents TextBox20 As System.Windows.Forms.TextBox
Friend WithEvents TextBox21 As System.Windows.Forms.TextBox
Friend WithEvents TextBox22 As System.Windows.Forms.TextBox
Friend WithEvents TextBox23 As System.Windows.Forms.TextBox
Friend WithEvents TextBox24 As System.Windows.Forms.TextBox
Friend WithEvents TextBox25 As System.Windows.Forms.TextBox
Friend WithEvents TextBox26 As System.Windows.Forms.TextBox
Friend WithEvents TextBox27 As System.Windows.Forms.TextBox
Friend WithEvents TextBox28 As System.Windows.Forms.TextBox
Friend WithEvents TextBox29 As System.Windows.Forms.TextBox
Friend WithEvents TextBox30 As System.Windows.Forms.TextBox
Friend WithEvents TextBox31 As System.Windows.Forms.TextBox
Friend WithEvents TextBox32 As System.Windows.Forms.TextBox
Friend WithEvents TextBox33 As System.Windows.Forms.TextBox
Friend WithEvents TextBox34 As System.Windows.Forms.TextBox
Friend WithEvents TextBox35 As System.Windows.Forms.TextBox
Friend WithEvents TextBox7 As System.Windows.Forms.TextBox
Friend WithEvents TextBox11 As System.Windows.Forms.TextBox
Friend WithEvents Label18 As System.Windows.Forms.Label
Friend WithEvents Label25 As System.Windows.Forms.Label
Friend WithEvents CheckBox3 As System.Windows.Forms.CheckBox
Friend WithEvents Label26 As System.Windows.Forms.Label
Friend WithEvents Label27 As System.Windows.Forms.Label
Friend WithEvents CheckBox6 As System.Windows.Forms.CheckBox
Friend WithEvents TextBox36 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox14 As System.Windows.Forms.GroupBox
Friend WithEvents TextBox37 As System.Windows.Forms.TextBox
Friend WithEvents Label30 As System.Windows.Forms.Label
Friend WithEvents Label31 As System.Windows.Forms.Label
Friend WithEvents TextBox38 As System.Windows.Forms.TextBox
Friend WithEvents Label32 As System.Windows.Forms.Label
Friend WithEvents Label33 As System.Windows.Forms.Label
Friend WithEvents TextBox39 As System.Windows.Forms.TextBox
Friend WithEvents Label34 As System.Windows.Forms.Label
Friend WithEvents TabPage4 As System.Windows.Forms.TabPage
Friend WithEvents TextBox40 As System.Windows.Forms.TextBox
Friend WithEvents TextBox41 As System.Windows.Forms.TextBox
Friend WithEvents TextBox42 As System.Windows.Forms.TextBox
Friend WithEvents TextBox43 As System.Windows.Forms.TextBox
Friend WithEvents TextBox44 As System.Windows.Forms.TextBox
Friend WithEvents TextBox45 As System.Windows.Forms.TextBox
Friend WithEvents TextBox46 As System.Windows.Forms.TextBox
Friend WithEvents TextBox47 As System.Windows.Forms.TextBox
Friend WithEvents TextBox48 As System.Windows.Forms.TextBox
Friend WithEvents ComboBox1 As System.Windows.Forms.ComboBox
Public WithEvents Button26 As System.Windows.Forms.Button
Friend WithEvents TextBox50 As System.Windows.Forms.TextBox
Friend WithEvents TextBox51 As System.Windows.Forms.TextBox
Friend WithEvents TextBox52 As System.Windows.Forms.TextBox
Friend WithEvents TextBox53 As System.Windows.Forms.TextBox
Friend WithEvents TextBox54 As System.Windows.Forms.TextBox
Friend WithEvents TextBox55 As System.Windows.Forms.TextBox
Friend WithEvents TextBox56 As System.Windows.Forms.TextBox
Friend WithEvents TextBox57 As System.Windows.Forms.TextBox
Friend WithEvents TextBox58 As System.Windows.Forms.TextBox
Friend WithEvents TextBox59 As System.Windows.Forms.TextBox
```

```
Friend WithEvents TextBox60 As System.Windows.Forms.TextBox
Friend WithEvents TextBox61 As System.Windows.Forms.TextBox
Friend WithEvents TextBox62 As System.Windows.Forms.TextBox
Friend WithEvents TextBox63 As System.Windows.Forms.TextBox
Friend WithEvents TextBox64 As System.Windows.Forms.TextBox
Friend WithEvents TextBox65 As System.Windows.Forms.TextBox
Friend WithEvents TextBox66 As System.Windows.Forms.TextBox
Friend WithEvents TextBox67 As System.Windows.Forms.TextBox
Friend WithEvents TextBox68 As System.Windows.Forms.TextBox
Friend WithEvents TextBox69 As System.Windows.Forms.TextBox
Friend WithEvents TextBox70 As System.Windows.Forms.TextBox
Friend WithEvents TextBox71 As System.Windows.Forms.TextBox
Friend WithEvents TextBox72 As System.Windows.Forms.TextBox
Friend WithEvents TextBox73 As System.Windows.Forms.TextBox
Friend WithEvents TextBox74 As System.Windows.Forms.TextBox
Friend WithEvents Label28 As System.Windows.Forms.Label
Friend WithEvents TabPage5 As System.Windows.Forms.TabPage
Friend WithEvents ListBox1 As System.Windows.Forms.ListBox
Friend WithEvents Label40 As System.Windows.Forms.Label
Friend WithEvents Label41 As System.Windows.Forms.Label
Friend WithEvents ListBox2 As System.Windows.Forms.ListBox
Friend WithEvents TextBox78 As System.Windows.Forms.TextBox
Friend WithEvents TextBox79 As System.Windows.Forms.TextBox
Friend WithEvents TextBox82 As System.Windows.Forms.TextBox
Friend WithEvents TextBox83 As System.Windows.Forms.TextBox
Friend WithEvents TextBox84 As System.Windows.Forms.TextBox
Friend WithEvents TextBox85 As System.Windows.Forms.TextBox
Friend WithEvents TextBox86 As System.Windows.Forms.TextBox
Friend WithEvents TextBox87 As System.Windows.Forms.TextBox
Friend WithEvents TextBox88 As System.Windows.Forms.TextBox
Friend WithEvents TextBox96 As System.Windows.Forms.TextBox
Friend WithEvents TextBox97 As System.Windows.Forms.TextBox
Friend WithEvents TextBox98 As System.Windows.Forms.TextBox
Friend WithEvents TextBox99 As System.Windows.Forms.TextBox
Friend WithEvents TextBox100 As System.Windows.Forms.TextBox
Friend WithEvents TextBox101 As System.Windows.Forms.TextBox
Friend WithEvents TextBox102 As System.Windows.Forms.TextBox
Friend WithEvents TextBox103 As System.Windows.Forms.TextBox
Friend WithEvents TextBox104 As System.Windows.Forms.TextBox
Friend WithEvents TextBox105 As System.Windows.Forms.TextBox
Friend WithEvents TextBox106 As System.Windows.Forms.TextBox
Friend WithEvents TextBox107 As System.Windows.Forms.TextBox
Friend WithEvents TextBox108 As System.Windows.Forms.TextBox
Friend WithEvents TextBox109 As System.Windows.Forms.TextBox
Friend WithEvents TextBox110 As System.Windows.Forms.TextBox
Friend WithEvents TextBox111 As System.Windows.Forms.TextBox
Public WithEvents Label11 As System.Windows.Forms.Label
Friend WithEvents TextBox91 As System.Windows.Forms.TextBox
Friend WithEvents Label57 As System.Windows.Forms.Label
Friend WithEvents TextBox92 As System.Windows.Forms.TextBox
Friend WithEvents TextBox93 As System.Windows.Forms.TextBox
Friend WithEvents TextBox94 As System.Windows.Forms.TextBox
Friend WithEvents Button28 As System.Windows.Forms.Button
Public WithEvents Button29 As System.Windows.Forms.Button
Friend WithEvents Button31 As System.Windows.Forms.Button
Friend WithEvents Button32 As System.Windows.Forms.Button
Friend WithEvents TextBox95 As System.Windows.Forms.TextBox
Friend WithEvents TextBox115 As System.Windows.Forms.TextBox
Friend WithEvents TrackBar1 As System.Windows.Forms.TrackBar
Friend WithEvents Label58 As System.Windows.Forms.Label
Friend WithEvents Label59 As System.Windows.Forms.Label
Friend WithEvents GroupBox15 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox16 As System.Windows.Forms.GroupBox
```

```
Friend WithEvents Label60 As System.Windows.Forms.Label
Friend WithEvents TextBox116 As System.Windows.Forms.TextBox
Friend WithEvents Button30 As System.Windows.Forms.Button
Friend WithEvents TabPage7 As System.Windows.Forms.TabPage
Friend WithEvents TextBox117 As System.Windows.Forms.TextBox
Friend WithEvents TextBox118 As System.Windows.Forms.TextBox
Friend WithEvents TextBox119 As System.Windows.Forms.TextBox
Friend WithEvents Label63 As System.Windows.Forms.Label
Friend WithEvents Label64 As System.Windows.Forms.Label
Friend WithEvents TextBox120 As System.Windows.Forms.TextBox
Friend WithEvents Label65 As System.Windows.Forms.Label
Friend WithEvents TextBox121 As System.Windows.Forms.TextBox
Friend WithEvents Label66 As System.Windows.Forms.Label
Friend WithEvents TextBox122 As System.Windows.Forms.TextBox
Friend WithEvents Label67 As System.Windows.Forms.Label
Friend WithEvents TextBox123 As System.Windows.Forms.TextBox
Friend WithEvents Button33 As System.Windows.Forms.Button
Friend WithEvents ListBox3 As System.Windows.Forms.ListBox
Friend WithEvents Label62 As System.Windows.Forms.Label
Friend WithEvents Label61 As System.Windows.Forms.Label
Friend WithEvents Label68 As System.Windows.Forms.Label
Friend WithEvents Button34 As System.Windows.Forms.Button
Friend WithEvents Label69 As System.Windows.Forms.Label
Friend WithEvents TextBox124 As System.Windows.Forms.TextBox
Friend WithEvents ListBox4 As System.Windows.Forms.ListBox
Friend WithEvents Label70 As System.Windows.Forms.Label
Friend WithEvents ListBox5 As System.Windows.Forms.ListBox
Friend WithEvents Label71 As System.Windows.Forms.Label
Friend WithEvents Button35 As System.Windows.Forms.Button
Friend WithEvents TabPage8 As System.Windows.Forms.TabPage
Friend WithEvents Label72 As System.Windows.Forms.Label
Friend WithEvents Label73 As System.Windows.Forms.Label
Friend WithEvents Label74 As System.Windows.Forms.Label
Friend WithEvents ListBox6 As System.Windows.Forms.ListBox
Friend WithEvents Action_Timer_1 As System.Windows.Forms.Timer
Friend WithEvents TextBox146 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox19 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox17 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox21 As System.Windows.Forms.GroupBox
Friend WithEvents Button38 As System.Windows.Forms.Button
Friend WithEvents Button39 As System.Windows.Forms.Button
Friend WithEvents State_Action_Real_Timer1 As System.Windows.Forms.Timer
Friend WithEvents State_Action_Rand_Timer1 As System.Windows.Forms.Timer
Friend WithEvents GroupBox22 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox23 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox25 As System.Windows.Forms.GroupBox
Friend WithEvents ListBox7 As System.Windows.Forms.ListBox
Friend WithEvents Label75 As System.Windows.Forms.Label
Friend WithEvents ListBox8 As System.Windows.Forms.ListBox
Friend WithEvents Label76 As System.Windows.Forms.Label
Friend WithEvents ListBox9 As System.Windows.Forms.ListBox
Friend WithEvents Label77 As System.Windows.Forms.Label
Friend WithEvents ListBox10 As System.Windows.Forms.ListBox
Friend WithEvents Label78 As System.Windows.Forms.Label
Friend WithEvents TextBox125 As System.Windows.Forms.TextBox
Friend WithEvents Label79 As System.Windows.Forms.Label
Friend WithEvents TextBox126 As System.Windows.Forms.TextBox
Friend WithEvents Label80 As System.Windows.Forms.Label
Friend WithEvents TextBox127 As System.Windows.Forms.TextBox
Friend WithEvents Label89 As System.Windows.Forms.Label
Friend WithEvents Button27 As System.Windows.Forms.Button
Friend WithEvents Button37 As System.Windows.Forms.Button
Friend WithEvents Button36 As System.Windows.Forms.Button
```

```
Friend WithEvents GroupBox27 As System.Windows.Forms.GroupBox
Friend WithEvents Label51 As System.Windows.Forms.Label
Friend WithEvents TextBox112 As System.Windows.Forms.TextBox
Friend WithEvents Label52 As System.Windows.Forms.Label
Friend WithEvents TextBox113 As System.Windows.Forms.TextBox
Public WithEvents Button40 As System.Windows.Forms.Button
Friend WithEvents Button45 As System.Windows.Forms.Button
Friend WithEvents GroupBox29 As System.Windows.Forms.GroupBox
Friend WithEvents Button47 As System.Windows.Forms.Button
Friend WithEvents Button49 As System.Windows.Forms.Button
Friend WithEvents Label54 As System.Windows.Forms.Label
Friend WithEvents TextBox132 As System.Windows.Forms.TextBox
Friend WithEvents ProgressBar1 As System.Windows.Forms.ProgressBar
Friend WithEvents Label55 As System.Windows.Forms.Label
Friend WithEvents Label91 As System.Windows.Forms.Label
Friend WithEvents ProgressBar2 As System.Windows.Forms.ProgressBar
Friend WithEvents Button50 As System.Windows.Forms.Button
Friend WithEvents Button51 As System.Windows.Forms.Button
Friend WithEvents Button53 As System.Windows.Forms.Button
Friend WithEvents Button54 As System.Windows.Forms.Button

'Sound play constants
Public Const SND_ASYNC = &H1 ' play asynchronously
Public Const SND_LOOP = &H8 ' loop the sound until next sndPlaySound
Public Const SND_NOSTOP = &H10 ' don't stop any currently playing sound
Public Const SND_NOWAIT = &H2000 ' don't wait if the driver is busy
'Declare function for playing sounds
Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal ByVallpszName As String, ByVal hModule
As Long, ByVal dwFlags As Long) As Long
Friend WithEvents Button55 As System.Windows.Forms.Button
Friend WithEvents Button56 As System.Windows.Forms.Button
Friend WithEvents GroupBox30 As System.Windows.Forms.GroupBox
Friend WithEvents Shaking_Timer_1 As System.Windows.Forms.Timer
Friend WithEvents Label101 As System.Windows.Forms.Label
Friend WithEvents TextBox142 As System.Windows.Forms.TextBox
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents Label8 As System.Windows.Forms.Label
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Timer2 As System.Windows.Forms.Timer
Friend WithEvents ToolTip1 As System.Windows.Forms.ToolTip
Friend WithEvents ToolTip2 As System.Windows.Forms.ToolTip

'Private miComPort As Integer
'Private WithEvents moRS232 As Rs232
'Private mlTicks As Long
'Private Delegate Sub CommEventUpdate(ByVal source As Rs232, ByVal mask As Rs232.EventMasks)
Friend WithEvents Robot_Operating As System.Windows.Forms.Timer
Friend WithEvents TextBox148 As System.Windows.Forms.TextBox
Public WithEvents Label103 As System.Windows.Forms.Label
Friend WithEvents TextBox153 As System.Windows.Forms.TextBox
Friend WithEvents TextBox154 As System.Windows.Forms.TextBox
Friend WithEvents TextBox155 As System.Windows.Forms.TextBox
Friend WithEvents Label104 As System.Windows.Forms.Label
Friend WithEvents Label106 As System.Windows.Forms.Label
Friend WithEvents TextBox81 As System.Windows.Forms.TextBox
Friend WithEvents TextBox156 As System.Windows.Forms.TextBox
Friend WithEvents TextBox157 As System.Windows.Forms.TextBox
Friend WithEvents Label109 As System.Windows.Forms.Label
Friend WithEvents ComboBox6 As System.Windows.Forms.ComboBox
Friend WithEvents TextBox159 As System.Windows.Forms.TextBox
Friend WithEvents TextBox160 As System.Windows.Forms.TextBox
Friend WithEvents Label110 As System.Windows.Forms.Label
```

```
Friend WithEvents TextBox161 As System.Windows.Forms.TextBox
Friend WithEvents Button59 As System.Windows.Forms.Button
Friend WithEvents State_Action_Best_Timer1 As System.Windows.Forms.Timer
Friend WithEvents GroupBox32 As System.Windows.Forms.GroupBox
Friend WithEvents ComboBox7 As System.Windows.Forms.ComboBox
Friend WithEvents AxWebBrowser3 As AxSHDocVw.AxWebBrowser
Friend WithEvents Button4 As System.Windows.Forms.Button
Friend WithEvents Button2 As System.Windows.Forms.Button
Public WithEvents Button3 As System.Windows.Forms.Button
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label6 As System.Windows.Forms.Label
Friend WithEvents Label7 As System.Windows.Forms.Label
Friend WithEvents Label9 As System.Windows.Forms.Label
Friend WithEvents Label10 As System.Windows.Forms.Label
Friend WithEvents Label12 As System.Windows.Forms.Label
Friend WithEvents GroupBox34 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox35 As System.Windows.Forms.GroupBox
Friend WithEvents ComboBox8 As System.Windows.Forms.ComboBox
Friend WithEvents TextBox90 As System.Windows.Forms.TextBox
Friend WithEvents Label113 As System.Windows.Forms.Label
Friend WithEvents Label114 As System.Windows.Forms.Label
Friend WithEvents TextBox136 As System.Windows.Forms.TextBox
Friend WithEvents Label115 As System.Windows.Forms.Label
Friend WithEvents ComboBox9 As System.Windows.Forms.ComboBox
Friend WithEvents Label116 As System.Windows.Forms.Label
Friend WithEvents Label117 As System.Windows.Forms.Label
Friend WithEvents ComboBox10 As System.Windows.Forms.ComboBox
Friend WithEvents TextBox137 As System.Windows.Forms.TextBox
Friend WithEvents Label118 As System.Windows.Forms.Label
Friend WithEvents GroupBox36 As System.Windows.Forms.GroupBox
Friend WithEvents Button60 As System.Windows.Forms.Button
Friend WithEvents Button61 As System.Windows.Forms.Button
Public WithEvents Button62 As System.Windows.Forms.Button
Friend WithEvents Button63 As System.Windows.Forms.Button
Friend WithEvents GroupBox37 As System.Windows.Forms.GroupBox
Friend WithEvents ComboBox11 As System.Windows.Forms.ComboBox
Friend WithEvents Label119 As System.Windows.Forms.Label
Friend WithEvents Label120 As System.Windows.Forms.Label
Friend WithEvents Label121 As System.Windows.Forms.Label
Friend WithEvents Label122 As System.Windows.Forms.Label
Friend WithEvents TextBox138 As System.Windows.Forms.TextBox
Friend WithEvents TextBox139 As System.Windows.Forms.TextBox
Friend WithEvents TextBox140 As System.Windows.Forms.TextBox
Friend WithEvents ComboBox12 As System.Windows.Forms.ComboBox
Friend WithEvents ComboBox13 As System.Windows.Forms.ComboBox
Friend WithEvents Label123 As System.Windows.Forms.Label
Friend WithEvents Label124 As System.Windows.Forms.Label
Friend WithEvents GroupBox18 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox28 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox33 As System.Windows.Forms.GroupBox
Friend WithEvents CheckBox10 As System.Windows.Forms.CheckBox
Friend WithEvents CheckBox7 As System.Windows.Forms.CheckBox
Friend WithEvents CheckBox11 As System.Windows.Forms.CheckBox
Friend WithEvents TextBox151 As System.Windows.Forms.TextBox
Friend WithEvents TextBox150 As System.Windows.Forms.TextBox
Friend WithEvents TextBox143 As System.Windows.Forms.TextBox
Friend WithEvents Button58 As System.Windows.Forms.Button
Friend WithEvents Button48 As System.Windows.Forms.Button
Friend WithEvents GroupBox31 As System.Windows.Forms.GroupBox
Friend WithEvents Button52 As System.Windows.Forms.Button
Friend WithEvents TextBox141 As System.Windows.Forms.TextBox
Friend WithEvents TextBox114 As System.Windows.Forms.TextBox
```

```
Friend WithEvents Button46 As System.Windows.Forms.Button
Friend WithEvents Button57 As System.Windows.Forms.Button
Friend WithEvents GroupBox26 As System.Windows.Forms.GroupBox
Friend WithEvents Label111 As System.Windows.Forms.Label
Friend WithEvents Label112 As System.Windows.Forms.Label
Friend WithEvents TextBox89 As System.Windows.Forms.TextBox
Friend WithEvents Label43 As System.Windows.Forms.Label
Friend WithEvents Label56 As System.Windows.Forms.Label
Friend WithEvents TextBox80 As System.Windows.Forms.TextBox
Friend WithEvents Label49 As System.Windows.Forms.Label
Friend WithEvents Label50 As System.Windows.Forms.Label
Friend WithEvents TextBox131 As System.Windows.Forms.TextBox
Friend WithEvents Label48 As System.Windows.Forms.Label
Friend WithEvents Label47 As System.Windows.Forms.Label
Friend WithEvents TextBox130 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox20 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox24 As System.Windows.Forms.GroupBox
Friend WithEvents Label93 As System.Windows.Forms.Label
Friend WithEvents TextBox134 As System.Windows.Forms.TextBox
Friend WithEvents Label102 As System.Windows.Forms.Label
Friend WithEvents TextBox144 As System.Windows.Forms.TextBox
Public WithEvents Label107 As System.Windows.Forms.Label
Friend WithEvents TextBox158 As System.Windows.Forms.TextBox
Public WithEvents Label108 As System.Windows.Forms.Label
Public WithEvents Label105 As System.Windows.Forms.Label
Friend WithEvents TextBox149 As System.Windows.Forms.TextBox
Public WithEvents Label53 As System.Windows.Forms.Label
Friend WithEvents TextBox129 As System.Windows.Forms.TextBox
Public WithEvents Label44 As System.Windows.Forms.Label
Friend WithEvents CheckBox9 As System.Windows.Forms.CheckBox
Friend WithEvents TextBox135 As System.Windows.Forms.TextBox
Friend WithEvents GroupBox38 As System.Windows.Forms.GroupBox
Friend WithEvents Label82 As System.Windows.Forms.Label
Friend WithEvents Label45 As System.Windows.Forms.Label
Friend WithEvents AxMSChart1 As AxMSChart20Lib.AxMSChart
Friend WithEvents AxMSChart3 As AxMSChart20Lib.AxMSChart
Friend WithEvents Button42 As System.Windows.Forms.Button
Friend WithEvents Label83 As System.Windows.Forms.Label
Friend WithEvents Label84 As System.Windows.Forms.Label
Friend WithEvents Label85 As System.Windows.Forms.Label
Friend WithEvents Label86 As System.Windows.Forms.Label
Friend WithEvents Label87 As System.Windows.Forms.Label
Friend WithEvents Label88 As System.Windows.Forms.Label
Friend WithEvents Label94 As System.Windows.Forms.Label
Friend WithEvents Label100 As System.Windows.Forms.Label
Friend WithEvents Label125 As System.Windows.Forms.Label
Friend WithEvents Label126 As System.Windows.Forms.Label
Friend WithEvents Label127 As System.Windows.Forms.Label
Friend WithEvents Label46 As System.Windows.Forms.Label
Friend WithEvents Label81 As System.Windows.Forms.Label
Friend WithEvents TextBox145 As System.Windows.Forms.TextBox
Friend WithEvents Label128 As System.Windows.Forms.Label
Friend WithEvents Label95 As System.Windows.Forms.Label
Friend WithEvents Label96 As System.Windows.Forms.Label
Friend WithEvents Label97 As System.Windows.Forms.Label
Friend WithEvents Label98 As System.Windows.Forms.Label
Friend WithEvents Label99 As System.Windows.Forms.Label
Friend WithEvents Label129 As System.Windows.Forms.Label
Friend WithEvents Label130 As System.Windows.Forms.Label
Friend WithEvents Label131 As System.Windows.Forms.Label
Friend WithEvents Label132 As System.Windows.Forms.Label
Friend WithEvents Label92 As System.Windows.Forms.Label
Friend WithEvents TextBox133 As System.Windows.Forms.TextBox
```

```
 Friend WithEvents Timer1 As System.Windows.Forms.Timer
 Friend WithEvents Label133 As System.Windows.Forms.Label
 Friend WithEvents TextBox147 As System.Windows.Forms.TextBox
 Friend WithEvents Label134 As System.Windows.Forms.Label
 Friend WithEvents CheckBox8 As System.Windows.Forms.CheckBox
 Friend WithEvents CheckBox12 As System.Windows.Forms.CheckBox
 Friend WithEvents CheckBox13 As System.Windows.Forms.CheckBox
 Friend WithEvents CheckBox14 As System.Windows.Forms.CheckBox
 Friend WithEvents TextBox128 As System.Windows.Forms.TextBox
 Friend WithEvents Label90 As System.Windows.Forms.Label
 Friend WithEvents ComboBox2 As System.Windows.Forms.ComboBox
 Friend WithEvents TextBox5 As System.Windows.Forms.TextBox
 Public WithEvents Label16 As System.Windows.Forms.Label
 Friend WithEvents TextBox49 As System.Windows.Forms.TextBox
 Friend WithEvents GroupBox4 As System.Windows.Forms.GroupBox
 Friend WithEvents Label29 As System.Windows.Forms.Label
 Friend WithEvents Label36 As System.Windows.Forms.Label
 Friend WithEvents ComboBox4 As System.Windows.Forms.ComboBox
 Friend WithEvents Label38 As System.Windows.Forms.Label
 Friend WithEvents ComboBox3 As System.Windows.Forms.ComboBox
 Friend WithEvents ComboBox5 As System.Windows.Forms.ComboBox
 Friend WithEvents Label35 As System.Windows.Forms.Label
 Friend WithEvents Label37 As System.Windows.Forms.Label
 Friend WithEvents ComboBox14 As System.Windows.Forms.ComboBox
 Friend WithEvents ComboBox15 As System.Windows.Forms.ComboBox
 Friend WithEvents ComboBox16 As System.Windows.Forms.ComboBox

<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
Me.components = New System.ComponentModel.Container
 Dim resources As System.Resources.ResourceManager = New System.Resources.ResourceManager(GetType(Form1))
Me.CmdDownLoad = New System.Windows.Forms.Button
Me.Button1 = New System.Windows.Forms.Button
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.TextBox2 = New System.Windows.Forms.TextBox
Me.Label2 = New System.Windows.Forms.Label
Me.Label3 = New System.Windows.Forms.Label
Me.Label4 = New System.Windows.Forms.Label
Me.Label5 = New System.Windows.Forms.Label
Me.Label6 = New System.Windows.Forms.Label
Me.TextBox3 = New System.Windows.Forms.TextBox
Me.Label7 = New System.Windows.Forms.Label
Me.Label8 = New System.Windows.Forms.Label
Me.Label9 = New System.Windows.Forms.Label
Me.Button4 = New System.Windows.Forms.Button
Me.Button5 = New System.Windows.Forms.Button
Me.Button6 = New System.Windows.Forms.Button
Me.CheckBox1 = New System.Windows.Forms.CheckBox
Me.CheckBox2 = New System.Windows.Forms.CheckBox
Me.Button7 = New System.Windows.Forms.Button
Me.TextBox4 = New System.Windows.Forms.TextBox
Me.Button9 = New System.Windows.Forms.Button
Me.Label10 = New System.Windows.Forms.Label
Me.Label11 = New System.Windows.Forms.Label
Me.Label12 = New System.Windows.Forms.Label
Me.Label13 = New System.Windows.Forms.Label
Me.Label14 = New System.Windows.Forms.Label
Me.Label15 = New System.Windows.Forms.Label
Me.Button10 = New System.Windows.Forms.Button
Me.Button2 = New System.Windows.Forms.Button
Me.Button11 = New System.Windows.Forms.Button
Me.Button12 = New System.Windows.Forms.Button
Me.Button13 = New System.Windows.Forms.Button
Me.Button14 = New System.Windows.Forms.Button
```

```
Me.Button15 = New System.Windows.Forms.Button
Me.Button16 = New System.Windows.Forms.Button
Me.Button17 = New System.Windows.Forms.Button
Me.Button18 = New System.Windows.Forms.Button
Me.Button19 = New System.Windows.Forms.Button
Me.Button20 = New System.Windows.Forms.Button
Me.Button8 = New System.Windows.Forms.Button
Me.Button21 = New System.Windows.Forms.Button
Me.Button22 = New System.Windows.Forms.Button
Me.Button23 = New System.Windows.Forms.Button
Me.Button24 = New System.Windows.Forms.Button
Me.Button25 = New System.Windows.Forms.Button
Me.TextBox6 = New System.Windows.Forms.TextBox
Me.GroupBox1 = New System.Windows.Forms.GroupBox
Me.GroupBox2 = New System.Windows.Forms.GroupBox
Me.GroupBox3 = New System.Windows.Forms.GroupBox
Me.GroupBox13 = New System.Windows.Forms.GroupBox
Me.GroupBox11 = New System.Windows.Forms.GroupBox
Me.TextBox11 = New System.Windows.Forms.TextBox
Me.Label18 = New System.Windows.Forms.Label
Me.Label25 = New System.Windows.Forms.Label
Me.CheckBox5 = New System.Windows.Forms.CheckBox
Me.Label17 = New System.Windows.Forms.Label
Me.Label1 = New System.Windows.Forms.Label
Me.CheckBox4 = New System.Windows.Forms.CheckBox
Me.TextBox10 = New System.Windows.Forms.TextBox
Me.Label23 = New System.Windows.Forms.Label
Me.Label24 = New System.Windows.Forms.Label
Me.GroupBox10 = New System.Windows.Forms.GroupBox
Me.GroupBox9 = New System.Windows.Forms.GroupBox
Me.GroupBox8 = New System.Windows.Forms.GroupBox
Me.GroupBox7 = New System.Windows.Forms.GroupBox
Me.GroupBox12 = New System.Windows.Forms.GroupBox
Me.Label20 = New System.Windows.Forms.Label
Me.Label22 = New System.Windows.Forms.Label
Me.TextBox9 = New System.Windows.Forms.TextBox
Me.Label21 = New System.Windows.Forms.Label
Me.Label19 = New System.Windows.Forms.Label
Me.TextBox8 = New System.Windows.Forms.TextBox
Me.GroupBox14 = New System.Windows.Forms.GroupBox
Me.Label28 = New System.Windows.Forms.Label
Me.ComboBox1 = New System.Windows.Forms.ComboBox
Me.Label33 = New System.Windows.Forms.Label
Me.TextBox39 = New System.Windows.Forms.TextBox
Me.Label34 = New System.Windows.Forms.Label
Me.Label32 = New System.Windows.Forms.Label
Me.TextBox38 = New System.Windows.Forms.TextBox
Me.Label31 = New System.Windows.Forms.Label
Me.Label30 = New System.Windows.Forms.Label
Me.TextBox37 = New System.Windows.Forms.TextBox
Me.Button26 = New System.Windows.Forms.Button
Me.CheckBox3 = New System.Windows.Forms.CheckBox
Me.Label26 = New System.Windows.Forms.Label
Me.Label27 = New System.Windows.Forms.Label
Me.CheckBox6 = New System.Windows.Forms.CheckBox
Me.GroupBox6 = New System.Windows.Forms.GroupBox
Me.TabControl1 = New System.Windows.Forms.TabControl
Me.TabPage8 = New System.Windows.Forms.TabPage
Me.CheckBox13 = New System.Windows.Forms.CheckBox
Me.Label92 = New System.Windows.Forms.Label
Me.TextBox133 = New System.Windows.Forms.TextBox
Me.Button42 = New System.Windows.Forms.Button
Me.GroupBox38 = New System.Windows.Forms.GroupBox
```

```
Me.Label94 = New System.Windows.Forms.Label
Me.Label129 = New System.Windows.Forms.Label
Me.Label130 = New System.Windows.Forms.Label
Me.Label131 = New System.Windows.Forms.Label
Me.Label132 = New System.Windows.Forms.Label
Me.Label95 = New System.Windows.Forms.Label
Me.Label96 = New System.Windows.Forms.Label
Me.Label97 = New System.Windows.Forms.Label
Me.Label98 = New System.Windows.Forms.Label
Me.Label99 = New System.Windows.Forms.Label
Me.Label128 = New System.Windows.Forms.Label
Me.TextBox145 = New System.Windows.Forms.TextBox
Me.Label81 = New System.Windows.Forms.Label
Me.Label46 = New System.Windows.Forms.Label
Me.Label127 = New System.Windows.Forms.Label
Me.Label126 = New System.Windows.Forms.Label
Me.Label125 = New System.Windows.Forms.Label
Me.Label100 = New System.Windows.Forms.Label
Me.Label83 = New System.Windows.Forms.Label
Me.Label88 = New System.Windows.Forms.Label
Me.Label87 = New System.Windows.Forms.Label
Me.Label86 = New System.Windows.Forms.Label
Me.Label85 = New System.Windows.Forms.Label
Me.Label84 = New System.Windows.Forms.Label
Me.Label45 = New System.Windows.Forms.Label
Me.Label82 = New System.Windows.Forms.Label
Me.AxMSChart3 = New AxMSChart20Lib.AxMSChart
Me.AxMSChart1 = New AxMSChart20Lib.AxMSChart
Me.GroupBox31 = New System.Windows.Forms.GroupBox
Me.Button52 = New System.Windows.Forms.Button
Me.TextBox141 = New System.Windows.Forms.TextBox
Me.TextBox114 = New System.Windows.Forms.TextBox
Me.Button57 = New System.Windows.Forms.Button
Me.GroupBox18 = New System.Windows.Forms.GroupBox
Me.TextBox146 = New System.Windows.Forms.TextBox
Me.GroupBox29 = New System.Windows.Forms.GroupBox
Me.ComboBox2 = New System.Windows.Forms.ComboBox
Me.Label109 = New System.Windows.Forms.Label
Me.ComboBox6 = New System.Windows.Forms.ComboBox
Me.Button50 = New System.Windows.Forms.Button
Me.Button51 = New System.Windows.Forms.Button
Me.ProgressBar2 = New System.Windows.Forms.ProgressBar
Me.Label91 = New System.Windows.Forms.Label
Me.Label55 = New System.Windows.Forms.Label
Me.ProgressBar1 = New System.Windows.Forms.ProgressBar
Me.Button45 = New System.Windows.Forms.Button
Me.Button40 = New System.Windows.Forms.Button
Me.Button47 = New System.Windows.Forms.Button
Me.Button54 = New System.Windows.Forms.Button
Me.Button53 = New System.Windows.Forms.Button
Me.Button56 = New System.Windows.Forms.Button
Me.GroupBox30 = New System.Windows.Forms.GroupBox
Me.Label101 = New System.Windows.Forms.Label
Me.TextBox142 = New System.Windows.Forms.TextBox
Me.Label54 = New System.Windows.Forms.Label
Me.TextBox132 = New System.Windows.Forms.TextBox
Me.GroupBox19 = New System.Windows.Forms.GroupBox
Me.GroupBox20 = New System.Windows.Forms.GroupBox
Me.TextBox135 = New System.Windows.Forms.TextBox
Me.GroupBox34 = New System.Windows.Forms.GroupBox
Me.GroupBox4 = New System.Windows.Forms.GroupBox
Me.Label37 = New System.Windows.Forms.Label
Me.ComboBox14 = New System.Windows.Forms.ComboBox
```

```
Me.ComboBox15 = New System.Windows.Forms.ComboBox
Me.ComboBox16 = New System.Windows.Forms.ComboBox
Me.Label35 = New System.Windows.Forms.Label
Me.ComboBox5 = New System.Windows.Forms.ComboBox
Me.ComboBox3 = New System.Windows.Forms.ComboBox
Me.Label29 = New System.Windows.Forms.Label
Me.Label36 = New System.Windows.Forms.Label
Me.ComboBox4 = New System.Windows.Forms.ComboBox
Me.Label38 = New System.Windows.Forms.Label
Me.GroupBox33 = New System.Windows.Forms.GroupBox
Me.CheckBox10 = New System.Windows.Forms.CheckBox
Me.CheckBox7 = New System.Windows.Forms.CheckBox
Me.CheckBox11 = New System.Windows.Forms.CheckBox
Me.GroupBox35 = New System.Windows.Forms.GroupBox
Me.ComboBox8 = New System.Windows.Forms.ComboBox
Me.TextBox90 = New System.Windows.Forms.TextBox
Me.Label113 = New System.Windows.Forms.Label
Me.Label114 = New System.Windows.Forms.Label
Me.TextBox136 = New System.Windows.Forms.TextBox
Me.Label115 = New System.Windows.Forms.Label
Me.ComboBox9 = New System.Windows.Forms.ComboBox
Me.Label116 = New System.Windows.Forms.Label
Me.Label117 = New System.Windows.Forms.Label
Me.ComboBox10 = New System.Windows.Forms.ComboBox
Me.TextBox137 = New System.Windows.Forms.TextBox
Me.Label118 = New System.Windows.Forms.Label
Me.GroupBox36 = New System.Windows.Forms.GroupBox
Me.Button60 = New System.Windows.Forms.Button
Me.Button61 = New System.Windows.Forms.Button
Me.Button62 = New System.Windows.Forms.Button
Me.Button63 = New System.Windows.Forms.Button
Me.GroupBox37 = New System.Windows.Forms.GroupBox
Me.ComboBox11 = New System.Windows.Forms.ComboBox
Me.Label119 = New System.Windows.Forms.Label
Me.Label120 = New System.Windows.Forms.Label
Me.Label121 = New System.Windows.Forms.Label
Me.Label122 = New System.Windows.Forms.Label
Me.TextBox138 = New System.Windows.Forms.TextBox
Me.TextBox139 = New System.Windows.Forms.TextBox
Me.TextBox140 = New System.Windows.Forms.TextBox
Me.ComboBox12 = New System.Windows.Forms.ComboBox
Me.ComboBox13 = New System.Windows.Forms.ComboBox
Me.Label123 = New System.Windows.Forms.Label
Me.Label124 = New System.Windows.Forms.Label
Me.GroupBox17 = New System.Windows.Forms.GroupBox
Me.AxWebBrowser3 = New AxSHDocVw.AxWebBrowser
Me.Button55 = New System.Windows.Forms.Button
Me.Button49 = New System.Windows.Forms.Button
Me.Button46 = New System.Windows.Forms.Button
Me.TabPage1 = New System.Windows.Forms.TabPage
Me.TextBox49 = New System.Windows.Forms.TextBox
Me.CheckBox12 = New System.Windows.Forms.CheckBox
Me.GroupBox24 = New System.Windows.Forms.GroupBox
Me.TextBox5 = New System.Windows.Forms.TextBox
Me.Label16 = New System.Windows.Forms.Label
Me.CheckBox14 = New System.Windows.Forms.CheckBox
Me.Label134 = New System.Windows.Forms.Label
Me.Label133 = New System.Windows.Forms.Label
Me.TextBox147 = New System.Windows.Forms.TextBox
Me.Label93 = New System.Windows.Forms.Label
Me.TextBox134 = New System.Windows.Forms.TextBox
Me.Label102 = New System.Windows.Forms.Label
Me.TextBox144 = New System.Windows.Forms.TextBox
```

```
Me.Label107 = New System.Windows.Forms.Label
Me.TextBox158 = New System.Windows.Forms.TextBox
Me.Label108 = New System.Windows.Forms.Label
Me.Label105 = New System.Windows.Forms.Label
Me.TextBox149 = New System.Windows.Forms.TextBox
Me.Label53 = New System.Windows.Forms.Label
Me.TextBox129 = New System.Windows.Forms.TextBox
Me.Label44 = New System.Windows.Forms.Label
Me.CheckBox9 = New System.Windows.Forms.CheckBox
Me.Label103 = New System.Windows.Forms.Label
Me.TextBox148 = New System.Windows.Forms.TextBox
Me.TextBox128 = New System.Windows.Forms.TextBox
Me.Label90 = New System.Windows.Forms.Label
Me.GroupBox26 = New System.Windows.Forms.GroupBox
Me.Label111 = New System.Windows.Forms.Label
Me.Label112 = New System.Windows.Forms.Label
Me.TextBox89 = New System.Windows.Forms.TextBox
Me.Label43 = New System.Windows.Forms.Label
Me.Label56 = New System.Windows.Forms.Label
Me.TextBox80 = New System.Windows.Forms.TextBox
Me.Label49 = New System.Windows.Forms.Label
Me.Label50 = New System.Windows.Forms.Label
Me.TextBox131 = New System.Windows.Forms.TextBox
Me.Label48 = New System.Windows.Forms.Label
Me.Label47 = New System.Windows.Forms.Label
Me.TextBox130 = New System.Windows.Forms.TextBox
Me.GroupBox28 = New System.Windows.Forms.GroupBox
Me.Button58 = New System.Windows.Forms.Button
Me.Button48 = New System.Windows.Forms.Button
Me.TextBox151 = New System.Windows.Forms.TextBox
Me.TextBox150 = New System.Windows.Forms.TextBox
Me.TextBox143 = New System.Windows.Forms.TextBox
Me.CheckBox8 = New System.Windows.Forms.CheckBox
Me.TabPage7 = New System.Windows.Forms.TabPage
Me.Label51 = New System.Windows.Forms.Label
Me.TextBox112 = New System.Windows.Forms.TextBox
Me.Button37 = New System.Windows.Forms.Button
Me.Button27 = New System.Windows.Forms.Button
Me.Label89 = New System.Windows.Forms.Label
Me.TextBox127 = New System.Windows.Forms.TextBox
Me.GroupBox25 = New System.Windows.Forms.GroupBox
Me.ListBox7 = New System.Windows.Forms.ListBox
Me.Label75 = New System.Windows.Forms.Label
Me.ListBox8 = New System.Windows.Forms.ListBox
Me.Label76 = New System.Windows.Forms.Label
Me.ListBox9 = New System.Windows.Forms.ListBox
Me.Label77 = New System.Windows.Forms.Label
Me.ListBox10 = New System.Windows.Forms.ListBox
Me.Label78 = New System.Windows.Forms.Label
Me.TextBox125 = New System.Windows.Forms.TextBox
Me.Label79 = New System.Windows.Forms.Label
Me.TextBox126 = New System.Windows.Forms.TextBox
Me.Label80 = New System.Windows.Forms.Label
Me.GroupBox23 = New System.Windows.Forms.GroupBox
Me.ListBox6 = New System.Windows.Forms.ListBox
Me.Label74 = New System.Windows.Forms.Label
Me.ListBox5 = New System.Windows.Forms.ListBox
Me.Label71 = New System.Windows.Forms.Label
Me.ListBox4 = New System.Windows.Forms.ListBox
Me.Label70 = New System.Windows.Forms.Label
Me.ListBox3 = New System.Windows.Forms.ListBox
Me.Label62 = New System.Windows.Forms.Label
Me.TextBox117 = New System.Windows.Forms.TextBox
```

```
Me.Label61 = New System.Windows.Forms.Label
Me.TextBox118 = New System.Windows.Forms.TextBox
Me.Label68 = New System.Windows.Forms.Label
Me.GroupBox22 = New System.Windows.Forms.GroupBox
Me.Label52 = New System.Windows.Forms.Label
Me.TextBox113 = New System.Windows.Forms.TextBox
Me.TextBox122 = New System.Windows.Forms.TextBox
Me.Label65 = New System.Windows.Forms.Label
Me.TextBox121 = New System.Windows.Forms.TextBox
Me.Label64 = New System.Windows.Forms.Label
Me.TextBox120 = New System.Windows.Forms.TextBox
Me.Label63 = New System.Windows.Forms.Label
Me.TextBox119 = New System.Windows.Forms.TextBox
Me.Label69 = New System.Windows.Forms.Label
Me.TextBox124 = New System.Windows.Forms.TextBox
Me.Label67 = New System.Windows.Forms.Label
Me.TextBox123 = New System.Windows.Forms.TextBox
Me.Label66 = New System.Windows.Forms.Label
Me.Button35 = New System.Windows.Forms.Button
Me.Button33 = New System.Windows.Forms.Button
Me.TabPage2 = New System.Windows.Forms.TabPage
Me.GroupBox5 = New System.Windows.Forms.GroupBox
Me.Label73 = New System.Windows.Forms.Label
Me.AxWebBrowser1 = New AxSHDocVw.AxWebBrowser
Me.AxWebBrowser2 = New AxSHDocVw.AxWebBrowser
Me.Label72 = New System.Windows.Forms.Label
Me.TabPage5 = New System.Windows.Forms.TabPage
Me.GroupBox32 = New System.Windows.Forms.GroupBox
Me.ComboBox7 = New System.Windows.Forms.ComboBox
Me.Button59 = New System.Windows.Forms.Button
Me.GroupBox27 = New System.Windows.Forms.GroupBox
Me.Label110 = New System.Windows.Forms.Label
Me.TextBox161 = New System.Windows.Forms.TextBox
Me.TextBox157 = New System.Windows.Forms.TextBox
Me.TextBox156 = New System.Windows.Forms.TextBox
Me.Label106 = New System.Windows.Forms.Label
Me.Label104 = New System.Windows.Forms.Label
Me.TextBox155 = New System.Windows.Forms.TextBox
Me.TextBox154 = New System.Windows.Forms.TextBox
Me.TextBox81 = New System.Windows.Forms.TextBox
Me.TextBox153 = New System.Windows.Forms.TextBox
Me.TextBox159 = New System.Windows.Forms.TextBox
Me.TextBox160 = New System.Windows.Forms.TextBox
Me.GroupBox21 = New System.Windows.Forms.GroupBox
Me.Button36 = New System.Windows.Forms.Button
Me.Button38 = New System.Windows.Forms.Button
Me.Button39 = New System.Windows.Forms.Button
Me.GroupBox16 = New System.Windows.Forms.GroupBox
Me.Button34 = New System.Windows.Forms.Button
Me.Button31 = New System.Windows.Forms.Button
Me.Button29 = New System.Windows.Forms.Button
Me.Button28 = New System.Windows.Forms.Button
Me.GroupBox15 = New System.Windows.Forms.GroupBox
Me.Button30 = New System.Windows.Forms.Button
Me.Label60 = New System.Windows.Forms.Label
Me.Label59 = New System.Windows.Forms.Label
Me.Label58 = New System.Windows.Forms.Label
Me.TrackBar1 = New System.Windows.Forms.TrackBar
Me.TextBox116 = New System.Windows.Forms.TextBox
Me.Button32 = New System.Windows.Forms.Button
Me.TextBox115 = New System.Windows.Forms.TextBox
Me.TextBox95 = New System.Windows.Forms.TextBox
Me.TextBox94 = New System.Windows.Forms.TextBox
```

```
Me.TextBox93 = New System.Windows.Forms.TextBox
Me.TextBox92 = New System.Windows.Forms.TextBox
Me.Label57 = New System.Windows.Forms.Label
Me.TextBox91 = New System.Windows.Forms.TextBox
Me.TextBox78 = New System.Windows.Forms.TextBox
Me.TextBox79 = New System.Windows.Forms.TextBox
Me.TextBox82 = New System.Windows.Forms.TextBox
Me.TextBox83 = New System.Windows.Forms.TextBox
Me.TextBox84 = New System.Windows.Forms.TextBox
Me.TextBox85 = New System.Windows.Forms.TextBox
Me.TextBox86 = New System.Windows.Forms.TextBox
Me.TextBox87 = New System.Windows.Forms.TextBox
Me.TextBox88 = New System.Windows.Forms.TextBox
Me.TextBox96 = New System.Windows.Forms.TextBox
Me.TextBox97 = New System.Windows.Forms.TextBox
Me.TextBox98 = New System.Windows.Forms.TextBox
Me.TextBox99 = New System.Windows.Forms.TextBox
Me.TextBox100 = New System.Windows.Forms.TextBox
Me.TextBox101 = New System.Windows.Forms.TextBox
Me.TextBox102 = New System.Windows.Forms.TextBox
Me.TextBox103 = New System.Windows.Forms.TextBox
Me.TextBox104 = New System.Windows.Forms.TextBox
Me.TextBox105 = New System.Windows.Forms.TextBox
Me.TextBox106 = New System.Windows.Forms.TextBox
Me.TextBox107 = New System.Windows.Forms.TextBox
Me.TextBox108 = New System.Windows.Forms.TextBox
Me.TextBox109 = New System.Windows.Forms.TextBox
Me.TextBox110 = New System.Windows.Forms.TextBox
Me.TextBox111 = New System.Windows.Forms.TextBox
Me.ListBox2 = New System.Windows.Forms.ListBox
Me.Label41 = New System.Windows.Forms.Label
Me.Label40 = New System.Windows.Forms.Label
Me.ListBox1 = New System.Windows.Forms.ListBox
Me.TabPage4 = New System.Windows.Forms.TabPage
Me.TextBox74 = New System.Windows.Forms.TextBox
Me.TextBox73 = New System.Windows.Forms.TextBox
Me.TextBox72 = New System.Windows.Forms.TextBox
Me.TextBox71 = New System.Windows.Forms.TextBox
Me.TextBox70 = New System.Windows.Forms.TextBox
Me.TextBox69 = New System.Windows.Forms.TextBox
Me.TextBox68 = New System.Windows.Forms.TextBox
Me.TextBox67 = New System.Windows.Forms.TextBox
Me.TextBox66 = New System.Windows.Forms.TextBox
Me.TextBox65 = New System.Windows.Forms.TextBox
Me.TextBox64 = New System.Windows.Forms.TextBox
Me.TextBox57 = New System.Windows.Forms.TextBox
Me.TextBox58 = New System.Windows.Forms.TextBox
Me.TextBox59 = New System.Windows.Forms.TextBox
Me.TextBox60 = New System.Windows.Forms.TextBox
Me.TextBox61 = New System.Windows.Forms.TextBox
Me.TextBox62 = New System.Windows.Forms.TextBox
Me.TextBox63 = New System.Windows.Forms.TextBox
Me.TextBox56 = New System.Windows.Forms.TextBox
Me.TextBox55 = New System.Windows.Forms.TextBox
Me.TextBox54 = New System.Windows.Forms.TextBox
Me.TextBox53 = New System.Windows.Forms.TextBox
Me.TextBox52 = New System.Windows.Forms.TextBox
Me.TextBox51 = New System.Windows.Forms.TextBox
Me.TextBox50 = New System.Windows.Forms.TextBox
Me.TextBox48 = New System.Windows.Forms.TextBox
Me.TextBox47 = New System.Windows.Forms.TextBox
Me.TextBox46 = New System.Windows.Forms.TextBox
Me.TextBox45 = New System.Windows.Forms.TextBox
```

```
Me.TextBox44 = New System.Windows.Forms.TextBox
Me.TextBox43 = New System.Windows.Forms.TextBox
Me.TextBox42 = New System.Windows.Forms.TextBox
Me.TextBox41 = New System.Windows.Forms.TextBox
Me.TextBox40 = New System.Windows.Forms.TextBox
Me.TabPage3 = New System.Windows.Forms.TabPage
Me.TextBox36 = New System.Windows.Forms.TextBox
Me.TextBox7 = New System.Windows.Forms.TextBox
Me.Button3 = New System.Windows.Forms.Button
Me.TextBox35 = New System.Windows.Forms.TextBox
Me.TextBox34 = New System.Windows.Forms.TextBox
Me.TextBox33 = New System.Windows.Forms.TextBox
Me.TextBox32 = New System.Windows.Forms.TextBox
Me.TextBox31 = New System.Windows.Forms.TextBox
Me.TextBox30 = New System.Windows.Forms.TextBox
Me.TextBox29 = New System.Windows.Forms.TextBox
Me.TextBox28 = New System.Windows.Forms.TextBox
Me.TextBox27 = New System.Windows.Forms.TextBox
Me.TextBox26 = New System.Windows.Forms.TextBox
Me.TextBox25 = New System.Windows.Forms.TextBox
Me.TextBox24 = New System.Windows.Forms.TextBox
Me.TextBox23 = New System.Windows.Forms.TextBox
Me.TextBox22 = New System.Windows.Forms.TextBox
Me.TextBox21 = New System.Windows.Forms.TextBox
Me.TextBox20 = New System.Windows.Forms.TextBox
Me.TextBox19 = New System.Windows.Forms.TextBox
Me.TextBox18 = New System.Windows.Forms.TextBox
Me.TextBox17 = New System.Windows.Forms.TextBox
Me.TextBox16 = New System.Windows.Forms.TextBox
Me.TextBox15 = New System.Windows.Forms.TextBox
Me.TextBox14 = New System.Windows.Forms.TextBox
Me.TextBox13 = New System.Windows.Forms.TextBox
Me.TextBox12 = New System.Windows.Forms.TextBox
Me.MainMenu1 = New System.Windows.Forms.MainMenu
Me.MenuItem1 = New System.Windows.Forms.MenuItem
Me.State_Action_Real_Timer1 = New System.Windows.Forms.Timer(Me.components)
Me.Action_Timer_1 = New System.Windows.Forms.Timer(Me.components)
Me.State_Action_Rand_Timer1 = New System.Windows.Forms.Timer(Me.components)
Me.Shaking_Timer_1 = New System.Windows.Forms.Timer(Me.components)
Me.Timer2 = New System.Windows.Forms.Timer(Me.components)
Me.ToolTip1 = New System.Windows.Forms.ToolTip(Me.components)
Me.ToolTip2 = New System.Windows.Forms.ToolTip(Me.components)
Me.Robot_Operating = New System.Windows.Forms.Timer(Me.components)
Me.State_Action_Best_Timer1 = New System.Windows.Forms.Timer(Me.components)
Me.Timer1 = New System.Windows.Forms.Timer(Me.components)
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
Me.GroupBox3.SuspendLayout()
Me.GroupBox13.SuspendLayout()
Me.GroupBox11.SuspendLayout()
Me.GroupBox10.SuspendLayout()
Me.GroupBox9.SuspendLayout()
Me.GroupBox8.SuspendLayout()
Me.GroupBox7.SuspendLayout()
Me.GroupBox12.SuspendLayout()
Me.GroupBox14.SuspendLayout()
Me.GroupBox6.SuspendLayout()
Me.TabControl1.SuspendLayout()
Me.TabPage8.SuspendLayout()
Me.GroupBox38.SuspendLayout()
 CType(Me.AxMSChart3, System.ComponentModel.ISupportInitialize).BeginInit()
 CType(Me.AxMSChart1, System.ComponentModel.ISupportInitialize).BeginInit()
Me.GroupBox31.SuspendLayout()
```

```
Me.GroupBox18.SuspendLayout()
Me.GroupBox29.SuspendLayout()
Me.GroupBox30.SuspendLayout()
Me.GroupBox19.SuspendLayout()
Me.GroupBox20.SuspendLayout()
Me.GroupBox34.SuspendLayout()
Me.GroupBox4.SuspendLayout()
Me.GroupBox33.SuspendLayout()
Me.GroupBox35.SuspendLayout()
Me.GroupBox36.SuspendLayout()
Me.GroupBox37.SuspendLayout()
Me.GroupBox17.SuspendLayout()
 CType(Me.AxWebBrowser3, System.ComponentModel.ISupportInitialize).BeginInit()
Me.TabPage1.SuspendLayout()
Me.GroupBox24.SuspendLayout()
Me.GroupBox26.SuspendLayout()
Me.GroupBox28.SuspendLayout()
Me.TabPage7.SuspendLayout()
Me.GroupBox25.SuspendLayout()
Me.GroupBox23.SuspendLayout()
Me.GroupBox22.SuspendLayout()
Me.TabPage2.SuspendLayout()
Me.GroupBox5.SuspendLayout()
 CType(Me.AxWebBrowser1, System.ComponentModel.ISupportInitialize).BeginInit()
 CType(Me.AxWebBrowser2, System.ComponentModel.ISupportInitialize).BeginInit()
Me.TabPage5.SuspendLayout()
Me.GroupBox32.SuspendLayout()
Me.GroupBox27.SuspendLayout()
Me.GroupBox21.SuspendLayout()
Me.GroupBox16.SuspendLayout()
Me.GroupBox15.SuspendLayout()
 CType(Me.TrackBar1, System.ComponentModel.ISupportInitialize).BeginInit()
Me.TabPage4.SuspendLayout()
Me.TabPage3.SuspendLayout()
Me.SuspendLayout()
 '
 'CmdDownLoad
 '
Me.CmdDownLoad.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192,
Byte))
Me.CmdDownLoad.Cursor = System.Windows.Forms.Cursors.Default
Me.CmdDownLoad.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.CmdDownLoad.ForeColor = System.Drawing.SystemColors.ControlText
Me.CmdDownLoad.Location = New System.Drawing.Point(8, 48)
Me.CmdDownLoad.Name = "CmdDownLoad"
Me.CmdDownLoad.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.CmdDownLoad.Size = New System.Drawing.Size(104, 40)
Me.CmdDownLoad.TabIndex = 1
Me.CmdDownLoad.Text = "Download Job"
 '
 'Button1
 '
Me.Button1.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button1.Cursor = System.Windows.Forms.Cursors.Default
Me.Button1.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button1.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button1.Location = New System.Drawing.Point(8, 24)
Me.Button1.Name = "Button1"
Me.Button1.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button1.Size = New System.Drawing.Size(104, 40)
Me.Button1.TabIndex = 4
```

```
Me.Button1.Text = "Open Communication"
'
'TextBox1
'
Me.TextBox1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox1.Location = New System.Drawing.Point(88, 32)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(64, 20)
Me.TextBox1.TabIndex = 8
Me.TextBox1.Text = ""
'
'TextBox2
'
Me.TextBox2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox2.Location = New System.Drawing.Point(88, 72)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.Size = New System.Drawing.Size(64, 20)
Me.TextBox2.TabIndex = 9
Me.TextBox2.Text = ""
'
'Label2
'
Me.Label2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label2.Location = New System.Drawing.Point(96, 16)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(32, 16)
Me.Label2.TabIndex = 11
Me.Label2.Text = "nCid"
'
'Label3
'
Me.Label3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label3.Location = New System.Drawing.Point(96, 56)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(32, 16)
Me.Label3.TabIndex = 12
Me.Label3.Text = "rc"
'
'Label4
'
Me.Label4.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label4.Location = New System.Drawing.Point(8, 32)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(64, 16)
Me.Label4.TabIndex = 13
Me.Label4.Text = "BscOpen"
'
'Label5
'
Me.Label5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label5.Location = New System.Drawing.Point(8, 72)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(64, 16)
Me.Label5.TabIndex = 14
Me.Label5.Text = "BscConnect"
'
'Label6
```

