Self-learning Visual Servoing of a Robot Manipulator using Explanation-based Fuzzy Neural Networks and Q-learning

by

Mehdi Sadeghzadeh

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of
Doctor of Philosophy
in
Engineering

Guelph, Ontario, Canada

ABSTRACT

SELF-LEARNING VISUAL SERVOING OF A ROBOT MANIPULATOR USING EXPLANATION-BASED FUZZY NEURAL NETWORKS AND Q-LEARNING

Mehdi Sadeghzadeh **Advisors: Professor David Calvert**
University of Guelph, 2014 **Professor Hussein A. Abdullah**

This thesis will show that the addition of Explanation-Based Fuzzy Neural Networks (EBFNN) to Q-learning improves the learning process of a self-learning visual servoing robot manipulator system. Two new self-learning visual servoing systems for robot manipulators are proposed based on the following methodologies:

- Self-learning visual servoing of a robot manipulator using a Q-learning algorithm and fuzzy neural networks.
- Self-learning visual servoing of a robot manipulator using EBFNN and a Q-learning algorithm.

Both learning methodologies do not require robot or camera models, or calibration. These systems apply Q-learning to find the optimal policy using reinforcement learning. This policy is used by the robot to reach a predetermined object that has been randomly placed in the environment. In the first system the Q-learning algorithm is implemented using fuzzy neural networks to estimate the Q-evaluation function for each robot action. This system learns the optimal policy in order to select the best basic action that maximizes the cumulative reward received at each time step. Simulation results demonstrate the effectiveness of the system to learn the highly non-linear mapping between the continuous work-space and the optimal action policy.

In the second system an analytical learning component is added to the induction learning. This system includes two main properties: on-line training and lifelong learning that are implemented by the Q-learning algorithm and the EBFNN respectively. It is demonstrated that the number of training samples, and therefore the training time for a specific robot positioning accuracy task, can be reduced using fuzzy explanation-based neural networks and the Q-learning algorithm. Background knowledge about the robot and its environment is transferred to the robot agent during the learning process using a set of neural networks which have been previously trained.

The on-line learning and real-time performances of these two systems are compared and simulation results show the effectiveness of the EBFNN to improve the learning process and performance of the self-learning visual servoing system. The T-test and Wilcoxon-Mann-Whitney U test are used to justify the statistical significance of the results.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisors, Dr. David Calvert and Dr. H. A. Abdullah, for their priceless guidance and support throughout every stage of this research. Without them this journey would not have been possible. A special thank you to Dr. Bob Dony for serving on my advisory committee and for his valuable advice in the area of Machine Vision Systems. I would also like to thank Dr. Medhat Moussa who introduced me to the visual servoing research topic. I am grateful for the unlimited support and help from Dr. Doug Joy, Associate Director of Graduate Studies in the School of Engineering.

To

my family:

my wife *Farah*, my son *Amir*, and my daughter *Parnia*;

whose love and encouragement helped accomplish this

thesis.

TABLE OF CONTENTS

Acknowledgements iv

Table of Contents vi

List of Symbols and Abbreviations xi

List of Tables xiv

List of Figures xv

Chapter One Introduction 1

1.1 Introduction 1

1.2 Research Contributions 7

1.3 Thesis Organization 9

Chapter Two Background 10

2.1 Visual Servoing Approaches 10

2.1.1 Model-based Visual Servoing 10

2.1.2 Model-free Visual Servoing 11

2.2 Vision-based Control Categorization 11

2.2.1 Position-based Visual Servoing (3D) 11

2.2.2 Image-based Visual Servoing (2D) 12

2.2.3 Hybrid Visual Servoing (2 ½ D) 13

2.2.4 Motion-based Visual Servoing (d2D/dt) 14

2.3 Camera Model in Machine Vision Systems 15

2.4 Camera Configurations in Visual Servoing Systems 17

2.4.1 Monocular Vision 17

2.4.2 Binocular Vision 18

2.4.3 Redundant Camera Systems 19

2.5 Image Feature Extraction Methods 19

2.5.1 Local Image Features Extraction 20

2.5.2 Global Image Features Extraction 20

2.6 Robot Manipulator Kinematics and Dynamics 21

2.7 Chapter Summary 22

Chapter Three Visual Servoing Problem Definition 23

3.1 Visual Servoing Problem Definition in Position-based Control Systems .. 25

3.2 Visual Servoing Problem Definition in Image-based Control Systems..... 28

3.3 Limitations and Problems in Visual Servoing Systems Implementation.... 32

3.4 Chapter Summary 35

Chapter Four Literature Survey in Visual Servoing 36

4.1 Research in Visual Servoing 36

4.2 Machine Learning in Robotics 49

4.3 Chapter Summary 55

Chapter Five Self-learning Visual Servoing of a Robot Manipulator Using Q-learning Algorithm and Fuzzy Neural Networks 57