```
'
Me.Label6.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label6.Location = New System.Drawing.Point(8, 104)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(80, 16)
Me.Label6.TabIndex = 16
Me.Label6.Text = "BscDownLoad"
'
'TextBox3
'
Me.TextBox3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox3.Location = New System.Drawing.Point(88, 104)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(64, 20)
Me.TextBox3.TabIndex = 15
Me.TextBox3.Text = ""
'
'Label7
'
Me.Label7.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label7.Location = New System.Drawing.Point(152, 72)
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(32, 16)
Me.Label7.TabIndex = 18
Me.Label7.Text = "(1)"
'
'Label8
'
Me.Label8.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label8.Location = New System.Drawing.Point(152, 40)
Me.Label8.Name = "Label8"
Me.Label8.Size = New System.Drawing.Size(48, 16)
Me.Label8.TabIndex = 19
Me.Label8.Text = "(not -1)"
'
'Label9
'
Me.Label9.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label9.Location = New System.Drawing.Point(152, 112)
Me.Label9.Name = "Label9"
Me.Label9.Size = New System.Drawing.Size(32, 16)
Me.Label9.TabIndex = 20
Me.Label9.Text = "(0)"
'
'Button4
'
Me.Button4.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button4.Cursor = System.Windows.Forms.Cursors.Default
Me.Button4.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button4.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button4.Location = New System.Drawing.Point(72, 136)
Me.Button4.Name = "Button4"
Me.Button4.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button4.Size = New System.Drawing.Size(56, 32)
Me.Button4.TabIndex = 21
Me.Button4.Text = "Clear"
'
```

```
'Button5
'
Me.Button5.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button5.Cursor = System.Windows.Forms.Cursors.Default
Me.Button5.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button5.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button5.Location = New System.Drawing.Point(8, 96)
Me.Button5.Name = "Button5"
Me.Button5.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button5.Size = New System.Drawing.Size(104, 40)
Me.Button5.TabIndex = 23
Me.Button5.Text = "Delete Job"
'
'Button6
'
Me.Button6.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button6.Cursor = System.Windows.Forms.Cursors.Default
Me.Button6.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button6.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button6.Location = New System.Drawing.Point(8, 144)
Me.Button6.Name = "Button6"
Me.Button6.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button6.Size = New System.Drawing.Size(104, 40)
Me.Button6.TabIndex = 24
Me.Button6.Text = "Run Job"
'
'CheckBox1
'
Me.CheckBox1.Checked = True
Me.CheckBox1.CheckState = System.Windows.Forms.CheckState.Checked
Me.CheckBox1.Location = New System.Drawing.Point(80, 136)
Me.CheckBox1.Name = "CheckBox1"
Me.CheckBox1.Size = New System.Drawing.Size(16, 16)
Me.CheckBox1.TabIndex = 25
Me.CheckBox1.Text = "Teach / Play"
'
'CheckBox2
'
Me.CheckBox2.Location = New System.Drawing.Point(80, 160)
Me.CheckBox2.Name = "CheckBox2"
Me.CheckBox2.Size = New System.Drawing.Size(16, 16)
Me.CheckBox2.TabIndex = 26
Me.CheckBox2.Text = "Servo"
'
'Button7
'
Me.Button7.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button7.Cursor = System.Windows.Forms.Cursors.Default
Me.Button7.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button7.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button7.Location = New System.Drawing.Point(128, 48)
Me.Button7.Name = "Button7"
Me.Button7.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button7.Size = New System.Drawing.Size(104, 40)
Me.Button7.TabIndex = 27
Me.Button7.Text = "Upload Job"
'
'TextBox4
'
```

```
Me.TextBox4.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox4.Location = New System.Drawing.Point(128, 24)
Me.TextBox4.Name = "TextBox4"
Me.TextBox4.Size = New System.Drawing.Size(104, 20)
Me.TextBox4.TabIndex = 28
Me.TextBox4.Text = "BAGS1.JBI"
 '
 'Button9
 '
Me.Button9.BackColor = System.Drawing.Color.Red
Me.Button9.Cursor = System.Windows.Forms.Cursors.Default
Me.Button9.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button9.ForeColor = System.Drawing.Color.Yellow
Me.Button9.Location = New System.Drawing.Point(128, 96)
Me.Button9.Name = "Button9"
Me.Button9.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button9.Size = New System.Drawing.Size(104, 40)
Me.Button9.TabIndex = 30
Me.Button9.Text = "Emergency Stop"
 '
 'Label10
 '
Me.Label10.Location = New System.Drawing.Point(104, 136)
Me.Label10.Name = "Label10"
Me.Label10.Size = New System.Drawing.Size(80, 16)
Me.Label10.TabIndex = 31
 '
 'Label11
 '
Me.Label11.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label11.Location = New System.Drawing.Point(40, 136)
Me.Label11.Name = "Label11"
Me.Label11.Size = New System.Drawing.Size(36, 16)
Me.Label11.TabIndex = 32
Me.Label11.Text = "Mode:"
 '
 'Label12
 '
Me.Label12.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label12.Location = New System.Drawing.Point(40, 160)
Me.Label12.Name = "Label12"
Me.Label12.Size = New System.Drawing.Size(36, 16)
Me.Label12.TabIndex = 33
Me.Label12.Text = "Servo:"
 '
 'Label13
 '
Me.Label13.Location = New System.Drawing.Point(104, 160)
Me.Label13.Name = "Label13"
Me.Label13.Size = New System.Drawing.Size(80, 16)
Me.Label13.TabIndex = 34
 '
 'Label14
 '
Me.Label14.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label14.Location = New System.Drawing.Point(8, 104)
Me.Label14.Name = "Label14"
Me.Label14.Size = New System.Drawing.Size(128, 16)
```

```
Me.Label14.TabIndex = 35
Me.Label14.Text = "Communication Status:"
'
'Label15
'
Me.Label15.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label15.Location = New System.Drawing.Point(136, 104)
Me.Label15.Name = "Label15"
Me.Label15.Size = New System.Drawing.Size(96, 16)
Me.Label15.TabIndex = 36
Me.Label15.Text = "Disconnected"
'
'Button10
'
Me.Button10.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button10.Cursor = System.Windows.Forms.Cursors.Default
Me.Button10.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button10.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button10.Location = New System.Drawing.Point(8, 56)
Me.Button10.Name = "Button10"
Me.Button10.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button10.Size = New System.Drawing.Size(50, 30)
Me.Button10.TabIndex = 38
Me.Button10.Text = "X-"
'
'Button2
'
Me.Button2.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button2.Cursor = System.Windows.Forms.Cursors.Default
Me.Button2.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button2.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button2.Location = New System.Drawing.Point(8, 24)
Me.Button2.Name = "Button2"
Me.Button2.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button2.Size = New System.Drawing.Size(50, 30)
Me.Button2.TabIndex = 37
Me.Button2.Text = "X+"
'
'Button11
'
Me.Button11.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button11.Cursor = System.Windows.Forms.Cursors.Default
Me.Button11.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button11.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button11.Location = New System.Drawing.Point(64, 56)
Me.Button11.Name = "Button11"
Me.Button11.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button11.Size = New System.Drawing.Size(50, 30)
Me.Button11.TabIndex = 40
Me.Button11.Text = "Y-"
'
'Button12
'
Me.Button12.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button12.Cursor = System.Windows.Forms.Cursors.Default
Me.Button12.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button12.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button12.Location = New System.Drawing.Point(64, 24)
```

```
Me.Button12.Name = "Button12"
Me.Button12.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button12.Size = New System.Drawing.Size(50, 30)
Me.Button12.TabIndex = 39
Me.Button12.Text = "Y+"
 '
 'Button13
 '
Me.Button13.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button13.Cursor = System.Windows.Forms.Cursors.Default
Me.Button13.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button13.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button13.Location = New System.Drawing.Point(120, 56)
Me.Button13.Name = "Button13"
Me.Button13.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button13.Size = New System.Drawing.Size(50, 30)
Me.Button13.TabIndex = 42
Me.Button13.Text = "Z-"
 '
 'Button14
 '
Me.Button14.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button14.Cursor = System.Windows.Forms.Cursors.Default
Me.Button14.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button14.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button14.Location = New System.Drawing.Point(120, 24)
Me.Button14.Name = "Button14"
Me.Button14.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button14.Size = New System.Drawing.Size(50, 30)
Me.Button14.TabIndex = 41
Me.Button14.Text = "Z+"
 '
 'Button15
 '
Me.Button15.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button15.Cursor = System.Windows.Forms.Cursors.Default
Me.Button15.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button15.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button15.Location = New System.Drawing.Point(8, 24)
Me.Button15.Name = "Button15"
Me.Button15.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button15.Size = New System.Drawing.Size(50, 30)
Me.Button15.TabIndex = 43
Me.Button15.Text = "Roll+"
 '
 'Button16
 '
Me.Button16.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button16.Cursor = System.Windows.Forms.Cursors.Default
Me.Button16.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button16.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button16.Location = New System.Drawing.Point(8, 56)
Me.Button16.Name = "Button16"
Me.Button16.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button16.Size = New System.Drawing.Size(50, 30)
Me.Button16.TabIndex = 44
Me.Button16.Text = "Roll-"
 '
 'Button17
```

```
'
Me.Button17.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button17.Cursor = System.Windows.Forms.Cursors.Default
Me.Button17.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button17.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button17.Location = New System.Drawing.Point(64, 56)
Me.Button17.Name = "Button17"
Me.Button17.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button17.Size = New System.Drawing.Size(50, 30)
Me.Button17.TabIndex = 46
Me.Button17.Text = "Pitch-"
'
'Button18
'
Me.Button18.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button18.Cursor = System.Windows.Forms.Cursors.Default
Me.Button18.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button18.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button18.Location = New System.Drawing.Point(64, 24)
Me.Button18.Name = "Button18"
Me.Button18.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button18.Size = New System.Drawing.Size(50, 30)
Me.Button18.TabIndex = 45
Me.Button18.Text = "Pitch+"
'
'Button19
'
Me.Button19.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button19.Cursor = System.Windows.Forms.Cursors.Default
Me.Button19.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button19.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button19.Location = New System.Drawing.Point(120, 56)
Me.Button19.Name = "Button19"
Me.Button19.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button19.Size = New System.Drawing.Size(50, 30)
Me.Button19.TabIndex = 48
Me.Button19.Text = "Yaw-"
'
'Button20
'
Me.Button20.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button20.Cursor = System.Windows.Forms.Cursors.Default
Me.Button20.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button20.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button20.Location = New System.Drawing.Point(120, 24)
Me.Button20.Name = "Button20"
Me.Button20.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button20.Size = New System.Drawing.Size(50, 30)
Me.Button20.TabIndex = 47
Me.Button20.Text = "Yaw+"
'
'Button8
'
Me.Button8.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(255, Byte))
Me.Button8.Cursor = System.Windows.Forms.Cursors.Default
Me.Button8.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button8.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button8.Location = New System.Drawing.Point(72, 24)
```

```
Me.Button8.Name = "Button8"
Me.Button8.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button8.Size = New System.Drawing.Size(56, 40)
Me.Button8.TabIndex = 49
Me.Button8.Text = "Home Center"
'
'Button21
'
Me.Button21.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(255, Byte))
Me.Button21.Cursor = System.Windows.Forms.Cursors.Default
Me.Button21.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button21.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button21.Location = New System.Drawing.Point(136, 24)
Me.Button21.Name = "Button21"
Me.Button21.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button21.Size = New System.Drawing.Size(56, 40)
Me.Button21.TabIndex = 50
Me.Button21.Text = "Home Right"
'
'Button22
'
Me.Button22.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(255, Byte))
Me.Button22.Cursor = System.Windows.Forms.Cursors.Default
Me.Button22.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button22.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button22.Location = New System.Drawing.Point(8, 24)
Me.Button22.Name = "Button22"
Me.Button22.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button22.Size = New System.Drawing.Size(56, 40)
Me.Button22.TabIndex = 51
Me.Button22.Text = "Home Left"
'
'Button23
'
Me.Button23.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button23.Cursor = System.Windows.Forms.Cursors.Default
Me.Button23.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button23.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button23.Location = New System.Drawing.Point(8, 24)
Me.Button23.Name = "Button23"
Me.Button23.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button23.Size = New System.Drawing.Size(50, 30)
Me.Button23.TabIndex = 52
Me.Button23.Text = "Open"
'
'Button24
'
Me.Button24.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button24.Cursor = System.Windows.Forms.Cursors.Default
Me.Button24.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button24.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button24.Location = New System.Drawing.Point(8, 56)
Me.Button24.Name = "Button24"
Me.Button24.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button24.Size = New System.Drawing.Size(50, 30)
Me.Button24.TabIndex = 53
Me.Button24.Text = "Close"
'
'Button25
```

```
'
Me.Button25.BackColor = System.Drawing.Color.Red
Me.Button25.Cursor = System.Windows.Forms.Cursors.Default
Me.Button25.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button25.ForeColor = System.Drawing.Color.Yellow
Me.Button25.Location = New System.Drawing.Point(168, 136)
Me.Button25.Name = "Button25"
Me.Button25.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button25.Size = New System.Drawing.Size(96, 48)
Me.Button25.TabIndex = 62
Me.Button25.Text = "Stop"
'
'TextBox6
'
Me.TextBox6.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox6.Location = New System.Drawing.Point(8, 24)
Me.TextBox6.Name = "TextBox6"
Me.TextBox6.Size = New System.Drawing.Size(104, 20)
Me.TextBox6.TabIndex = 63
Me.TextBox6.Text = "POLICY1.JBI"
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.CheckBox2)
Me.GroupBox1.Controls.Add(Me.Button1)
Me.GroupBox1.Controls.Add(Me.Label10)
Me.GroupBox1.Controls.Add(Me.Label11)
Me.GroupBox1.Controls.Add(Me.Label12)
Me.GroupBox1.Controls.Add(Me.Label13)
Me.GroupBox1.Controls.Add(Me.Label14)
Me.GroupBox1.Controls.Add(Me.Label15)
Me.GroupBox1.Controls.Add(Me.CheckBox1)
Me.GroupBox1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox1.Location = New System.Drawing.Point(16, 344)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(240, 184)
Me.GroupBox1.TabIndex = 64
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Communication"
'
'GroupBox2
'
Me.GroupBox2.Controls.Add(Me.Button7)
Me.GroupBox2.Controls.Add(Me.TextBox4)
Me.GroupBox2.Controls.Add(Me.Button5)
Me.GroupBox2.Controls.Add(Me.Button6)
Me.GroupBox2.Controls.Add(Me.TextBox6)
Me.GroupBox2.Controls.Add(Me.CmdDownLoad)
Me.GroupBox2.Controls.Add(Me.Button9)
Me.GroupBox2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox2.Location = New System.Drawing.Point(16, 536)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(240, 192)
Me.GroupBox2.TabIndex = 65
Me.GroupBox2.TabStop = False
Me.GroupBox2.Text = "Download / Upload"
'
'GroupBox3
'
```

```
Me.GroupBox3.Controls.Add(Me.GroupBox13)
Me.GroupBox3.Controls.Add(Me.GroupBox11)
Me.GroupBox3.Controls.Add(Me.GroupBox10)
Me.GroupBox3.Controls.Add(Me.GroupBox12)
Me.GroupBox3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox3.Location = New System.Drawing.Point(8, 16)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(488, 456)
Me.GroupBox3.TabIndex = 66
Me.GroupBox3.TabStop = False
Me.GroupBox3.Text = "Robot Setup"
'
'GroupBox13
'
Me.GroupBox13.Controls.Add(Me.Button21)
Me.GroupBox13.Controls.Add(Me.Button22)
Me.GroupBox13.Controls.Add(Me.Button8)
Me.GroupBox13.Location = New System.Drawing.Point(248, 40)
Me.GroupBox13.Name = "GroupBox13"
Me.GroupBox13.Size = New System.Drawing.Size(200, 72)
Me.GroupBox13.TabIndex = 83
Me.GroupBox13.TabStop = False
Me.GroupBox13.Text = "Home Positions"
'
'GroupBox11
'
Me.GroupBox11.Controls.Add(Me.TextBox11)
Me.GroupBox11.Controls.Add(Me.Label18)
Me.GroupBox11.Controls.Add(Me.Label25)
Me.GroupBox11.Controls.Add(Me.CheckBox5)
Me.GroupBox11.Controls.Add(Me.Label17)
Me.GroupBox11.Controls.Add(Me.Label1)
Me.GroupBox11.Controls.Add(Me.CheckBox4)
Me.GroupBox11.Controls.Add(Me.TextBox10)
Me.GroupBox11.Controls.Add(Me.Label23)
Me.GroupBox11.Controls.Add(Me.Label24)
Me.GroupBox11.Location = New System.Drawing.Point(56, 32)
Me.GroupBox11.Name = "GroupBox11"
Me.GroupBox11.Size = New System.Drawing.Size(168, 192)
Me.GroupBox11.TabIndex = 82
Me.GroupBox11.TabStop = False
Me.GroupBox11.Text = "Operational Mode"
'
'TextBox11
'
Me.TextBox11.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox11.Location = New System.Drawing.Point(16, 152)
Me.TextBox11.Name = "TextBox11"
Me.TextBox11.Size = New System.Drawing.Size(72, 20)
Me.TextBox11.TabIndex = 88
Me.TextBox11.Text = "5"
'
'Label18
'
Me.Label18.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label18.Location = New System.Drawing.Point(96, 152)
Me.Label18.Name = "Label18"
Me.Label18.Size = New System.Drawing.Size(24, 16)
Me.Label18.TabIndex = 90
Me.Label18.Text = "cm"
```

```
'
'Label25
'
Me.Label25.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label25.Location = New System.Drawing.Point(16, 128)
Me.Label25.Name = "Label25"
Me.Label25.Size = New System.Drawing.Size(88, 16)
Me.Label25.TabIndex = 89
Me.Label25.Text = "Wrist Step Size:"
'
'CheckBox5
'
Me.CheckBox5.Checked = True
Me.CheckBox5.CheckState = System.Windows.Forms.CheckState.Checked
Me.CheckBox5.Location = New System.Drawing.Point(32, 48)
Me.CheckBox5.Name = "CheckBox5"
Me.CheckBox5.Size = New System.Drawing.Size(16, 16)
Me.CheckBox5.TabIndex = 87
'
'Label17
'
Me.Label17.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label17.Location = New System.Drawing.Point(16, 24)
Me.Label17.Name = "Label17"
Me.Label17.Size = New System.Drawing.Size(80, 16)
Me.Label17.TabIndex = 86
Me.Label17.Text = "Incremental"
'
'Label1
'
Me.Label1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(96, 24)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(64, 16)
Me.Label1.TabIndex = 85
Me.Label1.Text = "Continious"
'
'CheckBox4
'
Me.CheckBox4.Location = New System.Drawing.Point(112, 48)
Me.CheckBox4.Name = "CheckBox4"
Me.CheckBox4.Size = New System.Drawing.Size(16, 16)
Me.CheckBox4.TabIndex = 84
'
'TextBox10
'
Me.TextBox10.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox10.Location = New System.Drawing.Point(16, 96)
Me.TextBox10.Name = "TextBox10"
Me.TextBox10.Size = New System.Drawing.Size(72, 20)
Me.TextBox10.TabIndex = 84
Me.TextBox10.Text = "10"
'
'Label23
'
Me.Label23.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label23.Location = New System.Drawing.Point(96, 96)
Me.Label23.Name = "Label23"
```

```
Me.Label23.Size = New System.Drawing.Size(24, 16)
Me.Label23.TabIndex = 86
Me.Label23.Text = "cm"
'
'Label24
'
Me.Label24.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label24.Location = New System.Drawing.Point(16, 72)
Me.Label24.Name = "Label24"
Me.Label24.Size = New System.Drawing.Size(88, 16)
Me.Label24.TabIndex = 85
Me.Label24.Text = "Arm Step Size:"
'
'GroupBox10
'
Me.GroupBox10.Controls.Add(Me.GroupBox9)
Me.GroupBox10.Controls.Add(Me.GroupBox8)
Me.GroupBox10.Controls.Add(Me.GroupBox7)
Me.GroupBox10.Controls.Add(Me.Button25)
Me.GroupBox10.Location = New System.Drawing.Point(16, 240)
Me.GroupBox10.Name = "GroupBox10"
Me.GroupBox10.Size = New System.Drawing.Size(456, 200)
Me.GroupBox10.TabIndex = 81
Me.GroupBox10.TabStop = False
Me.GroupBox10.Text = "Robot Joint Commands"
'
'GroupBox9
'
Me.GroupBox9.Controls.Add(Me.Button23)
Me.GroupBox9.Controls.Add(Me.Button24)
Me.GroupBox9.Location = New System.Drawing.Point(376, 24)
Me.GroupBox9.Name = "GroupBox9"
Me.GroupBox9.Size = New System.Drawing.Size(68, 96)
Me.GroupBox9.TabIndex = 80
Me.GroupBox9.TabStop = False
Me.GroupBox9.Text = "Gripper"
'
'GroupBox8
'
Me.GroupBox8.Controls.Add(Me.Button20)
Me.GroupBox8.Controls.Add(Me.Button19)
Me.GroupBox8.Controls.Add(Me.Button18)
Me.GroupBox8.Controls.Add(Me.Button17)
Me.GroupBox8.Controls.Add(Me.Button16)
Me.GroupBox8.Controls.Add(Me.Button15)
Me.GroupBox8.Location = New System.Drawing.Point(192, 24)
Me.GroupBox8.Name = "GroupBox8"
Me.GroupBox8.Size = New System.Drawing.Size(176, 96)
Me.GroupBox8.TabIndex = 79
Me.GroupBox8.TabStop = False
Me.GroupBox8.Text = "Wrist"
'
'GroupBox7
'
Me.GroupBox7.Controls.Add(Me.Button12)
Me.GroupBox7.Controls.Add(Me.Button10)
Me.GroupBox7.Controls.Add(Me.Button2)
Me.GroupBox7.Controls.Add(Me.Button11)
Me.GroupBox7.Controls.Add(Me.Button13)
Me.GroupBox7.Controls.Add(Me.Button14)
Me.GroupBox7.Location = New System.Drawing.Point(8, 24)
Me.GroupBox7.Name = "GroupBox7"
```

```
Me.GroupBox7.Size = New System.Drawing.Size(176, 96)
Me.GroupBox7.TabIndex = 78
Me.GroupBox7.TabStop = False
Me.GroupBox7.Text = "Arm"
'
'GroupBox12
'
Me.GroupBox12.Controls.Add(Me.Label20)
Me.GroupBox12.Controls.Add(Me.Label22)
Me.GroupBox12.Controls.Add(Me.TextBox9)
Me.GroupBox12.Controls.Add(Me.Label21)
Me.GroupBox12.Controls.Add(Me.Label19)
Me.GroupBox12.Controls.Add(Me.TextBox8)
Me.GroupBox12.Location = New System.Drawing.Point(256, 136)
Me.GroupBox12.Name = "GroupBox12"
Me.GroupBox12.Size = New System.Drawing.Size(176, 88)
Me.GroupBox12.TabIndex = 83
Me.GroupBox12.TabStop = False
Me.GroupBox12.Text = "Speed"
'
'Label20
'
Me.Label20.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label20.Location = New System.Drawing.Point(128, 56)
Me.Label20.Name = "Label20"
Me.Label20.Size = New System.Drawing.Size(40, 16)
Me.Label20.TabIndex = 74
Me.Label20.Text = "cm/sec"
'
'Label22
'
Me.Label22.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label22.Location = New System.Drawing.Point(8, 56)
Me.Label22.Name = "Label22"
Me.Label22.Size = New System.Drawing.Size(40, 16)
Me.Label22.TabIndex = 73
Me.Label22.Text = "Wrist:"
'
'TextBox9
'
Me.TextBox9.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox9.Location = New System.Drawing.Point(48, 56)
Me.TextBox9.Name = "TextBox9"
Me.TextBox9.Size = New System.Drawing.Size(72, 20)
Me.TextBox9.TabIndex = 72
Me.TextBox9.Text = "5"
'
'Label21
'
Me.Label21.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label21.Location = New System.Drawing.Point(128, 24)
Me.Label21.Name = "Label21"
Me.Label21.Size = New System.Drawing.Size(40, 16)
Me.Label21.TabIndex = 71
Me.Label21.Text = "cm/sec"
'
'Label19
'
```

```
Me.Label19.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label19.Location = New System.Drawing.Point(8, 24)
Me.Label19.Name = "Label19"
Me.Label19.Size = New System.Drawing.Size(32, 16)
Me.Label19.TabIndex = 70
Me.Label19.Text = "Arm:"
'
'TextBox8
'
Me.TextBox8.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox8.Location = New System.Drawing.Point(48, 24)
Me.TextBox8.Name = "TextBox8"
Me.TextBox8.Size = New System.Drawing.Size(72, 20)
Me.TextBox8.TabIndex = 69
Me.TextBox8.Text = "10"
'
'GroupBox14
'
Me.GroupBox14.Controls.Add(Me.Label28)
Me.GroupBox14.Controls.Add(Me.ComboBox1)
Me.GroupBox14.Controls.Add(Me.Label33)
Me.GroupBox14.Controls.Add(Me.TextBox39)
Me.GroupBox14.Controls.Add(Me.Label34)
Me.GroupBox14.Controls.Add(Me.Label32)
Me.GroupBox14.Controls.Add(Me.TextBox38)
Me.GroupBox14.Controls.Add(Me.Label31)
Me.GroupBox14.Controls.Add(Me.Label30)
Me.GroupBox14.Controls.Add(Me.TextBox37)
Me.GroupBox14.Controls.Add(Me.Button26)
Me.GroupBox14.Location = New System.Drawing.Point(368, 560)
Me.GroupBox14.Name = "GroupBox14"
Me.GroupBox14.Size = New System.Drawing.Size(216, 208)
Me.GroupBox14.TabIndex = 84
Me.GroupBox14.TabStop = False
Me.GroupBox14.Text = "Robot Shaking Commands"
Me.GroupBox14.Visible = False
'
'Label28
'
Me.Label28.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label28.Location = New System.Drawing.Point(16, 40)
Me.Label28.Name = "Label28"
Me.Label28.Size = New System.Drawing.Size(32, 16)
Me.Label28.TabIndex = 102
Me.Label28.Text = "Axis:"
'
'ComboBox1
'
Me.ComboBox1.Items.AddRange(New Object() {"X", "Y", "Z"})
Me.ComboBox1.Location = New System.Drawing.Point(80, 32)
Me.ComboBox1.Name = "ComboBox1"
Me.ComboBox1.Size = New System.Drawing.Size(80, 22)
Me.ComboBox1.TabIndex = 101
Me.ComboBox1.Text = "Axis"
'
'Label33
'
Me.Label33.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label33.Location = New System.Drawing.Point(160, 136)
```

```
Me.Label33.Name = "Label33"
Me.Label33.Size = New System.Drawing.Size(48, 16)
Me.Label33.TabIndex = 99
Me.Label33.Text = "cm / sec"
'
'TextBox39
'
Me.TextBox39.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox39.Location = New System.Drawing.Point(80, 128)
Me.TextBox39.Name = "TextBox39"
Me.TextBox39.Size = New System.Drawing.Size(72, 20)
Me.TextBox39.TabIndex = 98
Me.TextBox39.Text = "1500"
'
'Label34
'
Me.Label34.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label34.Location = New System.Drawing.Point(16, 136)
Me.Label34.Name = "Label34"
Me.Label34.Size = New System.Drawing.Size(48, 16)
Me.Label34.TabIndex = 97
Me.Label34.Text = "Speed:"
'
'Label32
'
Me.Label32.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label32.Location = New System.Drawing.Point(160, 104)
Me.Label32.Name = "Label32"
Me.Label32.Size = New System.Drawing.Size(24, 16)
Me.Label32.TabIndex = 96
Me.Label32.Text = "cm"
'
'TextBox38
'
Me.TextBox38.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox38.Location = New System.Drawing.Point(80, 96)
Me.TextBox38.Name = "TextBox38"
Me.TextBox38.Size = New System.Drawing.Size(72, 20)
Me.TextBox38.TabIndex = 95
Me.TextBox38.Text = "5"
'
'Label31
'
Me.Label31.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label31.Location = New System.Drawing.Point(16, 104)
Me.Label31.Name = "Label31"
Me.Label31.Size = New System.Drawing.Size(64, 16)
Me.Label31.TabIndex = 94
Me.Label31.Text = "Amplitude:"
'
'Label30
'
Me.Label30.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label30.Location = New System.Drawing.Point(16, 72)
Me.Label30.Name = "Label30"
Me.Label30.Size = New System.Drawing.Size(48, 16)
Me.Label30.TabIndex = 93
```

```
Me.Label30.Text = "Times:"
'
'TextBox37
'
Me.TextBox37.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox37.Location = New System.Drawing.Point(80, 64)
Me.TextBox37.Name = "TextBox37"
Me.TextBox37.Size = New System.Drawing.Size(72, 20)
Me.TextBox37.TabIndex = 92
Me.TextBox37.Text = "5"
'
'Button26
'
Me.Button26.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button26.Cursor = System.Windows.Forms.Cursors.Default
Me.Button26.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button26.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button26.Location = New System.Drawing.Point(88, 168)
Me.Button26.Name = "Button26"
Me.Button26.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button26.Size = New System.Drawing.Size(50, 30)
Me.Button26.TabIndex = 54
Me.Button26.Text = "Run"
'
'CheckBox3
'
Me.CheckBox3.Checked = True
Me.CheckBox3.CheckState = System.Windows.Forms.CheckState.Checked
Me.CheckBox3.Location = New System.Drawing.Point(544, 448)
Me.CheckBox3.Name = "CheckBox3"
Me.CheckBox3.Size = New System.Drawing.Size(16, 16)
Me.CheckBox3.TabIndex = 91
Me.CheckBox3.Visible = False
'
'Label26
'
Me.Label26.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label26.Location = New System.Drawing.Point(512, 424)
Me.Label26.Name = "Label26"
Me.Label26.Size = New System.Drawing.Size(104, 16)
Me.Label26.TabIndex = 90
Me.Label26.Text = "World Coordinates"
Me.Label26.Visible = False
'
'Label27
'
Me.Label27.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label27.Location = New System.Drawing.Point(624, 424)
Me.Label27.Name = "Label27"
Me.Label27.Size = New System.Drawing.Size(96, 16)
Me.Label27.TabIndex = 89
Me.Label27.Text = "Base Coordinates"
Me.Label27.Visible = False
'
'CheckBox6
'
Me.CheckBox6.Location = New System.Drawing.Point(656, 448)
Me.CheckBox6.Name = "CheckBox6"
Me.CheckBox6.Size = New System.Drawing.Size(16, 16)
```

```
Me.CheckBox6.TabIndex = 88
Me.CheckBox6.Visible = False
'
'GroupBox6
'
Me.GroupBox6.Controls.Add(Me.Label7)
Me.GroupBox6.Controls.Add(Me.Label8)
Me.GroupBox6.Controls.Add(Me.TextBox1)
Me.GroupBox6.Controls.Add(Me.TextBox2)
Me.GroupBox6.Controls.Add(Me.Label2)
Me.GroupBox6.Controls.Add(Me.Label3)
Me.GroupBox6.Controls.Add(Me.Label4)
Me.GroupBox6.Controls.Add(Me.Label5)
Me.GroupBox6.Controls.Add(Me.Label6)
Me.GroupBox6.Controls.Add(Me.TextBox3)
Me.GroupBox6.Controls.Add(Me.Label9)
Me.GroupBox6.Controls.Add(Me.Button4)
Me.GroupBox6.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox6.Location = New System.Drawing.Point(264, 352)
Me.GroupBox6.Name = "GroupBox6"
Me.GroupBox6.Size = New System.Drawing.Size(208, 176)
Me.GroupBox6.TabIndex = 69
Me.GroupBox6.TabStop = False
Me.GroupBox6.Text = "Messeges"
'
'TabControl1
'
Me.TabControl1.Controls.Add(Me.TabPage8)
Me.TabControl1.Controls.Add(Me.TabPage1)
Me.TabControl1.Controls.Add(Me.TabPage7)
Me.TabControl1.Controls.Add(Me.TabPage2)
Me.TabControl1.Controls.Add(Me.TabPage5)
Me.TabControl1.Controls.Add(Me.TabPage4)
Me.TabControl1.Controls.Add(Me.TabPage3)
Me.TabControl1.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.TabControl1.Location = New System.Drawing.Point(16, 8)
Me.TabControl1.Name = "TabControl1"
Me.TabControl1.SelectedIndex = 0
Me.TabControl1.Size = New System.Drawing.Size(1272, 824)
Me.TabControl1.TabIndex = 84
'
'TabPage8
'
Me.TabPage8.Controls.Add(Me.CheckBox13)
Me.TabPage8.Controls.Add(Me.Label92)
Me.TabPage8.Controls.Add(Me.TextBox133)
Me.TabPage8.Controls.Add(Me.Button42)
Me.TabPage8.Controls.Add(Me.GroupBox38)
Me.TabPage8.Controls.Add(Me.GroupBox31)
Me.TabPage8.Controls.Add(Me.GroupBox18)
Me.TabPage8.Controls.Add(Me.GroupBox29)
Me.TabPage8.Controls.Add(Me.GroupBox19)
Me.TabPage8.Controls.Add(Me.GroupBox17)
Me.TabPage8.Controls.Add(Me.Button55)
Me.TabPage8.Controls.Add(Me.Button49)
Me.TabPage8.Controls.Add(Me.Button46)
Me.TabPage8.Location = New System.Drawing.Point(4, 28)
Me.TabPage8.Name = "TabPage8"
Me.TabPage8.Size = New System.Drawing.Size(1264, 792)
Me.TabPage8.TabIndex = 7
Me.TabPage8.Text = "Human-Robot Collaboration"
'
```

```
'CheckBox13
'
Me.CheckBox13.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.CheckBox13.Location = New System.Drawing.Point(34, 84)
Me.CheckBox13.Name = "CheckBox13"
Me.CheckBox13.Size = New System.Drawing.Size(136, 16)
Me.CheckBox13.TabIndex = 281
Me.CheckBox13.Text = "Enable Matlab"
'
'Label92
'
Me.Label92.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label92.Location = New System.Drawing.Point(88, 106)
Me.Label92.Name = "Label92"
Me.Label92.Size = New System.Drawing.Size(40, 16)
Me.Label92.TabIndex = 280
Me.Label92.Text = "Policy:"
'
'TextBox133
'
Me.TextBox133.Enabled = False
Me.TextBox133.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox133.Location = New System.Drawing.Point(128, 104)
Me.TextBox133.Name = "TextBox133"
Me.TextBox133.Size = New System.Drawing.Size(32, 20)
Me.TextBox133.TabIndex = 279
Me.TextBox133.Text = "0"
'
'Button42
'
Me.Button42.BackColor = System.Drawing.Color.Red
Me.Button42.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button42.Location = New System.Drawing.Point(80, 708)
Me.Button42.Name = "Button42"
Me.Button42.Size = New System.Drawing.Size(64, 32)
Me.Button42.TabIndex = 278
Me.Button42.Text = "temp1"
Me.Button42.Visible = False
'
'GroupBox38
'
Me.GroupBox38.Controls.Add(Me.Label94)
Me.GroupBox38.Controls.Add(Me.Label129)
Me.GroupBox38.Controls.Add(Me.Label130)
Me.GroupBox38.Controls.Add(Me.Label131)
Me.GroupBox38.Controls.Add(Me.Label132)
Me.GroupBox38.Controls.Add(Me.Label95)
Me.GroupBox38.Controls.Add(Me.Label96)
Me.GroupBox38.Controls.Add(Me.Label97)
Me.GroupBox38.Controls.Add(Me.Label98)
Me.GroupBox38.Controls.Add(Me.Label99)
Me.GroupBox38.Controls.Add(Me.Label128)
Me.GroupBox38.Controls.Add(Me.TextBox145)
Me.GroupBox38.Controls.Add(Me.Label81)
Me.GroupBox38.Controls.Add(Me.Label46)
Me.GroupBox38.Controls.Add(Me.Label127)
Me.GroupBox38.Controls.Add(Me.Label126)
Me.GroupBox38.Controls.Add(Me.Label125)
Me.GroupBox38.Controls.Add(Me.Label100)
Me.GroupBox38.Controls.Add(Me.Label83)
```

```
Me.GroupBox38.Controls.Add(Me.Label88)
Me.GroupBox38.Controls.Add(Me.Label87)
Me.GroupBox38.Controls.Add(Me.Label86)
Me.GroupBox38.Controls.Add(Me.Label85)
Me.GroupBox38.Controls.Add(Me.Label84)
Me.GroupBox38.Controls.Add(Me.Label45)
Me.GroupBox38.Controls.Add(Me.Label82)
Me.GroupBox38.Controls.Add(Me.AxMSChart3)
Me.GroupBox38.Controls.Add(Me.AxMSChart1)
Me.GroupBox38.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.GroupBox38.Location = New System.Drawing.Point(248, 304)
Me.GroupBox38.Name = "GroupBox38"
Me.GroupBox38.Size = New System.Drawing.Size(344, 432)
Me.GroupBox38.TabIndex = 273
Me.GroupBox38.TabStop = False
Me.GroupBox38.Text = "System Performance"
'
'Label94
'
Me.Label94.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label94.Location = New System.Drawing.Point(258, 212)
Me.Label94.Name = "Label94"
Me.Label94.Size = New System.Drawing.Size(84, 32)
Me.Label94.TabIndex = 284
Me.Label94.Text = "Successful Policy Number"
'
'Label129
'
Me.Label129.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label129.Location = New System.Drawing.Point(24, 178)
Me.Label129.Name = "Label129"
Me.Label129.Size = New System.Drawing.Size(24, 16)
Me.Label129.TabIndex = 303
Me.Label129.Text = "0"
'
'Label130
'
Me.Label130.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label130.Location = New System.Drawing.Point(24, 152)
Me.Label130.Name = "Label130"
Me.Label130.Size = New System.Drawing.Size(24, 16)
Me.Label130.TabIndex = 302
Me.Label130.Text = "4"
'
'Label131
'
Me.Label131.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label131.Location = New System.Drawing.Point(24, 120)
Me.Label131.Name = "Label131"
Me.Label131.Size = New System.Drawing.Size(24, 16)
Me.Label131.TabIndex = 301
Me.Label131.Text = "8"
'
'Label132
'
Me.Label132.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label132.Location = New System.Drawing.Point(22, 92)
Me.Label132.Name = "Label132"
Me.Label132.Size = New System.Drawing.Size(24, 16)
Me.Label132.TabIndex = 300
Me.Label132.Text = "12"
'
'Label95
```

```
'
Me.Label95.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Bold)
Me.Label95.Location = New System.Drawing.Point(278, 194)
Me.Label95.Name = "Label95"
Me.Label95.Size = New System.Drawing.Size(24, 16)
Me.Label95.TabIndex = 298
Me.Label95.Text = "50"
'
'Label96
'
Me.Label96.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Bold)
Me.Label96.Location = New System.Drawing.Point(230, 194)
Me.Label96.Name = "Label96"
Me.Label96.Size = New System.Drawing.Size(24, 16)
Me.Label96.TabIndex = 299
Me.Label96.Text = "40"
'
'Label97
'
Me.Label97.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label97.Location = New System.Drawing.Point(182, 194)
Me.Label97.Name = "Label97"
Me.Label97.Size = New System.Drawing.Size(24, 16)
Me.Label97.TabIndex = 297
Me.Label97.Text = "30"
'
'Label98
'
Me.Label98.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label98.Location = New System.Drawing.Point(134, 194)
Me.Label98.Name = "Label98"
Me.Label98.Size = New System.Drawing.Size(24, 16)
Me.Label98.TabIndex = 296
Me.Label98.Text = "20"
'
'Label99
'
Me.Label99.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label99.Location = New System.Drawing.Point(86, 194)
Me.Label99.Name = "Label99"
Me.Label99.Size = New System.Drawing.Size(24, 16)
Me.Label99.TabIndex = 295
Me.Label99.Text = "10"
'
'Label128
'
Me.Label128.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label128.Location = New System.Drawing.Point(182, 264)
Me.Label128.Name = "Label128"
Me.Label128.Size = New System.Drawing.Size(24, 16)
Me.Label128.TabIndex = 279
Me.Label128.Text = "(%)"
'
'TextBox145
'
Me.TextBox145.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox145.Location = New System.Drawing.Point(150, 262)
Me.TextBox145.Name = "TextBox145"
Me.TextBox145.Size = New System.Drawing.Size(32, 20)
Me.TextBox145.TabIndex = 279
Me.TextBox145.Text = "0"
'
```

```
'Label81
'
Me.Label81.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label81.Location = New System.Drawing.Point(24, 378)
Me.Label81.Name = "Label81"
Me.Label81.Size = New System.Drawing.Size(24, 16)
Me.Label81.TabIndex = 294
Me.Label81.Text = "0"
'
'Label46
'
Me.Label46.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label46.Location = New System.Drawing.Point(22, 360)
Me.Label46.Name = "Label46"
Me.Label46.Size = New System.Drawing.Size(24, 16)
Me.Label46.TabIndex = 293
Me.Label46.Text = "20"
'
'Label127
'
Me.Label127.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label127.Location = New System.Drawing.Point(22, 340)
Me.Label127.Name = "Label127"
Me.Label127.Size = New System.Drawing.Size(24, 16)
Me.Label127.TabIndex = 292
Me.Label127.Text = "40"
'
'Label126
'
Me.Label126.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label126.Location = New System.Drawing.Point(22, 322)
Me.Label126.Name = "Label126"
Me.Label126.Size = New System.Drawing.Size(24, 16)
Me.Label126.TabIndex = 291
Me.Label126.Text = "60"
'
'Label125
'
Me.Label125.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label125.Location = New System.Drawing.Point(22, 302)
Me.Label125.Name = "Label125"
Me.Label125.Size = New System.Drawing.Size(24, 16)
Me.Label125.TabIndex = 290
Me.Label125.Text = "80"
'
'Label100
'
Me.Label100.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label100.Location = New System.Drawing.Point(20, 280)
Me.Label100.Name = "Label100"
Me.Label100.Size = New System.Drawing.Size(24, 16)
Me.Label100.TabIndex = 289
Me.Label100.Text = "100"
'
'Label83
'
Me.Label83.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label83.Location = New System.Drawing.Point(274, 400)
Me.Label83.Name = "Label83"
Me.Label83.Size = New System.Drawing.Size(54, 28)
Me.Label83.TabIndex = 279
Me.Label83.Text = "Learning Episode"
'
```

```
'Label88
'
Me.Label88.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Bold)
Me.Label88.Location = New System.Drawing.Point(280, 384)
Me.Label88.Name = "Label88"
Me.Label88.Size = New System.Drawing.Size(24, 16)
Me.Label88.TabIndex = 281
Me.Label88.Text = "50"
'
'Label87
'
Me.Label87.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Bold)
Me.Label87.Location = New System.Drawing.Point(232, 384)
Me.Label87.Name = "Label87"
Me.Label87.Size = New System.Drawing.Size(24, 16)
Me.Label87.TabIndex = 282
Me.Label87.Text = "40"
'
'Label86
'
Me.Label86.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label86.Location = New System.Drawing.Point(184, 384)
Me.Label86.Name = "Label86"
Me.Label86.Size = New System.Drawing.Size(24, 16)
Me.Label86.TabIndex = 281
Me.Label86.Text = "30"
'
'Label85
'
Me.Label85.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label85.Location = New System.Drawing.Point(136, 384)
Me.Label85.Name = "Label85"
Me.Label85.Size = New System.Drawing.Size(24, 16)
Me.Label85.TabIndex = 280
Me.Label85.Text = "20"
'
'Label84
'
Me.Label84.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label84.Location = New System.Drawing.Point(88, 384)
Me.Label84.Name = "Label84"
Me.Label84.Size = New System.Drawing.Size(24, 16)
Me.Label84.TabIndex = 279
Me.Label84.Text = "10"
'
'Label45
'
Me.Label45.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label45.Location = New System.Drawing.Point(2, 136)
Me.Label45.Name = "Label45"
Me.Label45.Size = New System.Drawing.Size(40, 16)
Me.Label45.TabIndex = 274
Me.Label45.Text = "[sec]"
'
'Label82
'
Me.Label82.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.Label82.Location = New System.Drawing.Point(2, 328)
Me.Label82.Name = "Label82"
Me.Label82.Size = New System.Drawing.Size(24, 16)
Me.Label82.TabIndex = 278
Me.Label82.Text = "[%]"
'
```

```
'AxMSChart3
'
Me.AxMSChart3.ContainingControl = Me
Me.AxMSChart3.DataSource = Nothing
Me.AxMSChart3.Location = New System.Drawing.Point(24, 224)
Me.AxMSChart3.Name = "AxMSChart3"
Me.AxMSChart3.OcxState = CType(resources.GetObject("AxMSChart3.OcxState"),
System.Windows.Forms.AxHost.State)
Me.AxMSChart3.Size = New System.Drawing.Size(296, 184)
Me.AxMSChart3.TabIndex = 278
'
'AxMSChart1
'
Me.AxMSChart1.ContainingControl = Me
Me.AxMSChart1.DataSource = Nothing
Me.AxMSChart1.Location = New System.Drawing.Point(24, 32)
Me.AxMSChart1.Name = "AxMSChart1"
Me.AxMSChart1.OcxState = CType(resources.GetObject("AxMSChart1.OcxState"),
System.Windows.Forms.AxHost.State)
Me.AxMSChart1.Size = New System.Drawing.Size(296, 184)
Me.AxMSChart1.TabIndex = 273
'
'GroupBox31
'
Me.GroupBox31.Controls.Add(Me.Button52)
Me.GroupBox31.Controls.Add(Me.TextBox141)
Me.GroupBox31.Controls.Add(Me.TextBox114)
Me.GroupBox31.Controls.Add(Me.Button57)
Me.GroupBox31.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.GroupBox31.Location = New System.Drawing.Point(16, 580)
Me.GroupBox31.Name = "GroupBox31"
Me.GroupBox31.Size = New System.Drawing.Size(216, 126)
Me.GroupBox31.TabIndex = 265
Me.GroupBox31.TabStop = False
Me.GroupBox31.Text = "Timer"
'
'Button52
'
Me.Button52.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(192, Byte), CType(128, Byte))
Me.Button52.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button52.Location = New System.Drawing.Point(120, 24)
Me.Button52.Name = "Button52"
Me.Button52.Size = New System.Drawing.Size(64, 32)
Me.Button52.TabIndex = 242
Me.Button52.Text = "Stop"
'
'TextBox141
'
Me.TextBox141.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox141.Location = New System.Drawing.Point(16, 72)
Me.TextBox141.Multiline = True
Me.TextBox141.Name = "TextBox141"
Me.TextBox141.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox141.Size = New System.Drawing.Size(112, 40)
Me.TextBox141.TabIndex = 197
Me.TextBox141.Text = ""
'
'TextBox114
'
Me.TextBox114.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.TextBox114.Location = New System.Drawing.Point(152, 80)
Me.TextBox114.Name = "TextBox114"
Me.TextBox114.Size = New System.Drawing.Size(40, 20)
Me.TextBox114.TabIndex = 193
Me.TextBox114.Text = "0"
 '
 'Button57
 '
Me.Button57.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(192, Byte), CType(128, Byte))
Me.Button57.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button57.Location = New System.Drawing.Point(40, 24)
Me.Button57.Name = "Button57"
Me.Button57.Size = New System.Drawing.Size(64, 32)
Me.Button57.TabIndex = 243
Me.Button57.Text = "Reset Timer"
 '
 'GroupBox18
 '
Me.GroupBox18.Controls.Add(Me.TextBox146)
Me.GroupBox18.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.GroupBox18.Location = New System.Drawing.Point(616, 14)
Me.GroupBox18.Name = "GroupBox18"
Me.GroupBox18.Size = New System.Drawing.Size(512, 94)
Me.GroupBox18.TabIndex = 259
Me.GroupBox18.TabStop = False
Me.GroupBox18.Text = "Collaboration Level"
 '
 'TextBox146
 '
Me.TextBox146.AllowDrop = True
Me.TextBox146.AutoSize = False
Me.TextBox146.BackColor = System.Drawing.SystemColors.Control
Me.TextBox146.BorderStyle = System.Windows.Forms.BorderStyle.None
Me.TextBox146.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox146.ForeColor = System.Drawing.Color.Red
Me.TextBox146.Location = New System.Drawing.Point(40, 32)
Me.TextBox146.Multiline = True
Me.TextBox146.Name = "TextBox146"
Me.TextBox146.Size = New System.Drawing.Size(424, 40)
Me.TextBox146.TabIndex = 181
Me.TextBox146.Text = "Autonomous Mode - No Human Intervention is Required!"
Me.TextBox146.TextAlign = System.Windows.Forms.HorizontalAlignment.Center
 '
 'GroupBox29
 '
Me.GroupBox29.Controls.Add(Me.ComboBox2)
Me.GroupBox29.Controls.Add(Me.Label109)
Me.GroupBox29.Controls.Add(Me.ComboBox6)
Me.GroupBox29.Controls.Add(Me.Button50)
Me.GroupBox29.Controls.Add(Me.Button51)
Me.GroupBox29.Controls.Add(Me.ProgressBar2)
Me.GroupBox29.Controls.Add(Me.Label91)
Me.GroupBox29.Controls.Add(Me.Label55)
Me.GroupBox29.Controls.Add(Me.ProgressBar1)
Me.GroupBox29.Controls.Add(Me.Button45)
Me.GroupBox29.Controls.Add(Me.Button40)
Me.GroupBox29.Controls.Add(Me.Button47)
Me.GroupBox29.Controls.Add(Me.Button54)
Me.GroupBox29.Controls.Add(Me.Button53)
Me.GroupBox29.Controls.Add(Me.Button56)
Me.GroupBox29.Controls.Add(Me.GroupBox30)
```

```
Me.GroupBox29.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox29.Location = New System.Drawing.Point(16, 136)
Me.GroupBox29.Name = "GroupBox29"
Me.GroupBox29.Size = New System.Drawing.Size(216, 434)
Me.GroupBox29.TabIndex = 190
Me.GroupBox29.TabStop = False
Me.GroupBox29.Text = "Control"
 '
 'ComboBox2
 '
Me.ComboBox2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox2.Items.AddRange(New Object() {"Regular", "Rotated"})
Me.ComboBox2.Location = New System.Drawing.Point(16, 112)
Me.ComboBox2.Name = "ComboBox2"
Me.ComboBox2.Size = New System.Drawing.Size(80, 22)
Me.ComboBox2.TabIndex = 243
 '
 'Label109
 '
Me.Label109.Font = New System.Drawing.Font("Arial", 8.25!, CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
Me.Label109.Location = New System.Drawing.Point(8, 136)
Me.Label109.Name = "Label109"
Me.Label109.Size = New System.Drawing.Size(80, 16)
Me.Label109.TabIndex = 242
Me.Label109.Text = "Reward Type"
 '
 'ComboBox6
 '
Me.ComboBox6.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox6.Items.AddRange(New Object() {"Peak", "Cummulative"})
Me.ComboBox6.Location = New System.Drawing.Point(8, 160)
Me.ComboBox6.Name = "ComboBox6"
Me.ComboBox6.Size = New System.Drawing.Size(96, 22)
Me.ComboBox6.TabIndex = 241
 '
 'Button50
 '
Me.Button50.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button50.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button50.Location = New System.Drawing.Point(112, 328)
Me.Button50.Name = "Button50"
Me.Button50.Size = New System.Drawing.Size(88, 48)
Me.Button50.TabIndex = 235
Me.Button50.Text = "System Creates Policy"
 '
 'Button51
 '
Me.Button51.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button51.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button51.Location = New System.Drawing.Point(16, 328)
Me.Button51.Name = "Button51"
Me.Button51.Size = New System.Drawing.Size(88, 48)
Me.Button51.TabIndex = 234
Me.Button51.Text = "Human Creates Policy"
 '
 'ProgressBar2
```

```
'
Me.ProgressBar2.Location = New System.Drawing.Point(16, 408)
Me.ProgressBar2.Name = "ProgressBar2"
Me.ProgressBar2.Size = New System.Drawing.Size(184, 16)
Me.ProgressBar2.TabIndex = 229
'
'Label91
'
Me.Label91.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label91.Location = New System.Drawing.Point(8, 384)
Me.Label91.Name = "Label91"
Me.Label91.Size = New System.Drawing.Size(112, 16)
Me.Label91.TabIndex = 228
Me.Label91.Text = "Policy Creation:"
'
'Label55
'
Me.Label55.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label55.Location = New System.Drawing.Point(8, 280)
Me.Label55.Name = "Label55"
Me.Label55.Size = New System.Drawing.Size(112, 16)
Me.Label55.TabIndex = 227
Me.Label55.Text = "RL Calculations:"
'
'ProgressBar1
'
Me.ProgressBar1.Location = New System.Drawing.Point(16, 304)
Me.ProgressBar1.Name = "ProgressBar1"
Me.ProgressBar1.Size = New System.Drawing.Size(184, 16)
Me.ProgressBar1.TabIndex = 226
'
'Button45
'
Me.Button45.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(255, Byte), CType(192, Byte))
Me.Button45.Cursor = System.Windows.Forms.Cursors.Default
Me.Button45.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button45.ForeColor = System.Drawing.SystemColors.ControlText
Me.Button45.Location = New System.Drawing.Point(24, 24)
Me.Button45.Name = "Button45"
Me.Button45.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button45.Size = New System.Drawing.Size(72, 40)
Me.Button45.TabIndex = 189
Me.Button45.Text = "Connect to Robot"
'
'Button40
'
Me.Button40.BackColor = System.Drawing.Color.Red
Me.Button40.Cursor = System.Windows.Forms.Cursors.Default
Me.Button40.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button40.ForeColor = System.Drawing.Color.Yellow
Me.Button40.Location = New System.Drawing.Point(120, 24)
Me.Button40.Name = "Button40"
Me.Button40.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button40.Size = New System.Drawing.Size(72, 40)
Me.Button40.TabIndex = 188
Me.Button40.Text = "Reset"
'
'Button47
'
```

```
Me.Button47.BackColor = System.Drawing.SystemColors.InactiveCaptionText
Me.Button47.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button47.Location = New System.Drawing.Point(104, 728)
Me.Button47.Name = "Button47"
Me.Button47.Size = New System.Drawing.Size(88, 48)
Me.Button47.TabIndex = 220
Me.Button47.Text = "Use the Rewarded  Policy"
 '
 'Button54
 '
Me.Button54.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button54.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button54.Location = New System.Drawing.Point(120, 72)
Me.Button54.Name = "Button54"
Me.Button54.Size = New System.Drawing.Size(72, 40)
Me.Button54.TabIndex = 238
Me.Button54.Text = "Execute Shaking"
 '
 'Button53
 '
Me.Button53.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button53.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button53.Location = New System.Drawing.Point(24, 72)
Me.Button53.Name = "Button53"
Me.Button53.Size = New System.Drawing.Size(72, 40)
Me.Button53.TabIndex = 236
Me.Button53.Text = "Grasp Bag"
 '
 'Button56
 '
Me.Button56.BackColor = System.Drawing.Color.FromArgb(CType(255, Byte), CType(192, Byte), CType(128, Byte))
Me.Button56.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button56.Location = New System.Drawing.Point(120, 144)
Me.Button56.Name = "Button56"
Me.Button56.Size = New System.Drawing.Size(72, 40)
Me.Button56.TabIndex = 240
Me.Button56.Text = "Calculate Reward"
 '
 'GroupBox30
 '
Me.GroupBox30.Controls.Add(Me.Label101)
Me.GroupBox30.Controls.Add(Me.TextBox142)
Me.GroupBox30.Controls.Add(Me.Label54)
Me.GroupBox30.Controls.Add(Me.TextBox132)
Me.GroupBox30.Font = New System.Drawing.Font("Arial", 9.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox30.Location = New System.Drawing.Point(16, 192)
Me.GroupBox30.Name = "GroupBox30"
Me.GroupBox30.Size = New System.Drawing.Size(184, 80)
Me.GroupBox30.TabIndex = 191
Me.GroupBox30.TabStop = False
Me.GroupBox30.Text = "Rewards"
 '
 'Label101
 '
Me.Label101.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label101.Location = New System.Drawing.Point(96, 24)
Me.Label101.Name = "Label101"
```

```
Me.Label101.Size = New System.Drawing.Size(80, 16)
Me.Label101.TabIndex = 225
Me.Label101.Text = "Total Time"
'
'TextBox142
'
Me.TextBox142.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox142.Location = New System.Drawing.Point(96, 48)
Me.TextBox142.Name = "TextBox142"
Me.TextBox142.Size = New System.Drawing.Size(72, 20)
Me.TextBox142.TabIndex = 224
Me.TextBox142.Text = "000"
'
'Label54
'
Me.Label54.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label54.Location = New System.Drawing.Point(8, 24)
Me.Label54.Name = "Label54"
Me.Label54.Size = New System.Drawing.Size(80, 16)
Me.Label54.TabIndex = 223
Me.Label54.Text = "Total Reward"
'
'TextBox132
'
Me.TextBox132.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox132.Location = New System.Drawing.Point(8, 48)
Me.TextBox132.Name = "TextBox132"
Me.TextBox132.Size = New System.Drawing.Size(72, 20)
Me.TextBox132.TabIndex = 222
Me.TextBox132.Text = "0"
'
'GroupBox19
'
Me.GroupBox19.Controls.Add(Me.GroupBox20)
Me.GroupBox19.Controls.Add(Me.GroupBox34)
Me.GroupBox19.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox19.Location = New System.Drawing.Point(622, 120)
Me.GroupBox19.Name = "GroupBox19"
Me.GroupBox19.Size = New System.Drawing.Size(496, 656)
Me.GroupBox19.TabIndex = 182
Me.GroupBox19.TabStop = False
Me.GroupBox19.Text = "Human-Robot Collaboration Control"
'
'GroupBox20
'
Me.GroupBox20.Controls.Add(Me.TextBox135)
Me.GroupBox20.Font = New System.Drawing.Font("Arial", 9.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox20.Location = New System.Drawing.Point(16, 576)
Me.GroupBox20.Name = "GroupBox20"
Me.GroupBox20.Size = New System.Drawing.Size(464, 72)
Me.GroupBox20.TabIndex = 203
Me.GroupBox20.TabStop = False
Me.GroupBox20.Text = "Policy Success Notification"
'
'TextBox135
'
Me.TextBox135.AllowDrop = True
Me.TextBox135.AutoSize = False
```