5.1 Introduction 57

5.2 Visual Servoing System Architecture 61

5.3 Proposed Learning Algorithm 62

5.3.1 Q-learning Algorithm 63

5.3.2 Adaptive Neuro Fuzzy Inference Systems 66

5.4 Simulation Results and Discussion 70

5.4.1 ANFIS Training and Validation Results 73

5.4.2 Real-time Visual Servoing Results 75

5.5 Chapter Summary 82

Chapter Six Self-learning Visual Servoing of a Robot Manipulator Using Explanation-Based Fuzzy Neural Networks and Q-Learning 84

6.1 Introduction 84

6.2 Q-learning EBFNN Visual Servoing System Architecture 88

6.3 Learning Methodology Based on Inductive and Analytical Learning Algorithms 89

6.3.1 The Q-learning Algorithm 90

6.3.2 Basic Action Model Neural Networks 90

6.3.3 Explanation-based Fuzzy Neural Networks 91

6.4 Simulation Results and Discussion 97

6.4.1 Action Model Neural Networks Training and Validation Results 97

6.4.2 Explanation-based Fuzzy Neural Networks Training and Validation Results 104

6.4.3 Q-learning EB-ANFIS Real-time Visual Servoing Results 106

6.4.4 Comparison of Q-learning EB-ANFIS with Q-learning ANFIS Visual Servoing Results 112

6.5 Chapter Summary 117

Chapter Seven Statistical Hypothesis Tests 119

7.1 Introduction 119

7.1.1 T-Test 120

7.1.2 Wilcoxon-Mann-Whitney U Test 120

7.2 Statistical Tests for QLEBFNN and QLFNN Visual Servoing Systems .. 121

7.2.1 Statistical Tests for QLEBFNN and QLFNN Centering Success Rates 122

7.2.2 Statistical Tests for QLEBFNN and QLFNN Reaching Success Rates 123

7.2.3 Statistical Tests for QLEBFNN and QLFNN Centering Basic Action Counts 123

7.2.4 Statistical Tests for QLEBFNN and QLFNN Reaching Basic Action Counts..... 124

7.3 Chapter Summary 127

Chapter Eight Conclusions and Research Future Directions 128

8.1 Conclusions 128

8.2 Research Future Directions 133

List of Papers 134

References Cited 135

LIST OF SYMBOLS AND ABBREVIATIONS

$\boldsymbol{A}$: Design Matrix

A : Action Space

$\boldsymbol{Err}$: Image Features Error

f : Number of Extracted Image Features

$\boldsymbol{F}$: Extracted Image Features Vector

$\boldsymbol{F_R}$: Reference Image Features

$\boldsymbol{I}$: Camera Image

$\boldsymbol{J}$: Robot Joint

Q(s,a) : Evaluation Function

Q_C : Centering Evaluation Values

Q_R : Reaching Evaluation Values

r : Immediate Reward

S : State Space

T : Boltzmann Constant

$V^{\pi}(s)$: Discounted Cumulative Reward

$\boldsymbol{y}$: Output Training Vector

$\boldsymbol{\tau}$: Robot Joints Torque Vector

π : Action Selection Policy

μ_i : Relative Importance

γ : Discount Rate

π^* : Optimal Policy

λ : Learning Rate

$\boldsymbol{\theta}$: Consequent Parameters Vector

η : ANFIS Learning Rate

μ_c : Relative Importance Constant

$\Delta\boldsymbol{\theta}_c$: Command Robot Joint Angles Changes

$\Delta\boldsymbol{B}_C/\Delta\boldsymbol{F}$: Centering Action NN Output Derivative

$\Delta\boldsymbol{B}_R/\Delta\boldsymbol{F}$: Reaching Action NN Output Derivative