```
Me.TextBox135.BackColor = System.Drawing.SystemColors.Control
Me.TextBox135.BorderStyle = System.Windows.Forms.BorderStyle.None
Me.TextBox135.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox135.ForeColor = System.Drawing.Color.Red
Me.TextBox135.Location = New System.Drawing.Point(20, 32)
Me.TextBox135.Multiline = True
Me.TextBox135.Name = "TextBox135"
Me.TextBox135.Size = New System.Drawing.Size(424, 24)
Me.TextBox135.TabIndex = 182
Me.TextBox135.Text = ""
Me.TextBox135.TextAlign = System.Windows.Forms.HorizontalAlignment.Center
 '
 'GroupBox34
 '
Me.GroupBox34.Controls.Add(Me.GroupBox4)
Me.GroupBox34.Controls.Add(Me.GroupBox33)
Me.GroupBox34.Controls.Add(Me.GroupBox35)
Me.GroupBox34.Controls.Add(Me.GroupBox36)
Me.GroupBox34.Controls.Add(Me.GroupBox37)
Me.GroupBox34.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox34.Location = New System.Drawing.Point(8, 40)
Me.GroupBox34.Name = "GroupBox34"
Me.GroupBox34.Size = New System.Drawing.Size(472, 528)
Me.GroupBox34.TabIndex = 202
Me.GroupBox34.TabStop = False
Me.GroupBox34.Text = "Human Decision Making - Suggest a Shaking Policy"
 '
 'GroupBox4
 '
Me.GroupBox4.Controls.Add(Me.Label37)
Me.GroupBox4.Controls.Add(Me.ComboBox14)
Me.GroupBox4.Controls.Add(Me.ComboBox15)
Me.GroupBox4.Controls.Add(Me.ComboBox16)
Me.GroupBox4.Controls.Add(Me.Label35)
Me.GroupBox4.Controls.Add(Me.ComboBox5)
Me.GroupBox4.Controls.Add(Me.ComboBox3)
Me.GroupBox4.Controls.Add(Me.Label29)
Me.GroupBox4.Controls.Add(Me.Label36)
Me.GroupBox4.Controls.Add(Me.ComboBox4)
Me.GroupBox4.Controls.Add(Me.Label38)
Me.GroupBox4.Location = New System.Drawing.Point(16, 336)
Me.GroupBox4.Name = "GroupBox4"
Me.GroupBox4.Size = New System.Drawing.Size(440, 136)
Me.GroupBox4.TabIndex = 236
Me.GroupBox4.TabStop = False
Me.GroupBox4.Text = "Q Table Weight Control (of X, Y, and Z) "
 '
 'Label37
 '
Me.Label37.Font = New System.Drawing.Font("Arial", 8.25!, CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
Me.Label37.Location = New System.Drawing.Point(304, 24)
Me.Label37.Name = "Label37"
Me.Label37.Size = New System.Drawing.Size(88, 16)
Me.Label37.TabIndex = 232
Me.Label37.Text = "Swing Control"
 '
 'ComboBox14
 '
```

```
Me.ComboBox14.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox14.ItemHeight = 14
Me.ComboBox14.Items.AddRange(New Object() {"Much Higher Swings", "Higher Swings", "No Change", "Lower
Swings", "Much Lower Swings"})
Me.ComboBox14.Location = New System.Drawing.Point(288, 40)
Me.ComboBox14.Name = "ComboBox14"
Me.ComboBox14.Size = New System.Drawing.Size(128, 22)
Me.ComboBox14.TabIndex = 231
'
'ComboBox15
'
Me.ComboBox15.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox15.ItemHeight = 14
Me.ComboBox15.Items.AddRange(New Object() {"Much Higher Swings", "Higher Swings", "No Change", "Lower
Swings", "Much Lower Swings"})
Me.ComboBox15.Location = New System.Drawing.Point(288, 72)
Me.ComboBox15.Name = "ComboBox15"
Me.ComboBox15.Size = New System.Drawing.Size(128, 22)
Me.ComboBox15.TabIndex = 230
'
'ComboBox16
'
Me.ComboBox16.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox16.ItemHeight = 14
Me.ComboBox16.Items.AddRange(New Object() {"Much Higher Swings", "Higher Swings", "No Change", "Lower
Swings", "Much Lower Swings"})
Me.ComboBox16.Location = New System.Drawing.Point(288, 104)
Me.ComboBox16.Name = "ComboBox16"
Me.ComboBox16.Size = New System.Drawing.Size(128, 22)
Me.ComboBox16.TabIndex = 229
'
'Label35
'
Me.Label35.Font = New System.Drawing.Font("Arial", 8.25!, CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
Me.Label35.Location = New System.Drawing.Point(152, 24)
Me.Label35.Name = "Label35"
Me.Label35.Size = New System.Drawing.Size(88, 16)
Me.Label35.TabIndex = 228
Me.Label35.Text = "Center Control"
'
'ComboBox5
'
Me.ComboBox5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox5.ItemHeight = 14
Me.ComboBox5.Items.AddRange(New Object() {"A Lot Higher", "A Little Higher", "Keep Current", "A Little Lower",
"A Lot Lower"})
Me.ComboBox5.Location = New System.Drawing.Point(136, 104)
Me.ComboBox5.Name = "ComboBox5"
Me.ComboBox5.Size = New System.Drawing.Size(128, 22)
Me.ComboBox5.TabIndex = 227
'
'ComboBox3
'
Me.ComboBox3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox3.ItemHeight = 14
```

```
Me.ComboBox3.Items.AddRange(New Object() {"A Lot Higher", "A Little Higher", "Keep Current", "A Little Lower",
"A Lot Lower"})
Me.ComboBox3.Location = New System.Drawing.Point(136, 40)
Me.ComboBox3.Name = "ComboBox3"
Me.ComboBox3.Size = New System.Drawing.Size(128, 22)
Me.ComboBox3.TabIndex = 226
'
'Label29
'
Me.Label29.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label29.Location = New System.Drawing.Point(40, 80)
Me.Label29.Name = "Label29"
Me.Label29.Size = New System.Drawing.Size(64, 16)
Me.Label29.TabIndex = 188
Me.Label29.Text = "Left-Right:"
'
'Label36
'
Me.Label36.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label36.Location = New System.Drawing.Point(16, 48)
Me.Label36.Name = "Label36"
Me.Label36.Size = New System.Drawing.Size(120, 16)
Me.Label36.TabIndex = 178
Me.Label36.Text = "Forward-Backward:"
'
'ComboBox4
'
Me.ComboBox4.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox4.ItemHeight = 14
Me.ComboBox4.Items.AddRange(New Object() {"A Lot Higher", "A Little Higher", "Keep Current", "A Little Lower",
"A Lot Lower"})
Me.ComboBox4.Location = New System.Drawing.Point(136, 72)
Me.ComboBox4.Name = "ComboBox4"
Me.ComboBox4.Size = New System.Drawing.Size(128, 22)
Me.ComboBox4.TabIndex = 225
'
'Label38
'
Me.Label38.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label38.Location = New System.Drawing.Point(40, 112)
Me.Label38.Name = "Label38"
Me.Label38.Size = New System.Drawing.Size(64, 16)
Me.Label38.TabIndex = 192
Me.Label38.Text = "Up-Down:"
'
'GroupBox33
'
Me.GroupBox33.Controls.Add(Me.CheckBox10)
Me.GroupBox33.Controls.Add(Me.CheckBox7)
Me.GroupBox33.Controls.Add(Me.CheckBox11)
Me.GroupBox33.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold)
Me.GroupBox33.Location = New System.Drawing.Point(16, 480)
Me.GroupBox33.Name = "GroupBox33"
Me.GroupBox33.Size = New System.Drawing.Size(440, 40)
Me.GroupBox33.TabIndex = 235
Me.GroupBox33.TabStop = False
Me.GroupBox33.Text = "Eliminate Axis"
'
'CheckBox10
```

```
'
Me.CheckBox10.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.CheckBox10.Location = New System.Drawing.Point(344, 16)
Me.CheckBox10.Name = "CheckBox10"
Me.CheckBox10.Size = New System.Drawing.Size(80, 16)
Me.CheckBox10.TabIndex = 259
Me.CheckBox10.Text = "Up-Down"
'
'CheckBox7
'
Me.CheckBox7.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.CheckBox7.Location = New System.Drawing.Point(256, 16)
Me.CheckBox7.Name = "CheckBox7"
Me.CheckBox7.Size = New System.Drawing.Size(80, 16)
Me.CheckBox7.TabIndex = 258
Me.CheckBox7.Text = "Left-Right"
'
'CheckBox11
'
Me.CheckBox11.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.CheckBox11.Location = New System.Drawing.Point(120, 16)
Me.CheckBox11.Name = "CheckBox11"
Me.CheckBox11.Size = New System.Drawing.Size(136, 16)
Me.CheckBox11.TabIndex = 257
Me.CheckBox11.Text = "Forward-Backward"
'
'GroupBox35
'
Me.GroupBox35.Controls.Add(Me.ComboBox8)
Me.GroupBox35.Controls.Add(Me.TextBox90)
Me.GroupBox35.Controls.Add(Me.Label113)
Me.GroupBox35.Controls.Add(Me.Label114)
Me.GroupBox35.Controls.Add(Me.TextBox136)
Me.GroupBox35.Controls.Add(Me.Label115)
Me.GroupBox35.Controls.Add(Me.ComboBox9)
Me.GroupBox35.Controls.Add(Me.Label116)
Me.GroupBox35.Controls.Add(Me.Label117)
Me.GroupBox35.Controls.Add(Me.ComboBox10)
Me.GroupBox35.Controls.Add(Me.TextBox137)
Me.GroupBox35.Controls.Add(Me.Label118)
Me.GroupBox35.Location = New System.Drawing.Point(16, 184)
Me.GroupBox35.Name = "GroupBox35"
Me.GroupBox35.Size = New System.Drawing.Size(440, 144)
Me.GroupBox35.TabIndex = 231
Me.GroupBox35.TabStop = False
Me.GroupBox35.Text = "Axis Speed Control (of X, Y, and Z) - 100 to 1500 mm / sec"
'
'ComboBox8
'
Me.ComboBox8.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox8.ItemHeight = 14
Me.ComboBox8.Items.AddRange(New Object() {"A Lot Faster", "A Little Faster", "Keep Current", "A Little Slower",
"A Lot Slower"})
Me.ComboBox8.Location = New System.Drawing.Point(168, 64)
Me.ComboBox8.Name = "ComboBox8"
Me.ComboBox8.Size = New System.Drawing.Size(128, 22)
Me.ComboBox8.TabIndex = 226
'
'TextBox90
'
Me.TextBox90.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.TextBox90.Location = New System.Drawing.Point(312, 104)
Me.TextBox90.Name = "TextBox90"
Me.TextBox90.Size = New System.Drawing.Size(40, 20)
Me.TextBox90.TabIndex = 193
Me.TextBox90.Text = ""
'
'Label113
'
Me.Label113.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label113.Location = New System.Drawing.Point(32, 72)
Me.Label113.Name = "Label113"
Me.Label113.Size = New System.Drawing.Size(136, 16)
Me.Label113.TabIndex = 188
Me.Label113.Text = "Left-Right Speed:"
'
'Label114
'
Me.Label114.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label114.Location = New System.Drawing.Point(360, 32)
Me.Label114.Name = "Label114"
Me.Label114.Size = New System.Drawing.Size(64, 24)
Me.Label114.TabIndex = 228
Me.Label114.Text = "mm / sec"
'
'TextBox136
'
Me.TextBox136.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox136.Location = New System.Drawing.Point(312, 24)
Me.TextBox136.Name = "TextBox136"
Me.TextBox136.Size = New System.Drawing.Size(40, 20)
Me.TextBox136.TabIndex = 180
Me.TextBox136.Text = ""
'
'Label115
'
Me.Label115.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label115.Location = New System.Drawing.Point(8, 32)
Me.Label115.Name = "Label115"
Me.Label115.Size = New System.Drawing.Size(152, 16)
Me.Label115.TabIndex = 178
Me.Label115.Text = "Forward-Backward Speed:"
'
'ComboBox9
'
Me.ComboBox9.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox9.ItemHeight = 14
Me.ComboBox9.Items.AddRange(New Object() {"A Lot Faster", "A Little Faster", "Keep Current", "A Little Slower",
"A Lot Slower"})
Me.ComboBox9.Location = New System.Drawing.Point(168, 24)
Me.ComboBox9.Name = "ComboBox9"
Me.ComboBox9.Size = New System.Drawing.Size(128, 22)
Me.ComboBox9.TabIndex = 225
'
'Label116
'
Me.Label116.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label116.Location = New System.Drawing.Point(360, 72)
```

```
Me.Label116.Name = "Label116"
Me.Label116.Size = New System.Drawing.Size(64, 24)
Me.Label116.TabIndex = 229
Me.Label116.Text = "mm / sec"
'
'Label117
'
Me.Label117.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label117.Location = New System.Drawing.Point(32, 112)
Me.Label117.Name = "Label117"
Me.Label117.Size = New System.Drawing.Size(128, 16)
Me.Label117.TabIndex = 192
Me.Label117.Text = "Up-Down Speed:"
'
'ComboBox10
'
Me.ComboBox10.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox10.ItemHeight = 14
Me.ComboBox10.Items.AddRange(New Object() {"A Lot Faster", "A Little Faster", "Keep Current", "A Little Slower",
"A Lot Slower"})
Me.ComboBox10.Location = New System.Drawing.Point(168, 104)
Me.ComboBox10.Name = "ComboBox10"
Me.ComboBox10.Size = New System.Drawing.Size(128, 22)
Me.ComboBox10.TabIndex = 227
'
'TextBox137
'
Me.TextBox137.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox137.Location = New System.Drawing.Point(312, 64)
Me.TextBox137.Name = "TextBox137"
Me.TextBox137.Size = New System.Drawing.Size(40, 20)
Me.TextBox137.TabIndex = 189
Me.TextBox137.Text = ""
'
'Label118
'
Me.Label118.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label118.Location = New System.Drawing.Point(360, 112)
Me.Label118.Name = "Label118"
Me.Label118.Size = New System.Drawing.Size(64, 24)
Me.Label118.TabIndex = 230
Me.Label118.Text = "mm / sec"
'
'GroupBox36
'
Me.GroupBox36.Controls.Add(Me.Button60)
Me.GroupBox36.Controls.Add(Me.Button61)
Me.GroupBox36.Controls.Add(Me.Button62)
Me.GroupBox36.Controls.Add(Me.Button63)
Me.GroupBox36.Font = New System.Drawing.Font("Arial", 9.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox36.Location = New System.Drawing.Point(56, 552)
Me.GroupBox36.Name = "GroupBox36"
Me.GroupBox36.Size = New System.Drawing.Size(256, 200)
Me.GroupBox36.TabIndex = 187
Me.GroupBox36.TabStop = False
Me.GroupBox36.Text = "Run Policy"
'
'Button60
```

```
'
Me.Button60.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button60.Location = New System.Drawing.Point(88, 80)
Me.Button60.Name = "Button60"
Me.Button60.Size = New System.Drawing.Size(80, 48)
Me.Button60.TabIndex = 169
Me.Button60.Text = "Grasping Point"
'
'Button61
'
Me.Button61.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button61.Location = New System.Drawing.Point(8, 24)
Me.Button61.Name = "Button61"
Me.Button61.Size = New System.Drawing.Size(112, 48)
Me.Button61.TabIndex = 168
Me.Button61.Text = "Grasp Bag"
'
'Button62
'
Me.Button62.BackColor = System.Drawing.Color.Red
Me.Button62.Cursor = System.Windows.Forms.Cursors.Default
Me.Button62.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button62.ForeColor = System.Drawing.Color.Yellow
Me.Button62.Location = New System.Drawing.Point(40, 136)
Me.Button62.Name = "Button62"
Me.Button62.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button62.Size = New System.Drawing.Size(168, 48)
Me.Button62.TabIndex = 166
Me.Button62.Text = "Emergency Stop"
'
'Button63
'
Me.Button63.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button63.Location = New System.Drawing.Point(136, 24)
Me.Button63.Name = "Button63"
Me.Button63.Size = New System.Drawing.Size(112, 48)
Me.Button63.TabIndex = 165
Me.Button63.Text = "Create Policy"
'
'GroupBox37
'
Me.GroupBox37.Controls.Add(Me.ComboBox11)
Me.GroupBox37.Controls.Add(Me.Label119)
Me.GroupBox37.Controls.Add(Me.Label120)
Me.GroupBox37.Controls.Add(Me.Label121)
Me.GroupBox37.Controls.Add(Me.Label122)
Me.GroupBox37.Controls.Add(Me.TextBox138)
Me.GroupBox37.Controls.Add(Me.TextBox139)
Me.GroupBox37.Controls.Add(Me.TextBox140)
Me.GroupBox37.Controls.Add(Me.ComboBox12)
Me.GroupBox37.Controls.Add(Me.ComboBox13)
Me.GroupBox37.Controls.Add(Me.Label123)
Me.GroupBox37.Controls.Add(Me.Label124)
Me.GroupBox37.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold)
Me.GroupBox37.Location = New System.Drawing.Point(16, 32)
Me.GroupBox37.Name = "GroupBox37"
Me.GroupBox37.Size = New System.Drawing.Size(440, 144)
Me.GroupBox37.TabIndex = 232
Me.GroupBox37.TabStop = False
```

```
Me.GroupBox37.Text = "Axis Amplitude (Step Size of X, Y, and Z) - 0 to 50 cm"
'
'ComboBox11
'
Me.ComboBox11.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox11.ItemHeight = 14
Me.ComboBox11.Items.AddRange(New Object() {"A Lot More", "A Little More", "Keep Current", "A Little Less", "A
Lot Less"})
Me.ComboBox11.Location = New System.Drawing.Point(184, 24)
Me.ComboBox11.Name = "ComboBox11"
Me.ComboBox11.Size = New System.Drawing.Size(128, 22)
Me.ComboBox11.TabIndex = 183
'
'Label119
'
Me.Label119.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label119.Location = New System.Drawing.Point(8, 32)
Me.Label119.Name = "Label119"
Me.Label119.Size = New System.Drawing.Size(176, 16)
Me.Label119.TabIndex = 182
Me.Label119.Text = "Forward-Backward Step Size:"
'
'Label120
'
Me.Label120.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label120.Location = New System.Drawing.Point(376, 112)
Me.Label120.Name = "Label120"
Me.Label120.Size = New System.Drawing.Size(40, 24)
Me.Label120.TabIndex = 224
Me.Label120.Text = "mm"
'
'Label121
'
Me.Label121.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label121.Location = New System.Drawing.Point(376, 72)
Me.Label121.Name = "Label121"
Me.Label121.Size = New System.Drawing.Size(40, 24)
Me.Label121.TabIndex = 223
Me.Label121.Text = "mm"
'
'Label122
'
Me.Label122.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label122.Location = New System.Drawing.Point(376, 32)
Me.Label122.Name = "Label122"
Me.Label122.Size = New System.Drawing.Size(40, 24)
Me.Label122.TabIndex = 222
Me.Label122.Text = "mm"
'
'TextBox138
'
Me.TextBox138.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox138.Location = New System.Drawing.Point(328, 104)
Me.TextBox138.Name = "TextBox138"
Me.TextBox138.Size = New System.Drawing.Size(40, 20)
Me.TextBox138.TabIndex = 221
Me.TextBox138.Text = ""
```

```
'
'TextBox139
'
Me.TextBox139.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox139.Location = New System.Drawing.Point(328, 64)
Me.TextBox139.Name = "TextBox139"
Me.TextBox139.Size = New System.Drawing.Size(40, 20)
Me.TextBox139.TabIndex = 220
Me.TextBox139.Text = ""
'
'TextBox140
'
Me.TextBox140.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox140.Location = New System.Drawing.Point(328, 24)
Me.TextBox140.Name = "TextBox140"
Me.TextBox140.Size = New System.Drawing.Size(40, 20)
Me.TextBox140.TabIndex = 219
Me.TextBox140.Text = ""
'
'ComboBox12
'
Me.ComboBox12.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox12.ItemHeight = 14
Me.ComboBox12.Items.AddRange(New Object() {"A Lot More", "A Little More", "Keep Current", "A Little Less", "A
Lot Less"})
Me.ComboBox12.Location = New System.Drawing.Point(184, 104)
Me.ComboBox12.Name = "ComboBox12"
Me.ComboBox12.Size = New System.Drawing.Size(128, 22)
Me.ComboBox12.TabIndex = 218
'
'ComboBox13
'
Me.ComboBox13.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox13.ItemHeight = 14
Me.ComboBox13.Items.AddRange(New Object() {"A Lot More", "A Little More", "Keep Current", "A Little Less", "A
Lot Less"})
Me.ComboBox13.Location = New System.Drawing.Point(184, 64)
Me.ComboBox13.Name = "ComboBox13"
Me.ComboBox13.Size = New System.Drawing.Size(128, 22)
Me.ComboBox13.TabIndex = 217
'
'Label123
'
Me.Label123.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label123.Location = New System.Drawing.Point(32, 112)
Me.Label123.Name = "Label123"
Me.Label123.Size = New System.Drawing.Size(144, 16)
Me.Label123.TabIndex = 198
Me.Label123.Text = "Up-Down Step Size:"
'
'Label124
'
Me.Label124.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label124.Location = New System.Drawing.Point(32, 72)
Me.Label124.Name = "Label124"
Me.Label124.Size = New System.Drawing.Size(144, 16)
Me.Label124.TabIndex = 195
```

```
Me.Label124.Text = "Left-Right Step Size:"
'
'GroupBox17
'
Me.GroupBox17.Controls.Add(Me.AxWebBrowser3)
Me.GroupBox17.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.GroupBox17.Location = New System.Drawing.Point(248, 16)
Me.GroupBox17.Name = "GroupBox17"
Me.GroupBox17.Size = New System.Drawing.Size(336, 280)
Me.GroupBox17.TabIndex = 182
Me.GroupBox17.TabStop = False
Me.GroupBox17.Text = "Visual Feedback"
'
'AxWebBrowser3
'
Me.AxWebBrowser3.ContainingControl = Me
Me.AxWebBrowser3.Enabled = True
Me.AxWebBrowser3.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.AxWebBrowser3.Location = New System.Drawing.Point(0, 16)
Me.AxWebBrowser3.OcxState = CType(resources.GetObject("AxWebBrowser3.OcxState"),
System.Windows.Forms.AxHost.State)
Me.AxWebBrowser3.Size = New System.Drawing.Size(360, 288)
Me.AxWebBrowser3.TabIndex = 59
'
'Button55
'
Me.Button55.BackColor = System.Drawing.Color.FromArgb(CType(192, Byte), CType(255, Byte), CType(192, Byte))
Me.Button55.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button55.Location = New System.Drawing.Point(128, 40)
Me.Button55.Name = "Button55"
Me.Button55.Size = New System.Drawing.Size(72, 40)
Me.Button55.TabIndex = 239
Me.Button55.Text = "Drop Bag"
'
'Button49
'
Me.Button49.BackColor = System.Drawing.SystemColors.InactiveCaptionText
Me.Button49.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button49.Location = New System.Drawing.Point(48, 40)
Me.Button49.Name = "Button49"
Me.Button49.Size = New System.Drawing.Size(72, 40)
Me.Button49.TabIndex = 218
Me.Button49.Text = "Initialize System"
'
'Button46
'
Me.Button46.BackColor = System.Drawing.Color.Yellow
Me.Button46.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button46.Location = New System.Drawing.Point(14, 706)
Me.Button46.Name = "Button46"
Me.Button46.Size = New System.Drawing.Size(56, 32)
Me.Button46.TabIndex = 241
Me.Button46.Text = "Record"
Me.Button46.Visible = False
'
'TabPage1
'
Me.TabPage1.Controls.Add(Me.TextBox49)
Me.TabPage1.Controls.Add(Me.CheckBox12)
```

```
Me.TabPage1.Controls.Add(Me.GroupBox24)
Me.TabPage1.Controls.Add(Me.GroupBox26)
Me.TabPage1.Controls.Add(Me.GroupBox28)
Me.TabPage1.Controls.Add(Me.GroupBox1)
Me.TabPage1.Controls.Add(Me.GroupBox2)
Me.TabPage1.Controls.Add(Me.GroupBox6)
Me.TabPage1.Controls.Add(Me.CheckBox8)
Me.TabPage1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TabPage1.Location = New System.Drawing.Point(4, 28)
Me.TabPage1.Name = "TabPage1"
Me.TabPage1.Size = New System.Drawing.Size(1264, 792)
Me.TabPage1.TabIndex = 0
Me.TabPage1.Text = "Development"
'
'TextBox49
'
Me.TextBox49.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox49.Location = New System.Drawing.Point(848, 616)
Me.TextBox49.Name = "TextBox49"
Me.TextBox49.Size = New System.Drawing.Size(60, 20)
Me.TextBox49.TabIndex = 272
Me.TextBox49.Text = "100"
'
'CheckBox12
'
Me.CheckBox12.Location = New System.Drawing.Point(824, 616)
Me.CheckBox12.Name = "CheckBox12"
Me.CheckBox12.Size = New System.Drawing.Size(16, 24)
Me.CheckBox12.TabIndex = 271
Me.CheckBox12.Text = "CheckBox12"
'
'GroupBox24
'
Me.GroupBox24.Controls.Add(Me.TextBox5)
Me.GroupBox24.Controls.Add(Me.Label16)
Me.GroupBox24.Controls.Add(Me.CheckBox14)
Me.GroupBox24.Controls.Add(Me.Label134)
Me.GroupBox24.Controls.Add(Me.Label133)
Me.GroupBox24.Controls.Add(Me.TextBox147)
Me.GroupBox24.Controls.Add(Me.Label93)
Me.GroupBox24.Controls.Add(Me.TextBox134)
Me.GroupBox24.Controls.Add(Me.Label102)
Me.GroupBox24.Controls.Add(Me.TextBox144)
Me.GroupBox24.Controls.Add(Me.Label107)
Me.GroupBox24.Controls.Add(Me.TextBox158)
Me.GroupBox24.Controls.Add(Me.Label108)
Me.GroupBox24.Controls.Add(Me.Label105)
Me.GroupBox24.Controls.Add(Me.TextBox149)
Me.GroupBox24.Controls.Add(Me.Label53)
Me.GroupBox24.Controls.Add(Me.TextBox129)
Me.GroupBox24.Controls.Add(Me.Label44)
Me.GroupBox24.Controls.Add(Me.CheckBox9)
Me.GroupBox24.Controls.Add(Me.Label103)
Me.GroupBox24.Controls.Add(Me.TextBox148)
Me.GroupBox24.Controls.Add(Me.TextBox128)
Me.GroupBox24.Controls.Add(Me.Label90)
Me.GroupBox24.Font = New System.Drawing.Font("Arial", 12.0!, System.Drawing.FontStyle.Bold)
Me.GroupBox24.Location = New System.Drawing.Point(16, 16)
Me.GroupBox24.Name = "GroupBox24"
Me.GroupBox24.Size = New System.Drawing.Size(664, 144)
Me.GroupBox24.TabIndex = 269
```

```
Me.GroupBox24.TabStop = False
Me.GroupBox24.Text = "System Papameter Configuration"
'
'TextBox5
'
Me.TextBox5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox5.Location = New System.Drawing.Point(522, 56)
Me.TextBox5.Name = "TextBox5"
Me.TextBox5.Size = New System.Drawing.Size(50, 20)
Me.TextBox5.TabIndex = 273
Me.TextBox5.Text = "238.586"
'
'Label16
'
Me.Label16.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label16.Location = New System.Drawing.Point(404, 58)
Me.Label16.Name = "Label16"
Me.Label16.Size = New System.Drawing.Size(112, 16)
Me.Label16.TabIndex = 274
Me.Label16.Text = "Reward Constant (c):"
'
'CheckBox14
'
Me.CheckBox14.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.CheckBox14.Location = New System.Drawing.Point(392, 114)
Me.CheckBox14.Name = "CheckBox14"
Me.CheckBox14.Size = New System.Drawing.Size(248, 16)
Me.CheckBox14.TabIndex = 268
Me.CheckBox14.Text = "Enable Disabling Robot When Bag is Empty"
'
'Label134
'
Me.Label134.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label134.Location = New System.Drawing.Point(386, 82)
Me.Label134.Name = "Label134"
Me.Label134.Size = New System.Drawing.Size(38, 16)
Me.Label134.TabIndex = 267
Me.Label134.Text = "grams"
'
'Label133
'
Me.Label133.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label133.Location = New System.Drawing.Point(234, 80)
Me.Label133.Name = "Label133"
Me.Label133.Size = New System.Drawing.Size(106, 16)
Me.Label133.TabIndex = 266
Me.Label133.Text = "One Object Weight:"
'
'TextBox147
'
Me.TextBox147.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox147.Location = New System.Drawing.Point(342, 78)
Me.TextBox147.Name = "TextBox147"
Me.TextBox147.Size = New System.Drawing.Size(36, 20)
Me.TextBox147.TabIndex = 265
Me.TextBox147.Text = "45"
'
'Label93
'
```

```
Me.Label93.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label93.Location = New System.Drawing.Point(8, 80)
Me.Label93.Name = "Label93"
Me.Label93.Size = New System.Drawing.Size(182, 16)
Me.Label93.TabIndex = 264
Me.Label93.Text = "Max Actions per Learning Episode:"
'
'TextBox134
'
Me.TextBox134.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox134.Location = New System.Drawing.Point(194, 78)
Me.TextBox134.Name = "TextBox134"
Me.TextBox134.Size = New System.Drawing.Size(32, 20)
Me.TextBox134.TabIndex = 263
Me.TextBox134.Text = "100"
'
'Label102
'
Me.Label102.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label102.Location = New System.Drawing.Point(8, 32)
Me.Label102.Name = "Label102"
Me.Label102.Size = New System.Drawing.Size(80, 16)
Me.Label102.TabIndex = 245
Me.Label102.Text = "Robot Status:"
'
'TextBox144
'
Me.TextBox144.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox144.Location = New System.Drawing.Point(108, 28)
Me.TextBox144.Name = "TextBox144"
Me.TextBox144.Size = New System.Drawing.Size(60, 20)
Me.TextBox144.TabIndex = 244
Me.TextBox144.Text = "Idle"
'
'Label107
'
Me.Label107.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label107.Location = New System.Drawing.Point(368, 32)
Me.Label107.Name = "Label107"
Me.Label107.Size = New System.Drawing.Size(88, 16)
Me.Label107.TabIndex = 254
Me.Label107.Text = "learning trials."
'
'TextBox158
'
Me.TextBox158.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox158.Location = New System.Drawing.Point(336, 32)
Me.TextBox158.Name = "TextBox158"
Me.TextBox158.Size = New System.Drawing.Size(24, 20)
Me.TextBox158.TabIndex = 253
Me.TextBox158.Text = "0"
'
'Label108
'
Me.Label108.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label108.Location = New System.Drawing.Point(184, 32)
Me.Label108.Name = "Label108"
```

```
Me.Label108.Size = New System.Drawing.Size(152, 16)
Me.Label108.TabIndex = 252
Me.Label108.Text = "Robot is autonomous for the"
'
'Label105
'
Me.Label105.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label105.Location = New System.Drawing.Point(192, 56)
Me.Label105.Name = "Label105"
Me.Label105.Size = New System.Drawing.Size(36, 16)
Me.Label105.TabIndex = 251
Me.Label105.Text = "(sec)"
'
'TextBox149
'
Me.TextBox149.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox149.Location = New System.Drawing.Point(160, 54)
Me.TextBox149.Name = "TextBox149"
Me.TextBox149.Size = New System.Drawing.Size(32, 20)
Me.TextBox149.TabIndex = 250
Me.TextBox149.Text = "0.25"
'
'Label53
'
Me.Label53.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label53.Location = New System.Drawing.Point(8, 56)
Me.Label53.Name = "Label53"
Me.Label53.Size = New System.Drawing.Size(152, 16)
Me.Label53.TabIndex = 249
Me.Label53.Text = "Scale Writing to File Interval:"
'
'TextBox129
'
Me.TextBox129.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox129.Location = New System.Drawing.Point(360, 54)
Me.TextBox129.Name = "TextBox129"
Me.TextBox129.Size = New System.Drawing.Size(32, 20)
Me.TextBox129.TabIndex = 257
Me.TextBox129.Text = "0"
'
'Label44
'
Me.Label44.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label44.Location = New System.Drawing.Point(232, 56)
Me.Label44.Name = "Label44"
Me.Label44.Size = New System.Drawing.Size(136, 16)
Me.Label44.TabIndex = 258
Me.Label44.Text = "Reward Value Threshold:"
'
'CheckBox9
'
Me.CheckBox9.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.CheckBox9.Location = New System.Drawing.Point(8, 112)
Me.CheckBox9.Name = "CheckBox9"
Me.CheckBox9.Size = New System.Drawing.Size(160, 16)
Me.CheckBox9.TabIndex = 256
Me.CheckBox9.Text = "Enable Sound Notifications"
```

```
'
'Label103
'
Me.Label103.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label103.Location = New System.Drawing.Point(196, 112)
Me.Label103.Name = "Label103"
Me.Label103.Size = New System.Drawing.Size(120, 16)
Me.Label103.TabIndex = 71
Me.Label103.Text = "Temporary Directory:"
'
'TextBox148
'
Me.TextBox148.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox148.Location = New System.Drawing.Point(322, 110)
Me.TextBox148.Name = "TextBox148"
Me.TextBox148.Size = New System.Drawing.Size(64, 20)
Me.TextBox148.TabIndex = 70
Me.TextBox148.Text = "d:/temp/"
'
'TextBox128
'
Me.TextBox128.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold)
Me.TextBox128.Location = New System.Drawing.Point(484, 78)
Me.TextBox128.Name = "TextBox128"
Me.TextBox128.Size = New System.Drawing.Size(32, 20)
Me.TextBox128.TabIndex = 272
Me.TextBox128.Text = "0.9"
'
'Label90
'
Me.Label90.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label90.Location = New System.Drawing.Point(432, 82)
Me.Label90.Name = "Label90"
Me.Label90.Size = New System.Drawing.Size(48, 16)
Me.Label90.TabIndex = 272
Me.Label90.Text = "Epsilon:"
'
'GroupBox26
'
Me.GroupBox26.Controls.Add(Me.Label111)
Me.GroupBox26.Controls.Add(Me.Label112)
Me.GroupBox26.Controls.Add(Me.TextBox89)
Me.GroupBox26.Controls.Add(Me.Label43)
Me.GroupBox26.Controls.Add(Me.Label56)
Me.GroupBox26.Controls.Add(Me.TextBox80)
Me.GroupBox26.Controls.Add(Me.Label49)
Me.GroupBox26.Controls.Add(Me.Label50)
Me.GroupBox26.Controls.Add(Me.TextBox131)
Me.GroupBox26.Controls.Add(Me.Label48)
Me.GroupBox26.Controls.Add(Me.Label47)
Me.GroupBox26.Controls.Add(Me.TextBox130)
Me.GroupBox26.Font = New System.Drawing.Font("Arial", 9.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox26.Location = New System.Drawing.Point(16, 168)
Me.GroupBox26.Name = "GroupBox26"
Me.GroupBox26.Size = New System.Drawing.Size(296, 168)
Me.GroupBox26.TabIndex = 189
Me.GroupBox26.TabStop = False
Me.GroupBox26.Text = "System Performance"
'
```

```
'Label111
'
Me.Label111.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label111.Location = New System.Drawing.Point(248, 128)
Me.Label111.Name = "Label111"
Me.Label111.Size = New System.Drawing.Size(32, 16)
Me.Label111.TabIndex = 201
Me.Label111.Text = "sec"
'
'Label112
'
Me.Label112.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label112.Location = New System.Drawing.Point(16, 120)
Me.Label112.Name = "Label112"
Me.Label112.Size = New System.Drawing.Size(136, 40)
Me.Label112.TabIndex = 200
Me.Label112.Text = "Average Successful Shaking Policies:"
'
'TextBox89
'
Me.TextBox89.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox89.Location = New System.Drawing.Point(152, 120)
Me.TextBox89.Name = "TextBox89"
Me.TextBox89.Size = New System.Drawing.Size(88, 22)
Me.TextBox89.TabIndex = 199
Me.TextBox89.Text = "0"
'
'Label43
'
Me.Label43.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label43.Location = New System.Drawing.Point(248, 88)
Me.Label43.Name = "Label43"
Me.Label43.Size = New System.Drawing.Size(32, 16)
Me.Label43.TabIndex = 198
Me.Label43.Text = "sec"
'
'Label56
'
Me.Label56.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label56.Location = New System.Drawing.Point(16, 80)
Me.Label56.Name = "Label56"
Me.Label56.Size = New System.Drawing.Size(136, 32)
Me.Label56.TabIndex = 197
Me.Label56.Text = "Last Successful Shaking (Bag Emptied):"
'
'TextBox80
'
Me.TextBox80.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox80.Location = New System.Drawing.Point(152, 80)
Me.TextBox80.Name = "TextBox80"
Me.TextBox80.Size = New System.Drawing.Size(88, 22)
Me.TextBox80.TabIndex = 196
Me.TextBox80.Text = "0"
'
'Label49
'
Me.Label49.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label49.Location = New System.Drawing.Point(248, 24)
Me.Label49.Name = "Label49"
Me.Label49.Size = New System.Drawing.Size(32, 16)
Me.Label49.TabIndex = 195
```

```
Me.Label49.Text = "(%)"
'
'Label50
'
Me.Label50.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label50.Location = New System.Drawing.Point(16, 24)
Me.Label50.Name = "Label50"
Me.Label50.Size = New System.Drawing.Size(128, 16)
Me.Label50.TabIndex = 194
Me.Label50.Text = "Actual Performance:"
'
'TextBox131
'
Me.TextBox131.Enabled = False
Me.TextBox131.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox131.Location = New System.Drawing.Point(152, 16)
Me.TextBox131.Name = "TextBox131"
Me.TextBox131.Size = New System.Drawing.Size(88, 22)
Me.TextBox131.TabIndex = 193
Me.TextBox131.Text = "0"
'
'Label48
'
Me.Label48.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label48.Location = New System.Drawing.Point(248, 56)
Me.Label48.Name = "Label48"
Me.Label48.Size = New System.Drawing.Size(32, 16)
Me.Label48.TabIndex = 192
Me.Label48.Text = "(%)"
'
'Label47
'
Me.Label47.Font = New System.Drawing.Font("Arial", 8.25!)
Me.Label47.Location = New System.Drawing.Point(16, 48)
Me.Label47.Name = "Label47"
Me.Label47.Size = New System.Drawing.Size(80, 16)
Me.Label47.TabIndex = 191
Me.Label47.Text = "Threshold:"
'
'TextBox130
'
Me.TextBox130.Font = New System.Drawing.Font("Arial", 9.75!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox130.Location = New System.Drawing.Point(152, 48)
Me.TextBox130.Name = "TextBox130"
Me.TextBox130.Size = New System.Drawing.Size(88, 22)
Me.TextBox130.TabIndex = 190
Me.TextBox130.Text = "60"
'
'GroupBox28
'
Me.GroupBox28.Controls.Add(Me.Button58)
Me.GroupBox28.Controls.Add(Me.Button48)
Me.GroupBox28.Controls.Add(Me.TextBox151)
Me.GroupBox28.Controls.Add(Me.TextBox150)
Me.GroupBox28.Controls.Add(Me.TextBox143)
Me.GroupBox28.Location = New System.Drawing.Point(520, 592)
Me.GroupBox28.Name = "GroupBox28"
Me.GroupBox28.Size = New System.Drawing.Size(248, 144)
Me.GroupBox28.TabIndex = 188
Me.GroupBox28.TabStop = False
Me.GroupBox28.Text = "Temp"
```

```
Me.GroupBox28.Visible = False
'
'Button58
'
Me.Button58.Location = New System.Drawing.Point(140, 80)
Me.Button58.Name = "Button58"
Me.Button58.Size = New System.Drawing.Size(64, 40)
Me.Button58.TabIndex = 257
Me.Button58.Text = "Button58"
Me.Button58.Visible = False
'
'Button48
'
Me.Button48.BackColor = System.Drawing.SystemColors.InactiveCaptionText
Me.Button48.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button48.Location = New System.Drawing.Point(44, 72)
Me.Button48.Name = "Button48"
Me.Button48.Size = New System.Drawing.Size(88, 48)
Me.Button48.TabIndex = 256
Me.Button48.Text = "Continue Algorithm"
Me.Button48.Visible = False
'
'TextBox151
'
Me.TextBox151.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox151.Location = New System.Drawing.Point(88, 32)
Me.TextBox151.Name = "TextBox151"
Me.TextBox151.Size = New System.Drawing.Size(60, 20)
Me.TextBox151.TabIndex = 250
Me.TextBox151.Text = ""
Me.TextBox151.Visible = False
'
'TextBox150
'
Me.TextBox150.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox150.Location = New System.Drawing.Point(16, 32)
Me.TextBox150.Name = "TextBox150"
Me.TextBox150.Size = New System.Drawing.Size(60, 20)
Me.TextBox150.TabIndex = 249
Me.TextBox150.Text = "0"
Me.TextBox150.Visible = False
'
'TextBox143
'
Me.TextBox143.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox143.Location = New System.Drawing.Point(168, 32)
Me.TextBox143.Name = "TextBox143"
Me.TextBox143.Size = New System.Drawing.Size(60, 20)
Me.TextBox143.TabIndex = 248
Me.TextBox143.Text = ""
Me.TextBox143.Visible = False
'
'CheckBox8
'
Me.CheckBox8.Checked = True
Me.CheckBox8.CheckState = System.Windows.Forms.CheckState.Checked
Me.CheckBox8.Location = New System.Drawing.Point(776, 616)
Me.CheckBox8.Name = "CheckBox8"
Me.CheckBox8.Size = New System.Drawing.Size(16, 16)
```

```
Me.CheckBox8.TabIndex = 37
Me.CheckBox8.Text = "Servo"
'
'TabPage7
'
Me.TabPage7.Controls.Add(Me.Label51)
Me.TabPage7.Controls.Add(Me.TextBox112)
Me.TabPage7.Controls.Add(Me.Button37)
Me.TabPage7.Controls.Add(Me.Button27)
Me.TabPage7.Controls.Add(Me.Label89)
Me.TabPage7.Controls.Add(Me.TextBox127)
Me.TabPage7.Controls.Add(Me.GroupBox25)
Me.TabPage7.Controls.Add(Me.GroupBox23)
Me.TabPage7.Controls.Add(Me.GroupBox22)
Me.TabPage7.Controls.Add(Me.Button35)
Me.TabPage7.Controls.Add(Me.Button33)
Me.TabPage7.Location = New System.Drawing.Point(4, 28)
Me.TabPage7.Name = "TabPage7"
Me.TabPage7.Size = New System.Drawing.Size(1264, 792)
Me.TabPage7.TabIndex = 6
Me.TabPage7.Text = "CQ(lamda)"
'
'Label51
'
Me.Label51.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label51.Location = New System.Drawing.Point(432, 96)
Me.Label51.Name = "Label51"
Me.Label51.Size = New System.Drawing.Size(160, 16)
Me.Label51.TabIndex = 219
Me.Label51.Text = "Number of Policies Performed:"
'
'TextBox112
'
Me.TextBox112.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox112.Location = New System.Drawing.Point(600, 96)
Me.TextBox112.Name = "TextBox112"
Me.TextBox112.Size = New System.Drawing.Size(32, 20)
Me.TextBox112.TabIndex = 218
Me.TextBox112.Text = "0"
'
'Button37
'
Me.Button37.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button37.Location = New System.Drawing.Point(632, 24)
Me.Button37.Name = "Button37"
Me.Button37.Size = New System.Drawing.Size(88, 48)
Me.Button37.TabIndex = 217
Me.Button37.Text = "Use the Policy with No Reward"
'
'Button27
'
Me.Button27.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button27.Location = New System.Drawing.Point(736, 24)
Me.Button27.Name = "Button27"
Me.Button27.Size = New System.Drawing.Size(88, 48)
Me.Button27.TabIndex = 216
Me.Button27.Text = "Use the Rewarded  Policy"
'
'Label89
```

```
'
Me.Label89.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label89.Location = New System.Drawing.Point(528, 560)
Me.Label89.Name = "Label89"
Me.Label89.Size = New System.Drawing.Size(88, 16)
Me.Label89.TabIndex = 215
Me.Label89.Text = "Final Q Values"
'
'TextBox127
'
Me.TextBox127.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox127.Location = New System.Drawing.Point(16, 584)
Me.TextBox127.Multiline = True
Me.TextBox127.Name = "TextBox127"
Me.TextBox127.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox127.Size = New System.Drawing.Size(1104, 176)
Me.TextBox127.TabIndex = 214
Me.TextBox127.Text = ""
'
'GroupBox25
'
Me.GroupBox25.Controls.Add(Me.ListBox7)
Me.GroupBox25.Controls.Add(Me.Label75)
Me.GroupBox25.Controls.Add(Me.ListBox8)
Me.GroupBox25.Controls.Add(Me.Label76)
Me.GroupBox25.Controls.Add(Me.ListBox9)
Me.GroupBox25.Controls.Add(Me.Label77)
Me.GroupBox25.Controls.Add(Me.ListBox10)
Me.GroupBox25.Controls.Add(Me.Label78)
Me.GroupBox25.Controls.Add(Me.TextBox125)
Me.GroupBox25.Controls.Add(Me.Label79)
Me.GroupBox25.Controls.Add(Me.TextBox126)
Me.GroupBox25.Controls.Add(Me.Label80)
Me.GroupBox25.Location = New System.Drawing.Point(424, 136)
Me.GroupBox25.Name = "GroupBox25"
Me.GroupBox25.Size = New System.Drawing.Size(376, 416)
Me.GroupBox25.TabIndex = 213
Me.GroupBox25.TabStop = False
Me.GroupBox25.Text = "Policy - With Reward"
'
'ListBox7
'
Me.ListBox7.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox7.ItemHeight = 14
Me.ListBox7.Location = New System.Drawing.Point(184, 56)
Me.ListBox7.Name = "ListBox7"
Me.ListBox7.Size = New System.Drawing.Size(56, 130)
Me.ListBox7.TabIndex = 181
'
'Label75
'
Me.Label75.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label75.Location = New System.Drawing.Point(184, 24)
Me.Label75.Name = "Label75"
Me.Label75.Size = New System.Drawing.Size(56, 16)
Me.Label75.TabIndex = 180
Me.Label75.Text = "Time (ms)"
'
'ListBox8
```

```
'
Me.ListBox8.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox8.ItemHeight = 14
Me.ListBox8.Location = New System.Drawing.Point(120, 56)
Me.ListBox8.Name = "ListBox8"
Me.ListBox8.Size = New System.Drawing.Size(48, 130)
Me.ListBox8.TabIndex = 178
'
'Label76
'
Me.Label76.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label76.Location = New System.Drawing.Point(120, 24)
Me.Label76.Name = "Label76"
Me.Label76.Size = New System.Drawing.Size(48, 16)
Me.Label76.TabIndex = 177
Me.Label76.Text = "Reward"
'
'ListBox9
'
Me.ListBox9.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox9.ItemHeight = 14
Me.ListBox9.Location = New System.Drawing.Point(64, 56)
Me.ListBox9.Name = "ListBox9"
Me.ListBox9.Size = New System.Drawing.Size(48, 130)
Me.ListBox9.TabIndex = 176
'
'Label77
'
Me.Label77.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label77.Location = New System.Drawing.Point(64, 24)
Me.Label77.Name = "Label77"
Me.Label77.Size = New System.Drawing.Size(40, 16)
Me.Label77.TabIndex = 175
Me.Label77.Text = "Action"
'
'ListBox10
'
Me.ListBox10.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox10.ItemHeight = 14
Me.ListBox10.Items.AddRange(New Object() {"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"})
Me.ListBox10.Location = New System.Drawing.Point(8, 56)
Me.ListBox10.Name = "ListBox10"
Me.ListBox10.Size = New System.Drawing.Size(48, 130)
Me.ListBox10.TabIndex = 170
'
'Label78
'
Me.Label78.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label78.Location = New System.Drawing.Point(16, 24)
Me.Label78.Name = "Label78"
Me.Label78.Size = New System.Drawing.Size(32, 16)
Me.Label78.TabIndex = 168
Me.Label78.Text = "State"
'
'TextBox125
'
```

```
Me.TextBox125.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox125.Location = New System.Drawing.Point(256, 56)
Me.TextBox125.Multiline = True
Me.TextBox125.Name = "TextBox125"
Me.TextBox125.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox125.Size = New System.Drawing.Size(96, 144)
Me.TextBox125.TabIndex = 149
Me.TextBox125.Text = ""
 '
 'Label79
 '
Me.Label79.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label79.Location = New System.Drawing.Point(280, 24)
Me.Label79.Name = "Label79"
Me.Label79.Size = New System.Drawing.Size(44, 16)
Me.Label79.TabIndex = 171
Me.Label79.Text = "Delta"
 '
 'TextBox126
 '
Me.TextBox126.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox126.Location = New System.Drawing.Point(8, 224)
Me.TextBox126.Multiline = True
Me.TextBox126.Name = "TextBox126"
Me.TextBox126.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox126.Size = New System.Drawing.Size(352, 176)
Me.TextBox126.TabIndex = 152
Me.TextBox126.Text = ""
 '
 'Label80
 '
Me.Label80.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label80.Location = New System.Drawing.Point(152, 208)
Me.Label80.Name = "Label80"
Me.Label80.Size = New System.Drawing.Size(56, 16)
Me.Label80.TabIndex = 172
Me.Label80.Text = "Q Values"
 '
 'GroupBox23
 '
Me.GroupBox23.Controls.Add(Me.ListBox6)
Me.GroupBox23.Controls.Add(Me.Label74)
Me.GroupBox23.Controls.Add(Me.ListBox5)
Me.GroupBox23.Controls.Add(Me.Label71)
Me.GroupBox23.Controls.Add(Me.ListBox4)
Me.GroupBox23.Controls.Add(Me.Label70)
Me.GroupBox23.Controls.Add(Me.ListBox3)
Me.GroupBox23.Controls.Add(Me.Label62)
Me.GroupBox23.Controls.Add(Me.TextBox117)
Me.GroupBox23.Controls.Add(Me.Label61)
Me.GroupBox23.Controls.Add(Me.TextBox118)
Me.GroupBox23.Controls.Add(Me.Label68)
Me.GroupBox23.Location = New System.Drawing.Point(16, 136)
Me.GroupBox23.Name = "GroupBox23"
Me.GroupBox23.Size = New System.Drawing.Size(376, 416)
Me.GroupBox23.TabIndex = 211
Me.GroupBox23.TabStop = False
Me.GroupBox23.Text = "Policy - No Reward"
 '
```

```
'ListBox6
'
Me.ListBox6.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox6.ItemHeight = 14
Me.ListBox6.Location = New System.Drawing.Point(184, 56)
Me.ListBox6.Name = "ListBox6"
Me.ListBox6.Size = New System.Drawing.Size(56, 130)
Me.ListBox6.TabIndex = 181
'
'Label74
'
Me.Label74.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label74.Location = New System.Drawing.Point(184, 24)
Me.Label74.Name = "Label74"
Me.Label74.Size = New System.Drawing.Size(56, 16)
Me.Label74.TabIndex = 180
Me.Label74.Text = "Time (ms)"
'
'ListBox5
'
Me.ListBox5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox5.ItemHeight = 14
Me.ListBox5.Location = New System.Drawing.Point(120, 56)
Me.ListBox5.Name = "ListBox5"
Me.ListBox5.Size = New System.Drawing.Size(48, 130)
Me.ListBox5.TabIndex = 178
'
'Label71
'
Me.Label71.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label71.Location = New System.Drawing.Point(120, 24)
Me.Label71.Name = "Label71"
Me.Label71.Size = New System.Drawing.Size(48, 16)
Me.Label71.TabIndex = 177
Me.Label71.Text = "Reward"
'
'ListBox4
'
Me.ListBox4.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox4.ItemHeight = 14
Me.ListBox4.Location = New System.Drawing.Point(64, 56)
Me.ListBox4.Name = "ListBox4"
Me.ListBox4.Size = New System.Drawing.Size(48, 130)
Me.ListBox4.TabIndex = 176
'
'Label70
'
Me.Label70.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label70.Location = New System.Drawing.Point(64, 24)
Me.Label70.Name = "Label70"
Me.Label70.Size = New System.Drawing.Size(40, 16)
Me.Label70.TabIndex = 175
Me.Label70.Text = "Action"
'
'ListBox3
'
```

```
Me.ListBox3.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox3.ItemHeight = 14
Me.ListBox3.Items.AddRange(New Object() {"0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"})
Me.ListBox3.Location = New System.Drawing.Point(8, 56)
Me.ListBox3.Name = "ListBox3"
Me.ListBox3.Size = New System.Drawing.Size(48, 130)
Me.ListBox3.TabIndex = 170
'
'Label62
'
Me.Label62.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label62.Location = New System.Drawing.Point(16, 24)
Me.Label62.Name = "Label62"
Me.Label62.Size = New System.Drawing.Size(32, 16)
Me.Label62.TabIndex = 168
Me.Label62.Text = "State"
'
'TextBox117
'
Me.TextBox117.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox117.Location = New System.Drawing.Point(256, 56)
Me.TextBox117.Multiline = True
Me.TextBox117.Name = "TextBox117"
Me.TextBox117.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox117.Size = New System.Drawing.Size(96, 144)
Me.TextBox117.TabIndex = 149
Me.TextBox117.Text = ""
'
'Label61
'
Me.Label61.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label61.Location = New System.Drawing.Point(280, 24)
Me.Label61.Name = "Label61"
Me.Label61.Size = New System.Drawing.Size(44, 16)
Me.Label61.TabIndex = 171
Me.Label61.Text = "Delta"
'
'TextBox118
'
Me.TextBox118.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox118.Location = New System.Drawing.Point(8, 224)
Me.TextBox118.Multiline = True
Me.TextBox118.Name = "TextBox118"
Me.TextBox118.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox118.Size = New System.Drawing.Size(352, 176)
Me.TextBox118.TabIndex = 152
Me.TextBox118.Text = ""
'
'Label68
'
Me.Label68.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label68.Location = New System.Drawing.Point(152, 208)
Me.Label68.Name = "Label68"
Me.Label68.Size = New System.Drawing.Size(56, 16)
Me.Label68.TabIndex = 172
Me.Label68.Text = "Q Values"
'
```

```
'GroupBox22
'
Me.GroupBox22.Controls.Add(Me.Label52)
Me.GroupBox22.Controls.Add(Me.TextBox113)
Me.GroupBox22.Controls.Add(Me.TextBox122)
Me.GroupBox22.Controls.Add(Me.Label65)
Me.GroupBox22.Controls.Add(Me.TextBox121)
Me.GroupBox22.Controls.Add(Me.Label64)
Me.GroupBox22.Controls.Add(Me.TextBox120)
Me.GroupBox22.Controls.Add(Me.Label63)
Me.GroupBox22.Controls.Add(Me.TextBox119)
Me.GroupBox22.Controls.Add(Me.Label69)
Me.GroupBox22.Controls.Add(Me.TextBox124)
Me.GroupBox22.Controls.Add(Me.Label67)
Me.GroupBox22.Controls.Add(Me.TextBox123)
Me.GroupBox22.Controls.Add(Me.Label66)
Me.GroupBox22.Location = New System.Drawing.Point(16, 16)
Me.GroupBox22.Name = "GroupBox22"
Me.GroupBox22.Size = New System.Drawing.Size(384, 104)
Me.GroupBox22.TabIndex = 210
Me.GroupBox22.TabStop = False
Me.GroupBox22.Text = "RL Parameters"
'
'Label52
'
Me.Label52.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label52.Location = New System.Drawing.Point(256, 48)
Me.Label52.Name = "Label52"
Me.Label52.Size = New System.Drawing.Size(68, 16)
Me.Label52.TabIndex = 178
Me.Label52.Text = "Punishment:"
'
'TextBox113
'
Me.TextBox113.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox113.Location = New System.Drawing.Point(336, 48)
Me.TextBox113.Name = "TextBox113"
Me.TextBox113.Size = New System.Drawing.Size(32, 20)
Me.TextBox113.TabIndex = 177
Me.TextBox113.Text = "-"
'
'TextBox122
'
Me.TextBox122.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox122.Location = New System.Drawing.Point(208, 24)
Me.TextBox122.Name = "TextBox122"
Me.TextBox122.Size = New System.Drawing.Size(32, 20)
Me.TextBox122.TabIndex = 161
Me.TextBox122.Text = "10"
'
'Label65
'
Me.Label65.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label65.Location = New System.Drawing.Point(8, 72)
Me.Label65.Name = "Label65"
Me.Label65.Size = New System.Drawing.Size(68, 16)
Me.Label65.TabIndex = 160
Me.Label65.Text = "Lambda:"
'
```

```
'TextBox121
'
Me.TextBox121.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox121.Location = New System.Drawing.Point(80, 72)
Me.TextBox121.Name = "TextBox121"
Me.TextBox121.Size = New System.Drawing.Size(32, 20)
Me.TextBox121.TabIndex = 159
Me.TextBox121.Text = "0.5"
'
'Label64
'
Me.Label64.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label64.Location = New System.Drawing.Point(8, 48)
Me.Label64.Name = "Label64"
Me.Label64.Size = New System.Drawing.Size(68, 16)
Me.Label64.TabIndex = 158
Me.Label64.Text = "Gamma:"
'
'TextBox120
'
Me.TextBox120.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox120.Location = New System.Drawing.Point(80, 48)
Me.TextBox120.Name = "TextBox120"
Me.TextBox120.Size = New System.Drawing.Size(32, 20)
Me.TextBox120.TabIndex = 157
Me.TextBox120.Text = "0.99"
'
'Label63
'
Me.Label63.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label63.Location = New System.Drawing.Point(8, 24)
Me.Label63.Name = "Label63"
Me.Label63.Size = New System.Drawing.Size(68, 16)
Me.Label63.TabIndex = 156
Me.Label63.Text = "Initial Alpha:"
'
'TextBox119
'
Me.TextBox119.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox119.Location = New System.Drawing.Point(80, 24)
Me.TextBox119.Name = "TextBox119"
Me.TextBox119.Size = New System.Drawing.Size(32, 20)
Me.TextBox119.TabIndex = 155
Me.TextBox119.Text = "0.05"
'
'Label69
'
Me.Label69.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label69.Location = New System.Drawing.Point(120, 72)
Me.Label69.Name = "Label69"
Me.Label69.Size = New System.Drawing.Size(88, 16)
Me.Label69.TabIndex = 174
Me.Label69.Text = "Learning Trials:"
'
'TextBox124
'
```

```
Me.TextBox124.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox124.Location = New System.Drawing.Point(208, 72)
Me.TextBox124.Name = "TextBox124"
Me.TextBox124.Size = New System.Drawing.Size(32, 20)
Me.TextBox124.TabIndex = 173
Me.TextBox124.Text = "10"
'
'Label67
'
Me.Label67.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label67.Location = New System.Drawing.Point(120, 48)
Me.Label67.Name = "Label67"
Me.Label67.Size = New System.Drawing.Size(68, 16)
Me.Label67.TabIndex = 164
Me.Label67.Text = "Reward:"
'
'TextBox123
'
Me.TextBox123.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox123.Location = New System.Drawing.Point(208, 48)
Me.TextBox123.Name = "TextBox123"
Me.TextBox123.Size = New System.Drawing.Size(32, 20)
Me.TextBox123.TabIndex = 163
Me.TextBox123.Text = "-"
'
'Label66
'
Me.Label66.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label66.Location = New System.Drawing.Point(120, 24)
Me.Label66.Name = "Label66"
Me.Label66.Size = New System.Drawing.Size(68, 16)
Me.Label66.TabIndex = 162
Me.Label66.Text = "Beta:"
'
'Button35
'
Me.Button35.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button35.Location = New System.Drawing.Point(528, 24)
Me.Button35.Name = "Button35"
Me.Button35.Size = New System.Drawing.Size(88, 48)
Me.Button35.TabIndex = 179
Me.Button35.Text = "Continue Algorithm"
'
'Button33
'
Me.Button33.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button33.Location = New System.Drawing.Point(424, 24)
Me.Button33.Name = "Button33"
Me.Button33.Size = New System.Drawing.Size(88, 48)
Me.Button33.TabIndex = 165
Me.Button33.Text = "Run Q(lambda)"
'
'TabPage2
'
Me.TabPage2.Controls.Add(Me.GroupBox5)
Me.TabPage2.Controls.Add(Me.GroupBox3)
Me.TabPage2.Controls.Add(Me.GroupBox14)
```

```
Me.TabPage2.Controls.Add(Me.Label26)
Me.TabPage2.Controls.Add(Me.Label27)
Me.TabPage2.Controls.Add(Me.CheckBox6)
Me.TabPage2.Controls.Add(Me.CheckBox3)
Me.TabPage2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TabPage2.Location = New System.Drawing.Point(4, 28)
Me.TabPage2.Name = "TabPage2"
Me.TabPage2.Size = New System.Drawing.Size(1264, 792)
Me.TabPage2.TabIndex = 1
Me.TabPage2.Text = "User Interface"
 '
 'GroupBox5
 '
Me.GroupBox5.Controls.Add(Me.Label73)
Me.GroupBox5.Controls.Add(Me.AxWebBrowser1)
Me.GroupBox5.Controls.Add(Me.AxWebBrowser2)
Me.GroupBox5.Controls.Add(Me.Label72)
Me.GroupBox5.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.GroupBox5.Location = New System.Drawing.Point(752, 0)
Me.GroupBox5.Name = "GroupBox5"
Me.GroupBox5.Size = New System.Drawing.Size(384, 656)
Me.GroupBox5.TabIndex = 69
Me.GroupBox5.TabStop = False
Me.GroupBox5.Text = "Visual Feeback"
Me.GroupBox5.Visible = False
 '
 'Label73
 '
Me.Label73.Font = New System.Drawing.Font("Arial", 8.25!, CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
Me.Label73.Location = New System.Drawing.Point(144, 336)
Me.Label73.Name = "Label73"
Me.Label73.Size = New System.Drawing.Size(80, 16)
Me.Label73.TabIndex = 76
Me.Label73.Text = "Overall View"
 '
 'AxWebBrowser1
 '
Me.AxWebBrowser1.ContainingControl = Me
Me.AxWebBrowser1.Enabled = True
Me.AxWebBrowser1.Location = New System.Drawing.Point(8, 40)
Me.AxWebBrowser1.OcxState = CType(resources.GetObject("AxWebBrowser1.OcxState"),
System.Windows.Forms.AxHost.State)
Me.AxWebBrowser1.Size = New System.Drawing.Size(360, 288)
Me.AxWebBrowser1.TabIndex = 59
 '
 'AxWebBrowser2
 '
Me.AxWebBrowser2.ContainingControl = Me
Me.AxWebBrowser2.Enabled = True
Me.AxWebBrowser2.Location = New System.Drawing.Point(8, 360)
Me.AxWebBrowser2.OcxState = CType(resources.GetObject("AxWebBrowser2.OcxState"),
System.Windows.Forms.AxHost.State)
Me.AxWebBrowser2.Size = New System.Drawing.Size(360, 288)
Me.AxWebBrowser2.TabIndex = 60
 '
 'Label72
 '
```

```
Me.Label72.Font = New System.Drawing.Font("Arial", 8.25!, CType((System.Drawing.FontStyle.Bold Or
System.Drawing.FontStyle.Underline), System.Drawing.FontStyle), System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
Me.Label72.Location = New System.Drawing.Point(144, 16)
Me.Label72.Name = "Label72"
Me.Label72.Size = New System.Drawing.Size(80, 16)
Me.Label72.TabIndex = 75
Me.Label72.Text = "Close View"
'
'TabPage5
'
Me.TabPage5.Controls.Add(Me.GroupBox32)
Me.TabPage5.Controls.Add(Me.GroupBox27)
Me.TabPage5.Controls.Add(Me.GroupBox21)
Me.TabPage5.Controls.Add(Me.GroupBox16)
Me.TabPage5.Controls.Add(Me.GroupBox15)
Me.TabPage5.Controls.Add(Me.TextBox115)
Me.TabPage5.Controls.Add(Me.TextBox95)
Me.TabPage5.Controls.Add(Me.TextBox94)
Me.TabPage5.Controls.Add(Me.TextBox93)
Me.TabPage5.Controls.Add(Me.TextBox92)
Me.TabPage5.Controls.Add(Me.Label57)
Me.TabPage5.Controls.Add(Me.TextBox91)
Me.TabPage5.Controls.Add(Me.TextBox78)
Me.TabPage5.Controls.Add(Me.TextBox79)
Me.TabPage5.Controls.Add(Me.TextBox82)
Me.TabPage5.Controls.Add(Me.TextBox83)
Me.TabPage5.Controls.Add(Me.TextBox84)
Me.TabPage5.Controls.Add(Me.TextBox85)
Me.TabPage5.Controls.Add(Me.TextBox86)
Me.TabPage5.Controls.Add(Me.TextBox87)
Me.TabPage5.Controls.Add(Me.TextBox88)
Me.TabPage5.Controls.Add(Me.TextBox96)
Me.TabPage5.Controls.Add(Me.TextBox97)
Me.TabPage5.Controls.Add(Me.TextBox98)
Me.TabPage5.Controls.Add(Me.TextBox99)
Me.TabPage5.Controls.Add(Me.TextBox100)
Me.TabPage5.Controls.Add(Me.TextBox101)
Me.TabPage5.Controls.Add(Me.TextBox102)
Me.TabPage5.Controls.Add(Me.TextBox103)
Me.TabPage5.Controls.Add(Me.TextBox104)
Me.TabPage5.Controls.Add(Me.TextBox105)
Me.TabPage5.Controls.Add(Me.TextBox106)
Me.TabPage5.Controls.Add(Me.TextBox107)
Me.TabPage5.Controls.Add(Me.TextBox108)
Me.TabPage5.Controls.Add(Me.TextBox109)
Me.TabPage5.Controls.Add(Me.TextBox110)
Me.TabPage5.Controls.Add(Me.TextBox111)
Me.TabPage5.Controls.Add(Me.ListBox2)
Me.TabPage5.Controls.Add(Me.Label41)
Me.TabPage5.Controls.Add(Me.Label40)
Me.TabPage5.Controls.Add(Me.ListBox1)
Me.TabPage5.Location = New System.Drawing.Point(4, 28)
Me.TabPage5.Name = "TabPage5"
Me.TabPage5.Size = New System.Drawing.Size(1264, 792)
Me.TabPage5.TabIndex = 4
Me.TabPage5.Text = "State-Action Space"
'
'GroupBox32
'
Me.GroupBox32.Controls.Add(Me.ComboBox7)
Me.GroupBox32.Controls.Add(Me.Button59)
Me.GroupBox32.Location = New System.Drawing.Point(16, 216)
```

```
Me.GroupBox32.Name = "GroupBox32"
Me.GroupBox32.Size = New System.Drawing.Size(304, 80)
Me.GroupBox32.TabIndex = 183
Me.GroupBox32.TabStop = False
Me.GroupBox32.Text = "Load Optimal Policy"
'
'ComboBox7
'
Me.ComboBox7.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ComboBox7.Items.AddRange(New Object() {"Optimal Policy1", "Optimal Policy2", "Optimal Policy3", "Policy
After System Crash"})
Me.ComboBox7.Location = New System.Drawing.Point(112, 32)
Me.ComboBox7.Name = "ComboBox7"
Me.ComboBox7.Size = New System.Drawing.Size(176, 22)
Me.ComboBox7.TabIndex = 183
Me.ComboBox7.Text = "ComboBox7"
'
'Button59
'
Me.Button59.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button59.Location = New System.Drawing.Point(16, 24)
Me.Button59.Name = "Button59"
Me.Button59.Size = New System.Drawing.Size(80, 48)
Me.Button59.TabIndex = 182
Me.Button59.Text = "Load Policy"
'
'GroupBox27
'
Me.GroupBox27.Controls.Add(Me.Label110)
Me.GroupBox27.Controls.Add(Me.TextBox161)
Me.GroupBox27.Controls.Add(Me.TextBox157)
Me.GroupBox27.Controls.Add(Me.TextBox156)
Me.GroupBox27.Controls.Add(Me.Label106)
Me.GroupBox27.Controls.Add(Me.Label104)
Me.GroupBox27.Controls.Add(Me.TextBox155)
Me.GroupBox27.Controls.Add(Me.TextBox154)
Me.GroupBox27.Controls.Add(Me.TextBox81)
Me.GroupBox27.Controls.Add(Me.TextBox153)
Me.GroupBox27.Controls.Add(Me.TextBox159)
Me.GroupBox27.Controls.Add(Me.TextBox160)
Me.GroupBox27.Location = New System.Drawing.Point(712, 24)
Me.GroupBox27.Name = "GroupBox27"
Me.GroupBox27.Size = New System.Drawing.Size(352, 128)
Me.GroupBox27.TabIndex = 181
Me.GroupBox27.TabStop = False
Me.GroupBox27.Text = "Current Shaking Parametes"
'
'Label110
'
Me.Label110.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label110.Location = New System.Drawing.Point(8, 64)
Me.Label110.Name = "Label110"
Me.Label110.Size = New System.Drawing.Size(112, 16)
Me.Label110.TabIndex = 190
Me.Label110.Text = "Arm Amplitudes (-):"
'
'TextBox161
'
Me.TextBox161.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
```

```
Me.TextBox161.Location = New System.Drawing.Point(272, 64)
Me.TextBox161.Name = "TextBox161"
Me.TextBox161.Size = New System.Drawing.Size(64, 20)
Me.TextBox161.TabIndex = 189
Me.TextBox161.Text = "-030.000"
'
'TextBox157
'
Me.TextBox157.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox157.Location = New System.Drawing.Point(272, 32)
Me.TextBox157.Name = "TextBox157"
Me.TextBox157.Size = New System.Drawing.Size(64, 20)
Me.TextBox157.TabIndex = 188
Me.TextBox157.Text = "030.000"
'
'TextBox156
'
Me.TextBox156.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox156.Location = New System.Drawing.Point(200, 32)
Me.TextBox156.Name = "TextBox156"
Me.TextBox156.Size = New System.Drawing.Size(64, 20)
Me.TextBox156.TabIndex = 187
Me.TextBox156.Text = "030.000"
'
'Label106
'
Me.Label106.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label106.Location = New System.Drawing.Point(8, 32)
Me.Label106.Name = "Label106"
Me.Label106.Size = New System.Drawing.Size(112, 16)
Me.Label106.TabIndex = 186
Me.Label106.Text = "Arm Amplitudes (+):"
'
'Label104
'
Me.Label104.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.Label104.Location = New System.Drawing.Point(32, 96)
Me.Label104.Name = "Label104"
Me.Label104.Size = New System.Drawing.Size(80, 16)
Me.Label104.TabIndex = 185
Me.Label104.Text = "Arm Speeds:"
'
'TextBox155
'
Me.TextBox155.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox155.Location = New System.Drawing.Point(272, 96)
Me.TextBox155.Name = "TextBox155"
Me.TextBox155.Size = New System.Drawing.Size(64, 20)
Me.TextBox155.TabIndex = 184
Me.TextBox155.Text = "1000"
'
'TextBox154
'
Me.TextBox154.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox154.Location = New System.Drawing.Point(200, 96)
Me.TextBox154.Name = "TextBox154"
Me.TextBox154.Size = New System.Drawing.Size(64, 20)
```

```
Me.TextBox154.TabIndex = 183
Me.TextBox154.Text = "1000"
'
'TextBox81
'
Me.TextBox81.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox81.Location = New System.Drawing.Point(128, 32)
Me.TextBox81.Name = "TextBox81"
Me.TextBox81.Size = New System.Drawing.Size(64, 20)
Me.TextBox81.TabIndex = 159
Me.TextBox81.Text = "030.000"
'
'TextBox153
'
Me.TextBox153.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox153.Location = New System.Drawing.Point(128, 96)
Me.TextBox153.Name = "TextBox153"
Me.TextBox153.Size = New System.Drawing.Size(64, 20)
Me.TextBox153.TabIndex = 182
Me.TextBox153.Text = "1000"
'
'TextBox159
'
Me.TextBox159.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox159.Location = New System.Drawing.Point(128, 64)
Me.TextBox159.Name = "TextBox159"
Me.TextBox159.Size = New System.Drawing.Size(64, 20)
Me.TextBox159.TabIndex = 192
Me.TextBox159.Text = "-030.000"
'
'TextBox160
'
Me.TextBox160.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(177, Byte))
Me.TextBox160.Location = New System.Drawing.Point(200, 64)
Me.TextBox160.Name = "TextBox160"
Me.TextBox160.Size = New System.Drawing.Size(64, 20)
Me.TextBox160.TabIndex = 191
Me.TextBox160.Text = "-030.000"
'
'GroupBox21
'
Me.GroupBox21.Controls.Add(Me.Button36)
Me.GroupBox21.Controls.Add(Me.Button38)
Me.GroupBox21.Controls.Add(Me.Button39)
Me.GroupBox21.Location = New System.Drawing.Point(16, 16)
Me.GroupBox21.Name = "GroupBox21"
Me.GroupBox21.Size = New System.Drawing.Size(224, 160)
Me.GroupBox21.TabIndex = 180
Me.GroupBox21.TabStop = False
Me.GroupBox21.Text = "Create Real Policy"
'
'Button36
'
Me.Button36.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button36.Location = New System.Drawing.Point(128, 32)
Me.Button36.Name = "Button36"
Me.Button36.Size = New System.Drawing.Size(80, 48)
Me.Button36.TabIndex = 182
```

```
Me.Button36.Text = "System Creates Policy"
'
'Button38
'
Me.Button38.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button38.Location = New System.Drawing.Point(24, 32)
Me.Button38.Name = "Button38"
Me.Button38.Size = New System.Drawing.Size(80, 48)
Me.Button38.TabIndex = 181
Me.Button38.Text = "Human Creates Policy"
'
'Button39
'
Me.Button39.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button39.Location = New System.Drawing.Point(72, 96)
Me.Button39.Name = "Button39"
Me.Button39.Size = New System.Drawing.Size(80, 48)
Me.Button39.TabIndex = 169
Me.Button39.Text = "Reset Policy"
'
'GroupBox16
'
Me.GroupBox16.Controls.Add(Me.Button34)
Me.GroupBox16.Controls.Add(Me.Button31)
Me.GroupBox16.Controls.Add(Me.Button29)
Me.GroupBox16.Controls.Add(Me.Button28)
Me.GroupBox16.Location = New System.Drawing.Point(256, 16)
Me.GroupBox16.Name = "GroupBox16"
Me.GroupBox16.Size = New System.Drawing.Size(200, 200)
Me.GroupBox16.TabIndex = 179
Me.GroupBox16.TabStop = False
Me.GroupBox16.Text = "Run Policy"
'
'Button34
'
Me.Button34.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button34.Location = New System.Drawing.Point(60, 80)
Me.Button34.Name = "Button34"
Me.Button34.Size = New System.Drawing.Size(80, 48)
Me.Button34.TabIndex = 169
Me.Button34.Text = "Shaking Point"
'
'Button31
'
Me.Button31.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button31.Location = New System.Drawing.Point(16, 24)
Me.Button31.Name = "Button31"
Me.Button31.Size = New System.Drawing.Size(80, 48)
Me.Button31.TabIndex = 168
Me.Button31.Text = "Grasp Bag"
'
'Button29
'
Me.Button29.BackColor = System.Drawing.Color.Red
Me.Button29.Cursor = System.Windows.Forms.Cursors.Default
Me.Button29.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button29.ForeColor = System.Drawing.Color.Yellow
Me.Button29.Location = New System.Drawing.Point(16, 136)
```

```
Me.Button29.Name = "Button29"
Me.Button29.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Button29.Size = New System.Drawing.Size(168, 48)
Me.Button29.TabIndex = 166
Me.Button29.Text = "Emergency Stop"
'
'Button28
'
Me.Button28.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button28.Location = New System.Drawing.Point(104, 24)
Me.Button28.Name = "Button28"
Me.Button28.Size = New System.Drawing.Size(80, 48)
Me.Button28.TabIndex = 165
Me.Button28.Text = "Execute Shaking"
'
'GroupBox15
'
Me.GroupBox15.Controls.Add(Me.Button30)
Me.GroupBox15.Controls.Add(Me.Label60)
Me.GroupBox15.Controls.Add(Me.Label59)
Me.GroupBox15.Controls.Add(Me.Label58)
Me.GroupBox15.Controls.Add(Me.TrackBar1)
Me.GroupBox15.Controls.Add(Me.TextBox116)
Me.GroupBox15.Controls.Add(Me.Button32)
Me.GroupBox15.Location = New System.Drawing.Point(472, 16)
Me.GroupBox15.Name = "GroupBox15"
Me.GroupBox15.Size = New System.Drawing.Size(224, 200)
Me.GroupBox15.TabIndex = 178
Me.GroupBox15.TabStop = False
Me.GroupBox15.Text = "Create Random Policy"
'
'Button30
'
Me.Button30.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button30.Location = New System.Drawing.Point(24, 136)
Me.Button30.Name = "Button30"
Me.Button30.Size = New System.Drawing.Size(80, 48)
Me.Button30.TabIndex = 181
Me.Button30.Text = "Create Policy"
'
'Label60
'
Me.Label60.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label60.Location = New System.Drawing.Point(16, 96)
Me.Label60.Name = "Label60"
Me.Label60.Size = New System.Drawing.Size(104, 16)
Me.Label60.TabIndex = 178
Me.Label60.Text = "Number of actions:"
'
'Label59
'
Me.Label59.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label59.Location = New System.Drawing.Point(184, 72)
Me.Label59.Name = "Label59"
Me.Label59.Size = New System.Drawing.Size(24, 16)
Me.Label59.TabIndex = 177
Me.Label59.Text = "500"
'
'Label58
```

```
'
Me.Label58.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label58.Location = New System.Drawing.Point(24, 72)
Me.Label58.Name = "Label58"
Me.Label58.Size = New System.Drawing.Size(8, 16)
Me.Label58.TabIndex = 176
Me.Label58.Text = "0"
'
'TrackBar1
'
Me.TrackBar1.LargeChange = 1
Me.TrackBar1.Location = New System.Drawing.Point(16, 24)
Me.TrackBar1.Maximum = 500
Me.TrackBar1.Name = "TrackBar1"
Me.TrackBar1.Size = New System.Drawing.Size(192, 45)
Me.TrackBar1.TabIndex = 175
Me.TrackBar1.Value = 25
'
'TextBox116
'
Me.TextBox116.Location = New System.Drawing.Point(128, 88)
Me.TextBox116.Name = "TextBox116"
Me.TextBox116.Size = New System.Drawing.Size(32, 26)
Me.TextBox116.TabIndex = 180
Me.TextBox116.Text = ""
'
'Button32
'
Me.Button32.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button32.Location = New System.Drawing.Point(120, 136)
Me.Button32.Name = "Button32"
Me.Button32.Size = New System.Drawing.Size(80, 48)
Me.Button32.TabIndex = 169
Me.Button32.Text = "Reset Policy"
'
'TextBox115
'
Me.TextBox115.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox115.Location = New System.Drawing.Point(224, 392)
Me.TextBox115.Name = "TextBox115"
Me.TextBox115.Size = New System.Drawing.Size(32, 20)
Me.TextBox115.TabIndex = 172
Me.TextBox115.Text = ""
'
'TextBox95
'
Me.TextBox95.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox95.Location = New System.Drawing.Point(224, 360)
Me.TextBox95.Name = "TextBox95"
Me.TextBox95.Size = New System.Drawing.Size(32, 20)
Me.TextBox95.TabIndex = 171
Me.TextBox95.Text = ""
'
'TextBox94
'
Me.TextBox94.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox94.Location = New System.Drawing.Point(816, 552)
Me.TextBox94.Multiline = True
```

```
Me.TextBox94.Name = "TextBox94"
Me.TextBox94.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox94.Size = New System.Drawing.Size(304, 64)
Me.TextBox94.TabIndex = 164
Me.TextBox94.Text = ""
'
'TextBox93
'
Me.TextBox93.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox93.Location = New System.Drawing.Point(816, 624)
Me.TextBox93.Multiline = True
Me.TextBox93.Name = "TextBox93"
Me.TextBox93.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox93.Size = New System.Drawing.Size(304, 72)
Me.TextBox93.TabIndex = 163
Me.TextBox93.Text = ""
'
'TextBox92
'
Me.TextBox92.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox92.Location = New System.Drawing.Point(816, 320)
Me.TextBox92.Multiline = True
Me.TextBox92.Name = "TextBox92"
Me.TextBox92.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox92.Size = New System.Drawing.Size(304, 64)
Me.TextBox92.TabIndex = 162
Me.TextBox92.Text = ""
'
'Label57
'
Me.Label57.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label57.Location = New System.Drawing.Point(280, 640)
Me.Label57.Name = "Label57"
Me.Label57.Size = New System.Drawing.Size(88, 16)
Me.Label57.TabIndex = 161
Me.Label57.Text = "Action Counter:"
'
'TextBox91
'
Me.TextBox91.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox91.Location = New System.Drawing.Point(376, 640)
Me.TextBox91.Name = "TextBox91"
Me.TextBox91.Size = New System.Drawing.Size(32, 20)
Me.TextBox91.TabIndex = 160
Me.TextBox91.Text = ""
'
'TextBox78
'
Me.TextBox78.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox78.Location = New System.Drawing.Point(816, 400)
Me.TextBox78.Multiline = True
Me.TextBox78.Name = "TextBox78"
Me.TextBox78.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox78.Size = New System.Drawing.Size(304, 64)
Me.TextBox78.TabIndex = 147
Me.TextBox78.Text = ""
'
'TextBox79
```

```
'
Me.TextBox79.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox79.Location = New System.Drawing.Point(488, 680)
Me.TextBox79.Multiline = True
Me.TextBox79.Name = "TextBox79"
Me.TextBox79.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox79.Size = New System.Drawing.Size(56, 24)
Me.TextBox79.TabIndex = 146
Me.TextBox79.Text = "END"
'
'TextBox82
'
Me.TextBox82.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox82.Location = New System.Drawing.Point(632, 656)
Me.TextBox82.Multiline = True
Me.TextBox82.Name = "TextBox82"
Me.TextBox82.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox82.Size = New System.Drawing.Size(64, 24)
Me.TextBox82.TabIndex = 143
Me.TextBox82.Text = "500"
'
'TextBox83
'
Me.TextBox83.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox83.Location = New System.Drawing.Point(488, 656)
Me.TextBox83.Multiline = True
Me.TextBox83.Name = "TextBox83"
Me.TextBox83.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox83.Size = New System.Drawing.Size(144, 24)
Me.TextBox83.TabIndex = 142
Me.TextBox83.Text = "IMOV P000 V="
'
'TextBox84
'
Me.TextBox84.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox84.Location = New System.Drawing.Point(488, 632)
Me.TextBox84.Multiline = True
Me.TextBox84.Name = "TextBox84"
Me.TextBox84.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox84.Size = New System.Drawing.Size(144, 24)
Me.TextBox84.TabIndex = 141
Me.TextBox84.Text = "NOP"
'
'TextBox85
'
Me.TextBox85.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox85.Location = New System.Drawing.Point(488, 608)
Me.TextBox85.Multiline = True
Me.TextBox85.Name = "TextBox85"
Me.TextBox85.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox85.Size = New System.Drawing.Size(144, 24)
Me.TextBox85.TabIndex = 140
Me.TextBox85.Text = "///GROUP1 RB1"
'
'TextBox86
'
Me.TextBox86.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.TextBox86.Location = New System.Drawing.Point(488, 584)
Me.TextBox86.Multiline = True
Me.TextBox86.Name = "TextBox86"
Me.TextBox86.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox86.Size = New System.Drawing.Size(144, 24)
Me.TextBox86.TabIndex = 139
Me.TextBox86.Text = "///ATTR SC,RW"
 '
 'TextBox87
 '
Me.TextBox87.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox87.Location = New System.Drawing.Point(488, 560)
Me.TextBox87.Multiline = True
Me.TextBox87.Name = "TextBox87"
Me.TextBox87.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox87.Size = New System.Drawing.Size(184, 24)
Me.TextBox87.TabIndex = 138
Me.TextBox87.Text = "///DATE 2053/11/21 21:42"
 '
 'TextBox88
 '
Me.TextBox88.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox88.Location = New System.Drawing.Point(488, 536)
Me.TextBox88.Multiline = True
Me.TextBox88.Name = "TextBox88"
Me.TextBox88.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox88.Size = New System.Drawing.Size(144, 24)
Me.TextBox88.TabIndex = 137
Me.TextBox88.Text = "//INST"
 '
 'TextBox96
 '
Me.TextBox96.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox96.Location = New System.Drawing.Point(992, 512)
Me.TextBox96.Multiline = True
Me.TextBox96.Name = "TextBox96"
Me.TextBox96.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox96.Size = New System.Drawing.Size(68, 24)
Me.TextBox96.TabIndex = 129
Me.TextBox96.Text = "0.00"
 '
 'TextBox97
 '
Me.TextBox97.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox97.Location = New System.Drawing.Point(920, 512)
Me.TextBox97.Multiline = True
Me.TextBox97.Name = "TextBox97"
Me.TextBox97.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox97.Size = New System.Drawing.Size(68, 24)
Me.TextBox97.TabIndex = 128
Me.TextBox97.Text = "0.00"
 '
 'TextBox98
 '
Me.TextBox98.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox98.Location = New System.Drawing.Point(848, 512)
Me.TextBox98.Multiline = True
Me.TextBox98.Name = "TextBox98"
```

```
Me.TextBox98.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox98.Size = New System.Drawing.Size(68, 24)
Me.TextBox98.TabIndex = 127
Me.TextBox98.Text = "0.00"
'
'TextBox99
'
Me.TextBox99.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox99.Location = New System.Drawing.Point(776, 512)
Me.TextBox99.Multiline = True
Me.TextBox99.Name = "TextBox99"
Me.TextBox99.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox99.Size = New System.Drawing.Size(72, 24)
Me.TextBox99.TabIndex = 126
Me.TextBox99.Text = "000.000"
'
'TextBox100
'
Me.TextBox100.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox100.Location = New System.Drawing.Point(704, 512)
Me.TextBox100.Multiline = True
Me.TextBox100.Name = "TextBox100"
Me.TextBox100.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox100.Size = New System.Drawing.Size(72, 24)
Me.TextBox100.TabIndex = 125
Me.TextBox100.Text = "000.000"
'
'TextBox101
'
Me.TextBox101.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox101.Location = New System.Drawing.Point(632, 512)
Me.TextBox101.Multiline = True
Me.TextBox101.Name = "TextBox101"
Me.TextBox101.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox101.Size = New System.Drawing.Size(72, 24)
Me.TextBox101.TabIndex = 124
Me.TextBox101.Text = "000.000"
'
'TextBox102
'
Me.TextBox102.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox102.Location = New System.Drawing.Point(488, 512)
Me.TextBox102.Multiline = True
Me.TextBox102.Name = "TextBox102"
Me.TextBox102.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox102.Size = New System.Drawing.Size(144, 24)
Me.TextBox102.TabIndex = 123
Me.TextBox102.Text = "P0000="
'
'TextBox103
'
Me.TextBox103.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox103.Location = New System.Drawing.Point(488, 488)
Me.TextBox103.Multiline = True
Me.TextBox103.Name = "TextBox103"
Me.TextBox103.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox103.Size = New System.Drawing.Size(216, 24)
Me.TextBox103.TabIndex = 122
```

```
Me.TextBox103.Text = "///RCONF 0,0,0,0,0,0,0,0"
'
'TextBox104
'
Me.TextBox104.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox104.Location = New System.Drawing.Point(488, 464)
Me.TextBox104.Multiline = True
Me.TextBox104.Name = "TextBox104"
Me.TextBox104.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox104.Size = New System.Drawing.Size(144, 24)
Me.TextBox104.TabIndex = 121
Me.TextBox104.Text = "///RECTAN"
'
'TextBox105
'
Me.TextBox105.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox105.Location = New System.Drawing.Point(488, 440)
Me.TextBox105.Multiline = True
Me.TextBox105.Name = "TextBox105"
Me.TextBox105.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox105.Size = New System.Drawing.Size(144, 24)
Me.TextBox105.TabIndex = 120
Me.TextBox105.Text = "///POSTYPE ROBOT"
'
'TextBox106
'
Me.TextBox106.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox106.Location = New System.Drawing.Point(488, 416)
Me.TextBox106.Multiline = True
Me.TextBox106.Name = "TextBox106"
Me.TextBox106.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox106.Size = New System.Drawing.Size(144, 24)
Me.TextBox106.TabIndex = 119
Me.TextBox106.Text = "///TOOL 0"
'
'TextBox107
'
Me.TextBox107.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox107.Location = New System.Drawing.Point(488, 392)
Me.TextBox107.Multiline = True
Me.TextBox107.Name = "TextBox107"
Me.TextBox107.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox107.Size = New System.Drawing.Size(144, 24)
Me.TextBox107.TabIndex = 118
Me.TextBox107.Text = "///NPOS 0,0,0,0,0,0"
'
'TextBox108
'
Me.TextBox108.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox108.Location = New System.Drawing.Point(488, 368)
Me.TextBox108.Multiline = True
Me.TextBox108.Name = "TextBox108"
Me.TextBox108.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox108.Size = New System.Drawing.Size(144, 24)
Me.TextBox108.TabIndex = 117
Me.TextBox108.Text = "//POS"
'
'TextBox109
```

```
'
Me.TextBox109.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox109.Location = New System.Drawing.Point(640, 344)
Me.TextBox109.Multiline = True
Me.TextBox109.Name = "TextBox109"
Me.TextBox109.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox109.Size = New System.Drawing.Size(144, 24)
Me.TextBox109.TabIndex = 116
Me.TextBox109.Text = "POLICY1"
'
'TextBox110
'
Me.TextBox110.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox110.Location = New System.Drawing.Point(488, 344)
Me.TextBox110.Multiline = True
Me.TextBox110.Name = "TextBox110"
Me.TextBox110.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox110.Size = New System.Drawing.Size(144, 24)
Me.TextBox110.TabIndex = 115
Me.TextBox110.Text = "//NAME "
'
'TextBox111
'
Me.TextBox111.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.TextBox111.Location = New System.Drawing.Point(488, 320)
Me.TextBox111.Multiline = True
Me.TextBox111.Name = "TextBox111"
Me.TextBox111.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox111.Size = New System.Drawing.Size(144, 24)
Me.TextBox111.TabIndex = 114
Me.TextBox111.Text = "/JOB"
'
'ListBox2
'
Me.ListBox2.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox2.ItemHeight = 14
Me.ListBox2.Location = New System.Drawing.Point(264, 336)
Me.ListBox2.Name = "ListBox2"
Me.ListBox2.Size = New System.Drawing.Size(200, 284)
Me.ListBox2.TabIndex = 91
'
'Label41
'
Me.Label41.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label41.Location = New System.Drawing.Point(304, 312)
Me.Label41.Name = "Label41"
Me.Label41.Size = New System.Drawing.Size(112, 16)
Me.Label41.TabIndex = 90
Me.Label41.Text = "Possible Action"
'
'Label40
'
Me.Label40.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label40.Location = New System.Drawing.Point(88, 312)
Me.Label40.Name = "Label40"
Me.Label40.Size = New System.Drawing.Size(80, 16)
Me.Label40.TabIndex = 87
```

```
Me.Label40.Text = "States"
'
'ListBox1
'
Me.ListBox1.Font = New System.Drawing.Font("Arial", 8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.ListBox1.ItemHeight = 14
Me.ListBox1.Location = New System.Drawing.Point(16, 336)
Me.ListBox1.Name = "ListBox1"
Me.ListBox1.Size = New System.Drawing.Size(200, 284)
Me.ListBox1.TabIndex = 9
'
'TabPage4
'
Me.TabPage4.Controls.Add(Me.TextBox74)
Me.TabPage4.Controls.Add(Me.TextBox73)
Me.TabPage4.Controls.Add(Me.TextBox72)
Me.TabPage4.Controls.Add(Me.TextBox71)
Me.TabPage4.Controls.Add(Me.TextBox70)
Me.TabPage4.Controls.Add(Me.TextBox69)
Me.TabPage4.Controls.Add(Me.TextBox68)
Me.TabPage4.Controls.Add(Me.TextBox67)
Me.TabPage4.Controls.Add(Me.TextBox66)
Me.TabPage4.Controls.Add(Me.TextBox65)
Me.TabPage4.Controls.Add(Me.TextBox64)
Me.TabPage4.Controls.Add(Me.TextBox57)
Me.TabPage4.Controls.Add(Me.TextBox58)
Me.TabPage4.Controls.Add(Me.TextBox59)
Me.TabPage4.Controls.Add(Me.TextBox60)
Me.TabPage4.Controls.Add(Me.TextBox61)
Me.TabPage4.Controls.Add(Me.TextBox62)
Me.TabPage4.Controls.Add(Me.TextBox63)
Me.TabPage4.Controls.Add(Me.TextBox56)
Me.TabPage4.Controls.Add(Me.TextBox55)
Me.TabPage4.Controls.Add(Me.TextBox54)
Me.TabPage4.Controls.Add(Me.TextBox53)
Me.TabPage4.Controls.Add(Me.TextBox52)
Me.TabPage4.Controls.Add(Me.TextBox51)
Me.TabPage4.Controls.Add(Me.TextBox50)
Me.TabPage4.Controls.Add(Me.TextBox48)
Me.TabPage4.Controls.Add(Me.TextBox47)
Me.TabPage4.Controls.Add(Me.TextBox46)
Me.TabPage4.Controls.Add(Me.TextBox45)
Me.TabPage4.Controls.Add(Me.TextBox44)
Me.TabPage4.Controls.Add(Me.TextBox43)
Me.TabPage4.Controls.Add(Me.TextBox42)
Me.TabPage4.Controls.Add(Me.TextBox41)
Me.TabPage4.Controls.Add(Me.TextBox40)
Me.TabPage4.Location = New System.Drawing.Point(4, 28)
Me.TabPage4.Name = "TabPage4"
Me.TabPage4.Size = New System.Drawing.Size(1264, 792)
Me.TabPage4.TabIndex = 3
Me.TabPage4.Text = "Shaking Editor"
'
'TextBox74
'
Me.TextBox74.Location = New System.Drawing.Point(456, 24)
Me.TextBox74.Multiline = True
Me.TextBox74.Name = "TextBox74"
Me.TextBox74.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox74.Size = New System.Drawing.Size(280, 256)
Me.TextBox74.TabIndex = 113
Me.TextBox74.Text = ""
```

```
'
'TextBox73
'
Me.TextBox73.Location = New System.Drawing.Point(16, 600)
Me.TextBox73.Multiline = True
Me.TextBox73.Name = "TextBox73"
Me.TextBox73.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox73.Size = New System.Drawing.Size(144, 24)
Me.TextBox73.TabIndex = 112
Me.TextBox73.Text = "END"
'
'TextBox72
'
Me.TextBox72.Location = New System.Drawing.Point(168, 568)
Me.TextBox72.Multiline = True
Me.TextBox72.Name = "TextBox72"
Me.TextBox72.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox72.Size = New System.Drawing.Size(64, 24)
Me.TextBox72.TabIndex = 111
Me.TextBox72.Text = "500"
'
'TextBox71
'
Me.TextBox71.Location = New System.Drawing.Point(16, 568)
Me.TextBox71.Multiline = True
Me.TextBox71.Name = "TextBox71"
Me.TextBox71.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox71.Size = New System.Drawing.Size(144, 24)
Me.TextBox71.TabIndex = 110
Me.TextBox71.Text = "IMOV P001 V="
'
'TextBox70
'
Me.TextBox70.Location = New System.Drawing.Point(168, 536)
Me.TextBox70.Multiline = True
Me.TextBox70.Name = "TextBox70"
Me.TextBox70.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox70.Size = New System.Drawing.Size(64, 24)
Me.TextBox70.TabIndex = 109
Me.TextBox70.Text = "500"
'
'TextBox69
'
Me.TextBox69.Location = New System.Drawing.Point(16, 536)
Me.TextBox69.Multiline = True
Me.TextBox69.Name = "TextBox69"
Me.TextBox69.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox69.Size = New System.Drawing.Size(144, 24)
Me.TextBox69.TabIndex = 108
Me.TextBox69.Text = "IMOV P000 V="
'
'TextBox68
'
Me.TextBox68.Location = New System.Drawing.Point(16, 504)
Me.TextBox68.Multiline = True
Me.TextBox68.Name = "TextBox68"
Me.TextBox68.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox68.Size = New System.Drawing.Size(144, 24)
Me.TextBox68.TabIndex = 107
Me.TextBox68.Text = "NOP"
'
'TextBox67
'
```

```
Me.TextBox67.Location = New System.Drawing.Point(16, 472)
Me.TextBox67.Multiline = True
Me.TextBox67.Name = "TextBox67"
Me.TextBox67.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox67.Size = New System.Drawing.Size(144, 24)
Me.TextBox67.TabIndex = 106
Me.TextBox67.Text = "///GROUP1 RB1"
 '
 'TextBox66
 '
Me.TextBox66.Location = New System.Drawing.Point(16, 440)
Me.TextBox66.Multiline = True
Me.TextBox66.Name = "TextBox66"
Me.TextBox66.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox66.Size = New System.Drawing.Size(144, 24)
Me.TextBox66.TabIndex = 105
Me.TextBox66.Text = "///ATTR SC,RW"
 '
 'TextBox65
 '
Me.TextBox65.Location = New System.Drawing.Point(16, 408)
Me.TextBox65.Multiline = True
Me.TextBox65.Name = "TextBox65"
Me.TextBox65.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox65.Size = New System.Drawing.Size(184, 24)
Me.TextBox65.TabIndex = 104
Me.TextBox65.Text = "///DATE 2053/11/21 21:42"
 '
 'TextBox64
 '
Me.TextBox64.Location = New System.Drawing.Point(16, 376)
Me.TextBox64.Multiline = True
Me.TextBox64.Name = "TextBox64"
Me.TextBox64.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox64.Size = New System.Drawing.Size(144, 24)
Me.TextBox64.TabIndex = 103
Me.TextBox64.Text = "//INST"
 '
 'TextBox57
 '
Me.TextBox57.Location = New System.Drawing.Point(600, 344)
Me.TextBox57.Multiline = True
Me.TextBox57.Name = "TextBox57"
Me.TextBox57.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox57.Size = New System.Drawing.Size(68, 24)
Me.TextBox57.TabIndex = 102
Me.TextBox57.Text = "0.00"
 '
 'TextBox58
 '
Me.TextBox58.Location = New System.Drawing.Point(520, 344)
Me.TextBox58.Multiline = True
Me.TextBox58.Name = "TextBox58"
Me.TextBox58.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox58.Size = New System.Drawing.Size(68, 24)
Me.TextBox58.TabIndex = 101
Me.TextBox58.Text = "0.00"
 '
 'TextBox59
 '
Me.TextBox59.Location = New System.Drawing.Point(440, 344)
Me.TextBox59.Multiline = True
Me.TextBox59.Name = "TextBox59"
```

```
Me.TextBox59.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox59.Size = New System.Drawing.Size(68, 24)
Me.TextBox59.TabIndex = 100
Me.TextBox59.Text = "0.00"
'
'TextBox60
'
Me.TextBox60.Location = New System.Drawing.Point(352, 344)
Me.TextBox60.Multiline = True
Me.TextBox60.Name = "TextBox60"
Me.TextBox60.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox60.Size = New System.Drawing.Size(72, 24)
Me.TextBox60.TabIndex = 99
Me.TextBox60.Text = "000.000"
'
'TextBox61
'
Me.TextBox61.Location = New System.Drawing.Point(264, 344)
Me.TextBox61.Multiline = True
Me.TextBox61.Name = "TextBox61"
Me.TextBox61.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox61.Size = New System.Drawing.Size(72, 24)
Me.TextBox61.TabIndex = 98
Me.TextBox61.Text = "000.000"
'
'TextBox62
'
Me.TextBox62.Location = New System.Drawing.Point(176, 344)
Me.TextBox62.Multiline = True
Me.TextBox62.Name = "TextBox62"
Me.TextBox62.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox62.Size = New System.Drawing.Size(72, 24)
Me.TextBox62.TabIndex = 97
Me.TextBox62.Text = "-100.000"
'
'TextBox63
'
Me.TextBox63.Location = New System.Drawing.Point(16, 344)
Me.TextBox63.Multiline = True
Me.TextBox63.Name = "TextBox63"
Me.TextBox63.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox63.Size = New System.Drawing.Size(144, 24)
Me.TextBox63.TabIndex = 96
Me.TextBox63.Text = "P0001="
'
'TextBox56
'
Me.TextBox56.Location = New System.Drawing.Point(600, 312)
Me.TextBox56.Multiline = True
Me.TextBox56.Name = "TextBox56"
Me.TextBox56.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox56.Size = New System.Drawing.Size(68, 24)
Me.TextBox56.TabIndex = 95
Me.TextBox56.Text = "0.00"
'
'TextBox55
'
Me.TextBox55.Location = New System.Drawing.Point(520, 312)
Me.TextBox55.Multiline = True
Me.TextBox55.Name = "TextBox55"
Me.TextBox55.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox55.Size = New System.Drawing.Size(68, 24)
Me.TextBox55.TabIndex = 94
```

```
Me.TextBox55.Text = "0.00"
'
'TextBox54
'
Me.TextBox54.Location = New System.Drawing.Point(440, 312)
Me.TextBox54.Multiline = True
Me.TextBox54.Name = "TextBox54"
Me.TextBox54.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox54.Size = New System.Drawing.Size(68, 24)
Me.TextBox54.TabIndex = 93
Me.TextBox54.Text = "0.00"
'
'TextBox53
'
Me.TextBox53.Location = New System.Drawing.Point(352, 312)
Me.TextBox53.Multiline = True
Me.TextBox53.Name = "TextBox53"
Me.TextBox53.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox53.Size = New System.Drawing.Size(72, 24)
Me.TextBox53.TabIndex = 92
Me.TextBox53.Text = "000.000"
'
'TextBox52
'
Me.TextBox52.Location = New System.Drawing.Point(264, 312)
Me.TextBox52.Multiline = True
Me.TextBox52.Name = "TextBox52"
Me.TextBox52.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox52.Size = New System.Drawing.Size(72, 24)
Me.TextBox52.TabIndex = 91
Me.TextBox52.Text = "000.000"
'
'TextBox51
'
Me.TextBox51.Location = New System.Drawing.Point(176, 312)
Me.TextBox51.Multiline = True
Me.TextBox51.Name = "TextBox51"
Me.TextBox51.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox51.Size = New System.Drawing.Size(72, 24)
Me.TextBox51.TabIndex = 90
Me.TextBox51.Text = "100.000"
'
'TextBox50
'
Me.TextBox50.Location = New System.Drawing.Point(16, 312)
Me.TextBox50.Multiline = True
Me.TextBox50.Name = "TextBox50"
Me.TextBox50.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox50.Size = New System.Drawing.Size(144, 24)
Me.TextBox50.TabIndex = 89
Me.TextBox50.Text = "P0000="
'
'TextBox48
'
Me.TextBox48.Location = New System.Drawing.Point(16, 256)
Me.TextBox48.Multiline = True
Me.TextBox48.Name = "TextBox48"
Me.TextBox48.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox48.Size = New System.Drawing.Size(216, 24)
Me.TextBox48.TabIndex = 87
Me.TextBox48.Text = "///RCONF 0,0,0,0,0,0,0,0"
'
'TextBox47
```

```
'
Me.TextBox47.Location = New System.Drawing.Point(16, 224)
Me.TextBox47.Multiline = True
Me.TextBox47.Name = "TextBox47"
Me.TextBox47.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox47.Size = New System.Drawing.Size(144, 24)
Me.TextBox47.TabIndex = 86
Me.TextBox47.Text = "///RECTAN"
'
'TextBox46
'
Me.TextBox46.Location = New System.Drawing.Point(16, 192)
Me.TextBox46.Multiline = True
Me.TextBox46.Name = "TextBox46"
Me.TextBox46.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox46.Size = New System.Drawing.Size(144, 24)
Me.TextBox46.TabIndex = 85
Me.TextBox46.Text = "///POSTYPE ROBOT"
'
'TextBox45
'
Me.TextBox45.Location = New System.Drawing.Point(16, 160)
Me.TextBox45.Multiline = True
Me.TextBox45.Name = "TextBox45"
Me.TextBox45.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox45.Size = New System.Drawing.Size(144, 24)
Me.TextBox45.TabIndex = 84
Me.TextBox45.Text = "///TOOL 0"
'
'TextBox44
'
Me.TextBox44.Location = New System.Drawing.Point(16, 128)
Me.TextBox44.Multiline = True
Me.TextBox44.Name = "TextBox44"
Me.TextBox44.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox44.Size = New System.Drawing.Size(144, 24)
Me.TextBox44.TabIndex = 83
Me.TextBox44.Text = "///NPOS 0,0,0,2,0,0"
'
'TextBox43
'
Me.TextBox43.Location = New System.Drawing.Point(16, 96)
Me.TextBox43.Multiline = True
Me.TextBox43.Name = "TextBox43"
Me.TextBox43.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox43.Size = New System.Drawing.Size(144, 24)
Me.TextBox43.TabIndex = 82
Me.TextBox43.Text = "//POS"
'
'TextBox42
'
Me.TextBox42.Location = New System.Drawing.Point(176, 64)
Me.TextBox42.Multiline = True
Me.TextBox42.Name = "TextBox42"
Me.TextBox42.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox42.Size = New System.Drawing.Size(144, 24)
Me.TextBox42.TabIndex = 81
Me.TextBox42.Text = "SHAKE1"
'
'TextBox41
'
Me.TextBox41.Location = New System.Drawing.Point(16, 64)
Me.TextBox41.Multiline = True
```

```
Me.TextBox41.Name = "TextBox41"
Me.TextBox41.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox41.Size = New System.Drawing.Size(144, 24)
Me.TextBox41.TabIndex = 80
Me.TextBox41.Text = "//NAME "
 '
 'TextBox40
 '
Me.TextBox40.Location = New System.Drawing.Point(16, 32)
Me.TextBox40.Multiline = True
Me.TextBox40.Name = "TextBox40"
Me.TextBox40.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox40.Size = New System.Drawing.Size(144, 24)
Me.TextBox40.TabIndex = 79
Me.TextBox40.Text = "/JOB"
 '
 'TabPage3
 '
Me.TabPage3.Controls.Add(Me.TextBox36)
Me.TabPage3.Controls.Add(Me.TextBox7)
Me.TabPage3.Controls.Add(Me.Button3)
Me.TabPage3.Controls.Add(Me.TextBox35)
Me.TabPage3.Controls.Add(Me.TextBox34)
Me.TabPage3.Controls.Add(Me.TextBox33)
Me.TabPage3.Controls.Add(Me.TextBox32)
Me.TabPage3.Controls.Add(Me.TextBox31)
Me.TabPage3.Controls.Add(Me.TextBox30)
Me.TabPage3.Controls.Add(Me.TextBox29)
Me.TabPage3.Controls.Add(Me.TextBox28)
Me.TabPage3.Controls.Add(Me.TextBox27)
Me.TabPage3.Controls.Add(Me.TextBox26)
Me.TabPage3.Controls.Add(Me.TextBox25)
Me.TabPage3.Controls.Add(Me.TextBox24)
Me.TabPage3.Controls.Add(Me.TextBox23)
Me.TabPage3.Controls.Add(Me.TextBox22)
Me.TabPage3.Controls.Add(Me.TextBox21)
Me.TabPage3.Controls.Add(Me.TextBox20)
Me.TabPage3.Controls.Add(Me.TextBox19)
Me.TabPage3.Controls.Add(Me.TextBox18)
Me.TabPage3.Controls.Add(Me.TextBox17)
Me.TabPage3.Controls.Add(Me.TextBox16)
Me.TabPage3.Controls.Add(Me.TextBox15)
Me.TabPage3.Controls.Add(Me.TextBox14)
Me.TabPage3.Controls.Add(Me.TextBox13)
Me.TabPage3.Controls.Add(Me.TextBox12)
Me.TabPage3.Location = New System.Drawing.Point(4, 28)
Me.TabPage3.Name = "TabPage3"
Me.TabPage3.Size = New System.Drawing.Size(1264, 792)
Me.TabPage3.TabIndex = 2
Me.TabPage3.Text = "Job Editor"
 '
 'TextBox36
 '
Me.TextBox36.Location = New System.Drawing.Point(176, 224)
Me.TextBox36.Multiline = True
Me.TextBox36.Name = "TextBox36"
Me.TextBox36.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox36.Size = New System.Drawing.Size(80, 24)
Me.TextBox36.TabIndex = 103
Me.TextBox36.Text = "ROBOT"
 '
 'TextBox7
 '
```

```
Me.TextBox7.Location = New System.Drawing.Point(160, 96)
Me.TextBox7.Multiline = True
Me.TextBox7.Name = "TextBox7"
Me.TextBox7.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox7.Size = New System.Drawing.Size(48, 24)
Me.TextBox7.TabIndex = 102
Me.TextBox7.Text = "X1"
'
'Button3
'
Me.Button3.Location = New System.Drawing.Point(440, 496)
Me.Button3.Name = "Button3"
Me.Button3.Size = New System.Drawing.Size(120, 32)
Me.Button3.TabIndex = 101
Me.Button3.Text = "Button3"
'
'TextBox35
'
Me.TextBox35.Location = New System.Drawing.Point(8, 544)
Me.TextBox35.Multiline = True
Me.TextBox35.Name = "TextBox35"
Me.TextBox35.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox35.Size = New System.Drawing.Size(184, 24)
Me.TextBox35.TabIndex = 100
Me.TextBox35.Text = "END"
'
'TextBox34
'
Me.TextBox34.Location = New System.Drawing.Point(200, 512)
Me.TextBox34.Multiline = True
Me.TextBox34.Name = "TextBox34"
Me.TextBox34.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox34.Size = New System.Drawing.Size(56, 24)
Me.TextBox34.TabIndex = 99
Me.TextBox34.Text = "100"
'
'TextBox33
'
Me.TextBox33.Location = New System.Drawing.Point(8, 512)
Me.TextBox33.Multiline = True
Me.TextBox33.Name = "TextBox33"
Me.TextBox33.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox33.Size = New System.Drawing.Size(184, 24)
Me.TextBox33.TabIndex = 98
Me.TextBox33.Text = "IMOV P000 V="
'
'TextBox32
'
Me.TextBox32.Location = New System.Drawing.Point(8, 480)
Me.TextBox32.Multiline = True
Me.TextBox32.Name = "TextBox32"
Me.TextBox32.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox32.Size = New System.Drawing.Size(184, 24)
Me.TextBox32.TabIndex = 97
Me.TextBox32.Text = "NOP"
'
'TextBox31
'
Me.TextBox31.Location = New System.Drawing.Point(8, 448)
Me.TextBox31.Multiline = True
Me.TextBox31.Name = "TextBox31"
Me.TextBox31.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox31.Size = New System.Drawing.Size(184, 24)
```

```
Me.TextBox31.TabIndex = 96
Me.TextBox31.Text = "///GROUP1 RB1"
'
'TextBox30
'
Me.TextBox30.Location = New System.Drawing.Point(8, 416)
Me.TextBox30.Multiline = True
Me.TextBox30.Name = "TextBox30"
Me.TextBox30.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox30.Size = New System.Drawing.Size(184, 24)
Me.TextBox30.TabIndex = 95
Me.TextBox30.Text = "///ATTR SC,RW"
'
'TextBox29
'
Me.TextBox29.Location = New System.Drawing.Point(8, 384)
Me.TextBox29.Multiline = True
Me.TextBox29.Name = "TextBox29"
Me.TextBox29.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox29.Size = New System.Drawing.Size(184, 24)
Me.TextBox29.TabIndex = 94
Me.TextBox29.Text = "///DATE 2053/11/21 21:42"
'
'TextBox28
'
Me.TextBox28.Location = New System.Drawing.Point(8, 352)
Me.TextBox28.Multiline = True
Me.TextBox28.Name = "TextBox28"
Me.TextBox28.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox28.Size = New System.Drawing.Size(184, 24)
Me.TextBox28.TabIndex = 93
Me.TextBox28.Text = "//INST"
'
'TextBox27
'
Me.TextBox27.Location = New System.Drawing.Point(528, 320)
Me.TextBox27.Multiline = True
Me.TextBox27.Name = "TextBox27"
Me.TextBox27.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox27.Size = New System.Drawing.Size(80, 24)
Me.TextBox27.TabIndex = 92
Me.TextBox27.Text = "0.00"
'
'TextBox26
'
Me.TextBox26.Location = New System.Drawing.Point(440, 320)
Me.TextBox26.Multiline = True
Me.TextBox26.Name = "TextBox26"
Me.TextBox26.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox26.Size = New System.Drawing.Size(80, 24)
Me.TextBox26.TabIndex = 91
Me.TextBox26.Text = "0.00"
'
'TextBox25
'
Me.TextBox25.Location = New System.Drawing.Point(352, 320)
Me.TextBox25.Multiline = True
Me.TextBox25.Name = "TextBox25"
Me.TextBox25.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox25.Size = New System.Drawing.Size(80, 24)
Me.TextBox25.TabIndex = 90
Me.TextBox25.Text = "0.00"
'
```

```
'TextBox24
'
Me.TextBox24.Location = New System.Drawing.Point(264, 320)
Me.TextBox24.Multiline = True
Me.TextBox24.Name = "TextBox24"
Me.TextBox24.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox24.Size = New System.Drawing.Size(80, 24)
Me.TextBox24.TabIndex = 89
Me.TextBox24.Text = "100.000"
'
'TextBox23
'
Me.TextBox23.Location = New System.Drawing.Point(176, 320)
Me.TextBox23.Multiline = True
Me.TextBox23.Name = "TextBox23"
Me.TextBox23.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox23.Size = New System.Drawing.Size(80, 24)
Me.TextBox23.TabIndex = 88
Me.TextBox23.Text = "100.000"
'
'TextBox22
'
Me.TextBox22.Location = New System.Drawing.Point(88, 320)
Me.TextBox22.Multiline = True
Me.TextBox22.Name = "TextBox22"
Me.TextBox22.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox22.Size = New System.Drawing.Size(80, 24)
Me.TextBox22.TabIndex = 87
Me.TextBox22.Text = "100.000"
'
'TextBox21
'
Me.TextBox21.Location = New System.Drawing.Point(8, 320)
Me.TextBox21.Multiline = True
Me.TextBox21.Name = "TextBox21"
Me.TextBox21.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox21.Size = New System.Drawing.Size(72, 24)
Me.TextBox21.TabIndex = 86
Me.TextBox21.Text = "P0000="
'
'TextBox20
'
Me.TextBox20.Location = New System.Drawing.Point(8, 288)
Me.TextBox20.Multiline = True
Me.TextBox20.Name = "TextBox20"
Me.TextBox20.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox20.Size = New System.Drawing.Size(376, 24)
Me.TextBox20.TabIndex = 85
Me.TextBox20.Text = "///RCONF 0,0,0,0,0,0,0,0"
'
'TextBox19
'
Me.TextBox19.Location = New System.Drawing.Point(8, 256)
Me.TextBox19.Multiline = True
Me.TextBox19.Name = "TextBox19"
Me.TextBox19.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox19.Size = New System.Drawing.Size(248, 24)
Me.TextBox19.TabIndex = 84
Me.TextBox19.Text = "///RECTAN"
'
'TextBox18
'
Me.TextBox18.Location = New System.Drawing.Point(8, 224)
```

```
Me.TextBox18.Multiline = True
Me.TextBox18.Name = "TextBox18"
Me.TextBox18.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox18.Size = New System.Drawing.Size(144, 24)
Me.TextBox18.TabIndex = 83
Me.TextBox18.Text = "///POSTYPE "
'
'TextBox17
'
Me.TextBox17.Location = New System.Drawing.Point(8, 184)
Me.TextBox17.Multiline = True
Me.TextBox17.Name = "TextBox17"
Me.TextBox17.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox17.Size = New System.Drawing.Size(144, 24)
Me.TextBox17.TabIndex = 82
Me.TextBox17.Text = "///TOOL 0"
'
'TextBox16
'
Me.TextBox16.Location = New System.Drawing.Point(8, 152)
Me.TextBox16.Multiline = True
Me.TextBox16.Name = "TextBox16"
Me.TextBox16.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox16.Size = New System.Drawing.Size(144, 24)
Me.TextBox16.TabIndex = 81
Me.TextBox16.Text = "///NPOS 0,0,0,1,0,0"
'
'TextBox15
'
Me.TextBox15.Location = New System.Drawing.Point(8, 128)
Me.TextBox15.Multiline = True
Me.TextBox15.Name = "TextBox15"
Me.TextBox15.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox15.Size = New System.Drawing.Size(144, 24)
Me.TextBox15.TabIndex = 80
Me.TextBox15.Text = "//POS"
'
'TextBox14
'
Me.TextBox14.Location = New System.Drawing.Point(8, 96)
Me.TextBox14.Multiline = True
Me.TextBox14.Name = "TextBox14"
Me.TextBox14.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox14.Size = New System.Drawing.Size(144, 24)
Me.TextBox14.TabIndex = 79
Me.TextBox14.Text = "//NAME "
'
'TextBox13
'
Me.TextBox13.Location = New System.Drawing.Point(8, 64)
Me.TextBox13.Multiline = True
Me.TextBox13.Name = "TextBox13"
Me.TextBox13.ScrollBars = System.Windows.Forms.ScrollBars.Both
Me.TextBox13.Size = New System.Drawing.Size(144, 24)
Me.TextBox13.TabIndex = 78
Me.TextBox13.Text = "/JOB"
'
'TextBox12
'
Me.TextBox12.Location = New System.Drawing.Point(664, 24)
Me.TextBox12.Multiline = True
Me.TextBox12.Name = "TextBox12"
Me.TextBox12.ScrollBars = System.Windows.Forms.ScrollBars.Both
```