PBVS : Position-Based Visual Servoing

IBVS : Image-Based Visual Servoing

GFE : Global Feature Extraction

LFE : Local Feature Extraction

NN : Neural Networks

BP : Back Propagation

RBF : Radial Basis Functions

FNN : Fuzzy Neural Networks

GA : Genetic Algorithm

EBNN : Explanation-Based Neural Network Learning

EBFNN : Explanation-Based Fuzzy Neural Networks

QLFNN : Q-Learning Fuzzy Neural Networks

QLEBFNN : Q-Learning Explanation-Based Fuzzy Neural Networks

ANFIS : Adaptive Neuro Fuzzy Interface System

PCA : Principle Component Analysis

TS-FNNC : *Takagi-Sugeno* Fuzzy Neural Network Controller

WNN : Wavelet Neural Network

PD : Proportional Derivative

AI : Artificial Intelligence

EBL : Explanation-Based Learning

NFS : Neuro-Fuzzy Systems

MLP : Multi Layers Perceptron

LSE : Least Square Estimator

SD : Steepest Descent

MSE : Mean Square Error

EB-ANFIS : Explanation-Based Adaptive Neuro Fuzzy Inference Systems

LIST OF TABLES

Table 4.1: The mean square error result.....39

Table 4.2: RMS-error for a completely visible yellow cube under different lighting conditions.....46

Table 4.3: RMS-error for a 20 percent visible yellow cube under different lighting conditions.....46

Table 4.4: RMS-error for a blue cube under different lighting conditions.....46

Table 4.5: RMS-error for a screw head under different lighting conditions.....46

Table 4.6: RMS-error for a ledge with 3 holes under different lighting conditions.....47

Table 5.1: Link Parameters of Puma 560.....71

Table 5.2: Centering training and validation samples and MSE.....74

Table 5.3: Reaching training and validation samples and MSE.....74

Table 6.1: Training and validation MSE for centering action models NNs.....103

Table 6.2: Training and validation MSE for reaching action models NNs.....103

Table 6.3: Centering training and validation samples and MSE for EB-ANFIS.....105

Table 6.4: Reaching training and validation samples and MSE for EB-ANFIS.....105

Table 6.5: Comparison of average basic action numbers for Q-learning EB-ANFIS and Q-learning ANFIS in reaching task.....116

Table 7.1: Statistical tests results for QLEBFNN and QLFNN.....127

LIST OF FIGURES

Figure 2.1: Model-based 3D visual servoing.....12

Figure 2.2: Image-based 2D visual serrvoing.....13

Figure 2.3: Hybrid 2 ½ D visual servoing.....14

Figure 2.4: Motion-based d2D/dt visual servoing.....15

Figure 2.5: Camera model.....16

Figure 3.1: Relevant coordinate frames for the eye-in-hand and stand-alone configurations.....24

Figure 4.1: Visual positioning features error for neural network and proportional controller methods 40

Figure 4.2: Relearning of the system after turning the camera (a) 0.1π, (b) 0.5π.....41

Figure 4.3: Neuro-fuzzy model for the task based mapping.....44

Figure 4.4: PCA neuro-fuzzy controller in off-line and on-line phases.....45

Figure 5.1: Architecture of the proposed visual servoing system.....61

Figure 5.2: First-order Sugeno fuzzy model with 4 inputs and 2 membership functions..69

Figure 5.3: Puma 560 robot.....71

Figure 5.4: Sample ANFIS final membership functions (a) centering, (b) reaching.....74

Figure 5.5: Target object images for different end-effector positions (a) reference image at desired grasping position, (b) near start point, (c) far start point, and (d) extremely far start point.....78

Figure 5.6: Robot manipulator positions (a) desired grasping position, (b) extremely far position.....78

Figure 5.7: Visual servoing centering average image features error (a) near start point, (b) far start point, and (c) extremely far start point.....79

Figure 5.8: Visual servoing reaching average image features error (a) near start point, (b) far start point, and (c) extremely far start point.....80

Figure 5.9: Visual servoing immediate reward (a) centering, (b) reaching.....81

Figure 6.1: Architecture of the Q-learning EBFNN visual servoing system.....89

Figure 6.2: Training performance curves for the centering action model neural networks: (a) Net1, (b) Net2, (c) Net3, (d) Net4, (e) Net5 and (f) Net6.....100

Figure 6.3: Training performance curves for the reaching action model neural networks: (a) Net1, (b) Net2, (c) Net3, (d) Net4, (e) Net5, (f) Net6, (g) Net7 and (h) Net8.....103

Figure 6.4: Sample EB-ANFIS final membership functions (a) centering, (b) reaching.....106

Figure 6.5: Robot manipulator positions (a) desired grasping position, (b) extremely far position..... 108

Figure 6.6: Q-learning EB-ANFIS visual servoing centering average image features error: (a) near start point, (b) far start point, and (c) extremely far start point.....109

Figure 6.7: Q-learning EB-ANFIS visual servoing reaching average image features error: (a) near start point, (b) far start point, and (c) extremely far start point.....110

Figure 6.8: Q-learning EB-ANFIS visual servoing immediate reward (a) centering, (b) reaching.....111

Figure 6.9: Comparison of Q-learning EB-ANFIS and Q-learning ANFIS real-time visual servoing performance (a) centering, (b) reaching.....115

Figure 6.10: Q-learning ANFIS visual servoing average image features error for extremely far start point a) centering, (b) reaching.....116

Figure 7.1: Centering basic action numbers (a) QLEBFNN and (b) QLFNN.....125

Figure 7.2: Reaching basic action numbers (a) QLEBFNN and (b) QLFNN.....126

CHAPTER ONE

INTRODUCTION

1.1 Introduction

This thesis investigates the improvement gained through the use of Explanation-Based Fuzzy Neural Networks (EBFNN) and Q-learning on the learning process and real-time performance of a self-learning visual servoing robot manipulator system. Two self-learning visual servoing systems are introduced. These systems are based on the:

1. Q-Learning Fuzzy Neural Networks (QLFNN).
2. Q-Learning Explanation-Based Fuzzy Neural Networks (QLEBFNN).

It will be demonstrated that the QLEBFNN, in comparison to the QLFNN, learns the visual servoing task in considerably fewer online training episodes. It also improves the real-time performance of the system by reducing the number of basic actions required to fulfill the visual servoing task.

Control of a robot using visual feedback is called visual servoing. In visual servoing, image features such as points, corners, lines and specific regions, can be used to adjust the manipulator gripper with a target object. A visual sensor is an element of a control system used to generate feedback about the state of the work-space. There have been more than three decades of research in visual servoing. These studies include visual servoing for a simple task, such as pick and place, to the sophisticated manipulation of objects. Visual servoing was first initiated by Hill and Park [1] in 1979. The term visual feedback was used instead of visual servoing prior to this introduction. A need for robotic systems with high degree of flexibility was the incentive for developing visual servoing

systems. Closed-loop control with visual feedback has been proposed to achieve higher levels of flexibility and accuracy in robotic systems.

Robotic systems have a vital role in intelligent control and automation of a huge numbers of industrial processes. One of the main limitations in existing robotic systems is the lack of insight about the work environment. This constraint is a main factor that limits the complexity of the tasks a robot can carry out. Existing robots are capable of accomplishing their jobs only within a completely or partially structured work-space. The target, obstacles and environment conditions must be predetermined. When a robot does not have the ability to observe the work-space, then the domain and flexibility of its duties will be limited to a set of rigid pre-defined tasks. Machine vision is one of the most convenient solutions to perceiving the real world for robotic systems. A machine vision system can be applied to recognize and locate a fixed or moving object. This capability is used in visual servoing systems to position, track and grasp the target. Machine vision can also be used in different navigation systems such as vehicles, submarines and aircrafts. Vision-based automatic control systems have a great potential to grow, and are one of the most promising research areas in robotics research.

Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) [2] are the two main categories of visual servoing systems. The image features are transformed to the Cartesian 3D space in PBVS while the IBVS does not perform this transformation. The image features error in IBVS is calculated in the image plane and transferred to the controller input. This allows for higher robustness between the camera and its image error. The IBVS does not require camera calibration.

There are two major divisions for image features extraction techniques: Global Feature Extraction (GFE) [3-5], and Local Feature Extraction (LFE) [6-10]. The basic geometric characteristics of the image, such as points, corners, edges, lines, specific areas or marks, are used in the local features extraction method. This type of image features extraction is extensively applied in the visual servoing system but is restricted by several parameters, such as the environment conditions, shape and texture of the object. Researchers in visual servoing have paid more attention to global features extraction techniques in recent years.