```
Me.TextBox12.Size = New System.Drawing.Size(424, 344)
Me.TextBox12.TabIndex = 76
Me.TextBox12.Text = ""
'
'MainMenu1
'
Me.MainMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem1})
'
'MenuItem1
'
Me.MenuItem1.Index = 0
Me.MenuItem1.Text = "Exit"
'
'State_Action_Real_Timer1
'
Me.State_Action_Real_Timer1.Interval = 250
'
'Action_Timer_1
'
Me.Action_Timer_1.Interval = 1
'
'State_Action_Rand_Timer1
'
Me.State_Action_Rand_Timer1.Interval = 250
'
'Shaking_Timer_1
'
Me.Shaking_Timer_1.Interval = 1
'
'Timer2
'
Me.Timer2.Enabled = True
Me.Timer2.Interval = 1
'
'Robot_Operating
'
Me.Robot_Operating.Interval = 500
'
'State_Action_Best_Timer1
'
Me.State_Action_Best_Timer1.Interval = 250
'
'Timer1
'
Me.Timer1.Interval = 10
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.SystemColors.Control
Me.ClientSize = New System.Drawing.Size(1168, 825)
Me.Controls.Add(Me.TabControl1)
Me.Cursor = System.Windows.Forms.Cursors.Default
Me.Font = New System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Location = New System.Drawing.Point(4, 30)
Me.Menu =Me.MainMenu1
Me.Name = "Form1"
Me.RightToLeft = System.Windows.Forms.RightToLeft.No
Me.Text = "Human-Robot Collaboration Learning System"
Me.WindowState = System.Windows.Forms.FormWindowState.Maximized
Me.GroupBox1.ResumeLayout(False)
Me.GroupBox2.ResumeLayout(False)
```

```
Me.GroupBox3.ResumeLayout(False)
Me.GroupBox13.ResumeLayout(False)
Me.GroupBox11.ResumeLayout(False)
Me.GroupBox10.ResumeLayout(False)
Me.GroupBox9.ResumeLayout(False)
Me.GroupBox8.ResumeLayout(False)
Me.GroupBox7.ResumeLayout(False)
Me.GroupBox12.ResumeLayout(False)
Me.GroupBox14.ResumeLayout(False)
Me.GroupBox6.ResumeLayout(False)
Me.TabControl1.ResumeLayout(False)
Me.TabPage8.ResumeLayout(False)
Me.GroupBox38.ResumeLayout(False)
 CType(Me.AxMSChart3, System.ComponentModel.ISupportInitialize).EndInit()
 CType(Me.AxMSChart1, System.ComponentModel.ISupportInitialize).EndInit()
Me.GroupBox31.ResumeLayout(False)
Me.GroupBox18.ResumeLayout(False)
Me.GroupBox29.ResumeLayout(False)
Me.GroupBox30.ResumeLayout(False)
Me.GroupBox19.ResumeLayout(False)
Me.GroupBox20.ResumeLayout(False)
Me.GroupBox34.ResumeLayout(False)
Me.GroupBox4.ResumeLayout(False)
Me.GroupBox33.ResumeLayout(False)
Me.GroupBox35.ResumeLayout(False)
Me.GroupBox36.ResumeLayout(False)
Me.GroupBox37.ResumeLayout(False)
Me.GroupBox17.ResumeLayout(False)
 CType(Me.AxWebBrowser3, System.ComponentModel.ISupportInitialize).EndInit()
Me.TabPage1.ResumeLayout(False)
Me.GroupBox24.ResumeLayout(False)
Me.GroupBox26.ResumeLayout(False)
Me.GroupBox28.ResumeLayout(False)
Me.TabPage7.ResumeLayout(False)
Me.GroupBox25.ResumeLayout(False)
Me.GroupBox23.ResumeLayout(False)
Me.GroupBox22.ResumeLayout(False)
Me.TabPage2.ResumeLayout(False)
Me.GroupBox5.ResumeLayout(False)
 CType(Me.AxWebBrowser1, System.ComponentModel.ISupportInitialize).EndInit()
 CType(Me.AxWebBrowser2, System.ComponentModel.ISupportInitialize).EndInit()
Me.TabPage5.ResumeLayout(False)
Me.GroupBox32.ResumeLayout(False)
Me.GroupBox27.ResumeLayout(False)
Me.GroupBox21.ResumeLayout(False)
Me.GroupBox16.ResumeLayout(False)
Me.GroupBox15.ResumeLayout(False)
 CType(Me.TrackBar1, System.ComponentModel.ISupportInitialize).EndInit()
Me.TabPage4.ResumeLayout(False)
Me.TabPage3.ResumeLayout(False)
Me.ResumeLayout(False)

    End Sub
#End Region

Dim Current_State As String
    'Dim Action As String
Dim Next_State As String
Dim Action_Counter_1 As Integer
Dim Action As String
Dim State_Action_Timer1_Rand_Counter_1 As Integer
Dim State_Action_Timer1_Real_Counter_1 As Integer
Dim State_Action_Timer1_Best_Counter_1 As Integer
```

```
Dim iEnd As Integer
Dim g As Graphics

#Region "Decs"

Dim xConn As sqlConn
    Public Shared item1 As New ListViewItem

Dim r As System.Object
Dim j As System.EventArgs
Dim err As System.Exception
Dim iCounter As Integer

#End Region

#Region "Form Functions"

#End Region
#Region "Upgrade Support "
    Private Shared m_vb6FormDefInstance As Form1
    Private Shared m_InitializingDefInstance As Boolean
    Public Shared Property DefInstance() As Form1
        Get
            If m_vb6FormDefInstance Is Nothing OrElse m_vb6FormDefInstance.IsDisposed Then
                m_InitializingDefInstance = True
                m_vb6FormDefInstance = New Form1
                m_InitializingDefInstance = False
            End If
            DefInstance = m_vb6FormDefInstance
        End Get
        Set(ByVal Value As Form1)
            m_vb6FormDefInstance = Value
        End Set
    End Property
#End Region

    'mode:  0...RS-232C   1...Ethernet
    Function Ms_BscOpenComm(ByVal mode%) As Integer
        '    Dim nCid As Integer
    Dim rc As Integer
    Dim IPAddrress As String
        Ms_BscOpenComm = -1
        If mode = 0 Then
            'Open the port.
            nCid = BscOpen(CurDir$, 1)

            If nCid < 0 Then GoTo Ms_BscOpenComm_Exit

            'Set serial communications parameters. ' Port, Rate, Parity, Bits, Stop
            rc = BscSetCom(nCid, 1, 9600, 0, 8, 0)

        Else
            'Open the Ethernet line.
            nCid = BscOpen(CurDir$, PACKETETHERNET)
            If nCid < 0 Then GoTo Ms_BscOpenComm_Exit

        End If
        If rc <> 1 Then
            rc = BscClose(nCid)
            nCid = -1
            GoTo Ms_BscOpenComm_Exit
        End If
```

```
        'Connect communications line.
        rc = BscConnect(nCid)
        If rc <> 1 Then
            rc = BscClose(nCid)
            nCid = -1
            GoTo Ms_BscOpenComm_Exit
        End If

Ms_BscOpenComm_Exit:
        Ms_BscOpenComm = nCid

        TextBox1.Text = nCid
        TextBox2.Text = rc

    End Function

    Function Ms_BscCloseComm(ByRef nCid As Short) As Short
    Dim rc As Short
        'Cut the communications line.
        rc = BscDisConnect(nCid)
        'Close the port.
        rc = BscClose(nCid)
        rc = BscEnforcedClose(nCid) ' New
        Ms_BscCloseComm = rc
        TextBox1.Text = nCid
    TextBox2.Text = rc
    End Function

    ' Global Declarations

Dim aaa As Double
Dim bbb As Double

Dim temp_row_1 As Integer

Dim Average_Successful_Shaking_Policies_Sum As Double
Dim Average_Successful_Shaking_Policies_Final As Double

Dim Percent_of_Successful_Policies As Double
Dim Number_of_Successful_Policies As Integer

Dim MatLab As Object

Dim Axis_Allowed_Counter As Integer

Dim Initial_Relative_Axis_Speed_X As Integer
Dim Initial_Relative_Axis_Speed_Y As Integer
Dim Initial_Relative_Axis_Speed_Z As Integer

Dim Initial_Relative_Axis_Amplitude_X As Integer
Dim Initial_Relative_Axis_Amplitude_Y As Integer
Dim Initial_Relative_Axis_Amplitude_Z As Integer

Dim Relative_Axis_Speed_X As Integer
Dim Relative_Axis_Speed_Y As Integer
Dim Relative_Axis_Speed_Z As Integer

Dim Relative_Axis_Amplitude_X As Integer
Dim Relative_Axis_Amplitude_Y As Integer
Dim Relative_Axis_Amplitude_Z As Integer

Dim Last_Relative_Axis_Amplitude_X As Integer
Dim Last_Relative_Axis_Amplitude_Y As Integer
```

```
Dim Last_Relative_Axis_Amplitude_Z As Integer

Dim Average_Successful_Shaking_Policies_Index As Integer

Dim allow_sound_flag As Integer

Dim Chosen_Best_Policy As String

Dim Length_of_Best_Policy As Integer

Dim eliminate_x_axis_flag As Integer
Dim eliminate_y_axis_flag As Integer
Dim eliminate_z_axis_flag As Integer

Dim Cummulative_Weight_Reward As Double
Dim Events_Weight_Reward As Double
Dim Events_Value_vector_String As String
Dim Cummulative_Value_vector_String As String
Dim Times_Value_vector_String As String

Dim counter_1 As Integer

    ' Times_vector
Dim Times_vector(0, 50) As Double
    ' Cummulative_Value_vector
Dim Cummulative_Value_vector(0, 50) As Double
    ' Events_Value_vector
Dim Events_Value_vector(0, 50) As Double

Dim Average_Successful_Shaking_Policies(0, 150) As Double

Dim Time_Now_1 As DateTime

Dim Finish_Flag As Integer
Dim column%, Row%

Dim temp1 As Integer

Dim temp_data As Double

Dim rc As Long

Dim Shaking_Time_1 As Integer
Dim Learning_Performance_1 As Integer

Dim system_performance_measure_1 As Double
Dim number_of_policies_that_were_not_rewarded As Integer
Dim number_of_policies_that_were_rewarded As Integer

Dim temp_output_file_name_1 As String
Dim output_reward_1 As Integer

Dim Reward(18, 35 - 18) As Double

Dim maxQ_V1(0, 35 - 18) As Double
Dim Q_Table(18, 35 - 18) As Double ' 19 states, 36 actions
Dim Q_Table_Rewarded(18, 35 - 18) As Double ' 19 states, 36 actions
Dim Q_Table_Final(18, 35 - 18) As Double ' 19 states, 36 actions

Dim Eligibility(18, 35 - 18) As Double ' 19 states, 36 actions
Dim Eligibility_Rewarded(18, 35 - 18) As Double ' 19 states, 36 actions

Dim delta(500, 0) As Double
```

```
Dim delta_Rewarded(500, 0) As Double

Dim alpha As Double
Dim gamma As Double
Dim lambda As Double
Dim learning_trial As Integer

Dim state_Q As Integer
Dim next_state_Q As Integer
Dim action_Q As Integer
Dim Action_Time As Integer
Dim startTime, endTime As DateTime
Dim Number_of_Policies_Performed As Integer
Dim Performance_Measure_1(0, 4) As Integer

Dim temp111(3, 3) As Integer

Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
MyBase.Load

    temp_row_1 = 1

    Average_Successful_Shaking_Policies_Sum = 0
    Initial_Plot_Graph_1()
    Initial_Plot_Graph_2()

    TextBox135.Text = ""

    Percent_of_Successful_Policies = 0
    Number_of_Successful_Policies = 0

    Axis_Allowed_Counter = 0

    Initial_Relative_Axis_Speed_X = 1000
    Initial_Relative_Axis_Speed_Y = 1000
    Initial_Relative_Axis_Speed_Z = 1000

    Initial_Relative_Axis_Amplitude_X = 30
    Initial_Relative_Axis_Amplitude_Y = 30
    Initial_Relative_Axis_Amplitude_Z = 30

    Relative_Axis_Amplitude_X = Initial_Relative_Axis_Amplitude_X
    Relative_Axis_Speed_X = Initial_Relative_Axis_Speed_X

    Relative_Axis_Amplitude_Y = Initial_Relative_Axis_Amplitude_Y
    Relative_Axis_Speed_Y = Initial_Relative_Axis_Speed_Y

    Relative_Axis_Amplitude_Z = Initial_Relative_Axis_Amplitude_Z
    Relative_Axis_Speed_Z = Initial_Relative_Axis_Speed_Z

    allow_sound_flag = 0

    Length_of_Best_Policy = 0

    eliminate_x_axis_flag = 0
    eliminate_y_axis_flag = 0
    eliminate_z_axis_flag = 0

    ComboBox6.SelectedIndex = 0
    ComboBox2.SelectedIndex = 0

    counter_1 = 0
    Cummulative_Weight_Reward = 0
```

```
        Events_Weight_Reward = 0

        Events_Value_vector_String = ""

        Finish_Flag = 0
        temp1 = 0

        GroupBox19.Enabled = False
        TabControl1.SelectedTab = TabPage8
        Learning_Performance_1 = 0

        Shaking_Time_1 = 0

        TabPage3.Enabled = False
        TabPage4.Enabled = False

        Button48.Enabled = False
        Button51.Enabled = False
        Button50.Enabled = False
        Button47.Enabled = False

        TextBox131.Text = "No Data"

        Show_States()

    Dim nullObject As System.Object = 0
    Dim str As String = ""
    Dim nullObjStr As System.Object = str
        Cursor.Current = Cursors.WaitCursor
        AxWebBrowser3.Navigate("http://www.ie.bgu.ac.il/kartoun/visual_feedback/camera_1.htm", nullObject,
nullObjStr, nullObjStr, nullObjStr)

        Action_Counter_1 = 0
        TextBox91.Text = Action_Counter_1.ToString
        State_Action_Timer1_Rand_Counter_1 = 0
        State_Action_Timer1_Real_Counter_1 = 0
        State_Action_Timer1_Best_Counter_1 = 0
        TextBox116.Text = TrackBar1.Value.ToString

        alpha = Val(TextBox119.Text)
        gamma = Val(TextBox120.Text)
        lambda = Val(TextBox121.Text)

        learning_trial = 0

        state_Q = 0 'Starting from Center
        initQFunction()

        ComboBox7.SelectedIndex = 0

        ComboBox8.SelectedIndex = 2
        ComboBox9.SelectedIndex = 2
        ComboBox10.SelectedIndex = 2

        ComboBox11.SelectedIndex = 2
        ComboBox12.SelectedIndex = 2
        ComboBox13.SelectedIndex = 2

        ComboBox3.SelectedIndex = 2
        ComboBox4.SelectedIndex = 2
        ComboBox5.SelectedIndex = 2

        ComboBox14.SelectedIndex = 2
```

```
        ComboBox15.SelectedIndex = 2
        ComboBox16.SelectedIndex = 2

        TextBox136.Text = Initial_Relative_Axis_Speed_X.ToString
        TextBox137.Text = Initial_Relative_Axis_Speed_Y.ToString
        TextBox90.Text = Initial_Relative_Axis_Speed_Z.ToString

        TextBox140.Text = Initial_Relative_Axis_Amplitude_X.ToString
        TextBox139.Text = Initial_Relative_Axis_Amplitude_Y.ToString
        TextBox138.Text = Initial_Relative_Axis_Amplitude_Z.ToString

        system_performance_measure_1 = 0
        number_of_policies_that_were_not_rewarded = 0
        number_of_policies_that_were_rewarded = 0

        Number_of_Policies_Performed = -1

    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Trial_Number", "1")
        pRegKey_Events.SetValue("Stop_Robot_Flag", "0")

        Average_Successful_Shaking_Policies_Index = 0

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        TextBox1.Text = Ms_BscOpenComm(0)
        If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
            Label15.Text = "Connected"
        Else
            Label15.Text = "Disconnected"
        End If
        CheckBox1.Checked = False
        CheckBox2.Checked = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        BscSelectMode(nCid, 1)
        BscServoOff(nCid)
        TextBox2.Text = Ms_BscCloseComm(0)
        TextBox2.Text = BscEnforcedClose(0)
        Label10.Text = "Teach"
        Label13.Text = "Off"
        CheckBox2.Checked = False
        If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
            Label15.Text = "Connected"
        Else
            Label15.Text = "Disconnected"
        End If
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        CheckBox1.Checked = True
        BscSelectMode(nCid, 1)
        BscServoOff(nCid)
        TextBox2.Text = Ms_BscCloseComm(0)
        TextBox2.Text = BscEnforcedClose(0)
        Label10.Text = "Teach"
        Label13.Text = "Off"
        CheckBox2.Checked = False
        If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
            Label15.Text = "Connected"
```

```
        Else
            Label15.Text = "Disconnected"
        End If
        Close()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
        TextBox1.Text = ""
        TextBox2.Text = ""
        TextBox3.Text = ""
    End Sub

    Private Sub CmdDownLoad_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CmdDownLoad.Click
        TextBox3.Text = BscSelectJob(nCid, TextBox6.Text)
        TextBox3.Text = BscDeleteJob(nCid)
        TextBox3.Text = ""
        TextBox3.Text = BscDownLoad(nCid, TextBox6.Text)
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
        TextBox3.Text = BscSelectJob(nCid, TextBox6.Text)
        TextBox3.Text = BscDeleteJob(nCid)
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
        TextBox3.Text = BscSelectJob(nCid, TextBox6.Text)
        TextBox3.Text = BscDeleteJob(nCid)
        TextBox3.Text = ""
        TextBox3.Text = BscDownLoad(nCid, TextBox6.Text)
        TextBox3.Text = BscSelOneCycle(nCid)
        BscHoldOff(nCid)
        BscSetMasterJob(nCid)
        BscSelectMode(nCid, 2)
        BscServoOn(nCid)
        BscStartJob(nCid)
    End Sub

    Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox1.CheckedChanged
        If CheckBox1.Checked = True Then
            BscSelectMode(nCid, 1)
            Label10.Text = "Teach"
            Label13.Text = "Off"
        End If
        If CheckBox1.Checked = False Then
            BscSelectMode(nCid, 2)
            Label10.Text = "Play"
            BscHoldOff(nCid)
        End If
    End Sub

    Private Sub CheckBox2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox2.CheckedChanged
        If CheckBox1.Checked = False Then
            If CheckBox2.Checked = False Then
                BscServoOff(nCid)
                Label13.Text = "Off"
            End If
            If CheckBox2.Checked = True Then
                BscServoOn(nCid)
                Label13.Text = "On"
            End If
```

```
    End If
End Sub

Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
    TextBox3.Text = BscSelectJob(nCid, TextBox4.Text)
    TextBox3.Text = BscUpLoad(nCid, TextBox4.Text)
End Sub

Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button9.Click
    Disconnect_Robot()
End Sub

Public Function Disconnect_Robot()
    CheckBox1.Checked = True
    BscSelectMode(nCid, 1)
    BscServoOff(nCid)
    TextBox2.Text = Ms_BscCloseComm(0)
    '   TextBox2.Text = BscEnforcedClose(0)
    Label10.Text = "Teach"
    Label13.Text = "Off"
    CheckBox2.Checked = False
    If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
        Label15.Text = "Connected"
    Else
        Label15.Text = "Disconnected"
    End If
End Function

Public Function Run_Program(ByVal e As String) As Integer
    TextBox143.Text = ""
    TextBox3.Text = BscSelectJob(nCid, e)
    TextBox3.Text = BscDeleteJob(nCid)
    TextBox3.Text = ""
    TextBox3.Text = BscDownLoad(nCid, e)
    If ((e = "CLOSE.JBI") Or (e = "OPEN.JBI")) Then

        BscHoldOff(nCid)
        TextBox3.Text = BscSelLoopCycle(nCid)

        BscSetMasterJob(nCid)
        BscSelectMode(nCid, 2)
        BscServoOn(nCid)
        'If Finish_Flag = 0 Then
        BscStartJob(nCid)
        '   TextBox143.Text = ""
        'Else
        '   TextBox143.Text = BscStartJob(nCid)
        'End If
        BscHoldOn(nCid)
    Else
        If CheckBox4.Checked = False Then
            TextBox3.Text = BscSelOneCycle(nCid)
            BscSetMasterJob(nCid)
            BscSelectMode(nCid, 2)
            BscServoOn(nCid)
            'If Finish_Flag = 0 Then
            BscStartJob(nCid)
            '   TextBox143.Text = ""
            'Else
            '   TextBox143.Text = BscStartJob(nCid)
            'End If
            BscHoldOff(nCid)
        Else
```

```
            TextBox3.Text = BscSelLoopCycle(nCid)
            BscSetMasterJob(nCid)
            BscSelectMode(nCid, 2)
            BscServoOn(nCid)
            'If Finish_Flag = 0 Then
            BscStartJob(nCid)
            '   TextBox143.Text = ""
            'Else
            '   TextBox143.Text = BscStartJob(nCid)
            'End If
            BscHoldOff(nCid)
        End If
    End If
End Function

Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    '  BscHoldOff(nCid)
    If CheckBox6.Checked = True Then
        Run_Program("BASER.JBI")
    Else
        TextBox7.Text = "X1"
    Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("00.000")
        TextBox22.Text = strFormatedNumber
        TextBox23.Text = "000.000"
        TextBox24.Text = "000.000"
        TextBox25.Text = "0.00"
        TextBox26.Text = "0.00"
        TextBox27.Text = "0.00"
        TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
        Run_Program_with_Parameters("X1")
    End If
End Sub

Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button10.Click
    ' BscHoldOff(nCid)
    If CheckBox6.Checked = True Then
        Run_Program("BASEL.JBI")
    Else
        TextBox7.Text = "X2"
    Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("-00.000")
        TextBox22.Text = strFormatedNumber
        TextBox23.Text = "000.000"
        TextBox24.Text = "000.000"
        TextBox25.Text = "0.00"
        TextBox26.Text = "0.00"
        TextBox27.Text = "0.00"
        TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
        Run_Program_with_Parameters("X2")
    End If
End Sub

Private Sub Button12_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button12.Click
    ' BscHoldOff(nCid)
    If CheckBox6.Checked = True Then
        Run_Program("STATIONU.JBI")
    Else
        TextBox7.Text = "Y1"
    Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("00.000")
        TextBox22.Text = "000.000"
        TextBox23.Text = strFormatedNumber
        TextBox24.Text = "000.000"
        TextBox25.Text = "0.00"
        TextBox26.Text = "0.00"
```

```
        TextBox27.Text = "0.00"
        TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
        Run_Program_with_Parameters("Y1")
    End If
End Sub

Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button11.Click
    '  BscHoldOff(nCid)
    If CheckBox6.Checked = True Then
        Run_Program("STATIOND.JBI")
    Else
        TextBox7.Text = "Y2"
    Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("-00.000")
        TextBox22.Text = "000.000"
        TextBox23.Text = strFormatedNumber
        TextBox24.Text = "000.000"
        TextBox25.Text = "0.00"
        TextBox26.Text = "0.00"
        TextBox27.Text = "0.00"
        TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
        Run_Program_with_Parameters("Y2")
    End If
End Sub

Private Sub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button14.Click
    '  BscHoldOff(nCid)
    TextBox7.Text = "Z1"
Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("00.000")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = strFormatedNumber
    TextBox25.Text = "0.00"
    TextBox26.Text = "0.00"
    TextBox27.Text = "0.00"
    TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
    Run_Program_with_Parameters("Z1")
End Sub

Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button13.Click
    ' BscHoldOff(nCid)
    TextBox7.Text = "Z2"
Dim strFormatedNumber As String = CLng((Val(TextBox10.Text) * 10).ToString).ToString("-00.000")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = strFormatedNumber
    TextBox25.Text = "0.00"
    TextBox26.Text = "0.00"
    TextBox27.Text = "0.00"
    TextBox34.Text = 10 * Val(TextBox8.Text).ToString()
    Run_Program_with_Parameters("Z2")
End Sub

Private Sub Button15_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button15.Click
    ' BscHoldOff(nCid)
    TextBox7.Text = "R1"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("0.00")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = "000.000"
    TextBox25.Text = strFormatedNumber
    TextBox26.Text = "0.00"
    TextBox27.Text = "0.00"
    TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
```

```
    Run_Program_with_Parameters("R1")
End Sub

Private Sub Button16_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button16.Click
    'BscHoldOff(nCid)
    TextBox7.Text = "R2"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("-0.00")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = "000.000"
    TextBox25.Text = strFormatedNumber
    TextBox26.Text = "0.00"
    TextBox27.Text = "0.00"
    TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
    Run_Program_with_Parameters("R2")
End Sub

Private Sub Button18_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button18.Click
    ' BscHoldOff(nCid)
    TextBox7.Text = "P1"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("0.00")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = "000.000"
    TextBox25.Text = "0.00"
    TextBox26.Text = strFormatedNumber
    TextBox27.Text = "0.00"
    TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
    Run_Program_with_Parameters("P1")
End Sub

Private Sub Button17_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button17.Click
    '  BscHoldOff(nCid)
    TextBox7.Text = "P2"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("-0.00")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = "000.000"
    TextBox25.Text = "0.00"
    TextBox26.Text = strFormatedNumber
    TextBox27.Text = "0.00"
    TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
    Run_Program_with_Parameters("P2")
End Sub

Private Sub Button20_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button20.Click
    '  BscHoldOff(nCid)
    TextBox7.Text = "YA1"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("0.00")
    TextBox22.Text = "000.000"
    TextBox23.Text = "000.000"
    TextBox24.Text = "000.000"
    TextBox25.Text = "0.00"
    TextBox26.Text = "0.00"
    TextBox27.Text = strFormatedNumber
    TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
    Run_Program_with_Parameters("YA1")
End Sub

Private Sub Button19_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button19.Click
    ' BscHoldOff(nCid)
    TextBox7.Text = "YA2"
Dim strFormatedNumber As String = CLng((Val(TextBox11.Text) * 1).ToString).ToString("-0.00")
```

```
        TextBox22.Text = "000.000"
        TextBox23.Text = "000.000"
        TextBox24.Text = "000.000"
        TextBox25.Text = "0.00"
        TextBox26.Text = "0.00"
        TextBox27.Text = strFormatedNumber
        TextBox34.Text = (10) * Val(TextBox9.Text).ToString()
        Run_Program_with_Parameters("YA2")
    End Sub

    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button8.Click
        Button21.Enabled = True
        Button22.Enabled = True
        Run_Program("HC.JBI")
    End Sub

    Private Sub Button21_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button21.Click
        Button22.Enabled = False
        Run_Program("HR.JBI")
    End Sub

    Private Sub Button22_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button22.Click
        Button21.Enabled = False
        Run_Program("HL.JBI")
    End Sub

    Private Sub Button25_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button25.Click
        BscHoldOn(nCid)
        BscHoldOn(nCid)
        BscHoldOn(nCid)
        BscHoldOn(nCid)
        BscHoldOn(nCid)
    End Sub

    Private Sub Button26_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Dim m
        m = CreateObject("Matlab.Application")
        m.Execute("type1")
    End Sub

    Private Sub CheckBox4_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox4.CheckedChanged
        If CheckBox4.Checked = True Then CheckBox5.Checked = False
        If CheckBox4.Checked = False Then CheckBox5.Checked = True
    End Sub

    Private Sub CheckBox5_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox5.CheckedChanged
        If CheckBox5.Checked = True Then CheckBox4.Checked = False
        If CheckBox5.Checked = False Then CheckBox4.Checked = True
    End Sub
    Public Function Run_Program_with_Parameters(ByVal e As String) As Integer
        BscHoldOff(nCid)
    Dim Arm_Increment As Integer
    Dim Wrist_Increment As Integer
    Dim Arm_Speed As Integer
    Dim Wrist_Speed As Integer

        Arm_Increment = Val(TextBox10.Text)
        Arm_Speed = Val(TextBox8.Text)
        Wrist_Speed = Val(TextBox9.Text)
        Wrist_Increment = Val(TextBox11.Text)
```

```
        If CheckBox3.Checked = True Then TextBox36.Text = "ROBOT"
        If CheckBox6.Checked = True Then TextBox36.Text = "PULSE"

        If ((Arm_Increment <= 0) Or (Arm_Increment) > 50) Then
            MsgBox("Arm increment should be more than 0 and less than 50 cm.")
        End If
        If ((Arm_Speed <= 0) Or (Arm_Speed) > 25) Then
            MsgBox("Arm speed should be more than 0 and less than 25 cm / sec.")
        End If
        If ((Wrist_Speed <= 0) Or (Wrist_Speed) > 20) Then
            MsgBox("Wrist speed should be more than 0 and less than 20 cm / sec.")
        End If
        If ((Wrist_Increment <= 0) Or (Wrist_Increment) > 5) Then
            MsgBox("Wrist increment should be more than 0 and less than 5 cm.")
        End If

        If (((Arm_Increment > 0) And (Arm_Increment) <= 50) And ((Arm_Speed > 0) And (Arm_Speed) <= 25) And
((Wrist_Speed > 0) And (Wrist_Speed) <= 20) And ((Wrist_Increment > 0) And (Wrist_Increment) <= 5)) Then
            TextBox12.Text = TextBox13.Text & Chr(13) & Chr(10) & TextBox14.Text & TextBox7.Text & Chr(13) &
Chr(10) & TextBox15.Text & Chr(13) & Chr(10) & TextBox16.Text & Chr(13) & Chr(10) & TextBox17.Text &
Chr(13) & Chr(10) & TextBox18.Text & TextBox36.Text & Chr(13) & Chr(10) & TextBox19.Text & Chr(13) &
Chr(10) & TextBox20.Text & Chr(13) & Chr(10) & TextBox21.Text & TextBox22.Text & "," & TextBox23.Text & ","
& TextBox24.Text & "," & TextBox25.Text & "," & TextBox26.Text & "," & TextBox27.Text & Chr(13) & Chr(10) &
TextBox28.Text & Chr(13) & Chr(10) & TextBox29.Text & Chr(13) & Chr(10) & TextBox30.Text & Chr(13) &
Chr(10) & TextBox31.Text & Chr(13) & Chr(10) & TextBox32.Text & Chr(13) & Chr(10) & TextBox33.Text &
TextBox34.Text & Chr(13) & Chr(10) & TextBox35.Text
            File.Delete(e + ".JBI")
        Dim fs As New FileStream(e + ".JBI", FileMode.OpenOrCreate, FileAccess.Write)
        Dim s As New StreamWriter(fs)

            s.WriteLine(TextBox12.Text)

            s.Close()

            If CheckBox4.Checked = False Then
                TextBox3.Text = BscSelOneCycle(nCid)
            Else
                TextBox3.Text = BscSelLoopCycle(nCid)
            End If

            TextBox3.Text = BscSelectJob(nCid, e + ".JBI")
            TextBox3.Text = BscDeleteJob(nCid)
            TextBox3.Text = ""
            TextBox3.Text = BscDownLoad(nCid, e + ".JBI")

            ' BscHoldOff(nCid)
            BscSetMasterJob(nCid)
            BscSelectMode(nCid, 2)
            BscServoOn(nCid)
            BscStartJob(nCid)
        End If
    End Function

    Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MenuItem1.Click
        CheckBox1.Checked = True
        BscSelectMode(nCid, 1)
        BscServoOff(nCid)
        TextBox2.Text = Ms_BscCloseComm(0)
        TextBox2.Text = BscEnforcedClose(0)
        Label10.Text = "Teach"
        Label13.Text = "Off"
        CheckBox2.Checked = False
```

```
        If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
            Label15.Text = "Connected"
        Else
            Label15.Text = "Disconnected"
        End If
    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Activate_Scale_Flag", "0")
        pRegKey_Events.SetValue("Events_Value", "0")
        pRegKey_Events.SetValue("Cummulative_Value", "0")
        Close()
    End Sub

    Private Sub Button23_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button23.Click
        'BscHoldOff(nCid)
        Run_Program("OPEN.JBI")
        'BscHoldOff(nCid)
    End Sub

    Private Sub Button24_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button24.Click
        'BscHoldOff(nCid)
        Run_Program("CLOSE.JBI")
        'BscHoldOff(nCid)
    End Sub

    Private Sub CheckBox3_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox3.CheckedChanged
        If CheckBox3.Checked = True Then CheckBox6.Checked = False
        If CheckBox3.Checked = False Then CheckBox6.Checked = True
        If CheckBox3.Checked = True Then TextBox36.Text = "ROBOT"
        If CheckBox6.Checked = True Then TextBox36.Text = "PULSE"
    End Sub

    Private Sub CheckBox6_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox6.CheckedChanged
        If CheckBox6.Checked = True Then CheckBox3.Checked = False
        If CheckBox6.Checked = False Then CheckBox3.Checked = True
        If CheckBox3.Checked = True Then TextBox36.Text = "ROBOT"
        If CheckBox6.Checked = True Then TextBox36.Text = "PULSE"
    End Sub

    Public Function Shake_1()
    Dim Amplitude As Integer
    Dim Speed As Integer
    Dim Times As Integer

        Amplitude = Val(TextBox38.Text)
        Speed = Val(TextBox39.Text)
        Times = Val(TextBox37.Text)

        If (((Amplitude > 0) And (Amplitude) <= 20) And ((Speed > 0) And (Speed) <= 1000) And ((Times > 0) And
(Times) <= 10)) Then

            If ComboBox1.SelectedItem = "X" Then
            Dim strFormatedNumber As String = CLng((Val(TextBox38.Text) * 10).ToString).ToString("00.000")

                TextBox51.Text = strFormatedNumber
                TextBox52.Text = "000.000"
                TextBox53.Text = "000.000"
                TextBox54.Text = "0.00"
                TextBox55.Text = "0.00"
                TextBox56.Text = "0.00"
```

```
        TextBox62.Text = "-" & strFormatedNumber
        TextBox60.Text = "000.000"
        TextBox61.Text = "000.000"
        TextBox57.Text = "0.00"
        TextBox58.Text = "0.00"
        TextBox59.Text = "0.00"

        TextBox70.Text = (1 * Val(TextBox39.Text)).ToString()
        TextBox72.Text = (1 * Val(TextBox39.Text)).ToString()

    ElseIf ComboBox1.SelectedItem = "Y" Then
    Dim strFormatedNumber_1 As String = CLng((Val(TextBox38.Text) * 10).ToString).ToString("00.000")

        TextBox51.Text = "000.000"
        TextBox52.Text = strFormatedNumber_1
        TextBox53.Text = "000.000"
        TextBox54.Text = "0.00"
        TextBox55.Text = "0.00"
        TextBox56.Text = "0.00"

        TextBox62.Text = "000.000"
        TextBox60.Text = "000.000"
        TextBox61.Text = "-" & strFormatedNumber_1
        TextBox57.Text = "0.00"
        TextBox58.Text = "0.00"
        TextBox59.Text = "0.00"

        TextBox70.Text = (1 * Val(TextBox39.Text)).ToString()
        TextBox72.Text = (1 * Val(TextBox39.Text)).ToString()

    ElseIf ComboBox1.SelectedItem = "Z" Then
    Dim strFormatedNumber_2 As String = CLng((Val(TextBox38.Text) * 10).ToString).ToString("00.000")

        TextBox51.Text = "000.000"
        TextBox52.Text = "000.000"
        TextBox53.Text = strFormatedNumber_2
        TextBox54.Text = "0.00"
        TextBox55.Text = "0.00"
        TextBox56.Text = "0.00"

        TextBox62.Text = "000.000"
        TextBox60.Text = "-" & strFormatedNumber_2
        TextBox61.Text = "000.000"
        TextBox57.Text = "0.00"
        TextBox58.Text = "0.00"
        TextBox59.Text = "0.00"

        TextBox70.Text = (1 * Val(TextBox39.Text)).ToString()
        TextBox72.Text = (1 * Val(TextBox39.Text)).ToString()
    Else
        MsgBox("Please choose an axis.")
    End If

    TextBox74.Text = TextBox40.Text & Chr(13) & Chr(10) & TextBox41.Text & TextBox42.Text & Chr(13) &
Chr(10) & TextBox43.Text & Chr(13) & Chr(10) & TextBox44.Text & Chr(13) & Chr(10) & TextBox45.Text &
Chr(13) & Chr(10) & TextBox46.Text & Chr(13) & Chr(10) & TextBox47.Text & Chr(13) & Chr(10) &
TextBox48.Text & Chr(13) & Chr(10) & TextBox50.Text & TextBox51.Text & "," & TextBox52.Text & "," &
TextBox53.Text & "," & TextBox54.Text & "," & TextBox55.Text & "," & TextBox56.Text & Chr(13) & Chr(10) &
TextBox63.Text & TextBox62.Text & "," & TextBox61.Text & "," & TextBox60.Text & "," & TextBox59.Text & "," &
TextBox58.Text & "," & TextBox57.Text & Chr(13) & Chr(10) & TextBox64.Text & Chr(13) & Chr(10) &
TextBox65.Text & Chr(13) & Chr(10) & TextBox66.Text & Chr(13) & Chr(10) & TextBox67.Text & Chr(13) &
```

```
Chr(10) & TextBox68.Text & Chr(13) & Chr(10) & TextBox69.Text & TextBox70.Text & Chr(13) & Chr(10) &
TextBox71.Text & TextBox72.Text & Chr(13) & Chr(10) & TextBox73.Text
            File.Delete(TextBox42.Text + ".JBI")
        Dim fs As New FileStream(TextBox42.Text + ".JBI", FileMode.OpenOrCreate, FileAccess.Write)
        Dim s As New StreamWriter(fs)
            s.WriteLine(TextBox74.Text)
            s.Close()
        End If
    End Function

    Public Function Shake_2()
        TextBox3.Text = BscSelectJob(nCid, TextBox42.Text + ".JBI")
        TextBox3.Text = BscDeleteJob(nCid)
        TextBox3.Text = ""
        TextBox3.Text = BscDownLoad(nCid, TextBox42.Text + ".JBI")
    End Function

    Public Function Shake_3()
        TextBox3.Text = BscSelOneCycle(nCid)
        BscHoldOff(nCid)
        BscSetMasterJob(nCid)
        BscSelectMode(nCid, 2)
        BscServoOn(nCid)
        BscStartJob(nCid)
    End Function

    Private Sub Button26_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button26.Click
    Dim counter_1 As Integer
        counter_1 = 1
    Dim Amplitude As Integer
    Dim Speed As Integer
    Dim Times As Integer

        Amplitude = Val(TextBox38.Text)
        Speed = Val(TextBox39.Text)
        Times = Val(TextBox37.Text)

        If ((Amplitude <= 0) Or (Amplitude) > 20) Then
            MsgBox("Amplitude should be more than 0 and less than 20 cm.")
        End If

        If ((Speed <= 0) Or (Speed) > 1000) Then
            MsgBox("Speed should be more than 0 and less than 1000 cm / sec.")
        End If

        If ((Times <= 0) Or (Times) > 10) Then
            MsgBox("Number of times should be more than 0 and less than 10.")
        End If

        If (((Amplitude > 0) And (Amplitude) <= 20) And ((Speed > 0) And (Speed) <= 1000) And ((Times > 0) And
(Times) <= 10)) And ((ComboBox1.SelectedItem = "X" Or ComboBox1.SelectedItem = "Y" Or
ComboBox1.SelectedItem = "Z")) Then
            Shake_1()
            Shake_2()
            For counter_1 = 1 To Val(TextBox37.Text)
                Shake_3()
            Next
        End If

    End Sub

    Public Function Show_States()
```

```
        xConn = New sqlConn
        xConn.connectMe("SELECT State FROM States Where State='" & "Center" & "'")
        ListBox1.Items.Clear()
        For iCounter = 0 To xConn.getData("State").Count - 1
            ListBox1.Items.Add(xConn.dataReturned.Item(iCounter))
        Next
        xConn.OLEConn.Close()
    End Function

    Public Function Show_States_1()
        xConn = New sqlConn
        xConn.connectMe("SELECT State FROM States_1 Where State='" & "Home" & "'")
        ListBox3.Items.Clear()
        ListBox10.Items.Clear()
        For iCounter = 0 To xConn.getData("State").Count - 1
            ListBox3.Items.Add(xConn.dataReturned.Item(iCounter))
            ListBox10.Items.Add(xConn.dataReturned.Item(iCounter))
        Next
        xConn.OLEConn.Close()
    End Function

    Private Sub ListBox1_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ListBox1.DoubleClick

        Current_State = ListBox1.SelectedItem()

        Try
            xConn = New sqlConn
            xConn.connectMe("SELECT * FROM States WHERE State ='" & ListBox1.SelectedItem & "';")
            ListBox2.Items.Clear()
            Try

                xConn.OLEComm.Connection = xConn.OLEConn
            Dim d As OleDb.OleDbDataReader = xConn.OLEComm.ExecuteReader()
                d.Read()
                For i As Integer = 1 To (36 - 18)

                    If d.IsDBNull(i + 1) = False Then

                        ListBox2.Items.Add(d(("Action" + i.ToString()).ToString))
                    End If
                Next
                Try
                    xConn.OLEConn.Close()
                Catch err As System.Exception
                    MsgBox(err.Message)
                End Try
            Catch err As System.Exception
                MsgBox(err.Message)
            End Try
        Catch err As System.Exception
            MsgBox(err.Message)
        End Try
    End Sub

    Private Sub ListBox2_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ListBox2.DoubleClick

        TextBox107.Text = "///NPOS 0,0,0," + (Action_Counter_1 + 1).ToString() + ",0,0"

        Action = ListBox2.SelectedItem()

        Try
```

```
        xConn = New sqlConn
        xConn.connectMe("SELECT * FROM Action_State WHERE Action ='" & ListBox2.SelectedItem & "';")

        If (ListBox2.SelectedItem() <> "Nothing") Then
            ListBox1.Items.Clear()
            Try
                xConn.OLEComm.Connection = xConn.OLEConn
            Dim d As OleDb.OleDbDataReader = xConn.OLEComm.ExecuteReader()
                d.Read()
                ListBox1.Items.Add(d(("State").ToString))
                Try
                    xConn.OLEConn.Close()
                Catch err As System.Exception
                    MsgBox(err.Message)
                End Try
            Catch err As System.Exception
                MsgBox(err.Message)
            End Try
        End If
    Catch err As System.Exception
        MsgBox(err.Message)

    End Try
    xConn.OLEConn.Close()

    'Center
    'Center - Vel1
    If Current_State = "Center" And Action = "Move_X_Plus_1_Vel1" Then
        Reset_Step()
        TextBox101.Text = (1 * Val(TextBox81.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "Center" And Action = "Move_X_Plus_2_Vel1" Then
        Reset_Step()
        TextBox101.Text = (2 * Val(TextBox81.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "Center" And Action = "Move_X_Plus_3_Vel1" Then
        Reset_Step()
        TextBox101.Text = (3 * Val(TextBox81.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "Center" And Action = "Move_X_Minus_1_Vel1" Then
        Reset_Step()
        TextBox101.Text = (1 * Val(TextBox159.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "Center" And Action = "Move_X_Minus_2_Vel1" Then
        Reset_Step()
        TextBox101.Text = (2 * Val(TextBox159.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "Center" And Action = "Move_X_Minus_3_Vel1" Then
        Reset_Step()
        TextBox101.Text = (3 * Val(TextBox159.Text)).ToString("000.000")
        TextBox82.Text = TextBox153.Text
    End If
```

```
'X
''' X_Plus - Vel1

If Current_State = "X_Plus_1" And Action = "Move_X_Minus_1_Vel1" Then
   Reset_Step()
   TextBox101.Text = (2 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Plus_2" And Action = "Move_X_Minus_2_Vel1" Then
   Reset_Step()
   TextBox101.Text = (4 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Plus_3" And Action = "Move_X_Minus_3_Vel1" Then
   Reset_Step()
   TextBox101.Text = (6 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Plus_1" And Action = "Move_Center_Vel1" Then
   Reset_Step()
   TextBox101.Text = (1 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Plus_2" And Action = "Move_Center_Vel1" Then
   Reset_Step()
   TextBox101.Text = (2 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Plus_3" And Action = "Move_Center_Vel1" Then
   Reset_Step()
   TextBox101.Text = (3 * Val(TextBox159.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

''' X_Minus - Vel1

If Current_State = "X_Minus_1" And Action = "Move_X_Plus_1_Vel1" Then
   Reset_Step()
   TextBox101.Text = (2 * Val(TextBox81.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Minus_2" And Action = "Move_X_Plus_2_Vel1" Then
   Reset_Step()
   TextBox101.Text = (4 * Val(TextBox81.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Minus_3" And Action = "Move_X_Plus_3_Vel1" Then
   Reset_Step()
   TextBox101.Text = (6 * Val(TextBox81.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
End If

If Current_State = "X_Minus_1" And Action = "Move_Center_Vel1" Then
   Reset_Step()
   TextBox101.Text = (1 * Val(TextBox81.Text)).ToString("000.000")
   TextBox82.Text = TextBox153.Text
```

```
    End If

    If Current_State = "X_Minus_2" And Action = "Move_Center_Vel1" Then
      Reset_Step()
      TextBox101.Text = (2 * Val(TextBox81.Text)).ToString("000.000")
      TextBox82.Text = TextBox153.Text
    End If

    If Current_State = "X_Minus_3" And Action = "Move_Center_Vel1" Then
      Reset_Step()
      TextBox101.Text = (3 * Val(TextBox81.Text)).ToString("000.000")
      TextBox82.Text = TextBox153.Text
    End If
    '  End If

    If Current_State = "Center" And Action = "Move_Y_Plus_1_Vel1" Then
      Reset_Step()
      TextBox100.Text = (1 * Val(TextBox156.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Center" And Action = "Move_Y_Plus_2_Vel1" Then
      Reset_Step()
      TextBox100.Text = (2 * Val(TextBox156.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Center" And Action = "Move_Y_Plus_3_Vel1" Then
      Reset_Step()
      TextBox100.Text = (3 * Val(TextBox156.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Center" And Action = "Move_Y_Minus_1_Vel1" Then
      Reset_Step()
      TextBox100.Text = (1 * Val(TextBox160.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Center" And Action = "Move_Y_Minus_2_Vel1" Then
      Reset_Step()
      TextBox100.Text = (2 * Val(TextBox160.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Center" And Action = "Move_Y_Minus_3_Vel1" Then
      Reset_Step()
      TextBox100.Text = (3 * Val(TextBox160.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    'Y
    ''' Y_Plus - Vel1

    If Current_State = "Y_Plus_1" And Action = "Move_Y_Minus_1_Vel1" Then
      Reset_Step()
      TextBox100.Text = (2 * Val(TextBox160.Text)).ToString("000.000")
      TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Plus_2" And Action = "Move_Y_Minus_2_Vel1" Then
      Reset_Step()
      TextBox100.Text = (4 * Val(TextBox160.Text)).ToString("000.000")
```

```
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Plus_3" And Action = "Move_Y_Minus_3_Vel1" Then
        Reset_Step()
        TextBox100.Text = (6 * Val(TextBox160.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Plus_1" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (1 * Val(TextBox160.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Plus_2" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (2 * Val(TextBox160.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Plus_3" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (3 * Val(TextBox160.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    ''' Y_Minus - Vel1

    If Current_State = "Y_Minus_1" And Action = "Move_Y_Plus_1_Vel1" Then
        Reset_Step()
        TextBox100.Text = (2 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Minus_2" And Action = "Move_Y_Plus_2_Vel1" Then
        Reset_Step()
        TextBox100.Text = (4 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Minus_3" And Action = "Move_Y_Plus_3_Vel1" Then
        Reset_Step()
        TextBox100.Text = (6 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Minus_1" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (1 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Minus_2" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (2 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
    End If

    If Current_State = "Y_Minus_3" And Action = "Move_Center_Vel1" Then
        Reset_Step()
        TextBox100.Text = (3 * Val(TextBox156.Text)).ToString("000.000")
        TextBox82.Text = TextBox154.Text
```

```
End If

If Current_State = "Center" And Action = "Move_Z_Plus_1_Vel1" Then
    Reset_Step()
    TextBox99.Text = (1 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Center" And Action = "Move_Z_Plus_2_Vel1" Then ""
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Center" And Action = "Move_Z_Plus_3_Vel1" Then
    Reset_Step()
    TextBox99.Text = (3 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Center" And Action = "Move_Z_Minus_1_Vel1" Then
    Reset_Step()
    TextBox99.Text = (1 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Center" And Action = "Move_Z_Minus_2_Vel1" Then
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Center" And Action = "Move_Z_Minus_3_Vel1" Then
    Reset_Step()
    TextBox99.Text = (3 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

''' Z_Plus - Vel1

If Current_State = "Z_Plus_1" And Action = "Move_Z_Minus_1_Vel1" Then
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Plus_2" And Action = "Move_Z_Minus_2_Vel1" Then
    Reset_Step()
    TextBox99.Text = (4 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Plus_3" And Action = "Move_Z_Minus_3_Vel1" Then
    Reset_Step()
    TextBox99.Text = (6 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Plus_1" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (1 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If
```

```
If Current_State = "Z_Plus_2" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Plus_3" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (3 * Val(TextBox161.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

''' Z_Minus - Vel1

If Current_State = "Z_Minus_1" And Action = "Move_Z_Plus_1_Vel1" Then
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Minus_2" And Action = "Move_Z_Plus_2_Vel1" Then
    Reset_Step()
    TextBox99.Text = (4 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Minus_3" And Action = "Move_Z_Plus_3_Vel1" Then
    Reset_Step()
    TextBox99.Text = (6 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Minus_1" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (1 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Minus_2" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (2 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If Current_State = "Z_Minus_3" And Action = "Move_Center_Vel1" Then
    Reset_Step()
    TextBox99.Text = (3 * Val(TextBox157.Text)).ToString("000.000")
    TextBox82.Text = TextBox155.Text
End If

If (ListBox2.SelectedItem() = "Nothing") Then
    TextBox101.Text = "000.000"
    TextBox100.Text = "000.000"
    TextBox99.Text = "000.000"
    TextBox98.Text = "00.00"
    TextBox97.Text = "00.00"
    TextBox96.Text = "00.00"
Else

    Action_Counter_1 = Action_Counter_1 + 1
    TextBox91.Text = Action_Counter_1.ToString
```

```
            If Action_Counter_1 < 10 Then
                TextBox102.Text = "P000" + Action_Counter_1.ToString() + "="
                TextBox93.Text = TextBox93.Text + "IMOV P00" + Action_Counter_1.ToString() + " V=" + TextBox82.Text
& Chr(13) & Chr(10)
            End If

            If ((Action_Counter_1 >= 10) And (Action_Counter_1 < 100)) Then
                TextBox102.Text = "P00" + Action_Counter_1.ToString() + "="
                TextBox93.Text = TextBox93.Text + "IMOV P0" + Action_Counter_1.ToString() + " V=" + TextBox82.Text
& Chr(13) & Chr(10)
            End If

            If ((Action_Counter_1 >= 100) And (Action_Counter_1 < 1000)) Then
                TextBox102.Text = "P0" + Action_Counter_1.ToString() + "="
                TextBox93.Text = TextBox93.Text + "IMOV P" + Action_Counter_1.ToString() + " V=" + TextBox82.Text &
Chr(13) & Chr(10)
            End If

            TextBox92.Text = TextBox111.Text & Chr(13) & Chr(10) & TextBox110.Text & TextBox109.Text & Chr(13)
& Chr(10) & TextBox108.Text & Chr(13) & Chr(10) & TextBox107.Text & Chr(13) & Chr(10) & TextBox106.Text &
Chr(13) & Chr(10) & TextBox105.Text & Chr(13) & Chr(10) & TextBox104.Text & Chr(13) & Chr(10) &
TextBox103.Text & Chr(13) & Chr(10)
            TextBox78.Text = TextBox78.Text & TextBox102.Text & TextBox101.Text & "," & TextBox100.Text & "," &
TextBox99.Text & "," & TextBox98.Text & "," & TextBox97.Text & "," & TextBox97.Text & Chr(13) & Chr(10)
            TextBox94.Text = TextBox88.Text & Chr(13) & Chr(10) & TextBox87.Text & Chr(13) & Chr(10) &
TextBox86.Text & Chr(13) & Chr(10) & TextBox85.Text & Chr(13) & Chr(10) & TextBox84.Text & Chr(13) &
Chr(10)

            Next_State = ListBox1.Items.Item(0)
            Current_State = Next_State
        End If
    End Sub
    Function Reset_Step()
        TextBox101.Text = "000.000"
        TextBox100.Text = "000.000"
        TextBox99.Text = "000.000"
        TextBox98.Text = "00.00"
        TextBox97.Text = "00.00"
        TextBox96.Text = "00.00"
    End Function

    Private Sub Button28_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        BscHoldOn(nCid)
    End Sub

    Private Sub Button28_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
        BscHoldOff(nCid)
    End Sub

    Private Sub Button28_Click_2(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button28.Click

        File.Delete(TextBox109.Text + ".JBI")
    Dim fs As New FileStream(TextBox109.Text + ".JBI", FileMode.OpenOrCreate, FileAccess.Write)
    Dim s As New StreamWriter(fs)

        s.WriteLine(TextBox92.Text + TextBox78.Text + TextBox94.Text + TextBox93.Text + TextBox79.Text)

        s.Close()
        Finish_Flag = 0
        ''''  Run_Program("GRASPH1" + ".JBI") ' Home
        '   System.Threading.Thread.Sleep(2000)
        Finish_Flag = 1
```

```
        Run_Program(TextBox109.Text + ".JBI")
        Finish_Flag = 0
    End Sub

    Private Sub Button29_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button29.Click
        Disconnect_Robot()
    End Sub

    Private Sub Button30_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        If ComboBox2.SelectedIndex = 0 Then
            Run_Program("GRASPH1" + ".JBI")
        Else
            Run_Program("GRASPH2" + ".JBI")
        End If
    End Sub

    Private Sub Button31_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button31.Click
        If ComboBox2.SelectedIndex = 0 Then
            Run_Program("GRASP1" + ".JBI")
        Else
            Run_Program("GRASP2" + ".JBI")
        End If
    End Sub

    Private Sub Button32_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button32.Click
        Reset_Policy_1()
    End Sub

    Function Reset_Policy_1()
        State_Action_Rand_Timer1.Enabled = False
        ListBox2.Items.Clear()
        Show_States()
    Dim nullObject As System.Object = 0
    Dim str As String = ""
    Dim nullObjStr As System.Object = str

        Action_Counter_1 = 0
        TextBox91.Text = Action_Counter_1.ToString
        State_Action_Timer1_Rand_Counter_1 = 1

        TextBox92.Text = ""
        TextBox78.Text = ""
        TextBox94.Text = ""
        TextBox93.Text = ""
    End Function

    Private Sub State_Action_Timer1_Rand_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles State_Action_Rand_Timer1.Tick
        On Error GoTo ErrorHandler

    Dim ListBox2_Lengh As Integer

        ListBox1.SetSelected(0, True)
        ListBox1_DoubleClick(sender, e)

        ListBox2_Lengh = ListBox2.Items.Count - 1

    Dim R1 As Integer
        '  R1 = Randomizer(0, 1)

    Dim R2 As Double
        Randomize(DateTime.Now.Second())
        R2 = Rnd(DateTime.Now.Millisecond())
```

```
        'MsgBox(R2)

        ' R2 = Randomizer(0, 1)



        If CheckBox8.Checked = False Then
            If R2 > 0.95 Then R2 = 1
        End If

        R1 = Abs(CInt(ListBox2_Lengh * R2))

        TextBox95.Text = R1.ToString
        TextBox115.Text = ListBox2_Lengh
        ListBox2.SetSelected(Abs(R1), True)
        ListBox2_DoubleClick(sender, e)

        State_Action_Timer1_Rand_Counter_1 = Val(TextBox91.Text)

        If State_Action_Timer1_Rand_Counter_1 >= TrackBar1.Value Then
            eliminate_x_axis_flag = 0
            eliminate_y_axis_flag = 0
            eliminate_z_axis_flag = 0
            State_Action_Rand_Timer1.Enabled = False
        Else

            State_Action_Timer1_Rand_Counter_1 = State_Action_Timer1_Rand_Counter_1 + 1
        End If
ErrorHandler:
    End Sub

    Private Sub State_Action_Timer1_Real_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles State_Action_Real_Timer1.Tick

        ''' Create the last best policy with human intervention
        ListBox1.SetSelected(0, True)
        ListBox1_DoubleClick(sender, e)

        ListBox2.SetSelected(ListBox4.Items.Item(State_Action_Timer1_Real_Counter_1), True)
        ListBox2_DoubleClick(sender, e)

        If Val(TextBox133.Text) > 0 Then
        Dim string_of_results As String
        Dim fs1 As New FileStream(TextBox148.Text & temp_output_file_name_1 & ".csv", FileMode.Append,
FileAccess.Write)
        Dim s1 As New StreamWriter(fs1)

            string_of_results = ListBox3.Items.Item(State_Action_Timer1_Real_Counter_1) & Chr(44) &
ListBox4.Items.Item(State_Action_Timer1_Real_Counter_1) & Chr(44) & TextBox132.Text & Chr(44) &
TextBox142.Text & Chr(44) & TextBox153.Text & Chr(44) & TextBox154.Text & Chr(44) & TextBox155.Text &
Chr(44) & TextBox81.Text & Chr(44) & TextBox156.Text & Chr(44) & TextBox157.Text & Chr(44)

            s1.WriteLine(string_of_results)
            s1.Close()
        End If

        State_Action_Timer1_Real_Counter_1 = State_Action_Timer1_Real_Counter_1 + 1
        ProgressBar2.Value = State_Action_Timer1_Real_Counter_1

        If State_Action_Timer1_Real_Counter_1 >= Val(TextBox134.Text) Then
            If Val(TextBox133.Text) > 0 Then
            Dim fs2 As New FileStream(TextBox148.Text & temp_output_file_name_1 & ".csv", FileMode.Append,
FileAccess.Write)
```

```
Dim s2 As New StreamWriter(fs2)

    's2.WriteLine(TextBox127.Text)

    '   s2.Write(Round(Q_Table_Final(0, 0), 2))
Dim Comma_Deliminated_Final_Q_Table As String


    Comma_Deliminated_Final_Q_Table = TextBox127.Text.Replace("  ", ",")
    Comma_Deliminated_Final_Q_Table = TextBox127.Text.Replace(", , ", ",")

   'Write2File(TextBox114.Text + ", " + Cummulative_Value + ", " + Events_Value, TextBox148.Text +
(Val(Trial_Number)).ToString + "_Trial_" + "Scale_Output.csv")
    s2.WriteLine(Comma_Deliminated_Final_Q_Table)

    'For i As Integer = 0 To 18
    's2.WriteLine(Q_Table(i, 0) + ", " + Q_Table(i, 1) + ", " + Q_Table(i, 2) + ", " + Q_Table(i, 3) + ", " +
Q_Table(i, 4) + ", " + Q_Table(i, 5) + ", " + Q_Table(i, 6) + ", " + Q_Table(i, 7) + ", " + Q_Table(i, 8) + ", " +
Q_Table(i, 9) + ", " + Q_Table(i, 10) + ", " + Q_Table(i, 11) + ", " + Q_Table(i, 12) + ", " + Q_Table(i, 13) + ", " +
Q_Table(i, 14) + ", " + Q_Table(i, 15) + ", " + Q_Table(i, 16) + ", " + Q_Table(i, 17))
    'Next

    '    For i As Integer = 0 To 18
    '   s2.WriteLine(Q_Table_Final(i, 0) + Q_Table_Final(i, 1))
    '   Next

    'Q_Table(0, 0) = 10.37 : Q_Table(0, 1) = 31.11 : Q_Table(0, 2) = 18.22 : Q_Table(0, 3) = 16.6 : Q_Table(0, 4)
= 37.78 : Q_Table(0, 5) = 354.48 : Q_Table(0, 6) = 11.1 : Q_Table(0, 7) = 4.23 : Q_Table(0, 8) = 12.71 : Q_Table(0, 9)
= 14.94 : Q_Table(0, 10) = 7.6 : Q_Table(0, 11) = 8.01 : Q_Table(0, 12) = 15.31 : Q_Table(0, 13) = 8.38 : Q_Table(0,
14) = 10.82 : Q_Table(0, 15) = 5.62 : Q_Table(0, 16) = 0.07 : Q_Table(0, 17) = 0.23
    'Q_Table(1, 0) = 0.24 : Q_Table(1, 1) = 0.13 : Q_Table(1, 2) = 1.24 : Q_Table(1, 3) = 0.28 : Q_Table(1, 4) =
0.02 : Q_Table(1, 5) = 25.79 : Q_Table(1, 6) = 0.14 : Q_Table(1, 7) = 0.2 : Q_Table(1, 8) = 0.2 : Q_Table(1, 9) = 0.19 :
Q_Table(1, 10) = 0.22 : Q_Table(1, 11) = 0.05 : Q_Table(1, 12) = 0.09 : Q_Table(1, 13) = 0.13 : Q_Table(1, 14) = 0.01 :
Q_Table(1, 15) = 0.08 : Q_Table(1, 16) = 0.08 : Q_Table(1, 17) = 0.17
    'Q_Table(2, 0) = 0.19 : Q_Table(2, 1) = 0.21 : Q_Table(2, 2) = 0.16 : Q_Table(2, 3) = 14.71 : Q_Table(2, 4) =
0.09 : Q_Table(2, 5) = 71.12 : Q_Table(2, 6) = 0.26 : Q_Table(2, 7) = 0.15 : Q_Table(2, 8) = 0.17 : Q_Table(2, 9) = 0.29
: Q_Table(2, 10) = 0.15 : Q_Table(2, 11) = 0.18 : Q_Table(2, 12) = 0.26 : Q_Table(2, 13) = 0.24 : Q_Table(2, 14) = 0.14
: Q_Table(2, 15) = 0.22 : Q_Table(2, 16) = 0.06 : Q_Table(2, 17) = 0.06
    'Q_Table(3, 0) = 0.22 : Q_Table(3, 1) = 0.27 : Q_Table(3, 2) = 0.04 : Q_Table(3, 3) = 0.2 : Q_Table(3, 4) =
45.58 : Q_Table(3, 5) = 26.32 : Q_Table(3, 6) = 0.04 : Q_Table(3, 7) = 0.14 : Q_Table(3, 8) = 0.29 : Q_Table(3, 9) =
0.14 : Q_Table(3, 10) = 0.21 : Q_Table(3, 11) = 0.17 : Q_Table(3, 12) = 0.28 : Q_Table(3, 13) = 0.18 : Q_Table(3, 14) =
0.01 : Q_Table(3, 15) = 0.22 : Q_Table(3, 16) = 0.28 : Q_Table(3, 17) = 0.01
    'Q_Table(4, 0) = 7.73 : Q_Table(4, 1) = 0.06 : Q_Table(4, 2) = 0.16 : Q_Table(4, 3) = 0.24 : Q_Table(4, 4) =
0.28 : Q_Table(4, 5) = 22.92 : Q_Table(4, 6) = 0.05 : Q_Table(4, 7) = 0.18 : Q_Table(4, 8) = 0.15 : Q_Table(4, 9) = 0.02
: Q_Table(4, 10) = 0.03 : Q_Table(4, 11) = 0.22 : Q_Table(4, 12) = 0.26 : Q_Table(4, 13) = 0.07 : Q_Table(4, 14) = 0.12
: Q_Table(4, 15) = 0.06 : Q_Table(4, 16) = 0.24 : Q_Table(4, 17) = 0.21
    'Q_Table(5, 0) = 0.21 : Q_Table(5, 1) = 24 : Q_Table(5, 2) = 0.23 : Q_Table(5, 3) = 0.15 : Q_Table(5, 4) =
0.19 : Q_Table(5, 5) = 62.43 : Q_Table(5, 6) = 0.02 : Q_Table(5, 7) = 0.18 : Q_Table(5, 8) = 0.17 : Q_Table(5, 9) = 0.27
: Q_Table(5, 10) = 0.21 : Q_Table(5, 11) = 0.22 : Q_Table(5, 12) = 0.18 : Q_Table(5, 13) = 0.02 : Q_Table(5, 14) = 0.19
: Q_Table(5, 15) = 0.06 : Q_Table(5, 16) = 0.05 : Q_Table(5, 17) = 0.25
    'Q_Table(6, 0) = 0.18 : Q_Table(6, 1) = 0 : Q_Table(6, 2) = 35.45 : Q_Table(6, 3) = 0.19 : Q_Table(6, 4) =
0.27 : Q_Table(6, 5) = 353.35 : Q_Table(6, 6) = 0.23 : Q_Table(6, 7) = 0.23 : Q_Table(6, 8) = 0.16 : Q_Table(6, 9) =
0.16 : Q_Table(6, 10) = 0.01 : Q_Table(6, 11) = 0.06 : Q_Table(6, 12) = 0.14 : Q_Table(6, 13) = 0.3 : Q_Table(6, 14) =
0.11 : Q_Table(6, 15) = 0.29 : Q_Table(6, 16) = 0.13 : Q_Table(6, 17) = 0.23
    'Q_Table(7, 0) = 0 : Q_Table(7, 1) = 0.24 : Q_Table(7, 2) = 34.17 : Q_Table(7, 3) = 0.19 : Q_Table(7, 4) =
0.29 : Q_Table(7, 5) = 17.98 : Q_Table(7, 6) = 0.11 : Q_Table(7, 7) = 0.01 : Q_Table(7, 8) = 0.07 : Q_Table(7, 9) = 0.11
: Q_Table(7, 10) = 0.08 : Q_Table(7, 11) = 0.15 : Q_Table(7, 12) = 0.25 : Q_Table(7, 13) = 0.26 : Q_Table(7, 14) = 0.08
: Q_Table(7, 15) = 0.09 : Q_Table(7, 16) = 0.14 : Q_Table(7, 17) = 0.28
    'Q_Table(8, 0) = 0.17 : Q_Table(8, 1) = 0.04 : Q_Table(8, 2) = 0.13 : Q_Table(8, 3) = 7.96 : Q_Table(8, 4) =
0.05 : Q_Table(8, 5) = 17.34 : Q_Table(8, 6) = 0.21 : Q_Table(8, 7) = 0.15 : Q_Table(8, 8) = 0.03 : Q_Table(8, 9) = 0.01
: Q_Table(8, 10) = 0.01 : Q_Table(8, 11) = 0.3 : Q_Table(8, 12) = 0.03 : Q_Table(8, 13) = 0.24 : Q_Table(8, 14) = 0.2 :
Q_Table(8, 15) = 0.24 : Q_Table(8, 16) = 0.12 : Q_Table(8, 17) = 0.27
```

```
            'Q_Table(9, 0) = 0.19 : Q_Table(9, 1) = 0.28 : Q_Table(9, 2) = 0 : Q_Table(9, 3) = 0.02 : Q_Table(9, 4) = 
23.86 : Q_Table(9, 5) = 17.7 : Q_Table(9, 6) = 0.28 : Q_Table(9, 7) = 0.16 : Q_Table(9, 8) = 0.07 : Q_Table(9, 9) = 0.04 
: Q_Table(9, 10) = 0.28 : Q_Table(9, 11) = 0.1 : Q_Table(9, 12) = 0.04 : Q_Table(9, 13) = 0.29 : Q_Table(9, 14) = 0.07 : 
Q_Table(9, 15) = 0.13 : Q_Table(9, 16) = 0.03 : Q_Table(9, 17) = 0.03
            'Q_Table(10, 0) = 0.25 : Q_Table(10, 1) = 0.17 : Q_Table(10, 2) = 34.17 : Q_Table(10, 3) = 0.16 : 
Q_Table(10, 4) = 0.23 : Q_Table(10, 5) = 23.98 : Q_Table(10, 6) = 0.21 : Q_Table(10, 7) = 0.01 : Q_Table(10, 8) = 0.23 
: Q_Table(10, 9) = 0.25 : Q_Table(10, 10) = 0.1 : Q_Table(10, 11) = 0.03 : Q_Table(10, 12) = 0.02 : Q_Table(10, 13) = 
0.28 : Q_Table(10, 14) = 0.09 : Q_Table(10, 15) = 0.07 : Q_Table(10, 16) = 0.2 : Q_Table(10, 17) = 0.03
            'Q_Table(11, 0) = 0.17 : Q_Table(11, 1) = 0.2 : Q_Table(11, 2) = 0.18 : Q_Table(11, 3) = 11.71 : Q_Table(11, 
4) = 0.27 : Q_Table(11, 5) = 12.38 : Q_Table(11, 6) = 0.17 : Q_Table(11, 7) = 0.18 : Q_Table(11, 8) = 0.24 : 
Q_Table(11, 9) = 0.29 : Q_Table(11, 10) = 0.26 : Q_Table(11, 11) = 0.13 : Q_Table(11, 12) = 0.22 : Q_Table(11, 13) = 
0.03 : Q_Table(11, 14) = 0.15 : Q_Table(11, 15) = 0.26 : Q_Table(11, 16) = 0.23 : Q_Table(11, 17) = 0.17
            'Q_Table(12, 0) = 0.05 : Q_Table(12, 1) = 0.1 : Q_Table(12, 2) = 0.1 : Q_Table(12, 3) = 0.09 : Q_Table(12, 4) 
= 20.66 : Q_Table(12, 5) = 26.91 : Q_Table(12, 6) = 0.3 : Q_Table(12, 7) = 0.04 : Q_Table(12, 8) = 0.21 : Q_Table(12, 
9) = 0.29 : Q_Table(12, 10) = 0.28 : Q_Table(12, 11) = 0.08 : Q_Table(12, 12) = 0.1 : Q_Table(12, 13) = 0.11 : 
Q_Table(12, 14) = 0.07 : Q_Table(12, 15) = 0.21 : Q_Table(12, 16) = 0.3 : Q_Table(12, 17) = 0.25
            'Q_Table(13, 0) = 0.16 : Q_Table(13, 1) = 0.06 : Q_Table(13, 2) = 12.58 : Q_Table(13, 3) = 0.19 : 
Q_Table(13, 4) = 0.06 : Q_Table(13, 5) = 20.17 : Q_Table(13, 6) = 0.16 : Q_Table(13, 7) = 0.22 : Q_Table(13, 8) = 0.04 
: Q_Table(13, 9) = 0.12 : Q_Table(13, 10) = 0.05 : Q_Table(13, 11) = 0.27 : Q_Table(13, 12) = 0.12 : Q_Table(13, 13) = 
0.25 : Q_Table(13, 14) = 0.25 : Q_Table(13, 15) = 0.11 : Q_Table(13, 16) = 0 : Q_Table(13, 17) = 0.1
            'Q_Table(14, 0) = 0.02 : Q_Table(14, 1) = 0.04 : Q_Table(14, 2) = 0.19 : Q_Table(14, 3) = 6.02 : Q_Table(14, 
4) = 0.11 : Q_Table(14, 5) = 14.18 : Q_Table(14, 6) = 0.18 : Q_Table(14, 7) = 0.16 : Q_Table(14, 8) = 0.3 : Q_Table(14, 
9) = 0.12 : Q_Table(14, 10) = 0.26 : Q_Table(14, 11) = 0.22 : Q_Table(14, 12) = 0.2 : Q_Table(14, 13) = 0.26 : 
Q_Table(14, 14) = 0.06 : Q_Table(14, 15) = 0.15 : Q_Table(14, 16) = 0.27 : Q_Table(14, 17) = 0.2
            'Q_Table(15, 0) = 0.25 : Q_Table(15, 1) = 0.09 : Q_Table(15, 2) = 0.06 : Q_Table(15, 3) = 0.23 : Q_Table(15, 
4) = 4.9 : Q_Table(15, 5) = 21.27 : Q_Table(15, 6) = 0.24 : Q_Table(15, 7) = 0.28 : Q_Table(15, 8) = 0.02 : Q_Table(15, 
9) = 0.26 : Q_Table(15, 10) = 0.14 : Q_Table(15, 11) = 0.2 : Q_Table(15, 12) = 0.2 : Q_Table(15, 13) = 0.19 : 
Q_Table(15, 14) = 0.22 : Q_Table(15, 15) = 0.05 : Q_Table(15, 16) = 0.09 : Q_Table(15, 17) = 0.13
            'Q_Table(16, 0) = 0.01 : Q_Table(16, 1) = 0.08 : Q_Table(16, 2) = 9.17 : Q_Table(16, 3) = 0.17 : Q_Table(16, 
4) = 0.19 : Q_Table(16, 5) = 15.23 : Q_Table(16, 6) = 0.16 : Q_Table(16, 7) = 0 : Q_Table(16, 8) = 0.09 : Q_Table(16, 
9) = 0.05 : Q_Table(16, 10) = 0.26 : Q_Table(16, 11) = 0.15 : Q_Table(16, 12) = 0.17 : Q_Table(16, 13) = 0.29 : 
Q_Table(16, 14) = 0.15 : Q_Table(16, 15) = 0.18 : Q_Table(16, 16) = 0.26 : Q_Table(16, 17) = 0.24
            'Q_Table(17, 0) = 0.14 : Q_Table(17, 1) = 0.22 : Q_Table(17, 2) = 0.06 : Q_Table(17, 3) = 3.09 : Q_Table(17, 
4) = 0.22 : Q_Table(17, 5) = 8.41 : Q_Table(17, 6) = 0.04 : Q_Table(17, 7) = 0.2 : Q_Table(17, 8) = 0.02 : Q_Table(17, 
9) = 0.2 : Q_Table(17, 10) = 0.04 : Q_Table(17, 11) = 0.14 : Q_Table(17, 12) = 0.29 : Q_Table(17, 13) = 0.14 : 
Q_Table(17, 14) = 0.21 : Q_Table(17, 15) = 0.17 : Q_Table(17, 16) = 0.28 : Q_Table(17, 17) = 0.18
            'Q_Table(18, 0) = 0.01 : Q_Table(18, 1) = 0.22 : Q_Table(18, 2) = 0.28 : Q_Table(18, 3) = 0.01 : Q_Table(18, 
4) = 2.89 : Q_Table(18, 5) = 6.55 : Q_Table(18, 6) = 0.16 : Q_Table(18, 7) = 0.24 : Q_Table(18, 8) = 0.28 : Q_Table(18, 
9) = 0.29 : Q_Table(18, 10) = 0.05 : Q_Table(18, 11) = 0.18 : Q_Table(18, 12) = 0.15 : Q_Table(18, 13) = 0.02 : 
Q_Table(18, 14) = 0.03 : Q_Table(18, 15) = 0.22 : Q_Table(18, 16) = 0.26 : Q_Table(18, 17) = 0.07


            s2.Close()
        End If
        State_Action_Real_Timer1.Enabled = False
        If allow_sound_flag = 1 Then
            rc = PlaySound(System.AppDomain.CurrentDomain.BaseDirectory & "policy_completed.wav", 0, 
SND_NOSTOP)
        End If
    End If
End Sub

Private Function Randomizer(ByVal iStart As Double, ByVal iEnd As Double) As Double
Dim iRandomValue As Double
    Randomize()
    iRandomValue = iStart + (Rnd() * (iEnd - iStart))
    Return iRandomValue
End Function

Private Sub TrackBar1_Scroll(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles 
TrackBar1.Scroll
    TextBox116.Text = TrackBar1.Value.ToString
End Sub
```

```
    Private Sub Button30_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button30.Click
        If TrackBar1.Value = 0 Then
            MsgBox("A policy length can not be zero!")
        Else
            Reset_Policy_1()
            State_Action_Rand_Timer1.Enabled = True
        End If
    End Sub

    Function Q_Lamda_Algorithm()
    Dim maxQ As Double
    Dim epsilon As Double
    Dim best_action_index As Integer
    Dim temp_state As Integer
    Dim temp_action As Integer
    Dim temp_maxQ_V1 As Double
    Dim R2 As Double
    Dim temp_state_Q
    Dim string_of_results As String

        epsilon = Val(TextBox128.Text)

        TextBox123.Text = TextBox132.Text

        If Val(TextBox133.Text) = "0" Then
            epsilon = 1
        End If

        Randomize(DateTime.Now.Second())
        R2 = Rnd(DateTime.Now.Millisecond())
        'MsgBox(R2)
        'R2 = Randomizer(0, 1)
        'MsgBox(R2)

        If CheckBox8.Checked = False Then
            If R2 > 0.95 Then R2 = 1
        End If

        Action_Timer_1.Enabled = True
        ProgressBar1.Value = learning_trial

        TextBox124.Text = TextBox134.Text
        While (learning_trial <= Val(TextBox124.Text) - 1)

            ListBox3.Items.Add(state_Q)

            '''Find_A_Different_Action:

            For reward_counter_1 As Integer = 0 To 18
                For reward_counter_2 As Integer = 0 To (35 - 18)
                    Reward(reward_counter_1, reward_counter_2) = Val(TextBox113.Text) ' / 5
                Next
            Next

            For i As Integer = 0 To (35 - 18)
                maxQ_V1(0, i) = 0
            Next
            temp_maxQ_V1 = 0

            ' Choose an action:
```

```
      Randomize(DateTime.Now.Second())
      R2 = Rnd(DateTime.Now.Millisecond())

      'R2 = Randomizer(0, 1)

      If CheckBox8.Checked = False Then
        If R2 > 0.95 Then R2 = 1
      End If


      For i As Integer = 0 To (35 - 18)
        maxQ_V1(0, i) = Q_Table(state_Q, i)
        temp_maxQ_V1 = Max(temp_maxQ_V1, maxQ_V1(0, i))
      Next

      For i As Integer = 0 To (35 - 18)
        If Q_Table(state_Q, i) = temp_maxQ_V1 Then
          best_action_index = i
        End If
      Next

      If (R2 > epsilon) Then
        action_Q = best_action_index
      Else
        action_Q = CInt(R2 * (35 - 18))
      End If

      ' Next state:

Find_A_Different_Action:
      Try
        xConn = New sqlConn
        xConn.connectMe("SELECT Action" & action_Q.ToString() & " FROM States_Numbers_Nothing Where
Id=" & state_Q.ToString)
        Try
          xConn.OLEComm.Connection = xConn.OLEConn
        Dim d As OleDb.OleDbDataReader = xConn.OLEComm.ExecuteReader()
          d.Read()
          temp_state_Q = d("Action" & action_Q.ToString())

          If temp_state_Q = "Nothing" Then
            xConn.OLEConn.Close()

            Randomize(DateTime.Now.Second())
            R2 = Rnd(DateTime.Now.Millisecond())

            '  R2 = Randomizer(0, 1)

            If CheckBox8.Checked = False Then
              If R2 > 0.95 Then R2 = 1
            End If

            action_Q = CInt(R2 * (35 - 18))
            GoTo Find_A_Different_Action
          Else
            next_state_Q = Val(d("Action" & action_Q.ToString()))
          End If
          Try
            xConn.OLEConn.Close()
          Catch err As System.Exception
            MsgBox(err.Message)
          End Try
        Catch err As System.Exception
```

```
          MsgBox(err.Message)
        End Try
      Catch err As System.Exception
        MsgBox(err.Message)
      End Try

      ListBox4.Items.Add(action_Q)
      '       ListBox5.Items.Add(Reward(state_Q, action_Q))
      ListBox5.Items.Add(TextBox132.Text)

      endTime = Now

      ListBox6.Items.Add(Round(endTime.Subtract(startTime).TotalMilliseconds))

      ListBox3.Refresh()
      ListBox4.Refresh()
      ListBox5.Refresh()
      ListBox6.Refresh()

      delta(learning_trial, 0) = Reward(state_Q, action_Q) + gamma * Q_Table(next_state_Q, action_Q) -
Q_Table(state_Q, action_Q)
      Eligibility(state_Q, action_Q) = Eligibility(state_Q, action_Q) + 1

      For state_i As Integer = 0 To 18
        For action_i As Integer = 0 To (35 - 18)
          If (state_i = state_Q) And (action_i = action_Q) Then
            Eligibility = MatLib.ScalarMultiply(lambda * gamma, Eligibility)
          Else
            Eligibility(state_i, action_i) = 0
          End If
          Q_Table(state_i, action_i) = Q_Table(state_i, action_i) + alpha * delta(learning_trial, 0) * Eligibility(state_i,
action_i)
        Next
      Next

      state_Q = next_state_Q
      learning_trial = learning_trial + 1
      Q_Lamda_Algorithm()

    End While

    TextBox117.Text = MatLib.PrintMat(delta) & vbCrLf & vbCrLf
    TextBox118.Text = MatLib.PrintMat(Q_Table) & vbCrLf & vbCrLf

    Button35.Enabled = True
  End Function

  Public Sub initQFunction()
    ListBox3.Items.Clear()
    ListBox4.Items.Clear()
    ListBox5.Items.Clear()
    ListBox6.Items.Clear()

    ListBox10.Items.Clear()
    ListBox9.Items.Clear()
    ListBox8.Items.Clear()
    ListBox7.Items.Clear()


    TextBox117.Text = ""
    TextBox118.Text = ""

    TextBox125.Text = ""
```

```
        TextBox126.Text = ""

    Dim j As Object
    Dim i As Object
    Dim k As Object
    Dim l As Object
    Dim m As Object
    Dim n As Object

    Dim R3 As Double

        For i = 0 To 18 ' 19 x 36
            For j = 0 To (35 - 18)
                Randomize(DateTime.Now.Millisecond())
                R3 = 0.3 * Rnd(DateTime.Now.Second())
                Q_Table(i, j) = R3
                Q_Table_Rewarded(i, j) = R3

                Q_Table_Final(i, j) = R3 '0
                Eligibility(i, j) = 0
                Eligibility_Rewarded(i, j) = 0

            Next j
        Next i

        '    If ComboBox7.SelectedItem = "Policy After System Crash" Then
'Q_Table(0, 0) = 0.27 : Q_Table(0, 1) = 0.25 : Q_Table(0, 2) = 0.18 : Q_Table(0, 3) = 0.29 : Q_Table(0, 4) = 0.2 :
Q_Table(0, 5) = 0.18 : Q_Table(0, 6) = 0.27 : Q_Table(0, 7) = 0.48 : Q_Table(0, 8) = 0.47 : Q_Table(0, 9) = 0.49 :
Q_Table(0, 10) = 0.22 : Q_Table(0, 11) = 0.49 : Q_Table(0, 12) = 0.06 : Q_Table(0, 13) = 0.08 : Q_Table(0, 14) = 0.08 :
Q_Table(0, 15) = 0.07 : Q_Table(0, 16) = 0.01 : Q_Table(0, 17) = 0.02
'Q_Table(1, 0) = 0.28 : Q_Table(1, 1) = 0.17 : Q_Table(1, 2) = 0.16 : Q_Table(1, 3) = 0.01 : Q_Table(1, 4) = 0.06 :
Q_Table(1, 5) = 0.2 : Q_Table(1, 6) = 0.18 : Q_Table(1, 7) = 0.23 : Q_Table(1, 8) = 0.24 : Q_Table(1, 9) = 0.23 :
Q_Table(1, 10) = 0.26 : Q_Table(1, 11) = 0.09 : Q_Table(1, 12) = 0.13 : Q_Table(1, 13) = 0.17 : Q_Table(1, 14) = 0.05 :
Q_Table(1, 15) = 0.12 : Q_Table(1, 16) = 0.12 : Q_Table(1, 17) = 0.21
 'Q_Table(2, 0) = 0.23 : Q_Table(2, 1) = 0.25 : Q_Table(2, 2) = 0.2 : Q_Table(2, 3) = 0.12 : Q_Table(2, 4) = 0.13 :
Q_Table(2, 5) = 0.15 : Q_Table(2, 6) = 0.3 : Q_Table(2, 7) = 0.19 : Q_Table(2, 8) = 0.2 : Q_Table(2, 9) = 0.02 :
Q_Table(2, 10) = 0.19 : Q_Table(2, 11) = 0.21 : Q_Table(2, 12) = 0.29 : Q_Table(2, 13) = 0.27 : Q_Table(2, 14) = 0.17 :
Q_Table(2, 15) = 0.25 : Q_Table(2, 16) = 0.1 : Q_Table(2, 17) = 0.1
'Q_Table(3, 0) = 0.26 : Q_Table(3, 1) = 0.01 : Q_Table(3, 2) = 0.08 : Q_Table(3, 3) = 0.24 : Q_Table(3, 4) = 0.06 :
Q_Table(3, 5) = 0.17 : Q_Table(3, 6) = 0.07 : Q_Table(3, 7) = 0.18 : Q_Table(3, 8) = 0.03 : Q_Table(3, 9) = 0.18 :
Q_Table(3, 10) = 0.24 : Q_Table(3, 11) = 0.21 : Q_Table(3, 12) = 0.01 : Q_Table(3, 13) = 0.22 : Q_Table(3, 14) = 0.05 :
Q_Table(3, 15) = 0.25 : Q_Table(3, 16) = 0.02 : Q_Table(3, 17) = 0.05
'Q_Table(4, 0) = 0.16 : Q_Table(4, 1) = 0.09 : Q_Table(4, 2) = 0.19 : Q_Table(4, 3) = 0.28 : Q_Table(4, 4) = 0.02 :
Q_Table(4, 5) = 0.14 : Q_Table(4, 6) = 0.09 : Q_Table(4, 7) = 0.22 : Q_Table(4, 8) = 0.19 : Q_Table(4, 9) = 0.06 :
Q_Table(4, 10) = 0.07 : Q_Table(4, 11) = 0.26 : Q_Table(4, 12) = 0.3 : Q_Table(4, 13) = 0.11 : Q_Table(4, 14) = 0.15 :
Q_Table(4, 15) = 0.09 : Q_Table(4, 16) = 0.28 : Q_Table(4, 17) = 0.25
'Q_Table(5, 0) = 0.25 : Q_Table(5, 1) = 0.14 : Q_Table(5, 2) = 0.27 : Q_Table(5, 3) = 0.18 : Q_Table(5, 4) = 0.22 :
Q_Table(5, 5) = 0.18 : Q_Table(5, 6) = 0.06 : Q_Table(5, 7) = 0.22 : Q_Table(5, 8) = 0.21 : Q_Table(5, 9) = 0 :
Q_Table(5, 10) = 0.25 : Q_Table(5, 11) = 0.26 : Q_Table(5, 12) = 0.21 : Q_Table(5, 13) = 0.06 : Q_Table(5, 14) = 0.22 :
Q_Table(5, 15) = 0.1 : Q_Table(5, 16) = 0.09 : Q_Table(5, 17) = 0.29
'Q_Table(6, 0) = 0.22 : Q_Table(6, 1) = 0.04 : Q_Table(6, 2) = 0.15 : Q_Table(6, 3) = 0.15 : Q_Table(6, 4) = 0.01 :
Q_Table(6, 5) = 0.14 : Q_Table(6, 6) = 0.27 : Q_Table(6, 7) = 0.27 : Q_Table(6, 8) = 0.19 : Q_Table(6, 9) = 0.19 :
Q_Table(6, 10) = 0.05 : Q_Table(6, 11) = 0.1 : Q_Table(6, 12) = 0.17 : Q_Table(6, 13) = 0.04 : Q_Table(6, 14) = 0.15 :
Q_Table(6, 15) = 0.03 : Q_Table(6, 16) = 0.17 : Q_Table(6, 17) = 0.27
'Q_Table(7, 0) = 0.04 : Q_Table(7, 1) = 0.27 : Q_Table(7, 2) = 0.6 : Q_Table(7, 3) = 0.22 : Q_Table(7, 4) = 0.03 :
Q_Table(7, 5) = 0.12 : Q_Table(7, 6) = 0.14 : Q_Table(7, 7) = 0.05 : Q_Table(7, 8) = 0.11 : Q_Table(7, 9) = 0.14 :
Q_Table(7, 10) = 0.12 : Q_Table(7, 11) = 0.19 : Q_Table(7, 12) = 0.29 : Q_Table(7, 13) = 0.29 : Q_Table(7, 14) = 0.12 :
Q_Table(7, 15) = 0.13 : Q_Table(7, 16) = 0.18 : Q_Table(7, 17) = 0.02
'Q_Table(8, 0) = 0.21 : Q_Table(8, 1) = 0.08 : Q_Table(8, 2) = 0.17 : Q_Table(8, 3) = 0.58 : Q_Table(8, 4) = 0.09 :
Q_Table(8, 5) = 0.17 : Q_Table(8, 6) = 0.25 : Q_Table(8, 7) = 0.19 : Q_Table(8, 8) = 0.07 : Q_Table(8, 9) = 0.04 :
Q_Table(8, 10) = 0.05 : Q_Table(8, 11) = 0.04 : Q_Table(8, 12) = 0.07 : Q_Table(8, 13) = 0.28 : Q_Table(8, 14) = 0.24 :
Q_Table(8, 15) = 0.28 : Q_Table(8, 16) = 0.16 : Q_Table(8, 17) = 0.01
```

```
'Q_Table(9, 0) = 0.23 : Q_Table(9, 1) = 0.02 : Q_Table(9, 2) = 0.04 : Q_Table(9, 3) = 0.06 : Q_Table(9, 4) = 0.35 :
Q_Table(9, 5) = 0.16 : Q_Table(9, 6) = 0.02 : Q_Table(9, 7) = 0.2 : Q_Table(9, 8) = 0.11 : Q_Table(9, 9) = 0.08 :
Q_Table(9, 10) = 0.01 : Q_Table(9, 11) = 0.13 : Q_Table(9, 12) = 0.08 : Q_Table(9, 13) = 0.02 : Q_Table(9, 14) = 0.1 :
Q_Table(9, 15) = 0.16 : Q_Table(9, 16) = 0.06 : Q_Table(9, 17) = 0.06
'Q_Table(10, 0) = 0.29 : Q_Table(10, 1) = 0.21 : Q_Table(10, 2) = 0.55 : Q_Table(10, 3) = 0.2 : Q_Table(10, 4) = 0.27 :
Q_Table(10, 5) = 0.16 : Q_Table(10, 6) = 0.24 : Q_Table(10, 7) = 0.04 : Q_Table(10, 8) = 0.26 : Q_Table(10, 9) = 0.29 :
Q_Table(10, 10) = 0.14 : Q_Table(10, 11) = 0.07 : Q_Table(10, 12) = 0.05 : Q_Table(10, 13) = 0.02 : Q_Table(10, 14) =
0.12 : Q_Table(10, 15) = 0.11 : Q_Table(10, 16) = 0.24 : Q_Table(10, 17) = 0.06
'Q_Table(11, 0) = 0.21 : Q_Table(11, 1) = 0.24 : Q_Table(11, 2) = 0.22 : Q_Table(11, 3) = 0.65 : Q_Table(11, 4) = 0 :
Q_Table(11, 5) = 0.16 : Q_Table(11, 6) = 0.21 : Q_Table(11, 7) = 0.22 : Q_Table(11, 8) = 0.27 : Q_Table(11, 9) = 0.03 :
Q_Table(11, 10) = 0 : Q_Table(11, 11) = 0.17 : Q_Table(11, 12) = 0.26 : Q_Table(11, 13) = 0.07 : Q_Table(11, 14) =
0.19 : Q_Table(11, 15) = 0.3 : Q_Table(11, 16) = 0.26 : Q_Table(11, 17) = 0.2
'Q_Table(12, 0) = 0.09 : Q_Table(12, 1) = 0.14 : Q_Table(12, 2) = 0.14 : Q_Table(12, 3) = 0.13 : Q_Table(12, 4) = 0.38 :
Q_Table(12, 5) = 0.15 : Q_Table(12, 6) = 0.03 : Q_Table(12, 7) = 0.08 : Q_Table(12, 8) = 0.25 : Q_Table(12, 9) = 0.03 :
Q_Table(12, 10) = 0.02 : Q_Table(12, 11) = 0.11 : Q_Table(12, 12) = 0.14 : Q_Table(12, 13) = 0.15 : Q_Table(12, 14) =
0.1 : Q_Table(12, 15) = 0.25 : Q_Table(12, 16) = 0.04 : Q_Table(12, 17) = 0.29
'Q_Table(13, 0) = 0.2 : Q_Table(13, 1) = 0.1 : Q_Table(13, 2) = 0.05 : Q_Table(13, 3) = 0.23 : Q_Table(13, 4) = 0.09 :
Q_Table(13, 5) = 0.16 : Q_Table(13, 6) = 0.2 : Q_Table(13, 7) = 0.26 : Q_Table(13, 8) = 0.08 : Q_Table(13, 9) = 0.16 :
Q_Table(13, 10) = 0.08 : Q_Table(13, 11) = 0.01 : Q_Table(13, 12) = 0.16 : Q_Table(13, 13) = 0.29 : Q_Table(13, 14) =
0.28 : Q_Table(13, 15) = 0.15 : Q_Table(13, 16) = 0.04 : Q_Table(13, 17) = 0.14
'Q_Table(14, 0) = 0.06 : Q_Table(14, 1) = 0.08 : Q_Table(14, 2) = 0.23 : Q_Table(14, 3) = 0.03 : Q_Table(14, 4) = 0.15 :
Q_Table(14, 5) = 0.15 : Q_Table(14, 6) = 0.22 : Q_Table(14, 7) = 0.2 : Q_Table(14, 8) = 0.03 : Q_Table(14, 9) = 0.16 :
Q_Table(14, 10) = 0 : Q_Table(14, 11) = 0.25 : Q_Table(14, 12) = 0.23 : Q_Table(14, 13) = 0.3 : Q_Table(14, 14) = 0.1
: Q_Table(14, 15) = 0.18 : Q_Table(14, 16) = 0.01 : Q_Table(14, 17) = 0.24
'Q_Table(15, 0) = 0.29 : Q_Table(15, 1) = 0.13 : Q_Table(15, 2) = 0.1 : Q_Table(15, 3) = 0.27 : Q_Table(15, 4) = 0.03 :
Q_Table(15, 5) = 0.16 : Q_Table(15, 6) = 0.28 : Q_Table(15, 7) = 0.01 : Q_Table(15, 8) = 0.06 : Q_Table(15, 9) = 0.3 :
Q_Table(15, 10) = 0.18 : Q_Table(15, 11) = 0.23 : Q_Table(15, 12) = 0.24 : Q_Table(15, 13) = 0.23 : Q_Table(15, 14) =
0.26 : Q_Table(15, 15) = 0.09 : Q_Table(15, 16) = 0.13 : Q_Table(15, 17) = 0.17
'Q_Table(16, 0) = 0.05 : Q_Table(16, 1) = 0.12 : Q_Table(16, 2) = 0.06 : Q_Table(16, 3) = 0.21 : Q_Table(16, 4) = 0.23 :
Q_Table(16, 5) = 0.18 : Q_Table(16, 6) = 0.2 : Q_Table(16, 7) = 0.04 : Q_Table(16, 8) = 0.13 : Q_Table(16, 9) = 0.09 :
Q_Table(16, 10) = 0.3 : Q_Table(16, 11) = 0.19 : Q_Table(16, 12) = 0.2 : Q_Table(16, 13) = 0.02 : Q_Table(16, 14) =
0.19 : Q_Table(16, 15) = 0.21 : Q_Table(16, 16) = 0.29 : Q_Table(16, 17) = 0.27
'Q_Table(17, 0) = 0.17 : Q_Table(17, 1) = 0.25 : Q_Table(17, 2) = 0.1 : Q_Table(17, 3) = 0.01 : Q_Table(17, 4) = 0.26 :
Q_Table(17, 5) = 0.08 : Q_Table(17, 6) = 0.08 : Q_Table(17, 7) = 0.24 : Q_Table(17, 8) = 0.05 : Q_Table(17, 9) = 0.23 :
Q_Table(17, 10) = 0.07 : Q_Table(17, 11) = 0.18 : Q_Table(17, 12) = 0.03 : Q_Table(17, 13) = 0.18 : Q_Table(17, 14) =
0.24 : Q_Table(17, 15) = 0.21 : Q_Table(17, 16) = 0.01 : Q_Table(17, 17) = 0.22
'Q_Table(18, 0) = 0.05 : Q_Table(18, 1) = 0.25 : Q_Table(18, 2) = 0.02 : Q_Table(18, 3) = 0.05 : Q_Table(18, 4) = 0.02 :
Q_Table(18, 5) = 0.13 : Q_Table(18, 6) = 0.19 : Q_Table(18, 7) = 0.28 : Q_Table(18, 8) = 0.02 : Q_Table(18, 9) = 0.03 :
Q_Table(18, 10) = 0.09 : Q_Table(18, 11) = 0.22 : Q_Table(18, 12) = 0.19 : Q_Table(18, 13) = 0.06 : Q_Table(18, 14) =
0.07 : Q_Table(18, 15) = 0.26 : Q_Table(18, 16) = 0.3 : Q_Table(18, 17) = 0.11

Q_Table_Rewarded = Q_Table
Q_Table_Final = Q_Table

'  End If

    For m = 0 To 4
      Performance_Measure_1(0, m) = 0
    Next

    startTime = Now
    TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
    TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
    TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    For n = 0 To 50
      Times_vector(0, n) = 0
      Cummulative_Value_vector(0, n) = 0
      Events_Value_vector(0, n) = 0
    Next

  End Sub
```

```
    Private Sub Button33_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button33.Click

        TextBox81.Text = TextBox140.Text
        TextBox156.Text = TextBox139.Text
        TextBox157.Text = TextBox138.Text

        TextBox153.Text = TextBox136.Text
        TextBox154.Text = TextBox137.Text
        TextBox155.Text = TextBox90.Text

        TextBox124.Text = TextBox134.Text
        ProgressBar1.Maximum = Val(TextBox124.Text)
        ProgressBar2.Maximum = Val(TextBox124.Text)
        Button33.Enabled = False
        initQFunction()

        learning_trial = 0

        Q_Lamda_Algorithm()

        learning_trial = 0
        state_Q = 0
        Q_Values_After_A_Reward_Was_Given_To_A_Policy()

        Write_Some_Policy_To_Database()

        Number_of_Policies_Performed = Number_of_Policies_Performed + 1
        TextBox112.Text = (Number_of_Policies_Performed).ToString
        TextBox133.Text = TextBox112.Text

    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Trial_Number", TextBox133.Text)

        If (ProgressBar1.Value = Val(TextBox124.Text)) Then
            If allow_sound_flag = 1 Then
                rc = PlaySound(System.AppDomain.CurrentDomain.BaseDirectory & "rl_completed.wav", 0, SND_NOSTOP)
            End If
        End If
    End Sub

    Private Sub Button34_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button34.Click
        If ComboBox2.SelectedIndex = 0 Then
            Run_Program("GRASPH1" + ".JBI")
        Else
            Run_Program("GRASPH2" + ".JBI")
        End If
    End Sub

    Private Sub Button35_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button35.Click

        ProgressBar1.Maximum = Val(TextBox124.Text)
        ProgressBar2.Maximum = Val(TextBox124.Text)

        ProgressBar1.Value = 0
        ProgressBar2.Value = 0
        TextBox124.Text = TextBox134.Text

        ListBox3.Items.Clear()
        ListBox4.Items.Clear()
        ListBox5.Items.Clear()
        ListBox6.Items.Clear()
```

```
        ListBox10.Items.Clear()
        ListBox9.Items.Clear()
        ListBox8.Items.Clear()
        ListBox7.Items.Clear()

        state_Q = 0
        learning_trial = 0
        Q_Lamda_Algorithm()

        If (ProgressBar1.Value = Val(TextBox124.Text)) Then
            If allow_sound_flag = 1 Then
                rc = PlaySound(System.AppDomain.CurrentDomain.BaseDirectory & "rl_completed.wav", 0, SND_NOSTOP)
            End If
        End If

        state_Q = 0
        learning_trial = 0
        Q_Values_After_A_Reward_Was_Given_To_A_Policy()

    End Sub

    Private Sub Action_Timer_1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Action_Timer_1.Tick
        Action_Time = Action_Time + Action_Timer_1.Interval
    End Sub

    Public Sub Q_Values_After_A_Reward_Was_Given_To_A_Policy()
    Dim maxQ As Double
    Dim epsilon As Double
    Dim best_action_index As Integer
    Dim temp_state As Integer
    Dim temp_action As Integer
    Dim temp_maxQ_V1 As Double
    Dim R2 As Double
    Dim temp_state_Q
    Dim string_of_results As String
        epsilon = Val(TextBox128.Text)

        TextBox123.Text = TextBox132.Text

        Action_Timer_1.Enabled = True
        TextBox124.Text = TextBox134.Text
        While (learning_trial <= Val(TextBox124.Text) - 1)

            state_Q = ListBox3.Items.Item(learning_trial)

            ListBox10.Items.Add(state_Q)

            For reward_counter_1 As Integer = 0 To 18
                For reward_counter_2 As Integer = 0 To (35 - 18)
                    Reward(reward_counter_1, reward_counter_2) = Val(TextBox123.Text)
                Next
            Next

            action_Q = ListBox4.Items.Item(learning_trial)

            ListBox9.Items.Add(action_Q)
            '''             ListBox8.Items.Add(Reward(state_Q, action_Q)) 'Here the actual positive reward is given.
            ListBox8.Items.Add(Val(TextBox132.Text))

            endTime = Now
```

```
            ListBox7.Items.Add(Round(endTime.Subtract(startTime).TotalMilliseconds))

            ListBox10.Refresh()
            ListBox9.Refresh()
            ListBox8.Refresh()
            ListBox7.Refresh()

            delta_Rewarded(learning_trial, 0) = Reward(state_Q, action_Q) + gamma * Q_Table_Rewarded(next_state_Q,
action_Q) - Q_Table_Rewarded(state_Q, action_Q)
            Eligibility_Rewarded(state_Q, action_Q) = Eligibility_Rewarded(state_Q, action_Q) + 1

            For state_i As Integer = 0 To 18
                For action_i As Integer = 0 To (35 - 18)
                    If (state_i = state_Q) And (action_i = action_Q) Then
                        Eligibility_Rewarded = MatLib.ScalarMultiply(lambda * gamma, Eligibility_Rewarded)
                    Else
                        Eligibility_Rewarded(state_i, action_i) = 0
                    End If
                    Q_Table_Rewarded(state_i, action_i) = Q_Table_Rewarded(state_i, action_i) + alpha *
delta_Rewarded(learning_trial, 0) * Eligibility_Rewarded(state_i, action_i)
                Next
            Next

            state_Q = next_state_Q
            learning_trial = learning_trial + 1
            Q_Values_After_A_Reward_Was_Given_To_A_Policy()

        End While

        TextBox125.Text = MatLib.PrintMat(delta_Rewarded) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Rewarded) & vbCrLf & vbCrLf

    End Sub

    Private Sub Button39_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button39.Click
        Reset_Policy_1()
    End Sub

    Private Sub Button38_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button38.Click
        Reset_Policy_1()

        ' Update listbox3 and listbox4 according to the best policy stored in the database

        ListBox3.Items.Clear()
        ListBox4.Items.Clear()

        ListBox10.Items.Clear()
        ListBox9.Items.Clear()
        TextBox124.Text = TextBox134.Text
        For i As Integer = 0 To Val(TextBox124.Text) - 1
            '           ProgressBar2.Value = i + 1
            xConn = New sqlConn
            'xConn.connectMe("SELECT State_Q FROM Best_Policy Where Id='" & i & "'")
            'xConn.connectMe("INSERT INTO Best_Policy(Id, State_Q, Action_Q, Reward) VALUES(" & i & "," &
string_listbox3 & "," & string_listbox4 & "," & string_listbox5 & " )")
            xConn.connectMe("SELECT State_Q FROM Best_Policy Where Id=" & i)
            For iCounter = 0 To xConn.getData("State_Q").Count - 1
                ListBox3.Items.Add(xConn.dataReturned.Item(iCounter))
                ListBox10.Items.Add(xConn.dataReturned.Item(iCounter))
            Next
            xConn.OLEConn.Close()

            xConn = New sqlConn
```

```
        'xConn.connectMe("SELECT Action_Q FROM Best_Policy Where Id='" & i & "'")
        xConn.connectMe("SELECT Action_Q FROM Best_Policy Where Id=" & i)
        For iCounter = 0 To xConn.getData("Action_Q").Count - 1
          ListBox4.Items.Add(xConn.dataReturned.Item(iCounter))
          ListBox9.Items.Add(xConn.dataReturned.Item(iCounter))
        Next
        xConn.OLEConn.Close()
      Next

      State_Action_Timer1_Real_Counter_1 = 0
      State_Action_Real_Timer1.Enabled = True
      temp_output_file_name_1 = TextBox133.Text & "_Trial_" & "Intervention"
    End Sub

    Private Sub Button37_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button37.Click
      number_of_policies_that_were_not_rewarded = number_of_policies_that_were_not_rewarded + 1
      system_performance_measure_1 = number_of_policies_that_were_rewarded /
(number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded)
      TextBox131.Text = (Learning_Performance_1 * 20).ToString

      output_reward_1 = Val(TextBox113.Text) '0 '-Val(TextBox123.Text) / 5
      Q_Table_Final = Q_Table
      ' Q_Table_Rewarded = Q_Table

      TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

      If ((number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded) >
Val(TextBox158.Text)) And (Val(TextBox131.Text) <= Val(TextBox130.Text)) Then '9
        TextBox146.Text = "Semi-Autonomous Mode - Human Suggests a Policy."
        TabControl1.SelectedTab = TabPage8
        Button51.Enabled = True
        Button50.Enabled = False
        GroupBox19.Enabled = True
      Else
        TextBox146.Text = "Autonomous Mode - No Human Intervention is Required."
        Button51.Enabled = False
        Button50.Enabled = True
        GroupBox19.Enabled = False
      End If

    End Sub

    Private Sub Button27_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button27.Click
      number_of_policies_that_were_rewarded = number_of_policies_that_were_rewarded + 1
      system_performance_measure_1 = number_of_policies_that_were_rewarded /
(number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded)
      TextBox131.Text = (Learning_Performance_1 * 20).ToString
      TextBox123.Text = TextBox132.Text
      output_reward_1 = Val(TextBox123.Text)
      Q_Table_Final = Q_Table_Rewarded
      ' Q_Table = Q_Table_Rewarded

      TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

      ''' Write last best successfuly policy to database
    Dim string_listbox3 As String
    Dim string_listbox4 As String
    Dim string_listbox5 As String
    Dim string_i As String
    Dim i As Integer

      '' delete records
      xConn.connectMe("Delete * FROM Best_Policy")
```

```
        xConn.OLEComm.ExecuteNonQuery()
        xConn.OLEConn.Close()
        xConn.OLEConn.Dispose()

        '' write best policy
        TextBox124.Text = TextBox134.Text
        For i = 0 To Val(TextBox124.Text) - 1
            string_listbox3 = ListBox3.Items.Item(i)
            string_listbox4 = ListBox4.Items.Item(i)
            string_listbox5 = ListBox5.Items.Item(i)
            xConn.connectMe("INSERT INTO Best_Policy(Id, State_Q, Action_Q, Reward) VALUES(" & i & "," &
string_listbox3 & "," & string_listbox4 & "," & string_listbox5 & " )")
            xConn.OLEComm.ExecuteNonQuery()
            xConn.OLEConn.Close()
            xConn.OLEConn.Dispose()
        Next

        If ((number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded) >
Val(TextBox158.Text)) And (Val(TextBox131.Text) <= Val(TextBox130.Text)) Then '9
            '''             MsgBox("Semi-Autonomous Mode - Human Suggests a Policy!")'
            TextBox146.Text = "Semi-Autonomous Mode - Human Suggests a Policy."
            TabControl1.SelectedTab = TabPage8
            Button51.Enabled = True
            Button50.Enabled = False
            GroupBox19.Enabled = True
        Else
            TextBox146.Text = "Autonomous Mode - No Human Intervention is Required."
            Button51.Enabled = False
            Button50.Enabled = True
            GroupBox19.Enabled = False
        End If
    End Sub

    Private Sub Button43_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Disconnect_Robot()
    End Sub

    Private Sub Button42_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Button31_Click(sender, e)
    End Sub

    Private Sub Button44_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Button38_Click(sender, e)
    End Sub

    Private Sub Button41_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Button34_Click(sender, e)
    End Sub

    Private Sub Button36_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button36.Click
        Reset_Policy_1()
        State_Action_Timer1_Real_Counter_1 = 0
        State_Action_Real_Timer1.Enabled = True
        temp_output_file_name_1 = TextBox133.Text & "_Trial"
    End Sub

    Public Sub Write_Some_Policy_To_Database()

        ''' Write last best successfuly policy to database
    Dim string_listbox3 As String
    Dim string_listbox4 As String
    Dim string_listbox5 As String
    Dim string_i As String
```

```
Dim i As Integer

    '' delete records
    xConn.connectMe("Delete * FROM Best_Policy")
    xConn.OLEComm.ExecuteNonQuery()
    xConn.OLEConn.Close()
    xConn.OLEConn.Dispose()

    '' write best policy
    TextBox124.Text = TextBox134.Text
    For i = 0 To Val(TextBox124.Text) - 1
        string_listbox3 = ListBox3.Items.Item(i)
        string_listbox4 = ListBox4.Items.Item(i)
        string_listbox5 = ListBox5.Items.Item(i)
        xConn.connectMe("INSERT INTO Best_Policy(Id, State_Q, Action_Q, Reward) VALUES(" & i & "," &
string_listbox3 & "," & string_listbox4 & "," & string_listbox5 & " )")
        xConn.OLEComm.ExecuteNonQuery()
        xConn.OLEConn.Close()
        xConn.OLEConn.Dispose()
    Next

End Sub

Private Sub Button45_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button45.Click
    '       Button1_Click(sender, e)
    TextBox1.Text = Ms_BscOpenComm(0)
    If TextBox1.Text <> "-1" And TextBox2.Text = "1" Then
        Label15.Text = "Connected"
    Else
        Label15.Text = "Disconnected"
    End If
    CheckBox1.Checked = False
    CheckBox2.Checked = True
End Sub

Private Sub Button40_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button40.Click
    'Button9_Click(sender, e)
    Disconnect_Robot()
End Sub

Private Sub Button49_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button49.Click
    TextBox135.Text = "Initializing!"
    If CheckBox13.Checked = True Then
        MatLab = CreateObject("Matlab.Application")
    End If

    Button49.Enabled = False
    TextBox134.Enabled = False
    Button33_Click_1(sender, e)
    Button50_Click(sender, e)
End Sub

Private Sub Button48_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Dim n As Integer

    Button48.Enabled = False
    Button35_Click(sender, e)

    If ((number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded) <=
Val(TextBox158.Text)) Then
        Button50.Enabled = True
    End If
```

```
        If ((number_of_policies_that_were_rewarded + number_of_policies_that_were_not_rewarded) >
Val(TextBox158.Text)) And (Val(TextBox131.Text) <= Val(TextBox130.Text)) Then '9
            Button51.Enabled = True
            Button50.Enabled = False
        Else
            Button51.Enabled = False
            Button50.Enabled = True
        End If

        For n = 0 To 50
            Times_vector(0, n) = 0
            Cummulative_Value_vector(0, n) = 0
            Events_Value_vector(0, n) = 0
        Next

    End Sub

    Private Sub Button46_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

    Dim m As Integer

        Button48.Enabled = True
        Button50.Enabled = True
        Button51.Enabled = True
        Button53.Enabled = True
        Button54.Enabled = True

        Performance_Measure_1(0, 4) = Performance_Measure_1(0, 3)
        Performance_Measure_1(0, 3) = Performance_Measure_1(0, 2)
        Performance_Measure_1(0, 2) = Performance_Measure_1(0, 1)
        Performance_Measure_1(0, 1) = Performance_Measure_1(0, 0)
        Performance_Measure_1(0, 0) = 0

        Learning_Performance_1 = Performance_Measure_1(0, 0) + Performance_Measure_1(0, 1) +
Performance_Measure_1(0, 2) + Performance_Measure_1(0, 3) + Performance_Measure_1(0, 4)

        Number_of_Policies_Performed = Number_of_Policies_Performed + 1
        TextBox112.Text = (Number_of_Policies_Performed).ToString
        TextBox133.Text = TextBox112.Text

    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Trial_Number", TextBox133.Text)

        Button37_Click(sender, e)
        Button51.Enabled = False
        Button50.Enabled = False
        Button53.Enabled = False
        Button47.Enabled = False
    End Sub

    Private Sub Button47_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button47.Click

    Dim m As Integer

        Performance_Measure_1(0, 4) = Performance_Measure_1(0, 3)
        Performance_Measure_1(0, 3) = Performance_Measure_1(0, 2)
        Performance_Measure_1(0, 2) = Performance_Measure_1(0, 1)
        Performance_Measure_1(0, 1) = Performance_Measure_1(0, 0)

        If Val(TextBox132.Text) >= Val(TextBox129.Text) Then
            Performance_Measure_1(0, 0) = 1
```

```
        Else
            Performance_Measure_1(0, 0) = 0
        End If

        Learning_Performance_1 = Performance_Measure_1(0, 0) + Performance_Measure_1(0, 1) +
Performance_Measure_1(0, 2) + Performance_Measure_1(0, 3) + Performance_Measure_1(0, 4)

        Number_of_Policies_Performed = Number_of_Policies_Performed + 1
        TextBox112.Text = (Number_of_Policies_Performed).ToString
        TextBox133.Text = TextBox112.Text

    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Trial_Number", TextBox133.Text)

        If Val(TextBox133.Text) > 0 Then
            Button27_Click(sender, e)
        End If

        Button53.Enabled = True
        Button48.Enabled = False
        Button51.Enabled = False
        Button50.Enabled = False
        Button47.Enabled = False

    End Sub

    Private Sub Button51_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button51.Click
        Button51.Enabled = False
        Button50.Enabled = False
        Button38_Click(sender, e)
        Button53.Enabled = True
        Button47.Enabled = True
        Button55.Enabled = True
        Button47_Click(sender, e)
    End Sub

    Private Sub Button50_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button50.Click
        Button51.Enabled = False
        Button50.Enabled = False
        Button36_Click(sender, e)
        Button53.Enabled = True
        Button47.Enabled = True
        Button55.Enabled = True
        Button47_Click(sender, e)
    End Sub

    Private Sub Button53_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button53.Click
        TextBox144.Text = "Grasping"
        Robot_Operating.Enabled = False
        TextBox114.Text = "0"
        TextBox141.Text = ""

        If allow_sound_flag = 1 Then
            rc = PlaySound(System.AppDomain.CurrentDomain.BaseDirectory & "grasping_a_bag.wav", 0,
SND_NOSTOP)
        End If

        Button31_Click(sender, e)
        Button54.Enabled = True
    End Sub

    Private Sub Button54_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button54.Click
```

```
        Button56.Enabled = True
        Cummulative_Weight_Reward = 0
        Events_Weight_Reward = 0

        Events_Value_vector_String = ""

        counter_1 = 0

        TextBox150.Text = "0"
    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Cummulative_Value", "0")
        pRegKey_Events.SetValue("Events_Value", "0")
        pRegKey_Events.SetValue("Time_Value", "0")
        pRegKey_Events.SetValue("Stop_Robot_Flag", "0")

        Robot_Operating.Enabled = True

        '     System.Threading.Thread.Sleep(250)

        Button57_Click(sender, e)

        If allow_sound_flag = 1 Then
            rc = PlaySound(System.AppDomain.CurrentDomain.BaseDirectory & "shaking_a_bag.wav", 0, SND_NOSTOP)
        End If

        Button28_Click_2(sender, e)
        Button56.Enabled = True

        TextBox114.Text = "0"
        Time_Now_1 = DateTime.Now

        Shaking_Timer_1.Enabled = True

    End Sub

    Private Sub Button55_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button55.Click
        Run_Program("OPEN.JBI")
    End Sub

    Private Sub Button56_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button56.Click
        Button56.Enabled = False
    Dim Wj As Double
        Wj = 0

        TextBox132.Text = ""
        TextBox135.Text = ""

        Events_Weight_Reward = 0

        Robot_Operating.Enabled = False

    Dim Result As String
    Dim Events_Value_vector_String_Splitted As Array
    Dim Cummulative_Value_vector_String_Splitted As Array
    Dim Times_Value_vector_String_Splitted As Array


        'Scale_Output_3_Screws
        'Times_Value_vector_String = "0.25    0.5     0.75    1       1.25    1.5     1.75    2       2.25    2.5
            2.75    3       3.25    3.5     3.75    4       4.25    4.5     4.75    5"
```

```
        'Cummulative_Value_vector_String = "0    0    0    88    0    0    0    0    0
            0    0    0    0    0    7.4    39.9    80.6    112.1    104.6    126.2"

        'Events_Value_vector_String = "0    0    0    6    0    0    0    0    0    0
            0    0    0    0    7.9    28.3    22.4    22.2    22.2    0"


        If ComboBox6.SelectedIndex = 0 Then
            TextBox132.Text = ""

            'Calling m-file from VB
            '  Dim MatLab As Object
            ' MatLab = CreateObject("Matlab.Application")

            Result = MatLab.Execute("cd " + TextBox148.Text)

            """"Tasks to perform:

            Result = MatLab.Execute("peakdetect([" + Events_Value_vector_String + "])")
            Result = Result.Replace(Chr(9), " ") ' Chr(9) = Tab  http://www.lookuptables.com/
            Result = Result.Replace("       ", " ")
            Result = Result.Replace("      ", " ")
            Result = Result.Replace("     ", " ")
            Result = Result.Replace("    ", " ")
            Result = Result.Replace("   ", " ")
            Result = Result.Replace("  ", " ")
            Result = Mid(Result, 10)

            Events_Value_vector_String_Splitted = QuoteSplit(Result, "    ")

            'Here I check if there are no events at the "Events_Value_vector_String" string
            'MsgBox(Mid(Result, 1, 6)) ' The result is "signal"

            If Mid(Result, 1, 6) = "signal" Then
                Events_Weight_Reward = 0
                TextBox132.Text = "0"
                GoTo no_events
            End If

            Write2File(Result, TextBox148.Text + "Result.txt")

            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace(Chr(9), " ") ' Chr(9) = Tab
http://www.lookuptables.com/
            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace("       ", " ")
            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace("      ", " ")
            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace("     ", " ")
            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace("    ", " ")
            Cummulative_Value_vector_String = Cummulative_Value_vector_String.Replace("  ", " ")

            Cummulative_Value_vector_String_Splitted = QuoteSplit(Cummulative_Value_vector_String, "    ")

            '           Write2File(Cummulative_Value_vector_String, TextBox148.Text +
"Cummulative_Value_vector_String.txt")

            Times_Value_vector_String = Times_Value_vector_String.Replace(Chr(9), " ") ' Chr(9) = Tab
http://www.lookuptables.com/
            Times_Value_vector_String = Times_Value_vector_String.Replace("       ", " ")
            Times_Value_vector_String = Times_Value_vector_String.Replace("      ", " ")
            Times_Value_vector_String = Times_Value_vector_String.Replace("     ", " ")
            Times_Value_vector_String = Times_Value_vector_String.Replace("    ", " ")
            Times_Value_vector_String = Times_Value_vector_String.Replace("  ", " ")

            Times_Value_vector_String_Splitted = QuoteSplit(Times_Value_vector_String, "    ")
```

```
            For i As Integer = 0 To (Events_Value_vector_String_Splitted.Length - 1)
                If i > 0 Then
                    Wj = Cummulative_Value_vector_String_Splitted(Events_Value_vector_String_Splitted(i - 1) - 1)

                Else
                    Wj = 0
                End If
                ' Events_Weight_Reward = Events_Weight_Reward +
Round(((Cummulative_Value_vector_String_Splitted(Events_Value_vector_String_Splitted(i) - 1) - Wj) /
Val(TextBox147.Text)), 0) / Times_Value_vector_String_Splitted(Events_Value_vector_String_Splitted(i) - 1)
                Events_Weight_Reward = Events_Weight_Reward +
(((Cummulative_Value_vector_String_Splitted(Events_Value_vector_String_Splitted(i) - 1) - Wj) /
Val(TextBox147.Text))) / Times_Value_vector_String_Splitted(Events_Value_vector_String_Splitted(i) - 1)

            Next
            '  MsgBox(Events_Weight_Reward)

            Events_Weight_Reward = Events_Weight_Reward * Val(TextBox5.Text) '238.585446



            TextBox132.Text = Round(Events_Weight_Reward, 2).ToString

            MsgBox(Events_Weight_Reward)

            If ((Val(TextBox132.Text)) < 5 And (Val(TextBox132.Text)) > -5) Then
                TextBox132.Text = 0
                Events_Weight_Reward = 0
            Else
                TextBox132.Text = Round(Events_Weight_Reward, 2).ToString
            End If

        End If
no_events:
        If ComboBox6.SelectedIndex = 1 Then

            'MsgBox(Cummulative_Value_vector_String_Splitted.Length)
            '''
            '           For i As Integer = 0 To (Cummulative_Value_vector_String_Splitted.Length - 1)
            'Cummulative_Weight_Reward = Cummulative_Weight_Reward +
Cummulative_Value_vector_String_Splitted(i) / Times_Value_vector_String_Splitted(i)
            'Next
            '''

            If ((Cummulative_Weight_Reward) < 5 And (Cummulative_Weight_Reward) > -5) Then
                Cummulative_Weight_Reward = 0
            Else
                TextBox132.Text = Round(Cummulative_Weight_Reward, 2)
            End If

        End If

        If TextBox132.Text = "" Then TextBox132.Text = 0

        TextBox142.Text = TextBox114.Text
        Button48.Enabled = True

        ListBox5.Items.Clear()
        ListBox8.Items.Clear()
        For i As Integer = 0 To (Val(TextBox134.Text) - 1)
            ListBox5.Items.Add(Val(TextBox132.Text))
            ListBox8.Items.Add(Val(TextBox132.Text))
```

```
        Next

        If Val(TextBox132.Text) > Val(TextBox129.Text) Then

            TextBox80.Text = TextBox114.Text

            Average_Successful_Shaking_Policies_Index = Average_Successful_Shaking_Policies_Index + 1
            Average_Successful_Shaking_Policies_Sum = Average_Successful_Shaking_Policies_Sum +
Val(TextBox114.Text) 'Average_Successful_Shaking_Policies(0, i))
            Average_Successful_Shaking_Policies_Final = Average_Successful_Shaking_Policies_Sum /
Average_Successful_Shaking_Policies_Index
            TextBox89.Text = Round(Average_Successful_Shaking_Policies_Final, 2).ToString

        Else
            '   Average_Successful_Shaking_Policies_Final = Average_Successful_Shaking_Policies_Final
        End If

        If Val(TextBox132.Text) > Val(TextBox129.Text) Then
            Number_of_Successful_Policies = Number_of_Successful_Policies + 1
            TextBox135.Text = "Policy was successful!"
        Else
            TextBox135.Text = "Policy failed!"
        End If

        Percent_of_Successful_Policies = Number_of_Successful_Policies / (Val(TextBox133.Text))

        Plot_Graph_2()

        If Average_Successful_Shaking_Policies_Index > 0 Then
            Initial_Plot_Graph_1()
            Plot_Graph_1()
        End If

        Button48_Click(sender, e)

    End Sub

    Private Sub Shaking_Timer_1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Shaking_Timer_1.Tick

    Dim Difference As TimeSpan
    Dim counter_1 As Integer
        Difference = Today.Now.Subtract(Time_Now_1)
        TextBox114.Text = Round(Val(Difference.Duration.TotalSeconds), 2).ToString

        TextBox151.Text = Difference.ToString

    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Time_Value", TextBox114.Text)

    Dim pRegKey1 As RegistryKey = Registry.CurrentUser
        pRegKey1 = pRegKey1.OpenSubKey("Uri\Digital_Scale", True)
    Dim val1 As Object = pRegKey1.GetValue("Stop_Robot_Flag")

    Dim Time_Value As Object = pRegKey1.GetValue("Time_Value")
    Dim Trial_Number As Object = pRegKey1.GetValue("Trial_Number")
    Dim Cummulative_Value As Object = pRegKey1.GetValue("Cummulative_Value")
    Dim Events_Value As Object = pRegKey1.GetValue("Events_Value")

        If ((Val(TextBox114.Text) Mod Val(TextBox149.Text)) = 0) Then
            TextBox150.Text = (Val(TextBox150.Text) + 1).ToString
```

```
            If Val(TextBox133.Text) > 0 Then
                Write2File(TextBox114.Text + ", " + Cummulative_Value + ", " + Events_Value, TextBox148.Text +
(Val(Trial_Number)).ToString + "_Trial_" + "Scale_Output.csv")
            End If

            ' Create vectors for rewards
            ' Times_vector
            Times_vector(0, counter_1) = Val(TextBox114.Text)
            'Times_Value_vector_String()
            Times_Value_vector_String = Times_Value_vector_String + " " + Times_vector(0, counter_1).ToString

            ' Cummulative_Value_vector
            Cummulative_Value_vector(0, counter_1) = Cummulative_Value
            Cummulative_Value_vector_String = Cummulative_Value_vector_String + " " + Cummulative_Value_vector(0,
counter_1).ToString

            ' Events_Value_vector
            Events_Value_vector(0, counter_1) = Events_Value
            Events_Value_vector_String = Events_Value_vector_String + " " + Events_Value_vector(0, counter_1).ToString

            'Cummulative Weight Reward
            Cummulative_Weight_Reward = Cummulative_Weight_Reward + (Cummulative_Value_vector(0, counter_1)) /
Times_vector(0, counter_1)
            counter_1 = counter_1 + 1

        Else
            '  TextBox150.Text = "0"
        End If

        If (Val(val1) = 1) Or ((Difference.Duration.TotalSeconds > 2.0) And (TextBox144.Text = "Idle")) Then
            System.Threading.Thread.Sleep(1000)

            'pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
            'pRegKey_Events.SetValue("Activate_Scale_Flag", "0")

            'Run_Program("OPEN.JBI")
            Button52_Click(sender, e)
            If CheckBox14.Checked = True Then
                Disconnect_Robot()
            End If
        End If

    End Sub

    Public Function isInteger(ByVal v As Object) As Boolean
        isInteger = IIf(VarType(v) = vbInteger, True, False)
    End Function

    Private Sub Button46_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
        TextBox141.Text = TextBox141.Text + (TextBox114.Text).ToString & Chr(13) & Chr(10)
    End Sub

    Private Sub Button52_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        TextBox141.Text = Chr(10) & TextBox141.Text + (TextBox114.Text).ToString & Chr(13) & Chr(10)
        Shaking_Time_1 = 0
        Shaking_Timer_1.Enabled = False
    End Sub

    Private Sub Button57_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Shaking_Timer_1.Enabled = False
        TextBox114.Text = "0"
        TextBox141.Text = ""
    Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
```

```
        pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
        pRegKey_Events.SetValue("Time_Value", TextBox114.Text)
    End Sub

    Private Sub Robot_Operating_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Robot_Operating.Tick
        If Val(TextBox133.Text) > 0 Then
            TextBox143.Text = BscIsPlayMode(nCid)
            If TextBox143.Text = "1" Then
                TextBox144.Text = "Operating"
            Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
                pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
                pRegKey_Events.SetValue("Activate_Scale_Flag", "1")
            Else
                TextBox144.Text = "Idle"
            Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
                pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
                'System.Threading.Thread.Sleep(500)
                pRegKey_Events.SetValue("Activate_Scale_Flag", "0")
            End If
        End If
    End Sub

    Private Sub Write2File(ByVal msg As String, ByVal filePath As String)
    Dim fs As FileStream = New FileStream(filePath, FileMode.Append, FileAccess.Write)
    Dim sw As StreamWriter = New StreamWriter(fs)
        sw.WriteLine(msg)
        sw.Flush()
        sw.Close()
        fs.Close()
    End Sub

    Public Function QuoteSplit(ByVal str As String, Optional ByVal splitChar As Char = ","c, Optional ByVal QuoteChar
As Char = """"c) As String()
        'Use double-quotes to escape the quote character. Example: Hello ""John"" will produce Hello "John"
    Dim quoteOpened As Boolean = False
    Dim al As New ArrayList
    Dim curStr As New System.Text.StringBuilder
        For i As Integer = 0 To str.Length - 1
        Dim c As Char = CChar(str.Substring(i, 1))
        Dim nextChar As String = "" ' Cannot use Char because it is a value type and cannot contain Nothing or empty
string
            If str.Length > (i + 1) Then nextChar = str.Substring(i + 1, 1)
            If quoteOpened Then
                'Look for ending quote character
                If (Not c = QuoteChar) Then
                    curStr.Append(c)
                ElseIf c = QuoteChar AndAlso Not nextChar = "" AndAlso nextChar = QuoteChar Then
                    curStr.Append(QuoteChar)
                    i += 1
                ElseIf c = QuoteChar Then
                    quoteOpened = False 'Clear
                End If
            Else 'If Not quoteOpened
                If c = splitChar Then
                    al.Add(curStr.ToString) 'Add to arraylist
                    curStr.Length = 0 'Clear current string
                ElseIf c = QuoteChar Then
                    quoteOpened = True
                    curStr.Length = 0 'Clear the current string, so if we have something like: , "Hello World" the result is "Hello
World" instead of " Hello World"
                Else
                    curStr.Append(c)
```

```
            End If
        End If
    Next
    al.Add(curStr.ToString) 'Add to arraylist
    curStr.Length = 0 'Clear current string

    Return CType(al.ToArray(GetType(String)), String())
  End Function

  Private Sub Button58_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
  Dim array1 As Array
  Dim string1 As String

    string1 = "10 12 14  145656   4    6           5         2"

    string1 = string1.Replace("      ", " ")
    string1 = string1.Replace("     ", " ")
    string1 = string1.Replace("    ", " ")
    string1 = string1.Replace("   ", " ")
    string1 = string1.Replace("  ", " ")

    MsgBox(string1)

    array1 = QuoteSplit(string1, " ")

    MsgBox(array1(0))
    MsgBox(array1(1))
    MsgBox(array1(2))
    MsgBox(array1(3))
    MsgBox(array1(4))
    MsgBox(array1(5))
    MsgBox(array1(6))
    MsgBox(array1(7))

  End Sub

  Private Sub ComboBox7_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox7.SelectedIndexChanged
    If ComboBox7.SelectedItem = "Optimal Policy1" Then
      Chosen_Best_Policy = "Best_Policy_1"
    End If

    If ComboBox7.SelectedItem = "Optimal Policy2" Then
      Chosen_Best_Policy = "Best_Policy_2"
    End If

    If ComboBox7.SelectedItem = "Optimal Policy3" Then
      Chosen_Best_Policy = "Best_Policy_3"
    End If

    If ComboBox7.SelectedItem = "Policy After System Crash" Then
      Chosen_Best_Policy = "Policy_After_System_Crash"
      Button49.Enabled = False
      MatLab = CreateObject("Matlab.Application")
    End If

  End Sub

  Private Sub Button59_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button59.Click
    Length_of_Best_Policy = 0
    State_Action_Timer1_Best_Counter_1 = 0
    ListBox3.Items.Clear()
    ListBox4.Items.Clear()
```

```
        xConn = New sqlConn
        xConn.connectMe("SELECT * FROM " & Chosen_Best_Policy)

        For iCounter = 1 To xConn.getData("State_Q").Count - 1
            ListBox1.Items.Add(xConn.dataReturned.Item(iCounter))
        Next

        Length_of_Best_Policy = ListBox1.Items.Count

        ListBox1.Items.Clear()
        xConn.OLEConn.Close()

        For i As Integer = 0 To Length_of_Best_Policy - 1

            xConn = New sqlConn

            xConn.connectMe("SELECT State_Q FROM " & Chosen_Best_Policy & " Where Id=" & i)

            For iCounter = 0 To xConn.getData("State_Q").Count - 1
                ListBox3.Items.Add(xConn.dataReturned.Item(iCounter))
            Next
            xConn.OLEConn.Close()

            xConn = New sqlConn
            xConn.connectMe("SELECT Action_Q FROM Best_Policy_1 Where Id=" & i)
            For iCounter = 0 To xConn.getData("Action_Q").Count - 1
                ListBox4.Items.Add(xConn.dataReturned.Item(iCounter))
            Next

            xConn.OLEConn.Close()

        Next
        Reset_Policy_1()
        State_Action_Best_Timer1.Enabled = True

    End Sub

    Private Sub State_Action_Best_Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles State_Action_Best_Timer1.Tick

        ListBox1.SetSelected(0, True)
        ListBox1_DoubleClick(sender, e)

        ListBox2.SetSelected(ListBox4.Items.Item(State_Action_Timer1_Best_Counter_1), True)
        ListBox2_DoubleClick(sender, e)

        State_Action_Timer1_Best_Counter_1 = State_Action_Timer1_Best_Counter_1 + 1

        If State_Action_Timer1_Best_Counter_1 >= Length_of_Best_Policy Then  'Val(TextBox134.Text)

            State_Action_Best_Timer1.Enabled = False

        End If

    End Sub

    Private Sub CheckBox9_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox9.CheckedChanged
        If CheckBox9.Checked = True Then
            allow_sound_flag = 1
        Else
            allow_sound_flag = 0
```

```
        End If
    End Sub

    Private Sub ComboBox9_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox9.SelectedIndexChanged
        If ComboBox9.SelectedIndex = 0 Then Relative_Axis_Speed_X = Relative_Axis_Speed_X + 500
        If ComboBox9.SelectedIndex = 1 Then Relative_Axis_Speed_X = Relative_Axis_Speed_X + 100
        If ComboBox9.SelectedIndex = 2 Then Relative_Axis_Speed_X = Relative_Axis_Speed_X
        If ComboBox9.SelectedIndex = 3 Then Relative_Axis_Speed_X = Relative_Axis_Speed_X - 100
        If ComboBox9.SelectedIndex = 4 Then Relative_Axis_Speed_X = Relative_Axis_Speed_X - 500

        If Relative_Axis_Speed_X >= 1500 Then Relative_Axis_Speed_X = 1500
        If Relative_Axis_Speed_X <= 100 Then Relative_Axis_Speed_X = 100

        TextBox136.Text = Relative_Axis_Speed_X.ToString
        TextBox153.Text = TextBox136.Text
    End Sub

    Private Sub ComboBox8_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox8.SelectedIndexChanged
        If ComboBox8.SelectedIndex = 0 Then Relative_Axis_Speed_Y = Relative_Axis_Speed_Y + 500
        If ComboBox8.SelectedIndex = 1 Then Relative_Axis_Speed_Y = Relative_Axis_Speed_Y + 100
        If ComboBox8.SelectedIndex = 2 Then Relative_Axis_Speed_Y = Relative_Axis_Speed_Y
        If ComboBox8.SelectedIndex = 3 Then Relative_Axis_Speed_Y = Relative_Axis_Speed_Y - 100
        If ComboBox8.SelectedIndex = 4 Then Relative_Axis_Speed_Y = Relative_Axis_Speed_Y - 500

        If Relative_Axis_Speed_Y >= 1500 Then Relative_Axis_Speed_Y = 1500
        If Relative_Axis_Speed_Y <= 100 Then Relative_Axis_Speed_Y = 100

        TextBox137.Text = Relative_Axis_Speed_Y.ToString
        TextBox154.Text = TextBox137.Text
    End Sub

    Private Sub ComboBox10_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox10.SelectedIndexChanged
        If ComboBox10.SelectedIndex = 0 Then Relative_Axis_Speed_Z = Relative_Axis_Speed_Z + 500
        If ComboBox10.SelectedIndex = 1 Then Relative_Axis_Speed_Z = Relative_Axis_Speed_Z + 100
        If ComboBox10.SelectedIndex = 2 Then Relative_Axis_Speed_Z = Relative_Axis_Speed_Z
        If ComboBox10.SelectedIndex = 3 Then Relative_Axis_Speed_Z = Relative_Axis_Speed_Z - 100
        If ComboBox10.SelectedIndex = 4 Then Relative_Axis_Speed_Z = Relative_Axis_Speed_Z - 500

        If Relative_Axis_Speed_Z >= 1500 Then Relative_Axis_Speed_Z = 1500
        If Relative_Axis_Speed_Z <= 100 Then Relative_Axis_Speed_Z = 100

        TextBox90.Text = Relative_Axis_Speed_Z.ToString
        TextBox155.Text = TextBox90.Text
    End Sub

    Private Sub ComboBox11_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox11.SelectedIndexChanged
        If ComboBox11.SelectedIndex = 0 Then Relative_Axis_Amplitude_X = Relative_Axis_Amplitude_X + 15
        If ComboBox11.SelectedIndex = 1 Then Relative_Axis_Amplitude_X = Relative_Axis_Amplitude_X + 5
        If ComboBox11.SelectedIndex = 2 Then Relative_Axis_Amplitude_X = Relative_Axis_Amplitude_X
        If ComboBox11.SelectedIndex = 3 Then Relative_Axis_Amplitude_X = Relative_Axis_Amplitude_X - 5
        If ComboBox11.SelectedIndex = 4 Then Relative_Axis_Amplitude_X = Relative_Axis_Amplitude_X - 15

        If Relative_Axis_Amplitude_X >= 50 Then Relative_Axis_Amplitude_X = 50
        If Relative_Axis_Amplitude_X <= 10 Then Relative_Axis_Amplitude_X = 10

        TextBox140.Text = Relative_Axis_Amplitude_X.ToString

        TextBox81.Text = TextBox140.Text
        TextBox159.Text = "-" + TextBox140.Text
```

```
    End Sub

    Private Sub ComboBox13_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox13.SelectedIndexChanged
        If ComboBox13.SelectedIndex = 0 Then Relative_Axis_Amplitude_Y = Relative_Axis_Amplitude_Y + 15
        If ComboBox13.SelectedIndex = 1 Then Relative_Axis_Amplitude_Y = Relative_Axis_Amplitude_Y + 5
        If ComboBox13.SelectedIndex = 2 Then Relative_Axis_Amplitude_Y = Relative_Axis_Amplitude_Y
        If ComboBox13.SelectedIndex = 3 Then Relative_Axis_Amplitude_Y = Relative_Axis_Amplitude_Y - 5
        If ComboBox13.SelectedIndex = 4 Then Relative_Axis_Amplitude_Y = Relative_Axis_Amplitude_Y - 15

        If Relative_Axis_Amplitude_Y >= 50 Then Relative_Axis_Amplitude_Y = 50
        If Relative_Axis_Amplitude_Y <= 10 Then Relative_Axis_Amplitude_Y = 10

        TextBox139.Text = Relative_Axis_Amplitude_Y.ToString

        TextBox156.Text = TextBox139.Text
        TextBox160.Text = "-" + TextBox139.Text

    End Sub

    Private Sub ComboBox12_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox12.SelectedIndexChanged
        If ComboBox12.SelectedIndex = 0 Then Relative_Axis_Amplitude_Z = Relative_Axis_Amplitude_Z + 15
        If ComboBox12.SelectedIndex = 1 Then Relative_Axis_Amplitude_Z = Relative_Axis_Amplitude_Z + 5
        If ComboBox12.SelectedIndex = 2 Then Relative_Axis_Amplitude_Z = Relative_Axis_Amplitude_Z
        If ComboBox12.SelectedIndex = 3 Then Relative_Axis_Amplitude_Z = Relative_Axis_Amplitude_Z - 5
        If ComboBox12.SelectedIndex = 4 Then Relative_Axis_Amplitude_Z = Relative_Axis_Amplitude_Z - 15

        If Relative_Axis_Amplitude_Z >= 50 Then Relative_Axis_Amplitude_Z = 50
        If Relative_Axis_Amplitude_Z <= 10 Then Relative_Axis_Amplitude_Z = 10

        TextBox138.Text = Relative_Axis_Amplitude_Z.ToString

        TextBox157.Text = TextBox138.Text
        TextBox161.Text = "-" + TextBox138.Text

    End Sub

    Private Sub CheckBox11_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox11.CheckedChanged
        If CheckBox11.Checked = True Then
            TextBox81.Text = "0"
            TextBox159.Text = "0"
            ComboBox11.Enabled = False
            TextBox140.Enabled = False
            ComboBox9.Enabled = False
            TextBox136.Enabled = False
            Axis_Allowed_Counter = Axis_Allowed_Counter + 1
        Else
            TextBox81.Text = TextBox140.Text
            TextBox159.Text = "-" + TextBox140.Text
            ComboBox11.Enabled = True
            TextBox140.Enabled = True
            ComboBox9.Enabled = True
            TextBox136.Enabled = True
            Axis_Allowed_Counter = Axis_Allowed_Counter - 1
        End If

        If Axis_Allowed_Counter > 2 Then CheckBox11.Checked = False

    End Sub
```

```
    Private Sub CheckBox7_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox7.CheckedChanged
        If CheckBox7.Checked = True Then
            TextBox156.Text = "0"
            TextBox160.Text = "0"
            ComboBox13.Enabled = False
            TextBox139.Enabled = False
            ComboBox8.Enabled = False
            TextBox137.Enabled = False
            Axis_Allowed_Counter = Axis_Allowed_Counter + 1
        Else
            TextBox156.Text = TextBox139.Text
            TextBox160.Text = "-" + TextBox139.Text
            ComboBox13.Enabled = True
            TextBox137.Enabled = True
            ComboBox8.Enabled = True
            TextBox137.Enabled = True
            Axis_Allowed_Counter = Axis_Allowed_Counter - 1
        End If

        If Axis_Allowed_Counter > 2 Then CheckBox7.Checked = False

    End Sub

    Private Sub CheckBox10_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CheckBox10.CheckedChanged
        If CheckBox10.Checked = True Then
            TextBox157.Text = "0"
            TextBox161.Text = "0"
            ComboBox12.Enabled = False
            TextBox138.Enabled = False
            ComboBox10.Enabled = False
            TextBox90.Enabled = False
            Axis_Allowed_Counter = Axis_Allowed_Counter + 1
        Else
            TextBox157.Text = TextBox138.Text
            TextBox161.Text = "-" + TextBox138.Text
            ComboBox12.Enabled = True
            TextBox138.Enabled = True
            ComboBox10.Enabled = True
            TextBox90.Enabled = True
            Axis_Allowed_Counter = Axis_Allowed_Counter - 1
        End If

        If Axis_Allowed_Counter > 2 Then CheckBox10.Checked = False

    End Sub

    Function Plot_Graph_2()

    Dim oSeries As MSChart20Lib.Series

        With AxMSChart3
            .ColumnLabel = "C2"
            .Column = 2
            .RowCount = 50

            If .Row <= 49 Then
                .Repaint = True
                .Data = Percent_of_Successful_Policies * 100
                If .Data = 0 Then .Data = 3
                With .Plot.SeriesCollection(2).DataPoints(-1)
                    .Brush.FillColor.Set(0, 0, 255)
```

```
            End With
            .Row = .Row + 1

        Else
            .Repaint = True
            .Data = Percent_of_Successful_Policies * 100
            If .Data = 0 Then .Data = 3
            With .Plot.SeriesCollection(2).DataPoints(-1)
                .Brush.FillColor.Set(0, 0, 255)
            End With

            GoTo exit1

        End If

exit1:
        TextBox145.Text = (Round(Percent_of_Successful_Policies * 100, 2)).ToString

    End With

  End Function

  Function Initial_Plot_Graph_2()
    With AxMSChart3
    Dim oSeries As MSChart20Lib.Series
        .RowCount = 50

        .chartType = AxMSChart3.chartType.VtChChartType2dStep
        .ColumnCount = 2
        .ColumnLabel = "C1"
        .Column = 1

        With .Plot.SeriesCollection(1).DataPoints(-1)
            .Brush.FillColor.Set(0, 0, 0)
        End With

        For i As Integer = 1 To 50
            .Row = i
            .Data = 0
        Next

        For Each oSeries In AxMSChart3.Plot.SeriesCollection
            oSeries.Pen.Width = 0
        Next oSeries

        .Row = 1
        .Repaint = True
    End With

  End Function

  Function Initial_Plot_Graph_1()
    With AxMSChart1
    Dim oSeries As MSChart20Lib.Series
        .RowCount = 50
        .chartType = AxMSChart1.chartType.VtChChartType2dArea
        .ColumnCount = 2
        .ColumnLabel = "C1"
        .Column = 1

        With .Plot.SeriesCollection(1).DataPoints(-1)
            .Brush.FillColor.Set(0, 0, 0)
        End With
```

```
        For Each oSeries In AxMSChart1.Plot.SeriesCollection
            oSeries.Pen.Width = 0 '50
        Next oSeries

        For i As Integer = 1 To 50
            .Row = i
            .Data = 0
        Next

        .ColumnLabel = "C2"
        .Column = 2

        With .Plot.SeriesCollection(2).DataPoints(-1)
            .Brush.FillColor.Set(0, 0, 0)
        End With

        For Each oSeries In AxMSChart1.Plot.SeriesCollection
            oSeries.Pen.Width = 0 '50
        Next oSeries

        For i As Integer = 1 To 50
            .Row = i
            .Data = 0
        Next

        .Row = 1
        .Repaint = True
    End With

End Function

Function Plot_Graph_1()
Dim oSeries As MSChart20Lib.Series

    With AxMSChart1
        .ColumnCount = 2
        .RowCount = 50

        ' Time for a sucessful policy
        .Column = 1
        .ColumnLabel = "C1"

        .Row = temp_row_1

        If .Row <= 49 Then
            .Repaint = True
            .Data = Val(TextBox114.Text) 'aaa
            With .Plot.SeriesCollection(1).DataPoints(-1)
                .Brush.FillColor.Set(255, 0, 0)
            End With
            .Row = .Row + 1
            temp_row_1 = .Row
        Else
            .Repaint = True
            .Data = Val(TextBox114.Text)
            With .Plot.SeriesCollection(2).DataPoints(-1)
                .Brush.FillColor.Set(255, 0, 0)
            End With
        End If

        .Repaint = True
```

```
        ' Plot threshold
        .Column = 2
        .ColumnLabel = "C2"
        .Row = 1

        With .Plot.SeriesCollection(2).DataPoints(-1)
          .Brush.FillColor.Set(255, 255, 0)
        End With

        For Each oSeries In AxMSChart1.Plot.SeriesCollection
          oSeries.Pen.Width = 0
        Next oSeries

        For i As Integer = 1 To 50
          .Row = i
          .Data = Val(TextBox89.Text) '5
        Next

        .Repaint = True
      End With

  End Function

  Private Sub Button42_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button42.Click
      aaa = Round(Randomizer(0, 1), 2) * 12
      Initial_Plot_Graph_1()
      Plot_Graph_1()
  End Sub

  Private Sub Button52_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button52.Click
      TextBox141.Text = Chr(10) & TextBox141.Text + (TextBox114.Text).ToString & Chr(13) & Chr(10)
      Shaking_Time_1 = 0
      Shaking_Timer_1.Enabled = False
  End Sub

  Private Sub Button57_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button57.Click
      Shaking_Timer_1.Enabled = False
      TextBox114.Text = "0"
      TextBox141.Text = ""
  Dim pRegKey_Events As RegistryKey = Registry.CurrentUser
      pRegKey_Events = pRegKey_Events.OpenSubKey("Uri\Digital_Scale", True)
      pRegKey_Events.SetValue("Time_Value", TextBox114.Text)
  End Sub

  Private Sub ComboBox3_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox3.SelectedIndexChanged
      If ComboBox3.SelectedIndex = 0 Then
        Q_Table_Rewarded(0, 0) = Q_Table_Rewarded(0, 0) * (1.5)
        Q_Table_Rewarded(0, 1) = Q_Table_Rewarded(0, 1) * (1.5)
        Q_Table_Rewarded(0, 2) = Q_Table_Rewarded(0, 2) * (1.5)
        Q_Table_Rewarded(0, 3) = Q_Table_Rewarded(0, 3) * (1.5)
        Q_Table_Rewarded(0, 4) = Q_Table_Rewarded(0, 4) * (1.5)
        Q_Table_Rewarded(0, 5) = Q_Table_Rewarded(0, 5) * (1.5)
      End If

      If ComboBox3.SelectedIndex = 1 Then
        Q_Table_Rewarded(0, 0) = Q_Table_Rewarded(0, 0) * (1.1)
        Q_Table_Rewarded(0, 1) = Q_Table_Rewarded(0, 1) * (1.1)
        Q_Table_Rewarded(0, 2) = Q_Table_Rewarded(0, 2) * (1.1)
        Q_Table_Rewarded(0, 3) = Q_Table_Rewarded(0, 3) * (1.1)
```

```
            Q_Table_Rewarded(0, 4) = Q_Table_Rewarded(0, 4) * (1.1)
            Q_Table_Rewarded(0, 5) = Q_Table_Rewarded(0, 5) * (1.1)
        End If

        If ComboBox3.SelectedIndex = 2 Then
            Q_Table_Rewarded(0, 0) = Q_Table_Rewarded(0, 0) * (1)
            Q_Table_Rewarded(0, 1) = Q_Table_Rewarded(0, 1) * (1)
            Q_Table_Rewarded(0, 2) = Q_Table_Rewarded(0, 2) * (1)
            Q_Table_Rewarded(0, 3) = Q_Table_Rewarded(0, 3) * (1)
            Q_Table_Rewarded(0, 4) = Q_Table_Rewarded(0, 4) * (1)
            Q_Table_Rewarded(0, 5) = Q_Table_Rewarded(0, 5) * (1)
        End If

        If ComboBox3.SelectedIndex = 3 Then
            Q_Table_Rewarded(0, 0) = Q_Table_Rewarded(0, 0) * (0.9)
            Q_Table_Rewarded(0, 1) = Q_Table_Rewarded(0, 1) * (0.9)
            Q_Table_Rewarded(0, 2) = Q_Table_Rewarded(0, 2) * (0.9)
            Q_Table_Rewarded(0, 3) = Q_Table_Rewarded(0, 3) * (0.9)
            Q_Table_Rewarded(0, 4) = Q_Table_Rewarded(0, 4) * (0.9)
            Q_Table_Rewarded(0, 5) = Q_Table_Rewarded(0, 5) * (0.9)
        End If

        If ComboBox3.SelectedIndex = 4 Then
            Q_Table_Rewarded(0, 0) = Q_Table_Rewarded(0, 0) * (0.5)
            Q_Table_Rewarded(0, 1) = Q_Table_Rewarded(0, 1) * (0.5)
            Q_Table_Rewarded(0, 2) = Q_Table_Rewarded(0, 2) * (0.5)
            Q_Table_Rewarded(0, 3) = Q_Table_Rewarded(0, 3) * (0.5)
            Q_Table_Rewarded(0, 4) = Q_Table_Rewarded(0, 4) * (0.5)
            Q_Table_Rewarded(0, 5) = Q_Table_Rewarded(0, 5) * (0.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    End Sub

    Private Sub ComboBox4_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox4.SelectedIndexChanged
        If ComboBox4.SelectedIndex = 0 Then
            Q_Table_Rewarded(0, 6) = Q_Table_Rewarded(0, 6) * (1.5)
            Q_Table_Rewarded(0, 7) = Q_Table_Rewarded(0, 7) * (1.5)
            Q_Table_Rewarded(0, 8) = Q_Table_Rewarded(0, 8) * (1.5)
            Q_Table_Rewarded(0, 9) = Q_Table_Rewarded(0, 9) * (1.5)
            Q_Table_Rewarded(0, 10) = Q_Table_Rewarded(0, 10) * (1.5)
            Q_Table_Rewarded(0, 11) = Q_Table_Rewarded(0, 11) * (1.5)
        End If

        If ComboBox4.SelectedIndex = 1 Then
            Q_Table_Rewarded(0, 6) = Q_Table_Rewarded(0, 6) * (1.1)
            Q_Table_Rewarded(0, 7) = Q_Table_Rewarded(0, 7) * (1.1)
            Q_Table_Rewarded(0, 8) = Q_Table_Rewarded(0, 8) * (1.1)
            Q_Table_Rewarded(0, 9) = Q_Table_Rewarded(0, 9) * (1.1)
            Q_Table_Rewarded(0, 10) = Q_Table_Rewarded(0, 10) * (1.1)
            Q_Table_Rewarded(0, 11) = Q_Table_Rewarded(0, 11) * (1.1)
        End If

        If ComboBox4.SelectedIndex = 2 Then
            Q_Table_Rewarded(0, 6) = Q_Table_Rewarded(0, 6) * (1)
            Q_Table_Rewarded(0, 7) = Q_Table_Rewarded(0, 7) * (1)
            Q_Table_Rewarded(0, 8) = Q_Table_Rewarded(0, 8) * (1)
            Q_Table_Rewarded(0, 9) = Q_Table_Rewarded(0, 9) * (1)
```

```
            Q_Table_Rewarded(0, 10) = Q_Table_Rewarded(0, 10) * (1)
            Q_Table_Rewarded(0, 11) = Q_Table_Rewarded(0, 11) * (1)
        End If

        If ComboBox4.SelectedIndex = 3 Then
            Q_Table_Rewarded(0, 6) = Q_Table_Rewarded(0, 6) * (0.9)
            Q_Table_Rewarded(0, 7) = Q_Table_Rewarded(0, 7) * (0.9)
            Q_Table_Rewarded(0, 8) = Q_Table_Rewarded(0, 8) * (0.9)
            Q_Table_Rewarded(0, 9) = Q_Table_Rewarded(0, 9) * (0.9)
            Q_Table_Rewarded(0, 10) = Q_Table_Rewarded(0, 10) * (0.9)
            Q_Table_Rewarded(0, 11) = Q_Table_Rewarded(0, 11) * (0.9)
        End If

        If ComboBox4.SelectedIndex = 4 Then
            Q_Table_Rewarded(0, 6) = Q_Table_Rewarded(0, 6) * (0.5)
            Q_Table_Rewarded(0, 7) = Q_Table_Rewarded(0, 7) * (0.5)
            Q_Table_Rewarded(0, 8) = Q_Table_Rewarded(0, 8) * (0.5)
            Q_Table_Rewarded(0, 9) = Q_Table_Rewarded(0, 9) * (0.5)
            Q_Table_Rewarded(0, 10) = Q_Table_Rewarded(0, 10) * (0.5)
            Q_Table_Rewarded(0, 11) = Q_Table_Rewarded(0, 11) * (0.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    End Sub

    Private Sub ComboBox5_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox5.SelectedIndexChanged
        If ComboBox5.SelectedIndex = 0 Then
            Q_Table_Rewarded(0, 12) = Q_Table_Rewarded(0, 12) * (1.5)
            Q_Table_Rewarded(0, 13) = Q_Table_Rewarded(0, 13) * (1.5)
            Q_Table_Rewarded(0, 14) = Q_Table_Rewarded(0, 14) * (1.5)
            Q_Table_Rewarded(0, 15) = Q_Table_Rewarded(0, 15) * (1.5)
            Q_Table_Rewarded(0, 16) = Q_Table_Rewarded(0, 16) * (1.5)
            Q_Table_Rewarded(0, 17) = Q_Table_Rewarded(0, 17) * (1.5)
        End If

        If ComboBox5.SelectedIndex = 1 Then
            Q_Table_Rewarded(0, 12) = Q_Table_Rewarded(0, 12) * (1.1)
            Q_Table_Rewarded(0, 13) = Q_Table_Rewarded(0, 13) * (1.1)
            Q_Table_Rewarded(0, 14) = Q_Table_Rewarded(0, 14) * (1.1)
            Q_Table_Rewarded(0, 15) = Q_Table_Rewarded(0, 15) * (1.1)
            Q_Table_Rewarded(0, 16) = Q_Table_Rewarded(0, 16) * (1.1)
            Q_Table_Rewarded(0, 17) = Q_Table_Rewarded(0, 17) * (1.1)
        End If

        If ComboBox5.SelectedIndex = 2 Then
            Q_Table_Rewarded(0, 12) = Q_Table_Rewarded(0, 12) * (1)
            Q_Table_Rewarded(0, 13) = Q_Table_Rewarded(0, 13) * (1)
            Q_Table_Rewarded(0, 14) = Q_Table_Rewarded(0, 14) * (1)
            Q_Table_Rewarded(0, 15) = Q_Table_Rewarded(0, 15) * (1)
            Q_Table_Rewarded(0, 16) = Q_Table_Rewarded(0, 16) * (1)
            Q_Table_Rewarded(0, 17) = Q_Table_Rewarded(0, 17) * (1)
        End If

        If ComboBox5.SelectedIndex = 3 Then
            Q_Table_Rewarded(0, 12) = Q_Table_Rewarded(0, 12) * (0.9)
            Q_Table_Rewarded(0, 13) = Q_Table_Rewarded(0, 13) * (0.9)
            Q_Table_Rewarded(0, 14) = Q_Table_Rewarded(0, 14) * (0.9)
            Q_Table_Rewarded(0, 15) = Q_Table_Rewarded(0, 15) * (0.9)
```

```
            Q_Table_Rewarded(0, 16) = Q_Table_Rewarded(0, 16) * (0.9)
            Q_Table_Rewarded(0, 17) = Q_Table_Rewarded(0, 17) * (0.9)
        End If

        If ComboBox5.SelectedIndex = 4 Then
            Q_Table_Rewarded(0, 12) = Q_Table_Rewarded(0, 12) * (0.5)
            Q_Table_Rewarded(0, 13) = Q_Table_Rewarded(0, 13) * (0.5)
            Q_Table_Rewarded(0, 14) = Q_Table_Rewarded(0, 14) * (0.5)
            Q_Table_Rewarded(0, 15) = Q_Table_Rewarded(0, 15) * (0.5)
            Q_Table_Rewarded(0, 16) = Q_Table_Rewarded(0, 16) * (0.5)
            Q_Table_Rewarded(0, 17) = Q_Table_Rewarded(0, 17) * (0.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    End Sub

    Private Sub ComboBox14_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox14.SelectedIndexChanged
        If ComboBox14.SelectedIndex = 0 Then
            Q_Table_Rewarded(3, 4) = Q_Table_Rewarded(3, 4) * (1.5)
            Q_Table_Rewarded(6, 2) = Q_Table_Rewarded(6, 2) * (1.5)
            Q_Table_Rewarded(2, 3) = Q_Table_Rewarded(2, 3) * (1.5)
            Q_Table_Rewarded(5, 1) = Q_Table_Rewarded(5, 1) * (1.5)
            Q_Table_Rewarded(1, 2) = Q_Table_Rewarded(1, 2) * (1.5)
            Q_Table_Rewarded(4, 0) = Q_Table_Rewarded(4, 0) * (1.5)
        End If

        If ComboBox14.SelectedIndex = 1 Then
            Q_Table_Rewarded(3, 4) = Q_Table_Rewarded(3, 4) * (1.1)
            Q_Table_Rewarded(6, 2) = Q_Table_Rewarded(6, 2) * (1.1)
            Q_Table_Rewarded(2, 3) = Q_Table_Rewarded(2, 3) * (1.1)
            Q_Table_Rewarded(5, 1) = Q_Table_Rewarded(5, 1) * (1.1)
            Q_Table_Rewarded(1, 2) = Q_Table_Rewarded(1, 2) * (1.1)
            Q_Table_Rewarded(4, 0) = Q_Table_Rewarded(4, 0) * (1.1)
        End If

        If ComboBox14.SelectedIndex = 2 Then
            Q_Table_Rewarded(3, 4) = Q_Table_Rewarded(3, 4) * (1)
            Q_Table_Rewarded(6, 2) = Q_Table_Rewarded(6, 2) * (1)
            Q_Table_Rewarded(2, 3) = Q_Table_Rewarded(2, 3) * (1)
            Q_Table_Rewarded(5, 1) = Q_Table_Rewarded(5, 1) * (1)
            Q_Table_Rewarded(1, 2) = Q_Table_Rewarded(1, 2) * (1)
            Q_Table_Rewarded(4, 0) = Q_Table_Rewarded(4, 0) * (1)
        End If

        If ComboBox14.SelectedIndex = 3 Then
            Q_Table_Rewarded(3, 4) = Q_Table_Rewarded(3, 4) * (0.9)
            Q_Table_Rewarded(6, 2) = Q_Table_Rewarded(6, 2) * (0.9)
            Q_Table_Rewarded(2, 3) = Q_Table_Rewarded(2, 3) * (0.9)
            Q_Table_Rewarded(5, 1) = Q_Table_Rewarded(5, 1) * (0.9)
            Q_Table_Rewarded(1, 2) = Q_Table_Rewarded(1, 2) * (0.9)
            Q_Table_Rewarded(4, 0) = Q_Table_Rewarded(4, 0) * (0.9)
        End If

        If ComboBox14.SelectedIndex = 4 Then
            Q_Table_Rewarded(3, 4) = Q_Table_Rewarded(3, 4) * (0.5)
            Q_Table_Rewarded(6, 2) = Q_Table_Rewarded(6, 2) * (0.5)
            Q_Table_Rewarded(2, 3) = Q_Table_Rewarded(2, 3) * (0.5)
            Q_Table_Rewarded(5, 1) = Q_Table_Rewarded(5, 1) * (0.5)
```

```
            Q_Table_Rewarded(1, 2) = Q_Table_Rewarded(1, 2) * (0.5)
            Q_Table_Rewarded(4, 0) = Q_Table_Rewarded(4, 0) * (0.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    End Sub

    Private Sub ComboBox15_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox15.SelectedIndexChanged
        If ComboBox15.SelectedIndex = 0 Then
            Q_Table_Rewarded(9, 4) = Q_Table_Rewarded(9, 4) * (1.5)
            Q_Table_Rewarded(12, 4) = Q_Table_Rewarded(12, 4) * (1.5)
            Q_Table_Rewarded(8, 3) = Q_Table_Rewarded(8, 3) * (1.5)
            Q_Table_Rewarded(11, 3) = Q_Table_Rewarded(11, 3) * (1.5)
            Q_Table_Rewarded(7, 2) = Q_Table_Rewarded(7, 2) * (1.5)
            Q_Table_Rewarded(10, 2) = Q_Table_Rewarded(10, 2) * (1.5)
        End If

        If ComboBox15.SelectedIndex = 1 Then
            Q_Table_Rewarded(9, 4) = Q_Table_Rewarded(9, 4) * (1.1)
            Q_Table_Rewarded(12, 4) = Q_Table_Rewarded(12, 4) * (1.1)
            Q_Table_Rewarded(8, 3) = Q_Table_Rewarded(8, 3) * (1.1)
            Q_Table_Rewarded(11, 3) = Q_Table_Rewarded(11, 3) * (1.1)
            Q_Table_Rewarded(7, 2) = Q_Table_Rewarded(7, 2) * (1.1)
            Q_Table_Rewarded(10, 2) = Q_Table_Rewarded(10, 2) * (1.1)
        End If

        If ComboBox15.SelectedIndex = 2 Then
            Q_Table_Rewarded(9, 4) = Q_Table_Rewarded(9, 4) * (1)
            Q_Table_Rewarded(12, 4) = Q_Table_Rewarded(12, 4) * (1)
            Q_Table_Rewarded(8, 3) = Q_Table_Rewarded(8, 3) * (1)
            Q_Table_Rewarded(11, 3) = Q_Table_Rewarded(11, 3) * (1)
            Q_Table_Rewarded(7, 2) = Q_Table_Rewarded(7, 2) * (1)
            Q_Table_Rewarded(10, 2) = Q_Table_Rewarded(10, 2) * (1)
        End If

        If ComboBox15.SelectedIndex = 3 Then
            Q_Table_Rewarded(9, 4) = Q_Table_Rewarded(9, 4) * (0.9)
            Q_Table_Rewarded(12, 4) = Q_Table_Rewarded(12, 4) * (0.9)
            Q_Table_Rewarded(8, 3) = Q_Table_Rewarded(8, 3) * (0.9)
            Q_Table_Rewarded(11, 3) = Q_Table_Rewarded(11, 3) * (0.9)
            Q_Table_Rewarded(7, 2) = Q_Table_Rewarded(7, 2) * (0.9)
            Q_Table_Rewarded(10, 2) = Q_Table_Rewarded(10, 2) * (0.9)
        End If

        If ComboBox15.SelectedIndex = 4 Then
            Q_Table_Rewarded(9, 4) = Q_Table_Rewarded(9, 4) * (1.5)
            Q_Table_Rewarded(12, 4) = Q_Table_Rewarded(12, 4) * (1.5)
            Q_Table_Rewarded(8, 3) = Q_Table_Rewarded(8, 3) * (1.5)
            Q_Table_Rewarded(11, 3) = Q_Table_Rewarded(11, 3) * (1.5)
            Q_Table_Rewarded(7, 2) = Q_Table_Rewarded(7, 2) * (1.5)
            Q_Table_Rewarded(10, 2) = Q_Table_Rewarded(10, 2) * (1.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
```

```
    End Sub

    Private Sub ComboBox16_OnChange(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ComboBox16.SelectedIndexChanged
        If ComboBox16.SelectedIndex = 0 Then
            Q_Table_Rewarded(15, 4) = Q_Table_Rewarded(15, 4) * (1.5)
            Q_Table_Rewarded(18, 4) = Q_Table_Rewarded(18, 4) * (1.5)
            Q_Table_Rewarded(14, 3) = Q_Table_Rewarded(14, 3) * (1.5)
            Q_Table_Rewarded(17, 3) = Q_Table_Rewarded(17, 3) * (1.5)
            Q_Table_Rewarded(13, 2) = Q_Table_Rewarded(13, 2) * (1.5)
            Q_Table_Rewarded(16, 2) = Q_Table_Rewarded(16, 2) * (1.5)
        End If

        If ComboBox16.SelectedIndex = 1 Then
            Q_Table_Rewarded(15, 4) = Q_Table_Rewarded(15, 4) * (1.1)
            Q_Table_Rewarded(18, 4) = Q_Table_Rewarded(18, 4) * (1.1)
            Q_Table_Rewarded(14, 3) = Q_Table_Rewarded(14, 3) * (1.1)
            Q_Table_Rewarded(17, 3) = Q_Table_Rewarded(17, 3) * (1.1)
            Q_Table_Rewarded(13, 2) = Q_Table_Rewarded(13, 2) * (1.1)
            Q_Table_Rewarded(16, 2) = Q_Table_Rewarded(16, 2) * (1.1)
        End If

        If ComboBox16.SelectedIndex = 2 Then
            Q_Table_Rewarded(15, 4) = Q_Table_Rewarded(15, 4) * (1)
            Q_Table_Rewarded(18, 4) = Q_Table_Rewarded(18, 4) * (1)
            Q_Table_Rewarded(14, 3) = Q_Table_Rewarded(14, 3) * (1)
            Q_Table_Rewarded(17, 3) = Q_Table_Rewarded(17, 3) * (1)
            Q_Table_Rewarded(13, 2) = Q_Table_Rewarded(13, 2) * (1)
            Q_Table_Rewarded(16, 2) = Q_Table_Rewarded(16, 2) * (1)
        End If

        If ComboBox16.SelectedIndex = 3 Then
            Q_Table_Rewarded(15, 4) = Q_Table_Rewarded(15, 4) * (0.9)
            Q_Table_Rewarded(18, 4) = Q_Table_Rewarded(18, 4) * (0.9)
            Q_Table_Rewarded(14, 3) = Q_Table_Rewarded(14, 3) * (0.9)
            Q_Table_Rewarded(17, 3) = Q_Table_Rewarded(17, 3) * (0.9)
            Q_Table_Rewarded(13, 2) = Q_Table_Rewarded(13, 2) * (0.9)
            Q_Table_Rewarded(16, 2) = Q_Table_Rewarded(16, 2) * (0.9)
        End If

        If ComboBox16.SelectedIndex = 4 Then
            Q_Table_Rewarded(15, 4) = Q_Table_Rewarded(15, 4) * (0.5)
            Q_Table_Rewarded(18, 4) = Q_Table_Rewarded(18, 4) * (0.5)
            Q_Table_Rewarded(14, 3) = Q_Table_Rewarded(14, 3) * (0.5)
            Q_Table_Rewarded(17, 3) = Q_Table_Rewarded(17, 3) * (0.5)
            Q_Table_Rewarded(13, 2) = Q_Table_Rewarded(13, 2) * (0.5)
            Q_Table_Rewarded(16, 2) = Q_Table_Rewarded(16, 2) * (0.5)
        End If

        Q_Table_Final = Q_Table_Rewarded
        TextBox127.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox118.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf
        TextBox126.Text = MatLib.PrintMat(Q_Table_Final) & vbCrLf & vbCrLf

    End Sub

End Class
```

Appendix XII. Bag Classification using Support Vector Machines - Source Code

feature_extraction_4_bags_9.m

```
close all;
clear all;
home;

vector_matrix_train=[0 0 0 0 0 0 0 0 0 0];
vector_matrix_test=[0 0 0 0 0 0 0 0 0 0];

vector_matrix_train_new=[0 0 0 0 0 0 0 0 0 0];
vector_matrix_test_new=[0 0 0 0 0 0 0 0 0 0];

% Class1 training feature extraction

for i=1:20
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class1_train\class1_train_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu-0.02); % +0.12

e=~e;
objsize=10000;
objclean

%imshow(e);

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',10,10);

e = imdilate(e,se);
e = imerode(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

%moments(e,1);
[centroid, theta, roundness,figNo] = moments(e,1);
```

```
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 1];
vector_matrix_train=[vector_matrix_train;current_vector];

subplot(5,4,i), imshow(e);

end

% Class2 training feature extraction

figure,

for i=1:20
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class2_train\class2_train_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu+0.02);

e=~e;
objsize=20000;
objclean

%imshow(e);

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',10,10);

e = imdilate(e,se);
e = imerode(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;
```

```
current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 2];
vector_matrix_train=[vector_matrix_train;current_vector];

subplot(5,4,i), imshow(e);

end

% Class3 training feature extraction

figure,

for i=1:20
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class3_train\class3_train_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu-0.02); %+0.08

objsize=20000;
objclean

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',4,4);

e = imdilate(e,se);
e = imerode(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*(stats.Area)/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 3];
vector_matrix_train=[vector_matrix_train;current_vector];

subplot(5,4,i), imshow(e);
```

```
end

% Class4 training feature extraction

figure,

for i=1:20
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class4_train\class4_train_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu);

e=~e;
objsize=20000;
objclean

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',4,4);

e = imdilate(e,se);
e = imerode(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 4];
vector_matrix_train=[vector_matrix_train;current_vector];

subplot(5,4,i), imshow(e);

end

% Class1 testing feature extraction
```

```
figure,

for i=1:10
    current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class1_test\class1_test_';
    current_pic2=int2str(i);
    current_pic3='.jpg';
    current_pic_final=[current_pic1 current_pic2 current_pic3];
    a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu); %+0.12

e=~e;
objsize=15000;
objclean

se = strel('line',10,10);

e = imdilate(e,se);
e = imerode(e,se);

%imshow(e);

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 1];
vector_matrix_test=[vector_matrix_test;current_vector];

subplot(5,2,i), imshow(e);

end

% Class2 testing feature extraction

figure,
```

```
for i=1:10
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class2_test\class2_test_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu);

e=~e;
objsize=20000;
objclean

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',10,10);
e = imerode(e,se);
e = imdilate(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;
current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
     stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 2];
vector_matrix_test=[vector_matrix_test;current_vector];

subplot(5,2,i), imshow(e);

end

% Class3 testing feature extraction

figure,

for i=1:10
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class3_test\class3_test_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);
```

```
tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu+0.02); %+0.08

objsize=20000;
objclean

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

se = strel('line',4,4);

e = imdilate(e,se);
e = imerode(e,se);

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
    stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 3];
vector_matrix_test=[vector_matrix_test;current_vector];

subplot(5,2,i), imshow(e);

end

% Class4 testing feature extraction

figure,

for i=1:10
   current_pic1='D:\phd\Experiments\svm_experiment_4_bags_120_samples\class4_test\class4_test_';
   current_pic2=int2str(i);
   current_pic3='.jpg';
   current_pic_final=[current_pic1 current_pic2 current_pic3];
   a=imread(current_pic_final);

tOtsu = graythresh(a); % Built-in function uses Otsu's method
iThreshOtsu = round(tOtsu*256); % Convert from 0..1 to 0..256
hist = imhist(a); % Get histogram
P = hist'/sum(hist); % estimate PDF
```

```
vw(1:256) = 0; % Holds results
for iThresh = iThreshOtsu-10:iThreshOtsu+10
q1 = sum(P(1:iThresh));
q2 = sum(P(iThresh+1:256));
u1 = sum([1:iThresh].*P(1:iThresh))/q1;
u2 = sum([iThresh+1:256].*P(iThresh+1:256))/q2;
v1 = sum((([1:iThresh]-u1).^2).*P(1:iThresh))/q1;;
v2 = sum((([iThresh+1:256]-u2).^2).*P(iThresh+1:256))/q2;
vw(iThresh) = q1*v1 + q2*v2;
end
vw(iThreshOtsu-10:iThreshOtsu+10);

e=im2bw(a,tOtsu);

e=~e;
objsize=20000;
objclean

e = bwfill(e,'holes');

e=medfilt2(e);

objsize=10000;
objclean

L = bwlabel(e);
stats = imfeature(L,'All');

[centroid, theta, roundness,figNo] = moments(e,1);
ConvexPerimeter = (4*pi*stats.Area/roundness)^0.5;

current_vector=[stats.Area stats.BoundingBox(3)/stats.BoundingBox(4) stats.MajorAxisLength stats.MinorAxisLength
    stats.Eccentricity  stats.EquivDiameter stats.Extent roundness ConvexPerimeter 4];
vector_matrix_test=[vector_matrix_test;current_vector];

subplot(5,2,i), imshow(e);

end
home;

train_test_vector=[vector_matrix_train;vector_matrix_test];

for i=1:9
   for j=2:21
   vector_matrix_train_new(j,i)=vector_matrix_train(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=22:41
   vector_matrix_train_new(j,i)=vector_matrix_train(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=42:61
   vector_matrix_train_new(j,i)=vector_matrix_train(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=62:81
   vector_matrix_train_new(j,i)=vector_matrix_train(j,i)/max(train_test_vector(2:121,i))
```

```
end
end

for i=1:9
   for j=2:11
   vector_matrix_test_new(j,i)=vector_matrix_test(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=12:21
   vector_matrix_test_new(j,i)=vector_matrix_test(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=22:31
   vector_matrix_test_new(j,i)=vector_matrix_test(j,i)/max(train_test_vector(2:121,i))
end
end

for i=1:9
   for j=32:41
   vector_matrix_test_new(j,i)=vector_matrix_test(j,i)/max(train_test_vector(2:121,i))
end
end

vector_matrix_train
vector_matrix_test

Labels_train=vector_matrix_train(2:81,10)'
Samples_train=vector_matrix_train_new(2:81,1:9)'

Labels_test=vector_matrix_test(2:41,10)'
Samples_test=vector_matrix_test_new(2:41,1:9)'

save results Labels_train Samples_train Labels_test Samples_test
```

moments.m

```
  % MOMENTS
  %
  % Function calculates the moments of a binary image and returns
  % the centroid, the angle of axis of minimum inertia, and a measure
  % of 'roundness'.  The function assumes that there is only one object
  % in the binary image.
  %
  % function [centroid, theta, roundness] = moments(im)
  %
  % Argument:  im   - a binary image containing values of 0 or 1
  %
  % Returns:   centroid  - a 2 element vector
  %            theta     - the angle of axis of minimum inertia (radians)
  %            roundness - ratio of minimum inertia/maximum inertia.
  %
  % Note that positive x is to the right and positive y is downwards
  % thus angles are positive clockwise.
  %
  % The function also displays the image and overlays the position of
  % the centroid and the axis of minimum inertia.

  function [centroid, theta, roundness,figNo] = moments(im,figNo)
  [rows,cols] = size(im);
```

```
  x = ones(rows,1)*[1:cols];    % Matrix with each pixel set to its x coordinate
  y = [1:rows]'*ones(1,cols);   %   "    "    "    "    "  "  "  y    "

  area = sum(sum(im));
if area == 0
  meanx =0; meany =0;
else
  meanx = sum(sum(double(im) .* x))/area;
  meany = sum(sum(double(im) .* y))/area;
end
  centroid = [meanx meany];

  xp=x-(ones(size(im))*meanx);
  yp=y-(ones(size(im))*meany);

  a = sum(sum(double(im) .* xp .* xp));
  b = 2*sum(sum(double(im) .* xp .* yp));
  c = sum(sum(double(im) .* yp .* yp));

  theta = 0.5* atan2(b,(a-c));

if a-c == 0
  roundness = 1;
else
  cos2theta = (a-c)/sqrt(b^2+(a-c)^2);
  sin2theta = b/sqrt(b^2+(a-c)^2);
  Imin = 0.5*(c+a)-0.5*(a-c)*(cos2theta)-0.5*b*(sin2theta); % Positive solutions minimize I
  Imax = 0.5*(c+a)-0.5*(a-c)*(-cos2theta)-0.5*b*(-sin2theta); % Negative solutions maximize I

  roundness = Imin/Imax;
end

  t = double(im) .* x *cos(theta) + double(im) .* y *sin(theta);
  rho = meany*cos(theta)-meanx*sin(theta);
  x0 = -rho*sin(theta) + t*cos(theta);
  y0 =  rho*cos(theta) + t*sin(theta);

if nargin == 2
    if figNo              % We have a valid figure number
%      figure(figNo);     % Reuse or create a figure window with this number
%           imshow(im);
%           line(x0,y0);
    else
%      newWindow=0;       % figNo == 0
   end
else
%    return (roundness)
end
```

objclean.m

```
% This subroutine labels the number of objects in the image, counts the number of
% pixels in each object and deletes the objects that have less than 'objsize'

[lbw num]=bwlabel(e);
num;

for n=1:num
  bincount(n,1)=n; % represents a label name
  bincount(n,2)=sum(sum(lbw==n)); % represents an area of a label
end
```

```
lbwnew=lbw;

for m=1:num
   if bincount(m,2)<=objsize
      small=find(lbwnew==m);
      lbwnew(small)=0;
   end
end

e=lbwnew;
%figure
%imshow(rclean)
```

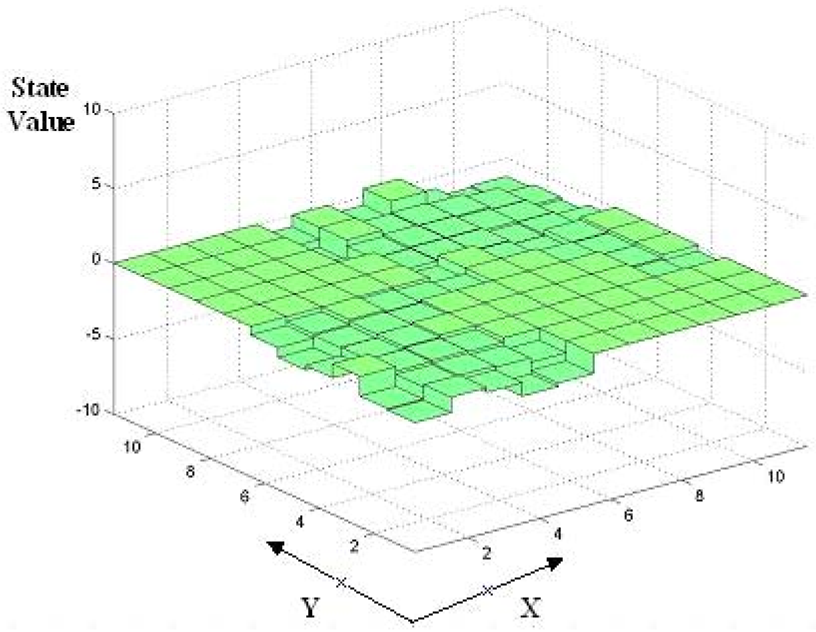
rbf_svm.m

```
close all;
clear all;
home
clc

iterations=10;

%%load results;
load matrix1;

for i=2:2 %513
load results;

% cross 1
Samples_train=[Samples_train(:,1:20) Samples_train(:,21:40) Samples_train(:,41:60) Samples_train(:,61:80)];
Labels_train=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4];
Samples_test=[Samples_test(:,1:10) Samples_test(:,11:20) Samples_test(:,21:30) Samples_test(:,31:40)];
Labels_test=[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4];

% cross 2
%Samples_train=[Samples_train(:,11:20) Samples_test(:,1:10) Samples_train(:,31:40) Samples_test(:,11:20)
      Samples_train(:,51:60) Samples_test(:,21:30) Samples_train(:,71:80) Samples_test(:,31:40)];
%Labels_train=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4];
%Samples_test=[Samples_train(:,1:10) Samples_train(:,21:30) Samples_train(:,41:50) Samples_train(:,61:70)];
%Labels_test=[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4];

% cross 3
%Samples_train=[Samples_train(:,1:10) Samples_test(:,1:10) Samples_train(:,21:30) Samples_test(:,11:20)
      Samples_train(:,41:50) Samples_test(:,21:30) Samples_train(:,61:70) Samples_test(:,31:40)];
%Labels_train=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4];
%Samples_test=[Samples_train(:,11:20) Samples_train(:,31:40) Samples_train(:,51:60) Samples_train(:,71:80)];
%Labels_test=[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4];

%%'start stam'
current_matrix1_vector=matrix1(i,:);
current_matrix1_vector=[0 1 1 0 0 0 1 1 0];
b=find(current_matrix1_vector>0);
c=size(b);
d=c(2);

if (d==4) % number of features

if (current_matrix1_vector(1)==0)
Samples_train(1,:)=0;
```

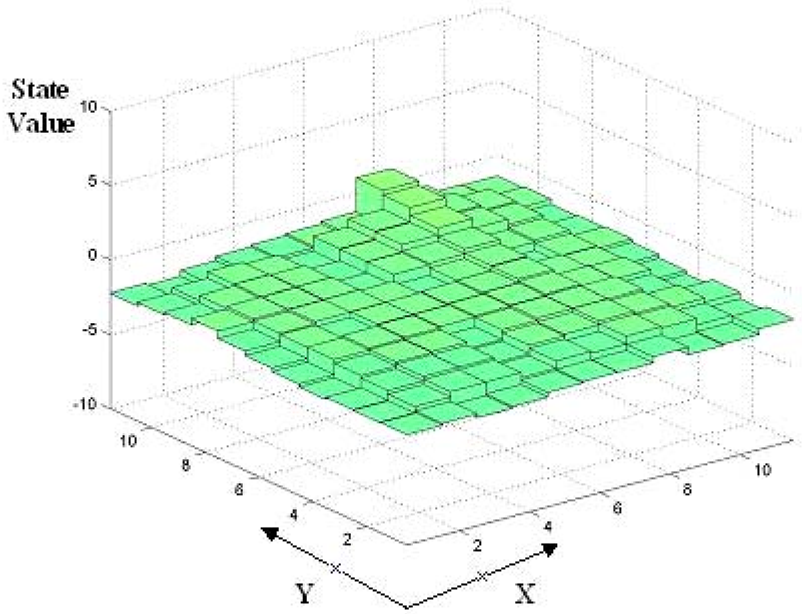
```
Samples_test(1,:)=0;
end

if (current_matrix1_vector(2)==0)
Samples_train(2,:)=0;
Samples_test(2,:)=0;
end

if (current_matrix1_vector(3)==0)
Samples_train(3,:)=0;
Samples_test(3,:)=0;
end

if (current_matrix1_vector(4)==0)
Samples_train(4,:)=0;
Samples_test(4,:)=0;
end

if (current_matrix1_vector(5)==0)
Samples_train(5,:)=0;
Samples_test(5,:)=0;
end

if (current_matrix1_vector(6)==0)
Samples_train(6,:)=0;
Samples_test(6,:)=0;
end

if (current_matrix1_vector(7)==0)
Samples_train(7,:)=0;
Samples_test(7,:)=0;
end

if (current_matrix1_vector(8)==0)
Samples_train(8,:)=0;
Samples_test(8,:)=0;
end

if (current_matrix1_vector(9)==0)
Samples_train(9,:)=0;
Samples_test(9,:)=0;
end

for Degree=9:9

[AlphaY, SVs, Bias, Parameters, nSV, nLabel] = polySVC(Samples_train, Labels_train, Degree); %polySVC or rbfSVC

[ClassRate, DecisionValue, Ns, ConfMatrix, PreLabels]= SVMTest(Samples_test, Labels_test, AlphaY, SVs,
    Bias,Parameters, nSV, nLabel);

ConfMatrix;

ClassRate_vector(Degree)=ClassRate;
ClassRate

end
end
end
end
```

Appendix XIII. Multiple Mobile Robot Navigation - Value Maps Example

Examples for state-action values of one of the simulation runs[1] described in Section 5.4 are given for several of the learning episodes:

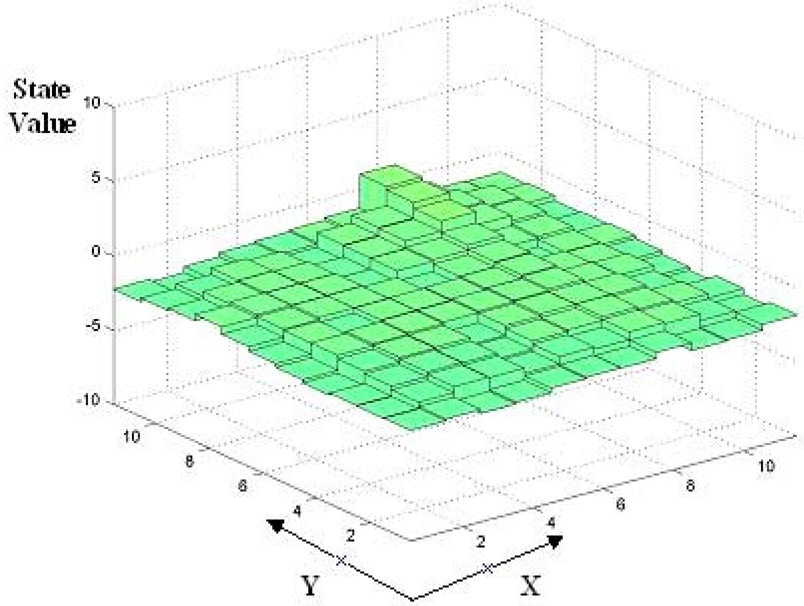


(a) State-action value map after one learning episode



(b) State-action value map after fifty episodes

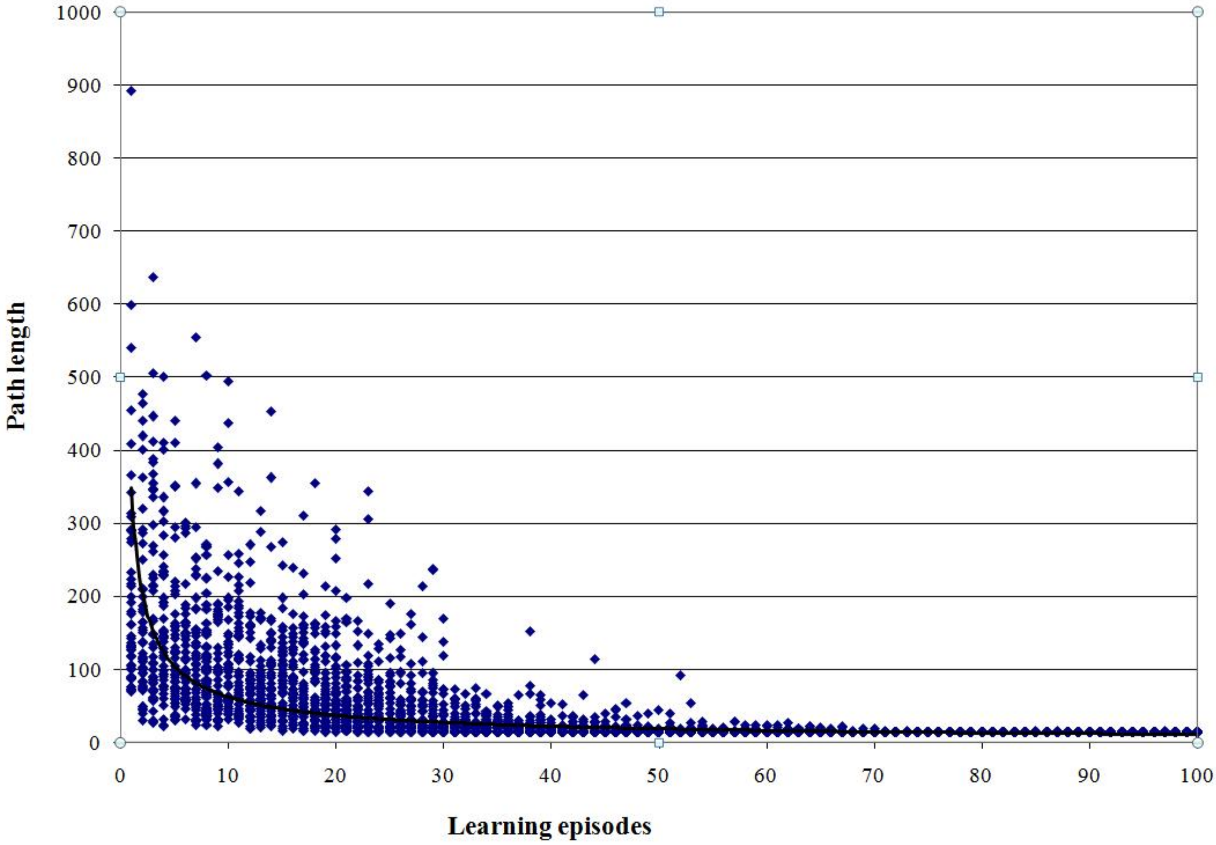
[1] One simulation run contains 100 learning episodes. Each learning episode consists of placing the robot at a starting location in the environment. The robot explores the environment. A learning episode ends when the robot reaches the target.



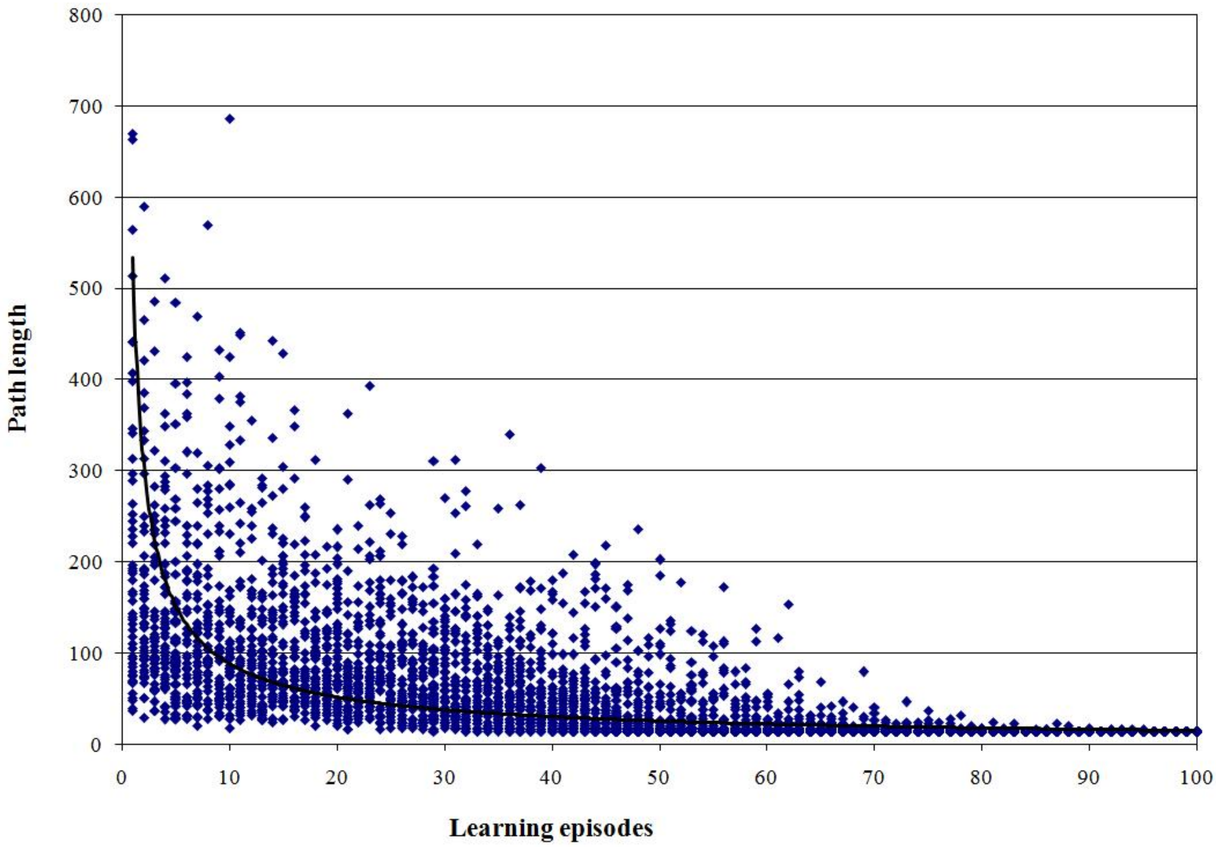
(c) State-action value map after 100 episodes

Fig. XIII.1 An 11 × 11 world state-action value maps

Appendix XIV. Multiple Mobile Robot Navigation - Detailed Scatter Plots



(a) Robot1 - *CQ(λ)*



(b) Robot2 - *Q(λ)*

Fig. XIV.1 Fifty simulation runs for convergence of two robots

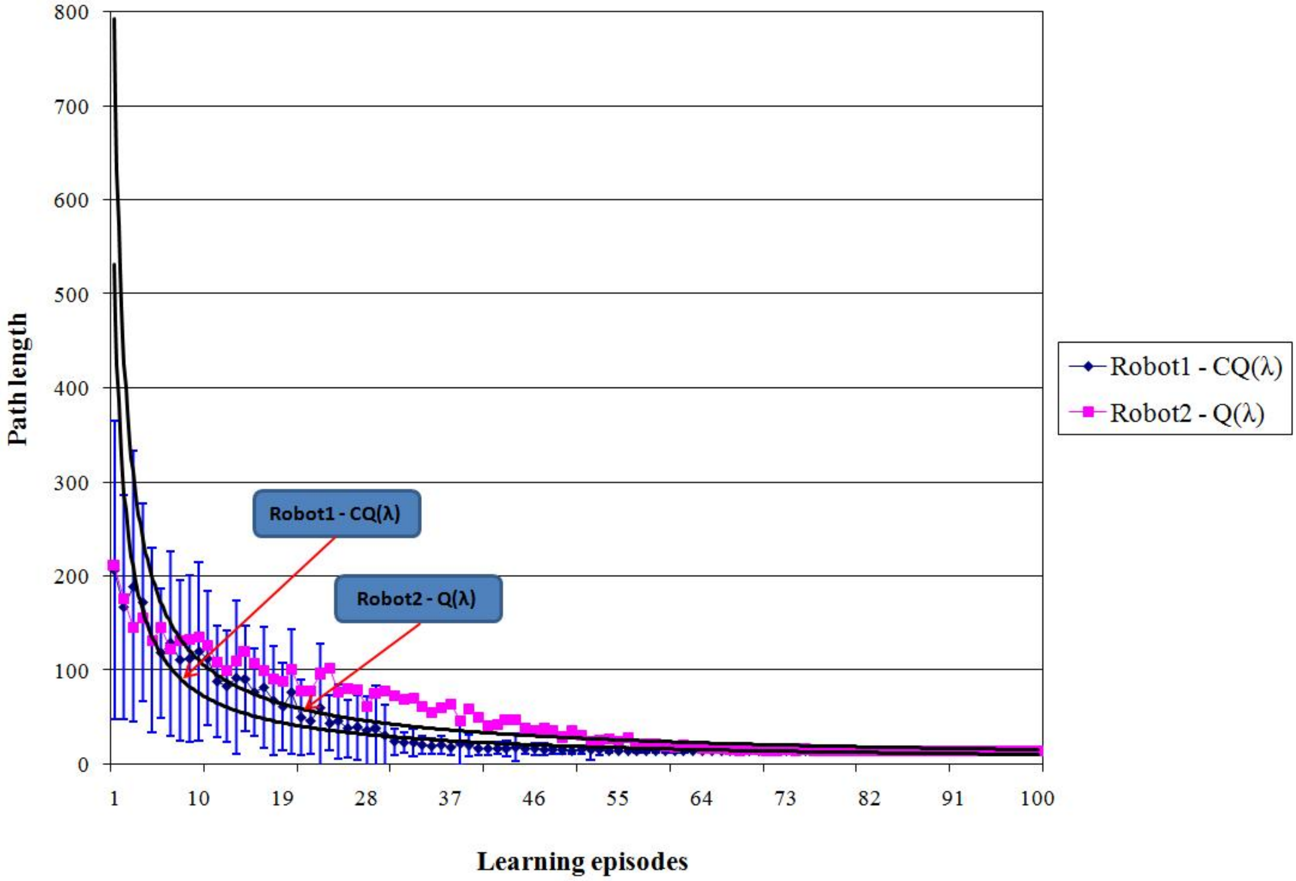
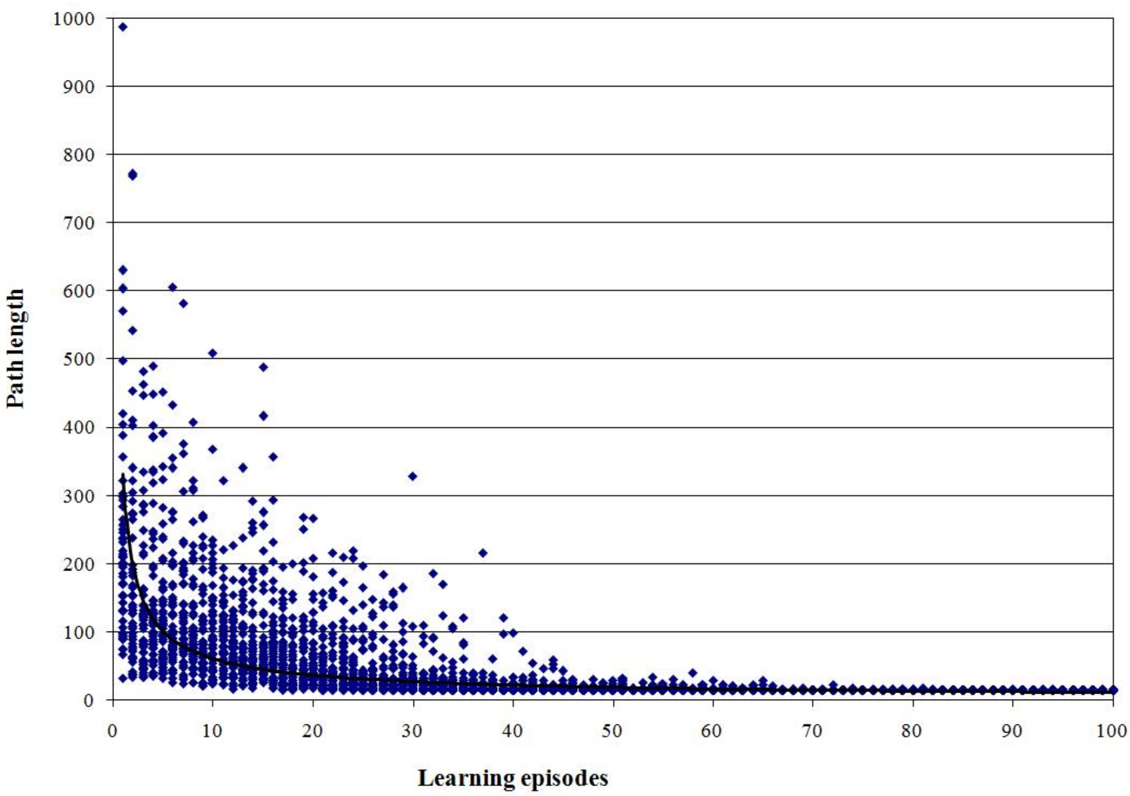
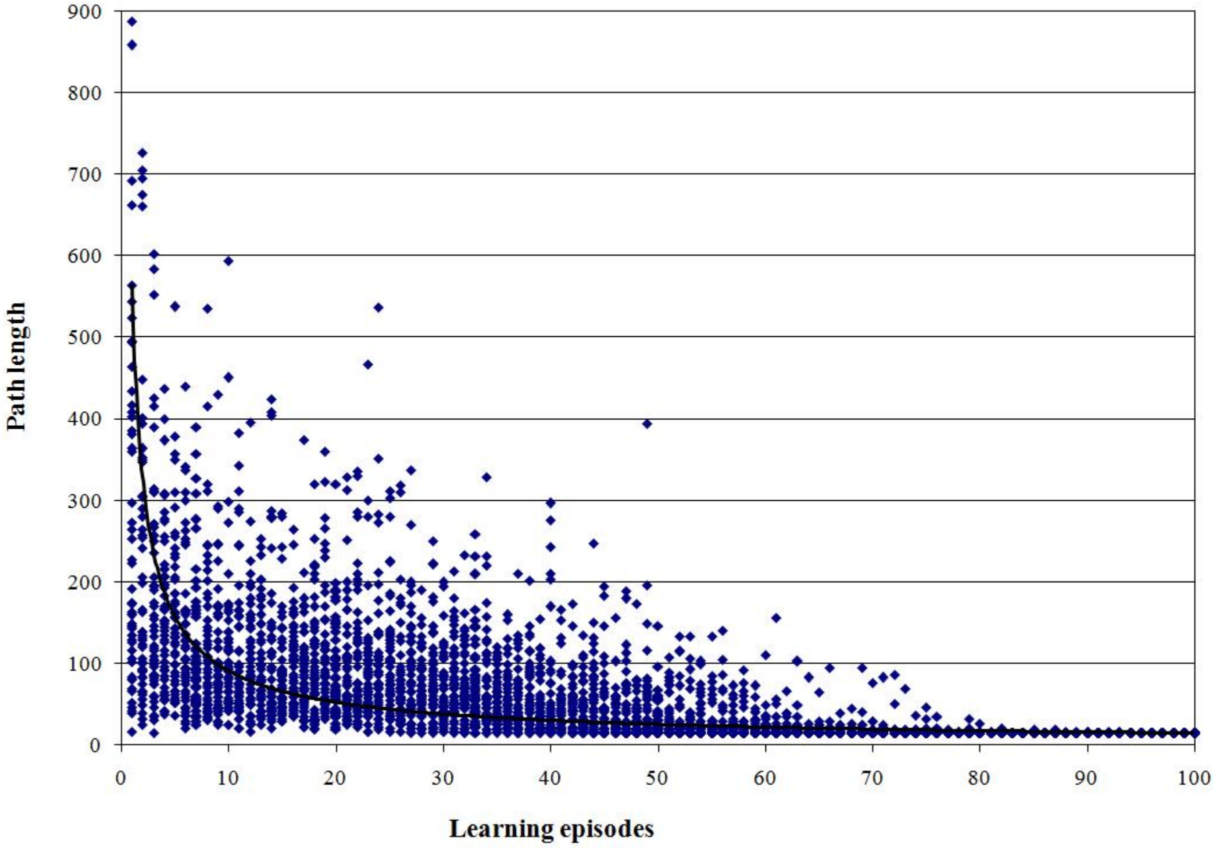


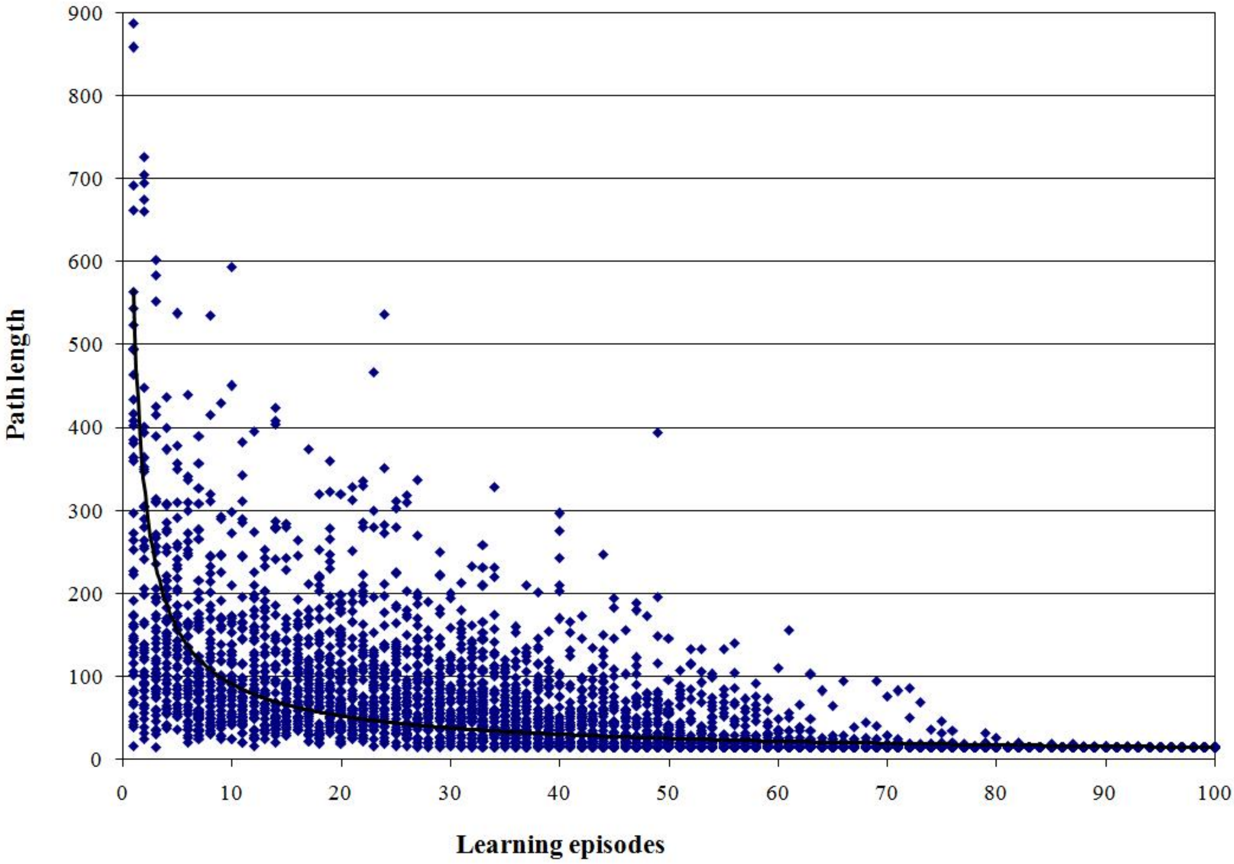
Fig. XIV.2 Fifty simulation runs for convergence of two robots - mean curves comparison



(a) Robot1 - *CQ(λ)*



(b) Robot2 - *Q(λ)*



(c) Robot3 - *Q(λ)*

Fig. XIV.3 Fifty simulation runs for convergence of three robots

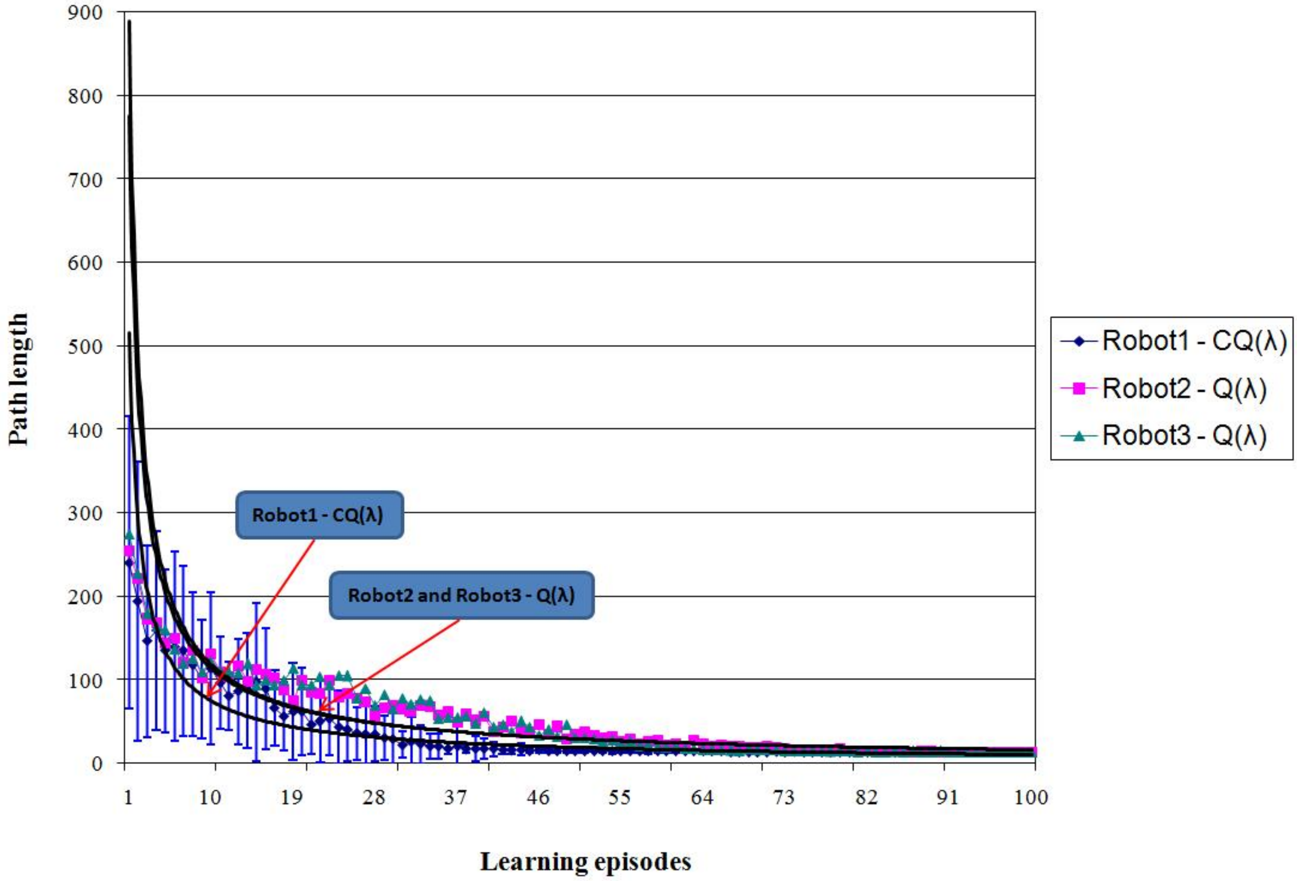


Fig. XIV.4 Fifty simulation runs for convergence of three robots - mean curves comparison

תקציר

על מנת לקדם את השימוש ברובוטים במשימות יום-יומיות הרובוטים חייבים להתנהל בסביבות בלתי מובנות, בלתי צפויות ומשתנות. דבר זה דורש מהם לפעול בצורה עצמאית וללמוד כיצד להגיב לסביבתם וכיצד הסביבה בהם הם פועלים מגיבה בהתאם לפעולות אותן הם מבצעים.

אחת הגישות ללמידה נקראת שיטת החיזוקים (Reinforcement Learning). בשיטה זו הרובוט מונחה על ידי קבלת חיזוקים מן הסביבה בה הוא מתפקד. חיזוקים אלו מספקים לרובוט אינדיקציה לגבי רמת התפקוד שלו לשם ביצוע משימה נתונה. אלגוריתמים נפוצים המבוססים על שיטת החיזוקים ושפותחו על מערכות רובוטיות רבות כוללים שיטה הנקראת "למידת Q" ואת הגרסה המשופרת שלה הנקראת "למידת $Q(\lambda)$". אלגוריתמים אלו אינם דורשים את מודל הבעיה כתנאי לפתרונה. בנוסף הם אינם תלויים במדיניות למידה מסוימת והם יכולים לבחור ביצוע פעולות על פי כמה אסטרטגיות של שליטה. למרות שנעשה שימוש בשיטות "למידת Q" ו - "למידת $Q(\lambda)$" באפליקציות רובוטיות רבות, קיימת הצדקה לשיפורן. חסרונותיהן כוללים: א) זמן חישוב רב הדרוש לקבלת פתרון אופטימאלי, ו - ב) זמני לימוד ארוכים הדרושים להתכנסות הפתרון.

עבודת מחקר זו מציגה פיתוח של אלגוריתם חדש הנקרא $CQ(\lambda)$. $CQ(\lambda)$ הינו אלגוריתם למידה המהווה הרחבה של אלגוריתם למידה פופולארי, ה - $Q(\lambda)$. האלגוריתם המוצע מאפשר שיתוף פעולה של למידה בין כמה סוכנים הפועלים בסביבה לשם ביצוע משימה. בנוסף, שימוש בטכניקה זו מאפשר שילוב של אדם בתהליך ביצוע המשימה, דבר הצפוי לקדם את תהליך הלמידה תוך כדי שימוש באינטליגנציה ובניסיון האדם. כאשר מערכת כוללת רובוט אחד ואדם בלבד, פונקציית הלמידה של הרובוט מתעדכנת תוך כדי קבלת ידע הן מהאדם והן מאינטראקציה עם הסביבה.

אלגוריתם ה - $CQ(\lambda)$, פותח, נבדק ויושם עבור שתי גישות: א) למידה תוך כדי שימוש בכמה סוכנים, ו - ב) למידה במערכות משולבות אדם-רובוט. בגישה הראשונה, שיתוף הפעולה בין הסוכנים מערב את בחירת ערך מצב-הפעולה (ערך ה - Q) הגבוה ביותר מתוך כל הסוכנים הקיימים במערכת עבור כל צעד למידה. בגישה השנייה, שתי רמות של שיתוף פעולה הוגדרו עבור מערכות למידה משולבות אדם-רובוט: א) אוטונומיה מלאה - הרובוט מחליט בעצמו אילו פעולות לבצע ומתפקד בצורה אוטונומית על פי פונקציית הלמידה של אלגוריתם ה - $Q(\lambda)$, ו - ב) אוטונומיה חלקית - האדם מנחה את הרובוט והרובוט משתמש בידע זה ומעדכן את פונקציית הלמידה שלו. שימוש ברמת האוטונומיה החלקית מאפשר לרובוט מידה מסוימת של מודעות עצמית בכדי לבחור בצורה אדפטיבית את רמת שיתוף הפעולה; ברמה של אוטונומיה מלאה (ביצוע המטלה בצורה עצמאית) או ברמה של אוטונומיה חלקית בה הרובוט פונה לאדם לקבלת סיוע. תכונת מודעות הרובוט המהווה למעשה גישה של אוטונומיות דינאמית הודגמה תוך שימוש ברובוט נייד ובזרוע רובוטית.

ניסויים מקיפים תוך שימוש במערכות רובוטיות שונות על מגוון של יישומים בוצעו בכדי להדגים את מעלותיו ואת מגרעותיו של אלגוריתם ה - $CQ(\lambda)$. אפליקציות ייעודיות פותחו והודגמו בכדי לבחון את ביצועי האלגוריתם בהקשר של סביבה נבונה תוך כדי שימוש ברובוט נייד המבצע משימת ניווט ובזרוע רובוטית לביצוע משימה של בדיקת תיק החשוד כמכיל חומרי חבלה לא קונבנציונאליים.

תוצאות המחקר הראו את עליונות אלגוריתם ה - $CQ(\lambda)$ בהשוואה לאלגוריתם ה - $Q(\lambda)$ הסטנדרטי. השיטה המוצעת צפויה להפחית הן את מספרן של איטראציות הלמידה והן את משך הזמן הדרוש ללמוד לבצע משימה מבוססת רובוט.

שיטה

אלגוריתם ה - $CQ(\lambda)$ פותח, נבדק ויושם עבור שתי גישות:

1) <u>למידה תוך כדי שימוש בכמה סוכנים</u>

אלגוריתם ה - $CQ(\lambda)$ פותח עבור מספר סוכנים לומדים ומבוסס על כך שערכו של צמד המצב-פעולה של סוכן שיתופי מעודכן על פי ביצועיו של הסוכן הטוב ביותר הקיים במערכת; שיתוף הפעולה מתבצע באמצעות שימוש בערך ה - Q הגבוה ביותר מבין כל הסוכנים הקיימים במערכת. תוך שימוש בטכניקה זו, ערכי ה - Q של הסוכן השיתופי יהיו בעלי ערך מיטבי.

2) <u>למידה במערכות משולבות אדם-רובוט</u>

בגישה זו, שתי רמות של שיתוף פעולה הוגדרו עבור מערכות למידה משולבות אדם-רובוט: א) אוטונומיה מלאה - הרובוט מחליט בעצמו אילו פעולות לבצע ומתפקד בצורה אוטונומית על פי פונקציית הלמידה של אלגוריתם ה - $Q(\lambda)$, ו - ב) אוטונומיה חלקית - האדם מנחה את הרובוט והרובוט משתמש בידע זה ומעדכן את פונקציית הלמידה שלו. שימוש ברמת האוטונומיה החלקית מאפשר לרובוט מידה מסוימת של מודעות עצמית בכדי לבחור בצורה אדפטיבית את רמת שיתוף הפעולה; ברמה של אוטונומיה מלאה (ביצוע המטלה בצורה עצמאית) או ברמה של אוטונומיה חלקית בה הרובוט פונה לאדם לקבלת סיוע. תכונת מודעות הרובוט מהווה למעשה גישה של אוטונומיות דינאמית הודגמה תוך שימוש ברובוט נייד ובזרוע רובוטית. במהלך תהליך הלמידה, הרובוט בוחן את רמת הביצוע שלו. התהליך מתבצע באמצעות הגדרת הפרמטר Λ, סף קבלה מינימלי אשר מעליו האדם מתבקש להתערב בפעילות הרובוטית. לאחר כל ביצוע של אפיזודת למידה, הפרמטר Λ מושווה לערך L_{ave}, ממוצע נע של רמת ביצוע הרובוט עבור N אפיזודות הלימוד האחרונות. על פי סף זה הרובוט מחליט אם לתפקד בצורה אוטונומית או לבקש סיוע מהאדם.

ניתוח תיאורטי בוצע עבור שתי הגישות. עבור גישת ריבוי הסוכנים מערכת הלמידה מורכבת מסוכן שיתופי, Q_C, ומכמה סוכנים מסוג $Q(\lambda)$ הלומדים לבצע את המשימה בצורה עצמאית. הודגם מתמטית שפונקציית הלמידה של הסוכן השיתופי מתכנסת מהר יותר מפונקציות הלמידה של הסוכנים העצמאיים. עבור גישת האדם-רובוט, נערך דיון תאורטי על מנת להסביר לימוד מסוג Q שיתופי הוא הינו מקרה פרטי של "למידת Q" ולכן פונקציית הלימוד של Q שיתופי תתכנס לפתרון אופטימלי בהסתברות 1.

אלגוריתם ה - $CQ(\lambda)$ המוצג בעבודה זו נבדק ונבחן על שלוש מערכות אשר פותחו במיוחד עבור עבודה זו:

1) <u>ניווט במערכת מרובת רובוטים (סימולציה)</u>

המערכת מורכבת מכמה רובוטים ניידים במודל סימולציה. הרובוטים לומדים לנווט בעולם דו-מימדי אשר מכיל אזורים לא רצויים ומטרתם היא להגיע למטרה במסלול האופטימלי. המערכת מורכבת ברובוט שיתופי אחד, Q_C, ומכמה סוכנים מסוג $Q(\lambda)$ הלומדים לבצע את המשימה בצורה עצמאית.

ביצועי המערכת הוערכו תוך שימוש במדדי הביצוע: א) $N_{t_{no}}$ - התכנסות לפתרון הקרוב לאופטימלי - ממוצע N המסלולים האחרונים, ו - ב) N_{t_o} - התכנסות לפתרון אופטימלי - מספר איטראציות הלמידה הדרוש להגיע לפתרון האופטימלי ולחזור עליו מספר אין סופי של פעמים.

2) <u>מערכת שיתוף פעולה בין אדם לרובוט למשימת ריקון תיק</u>

המערכת מורכבת מזרוע רובוטית מסוג Motoman UP-6 בעלת שש דרגות חופש, תיק המכיל מספר אובייקטים, מצלמה דיגיטלית אשר תפקידה לשלוח מידע ויזואלי מהסביבה הרובוטית אל ממשק האדם, משטח בחינה עליו מונח התיק החשוד ומד משקל דיגיטלי אשר תפקידו למדוד את ערכי ה - "פרסים" (Rewards). משימת הרובוט הינה לאחוז ולהרים תיק חשוד המונח על משטח הבחינה וללמוד לרוקנו מתוכנו בזמן הקצר ביותר תוך כדי אינטאקציה עם הסביבה וקבלת סיוע מהאדם.

ביצועי המערכת הוערכו תוך שימוש במדדי הביצוע: א) זמן ממוצע לסיים (או לסיים באופן חלקי) את ריקון התיק מתוכנו, ב) ממוצע ה - "פרסים" - מדד המציין את השתפרות תהליך למידה, ו - ג) שיעור התערבות האדם - מדד המייצג את אחוז השתתפות האדם מתוך כל אפיזודות הלמידה; ככל שמדד זה נמוך יותר כך גדלה מידת האוטונומיות של הרובוט. שלוש פונקציות פרסים הוגדרו כדלקמן: א) פונקצית פרסים ליניארית, ב) פונקצית פרסים מצטברת ו - ג) פונקציית פרסים המבוססת על אירועים.

3) <u>מערכת שיתוף פעולה בין אדם לרובוט נייד למשימת ניווט</u>

המערכת מורכבת מרובוט נייד מסוג Evolution Robotics ER-1 המצוייד במחשב נישא ובמצלמה. הרובוט לומד לנווט את דרכו אל עבר מטרה הממוקמת בעולם דו-מימדי. האדם נמצא רחוק מהסביבה הרובוטית. תחת תנאים שהוגדרו מראש הרובוט מחליט אם לפנות לאדם לקבלת עיצה או לנווט בצורה אוטונומית. הרובוט לומד תוך כדי אינטראקציה עם הסביבה ועל ידי רכישת ידע מהאדם. מטרת מערכת הלמידה הינה לאפשר לרובוט להתחיל מסלול ניווט מכל נקודה בעולם ולהגיע למטרה תוך בחירת המסלול הקצר ביותר והמנעות מאזורי ניווט שהוגדרו מראש כלא רצויים.

ביצועי המערכת הוערכו תוך שימוש במדדי הביצוע: א) המספר הממוצע של צעדים הדרושים להגיע אל תא המטרה בצורה אופטימאלית, ב) המספר הממוצע של צעדים הדרוש להגיע אל תא המטרה בצורה ברת ביצוע (פיסאבילית) ו - ג) שיעור התערבות האדם - מדידת תדירות הפעמים בה האדם שיתף פעולה עם הרובוט.

<u>ניתוח ותוצאות</u>

האצת ביצועי תהליך למידה תוך שיתוף פעולה בין כמה סוכנים הודגמה תוך כדי שימוש בסימולציות של רובוטים עבור משימת ניווט. 50 הרצות סימולציה תוך שימוש בשני רובוטים הראו שיפור של 17.02% בממוצע עבור מספר אפיזודות הלמידה הדרושות להשגת התכנסות לפתרון אופטימאלי ושיפור של 32.98% בממוצע עבור מספר אפיזודות הלמידה הדרושות להשגת פתרון פיסאבילי בהשוואה לאלגוריתם ה - $Q(\lambda)$.

מבחני מובהקות סטטיסטית אשר בוצעו לצורך השוואה בין ביצועי שני הרובוטים הן עבור התכנסות לפתרון הקרוב לאופטימלי והן עבור התכנסות לפתרון אופטימלי הראו שקיים הבדל משמעותי בין הרובוט השיתופי (ה - $CQ(\lambda)$) לבין הרובוט העצמאי (ה - $Q(\lambda)$). במערכת נוספת המורכבת משלושה רובוטים; הראשון שיתופי ולומד על פי אלגוריתם ה - $CQ(\lambda)$ ואילו השניים האחרים פועלים על פי אלגוריתם ה - $Q(\lambda)$, מבחני מובהקות סטטיסטית הראו שלא קיים הבדל בין ביצועי שני הרובוטים הלומדים על פי $Q(\lambda)$ הן עבור התכנסות לפתרון הקרוב לאופטימלי והן עבור התכנסות לפתרון אופטימלי. מבחני מובהקות סטטיסטית נוספים הראו שקיים הבדל הן עבור התכנסות לפתרון הקרוב לאופטימלי והן עבור התכנסות לפתרון אופטימלי בין רובוט ה - $CQ(\lambda)$ לבין השניים הנוספים. בנוסף, נמצא שלא קיים כל הבדל בביצועי רובוט ה - $CQ(\lambda)$ בין אם רובוט נוסף מסייע לו או בין אם שני רובוטים נוספים מסייעים לו. עליונות אלגוריתם ה -

$CQ(\lambda)$ על אלגוריתם ה - $Q(\lambda)$ הודגמה עבור שני המקרים. בנוסף, הודגם שביצועי רובוט המשתף פעולה עם שני רובוטים נוספים אינו טוב יותר מזו של רובוט המשתף פעולה עם רובוט אחד נוסף בלבד.

מנקודת מבט של שיעור למידה (Learning Rate) הודגם שהשתפרות הלמידה של סוכני $Q(\lambda)$ (סוכנים עצמאים) רב יותר מאשר סוכן ה - $CQ(\lambda)$ (הסוכן השיתופי) עבור שני המקרים שתוארו; ככל ששיעור הלמידה נמוך יותר כך גבוהה יותר השתפרות למידת הסוכן. למרות שהניסויים שבוצעו הדגימו שסוכן ה - $CQ(\lambda)$ לומד מהר יותר מסוכני ה - $Q(\lambda)$ ומגיע לפתרון אופטימאלי מהר יותר, שיעור השתפרות סוכני ה - $Q(\lambda)$ גבוה יותר. ההסבר לכך נובע מכל שסוכני ה - $Q(\lambda)$ לומדים בצורה יעילה פחות מסוכן ה - $CQ(\lambda)$ בשלבי הלימוד ההתחלתיים אך מאחר וכל הסוכנים (השיתופי והעצמאיים) מתכנסים בסופו של דבר לאותו פתרון אופטימלי (לאחר הרבה מאד אפיזודות למידה) אזי על הסוכנים העצמאיים להשיג את ביצועי הסוכן השיתופי.

עבור משימת ניעור התיק, התוצאות שהתקבלו מראות שהלמידה הייתה מהירה יותר כאשר אדם נתבקש להתערב בתהליך הלמידה של הרובוט. השוואה בין ביצועי $Q(\lambda)$ ל - $CQ(\lambda)$ תוך שימוש בפונקציית פרסים ליניארית לאורך 25 אפיזודות למידה הצביעה על שיפור של 45.5% עבור ממוצע הפרסים לטובת אלגוריתם ה - $CQ(\lambda)$ בעוד שיעור התערבות האדם הייתה 86.7%. השוואה בין ביצועי $Q(\lambda)$ ל - $CQ(\lambda)$ תוך שימוש בפונקצית פרסים מצטברת הראתה שהמשך הממוצע לרוקן (או לרוקן באופן חלקי) את תוכנו של תיק חשוד פחת ב - 16.6% וממוצע הפרסים גדל ב - 25.76%. שיעור התערבות האדם עבור למידת ה - $CQ(\lambda)$ עמד על 30%. השוואה בין ביצועי $Q(\lambda)$ ל - $CQ(\lambda)$ תוך שימוש בפונקצית פרסים המבוססת על אירועים הראתה שהמשך הממוצע לרוקן (או לרוקן באופן חלקי) את תוכנו של תיק חשוד פחת ב - 34.3% וממוצע הפרסים גדל ב - 30.04%. שיעור התערבות האדם עבור למידת ה - $CQ(\lambda)$ עמד על 20%.

עבור שלושת פונקציות הפרסים, הן עבור $Q(\lambda)$ והן עבור $CQ(\lambda)$, הרובוט מתחיל להתנסות בסביבתו על ידי ביצוע אפיזודת נענוע המורכבת מפעולות אקראיות בציר ה - X, ה - Y, וה - Z. אינטואיטיבית, נענוע אנכי נראה כאופציה הטובה ביותר, אך ניסויים קבעו שאסטרטגיות נענוע המורכבות ברובן מנענוע לאורך ציר ה - Y תוך שילוב מעט פעולות על ציר ה - X היו האפקטיביות ביותר. אסטרטגיות כאלו גרמו לשקית הפלסטיק לעיתים "להסתבך", גם עקב העובדה שרוב המשקל בשקית רוכז רוב הזמן במקום אחד. הסבר אפשרי לכך הינו המבנה היחודי של קשר השקית (פתח השקית); משיכה ידנית של שתי ידיות השקית לאורך הציר האופקי תגרום לקשר להשתחרר מהר יותר.

בכדי לפרש את התוצאות שהושגו בניסוי זה ובכדי להראות שלא היו השפעות סובייקטיביות, פותח מודל פיסיקאלי של פתיחת קשר של שקית פלסטיק על ידי רובוט. המודל מסביר את התוצאות שהושגו עבור שלושת פונקציות הפרסים. הוכח שתאוצת האובייקטים בשקית מתפתחת לאורך זמן ולכן כדאי לפתוח אותה תוך כדי הפעלת כוחות בצורה רציפה תוך כדי שמירה על מרחק גדול ככל האפשר לאורך ציר ה - Y. באופן אידיאלי, רצוי להאיץ את הזרוע הרובוטית בתאוצה הקרובה לתאוצת כוח הכובד כלפי מטה תוך כדי נענועה לאורך ציר ה - Y בכדי להתגבר על רוב כוחות החיכוך.

עבור משימת הניווט שהתבצעה על ידי רובוט נייד, ניסויים הצביעו על שיפור למידה משמעותי תוך שימוש באלגוריתם ה - $CQ(\lambda)$ בהשוואה לאלגוריתם ה - $Q(\lambda)$ (למידה ללא התערבות אדם). באופן ספציפי, עבור פתרון פיסאבילי ופתרון אופטימאלי, הושג שיפור של 23.07% ו - 18.56% בהתאמה תוך שימוש בסף קבלה של $\Lambda = 8$, ובפרמטרים $\gamma = 0.99$ ו - $\lambda = 0.75$ בעוד האדם נתבקש להתערב ב - 30% מאפיזודות הלמידה. ניסויים כללו דרגות שונות של אוטונומיה רובוטית

הראו כצפוי שככל שהערך Λ היה גבוה יותר כך פחתה רמת שיתוף הפעולה בין הרובוט לאדם. עבור שימוש בפרמטרים $\gamma = 0.99$ ו - $\lambda = 0.75$ שיפור הלמידה היה הגבוה ביותר. השילוב של ערכים גבוהים של γ ו - λ הביא לשיפור הלמידה הגבוה ביותר הינו הודות לעובדה שבחירת ערך גבוה של λ מאפשר עדכון ארוך יותר של רצף ערכי זוגות המצב-פעולה תוך שמירה על משך זמן חישובי סביר הדרוש לפתרון הבעיה. בניסויים אחרים תוך שימוש במגוון רחב של ערכי γ ו - λ לא הייתה עקביות בין בהשגת פתרונות המתאימים לכל ערכי הסף. ניתן להסביר חוסר עקביות זה בכך שלעיתים התערבות האדם פגמה ביכולת הרובוט לחקור את הסביבה בצורה אוטונומית (ביצוע Exploration) והוא למעשה אולץ לנצל את ידע האדם (ביצוע Exploitation) כך שבמקרים מסויימים לא חל שיפור ביכולת למידת הרובוט.

<u>מסקנות</u>

התרומה העיקרית של עבודה זו הינה פיתוח של שיטת לימוד חדשה. האלגוריתם המוצע, ה - $CQ(\lambda)$, מאפשר שיתוף פעולה של למידה במערכות בהן קיים יותר מסוכן לומד אחד, או במערכות משולבות אדם-רובוט בהן למידת הרובוט מואצת תוך שימוש באינטליגנציית ונסיון האדם.

ניסויים מקיפים בוצעו על מערכות רובוטיות שונות שפותחו במיוחד עבור עבודה זו במטרה להדגים את יתרונות והסרונות אלגוריתם ה - $CQ(\lambda)$. המערכות שפותחו משמשות כמתקני ניסוי עבור בחינת אלגוריתם ה - $CQ(\lambda)$ בהקשר של סביבה אינטליגנטית תוך שימוש ברובוט נייד עבור משימת ביצוע ניווט ובזרוע רובוטית עבור ביצוע משימה של בדיקת תיק חשוד. תוצאות הניסויים הראו את עליונות אלגוריתם ה - $CQ(\lambda)$ על אלגוריתם ה - $Q(\lambda)$ הסטנדרטי בהקשר של האצת ביצועי למידה.

מילות מפתח: למידה באמצעות חיזוקים, למידה רובוטית, שיתוף פעולה בין אדם לרובוט

העבודה בוצעה בהדרכתם של פרופ' הלמן שטרן ופרופ' יעל אידן

המחלקה להנדסת תעשיה וניהול

הפקולטה למדעי ההנדסה

שיטות לשיתוף פעולה בין אדם לרובוט במערכות לומדות

מחקר לשם מילוי חלקי של הדרישות לקבלת

''דוקטורט לפילוסופיה''

מאת

אורי קרטון

הוגש לסנאט אוניברסיטת בן-גוריון בנגב

אישור מנחה פרופ' הלמן שטרן ______________________

אישור מנחה פרופ' יעל אידן ______________________

אישור דיקן בית הספר ללימודי מחקר מתקדמים ______________________

תשס"ז 2007

באר-שבע

שיטות לשיתוף פעולה בין אדם לרובוט במערכות לומדות

מחקר לשם מילוי חלקי של הדרישות לקבלת
''דוקטורט לפילוסופיה''

מאת
אורי קרטון

הוגש לסנאט אוניברסיטת בן-גוריון בנגב

תשס"ז 2007

באר-שבע