With respect to controller selection, various stable and robust controllers have been developed for visual servoing. A vast range of machine learning algorithms [11], such as Neural Networks (NN) with Back Propagation (BP), Radial Basis Functions (RBF), Fuzzy Neural Networks (FNN) and Genetic Algorithms (GAs), are used in the design of controllers for robot manipulators.

A powerful methodology used to design and implement an autonomous robotic system is the use of learning methods. In the area of self-learning robotics the number of training samples is often limited. Self-learning can also be used in Adaptive Visual Servoing. Explanation-Based Neural Network (EBNN) learning is proposed [12] as a reliable approach for generalization that needs a lower amount of training data. This feature relies on previously learned knowledge. Research on robot learning has primarily concentrated on learning particular tasks separately. A robot normally needs to do different duties during its lifetime. This provides the opportunity to transfer knowledge between different tasks. The robot can learn the invariants of the environment and of the different tasks. This learned knowledge is task independent and is named the Domain

Theory. The Domain Theory can be applied to bias generalization in order to reduce the requirement for real world experimentation. The EBNN can be used to establish this background knowledge as domain theory.

The main limitations of traditional visual servoing systems are the need for inverse kinematics or Jacobian matrix calculation. For this purpose accurate mathematical models of the robot and camera are necessary. The main problem in existing learning-based visual servoing systems is that they mostly use a supervised learning methodology. The training is performed in off-line mode and needs thousands of training samples for convergence. Two novel visual servoing systems are proposed to overcome these problems and limitations. Not only do the proposed systems not require robot and camera models or inverse kinematics and Jacobian calculations, but they also implement an unsupervised learning approach in an on-line mode. The experimental results demonstrate the achievement of higher adaptability and flexibility to learn and accomplish accurate positioning tasks.

Two novel self-learning visual servoing systems using the Q-learning algorithm [13] are introduced in this research. In the first system Q-learning is implemented using fuzzy neural networks. EBFNN are employed for Q-learning implementation in the second system.

These systems apply Q-learning to find the optimal policy through reinforcement learning. This policy is used by the robot to reach a predetermined object that has been randomly placed in the environment. The proposed visual servoing systems do not need robot or camera models, or require calibration because they do not need a robot and camera mathematical model for their controller design. Traditional visual servoing

systems always require the calculation of robot inverse kinematics and the Jacobian matrix. The proposed systems are not based on these calculations and therefore do not need a robot model. Camera calibration is the procedure of estimating the camera's intrinsic and extrinsic parameters relating to the world coordinate system. The proposed systems are not of the position-based visual serrvoing type which needs a camera model and calibration in its design.

In the first system, the Q-learning algorithm is implemented using fuzzy neural networks to estimate the Q-evaluation function for each robot action. Each fuzzy neural network is trained using the input state and the Q-value for the basic action in on-line training episodes. The input state consists of the extracted image features. A camera mounted on the robot end-effector captures the target image at each time step and sends it to a feature extraction unit. The Harris Corner Detector algorithm is utilized for object recognition. This system learns the optimal policy which is necessary to select the best action that maximizes the cumulative reward received at each time step.

Training is performed on-line in robotic self-learning systems. A smaller number of training samples and a shorter training time to achieve a specific positioning accuracy are desirable in these systems. The second proposed system includes two main properties: on-line self training and lifelong learning that are implemented by the Q-learning algorithm and explanation-based fuzzy neural networks respectively. It will be demonstrated that the numbers of training samples (learning episodes), and therefore the training time required to achieve a specific positioning accuracy, can be reduced using fuzzy explanation-based neural networks with Q-learning when compared to fuzzy neural networks and Q-learning. Background knowledge about the robot and its environment is

transferred to the robot during the learning process using a set of previously trained neural networks.

The proposed self-learning visual servoing systems will implement new architectures and control strategies. The control strategy of robot joints is based on defined basic actions. The definition of these basic actions makes it possible to use EBFNNs to transfer knowledge for use in the visual servoing task. The focus of this work is on the necessary learning algorithms for this system. The learning algorithms include two main components: Q-learning and the EBFNN. The key point to understanding why EBFNNs are appropriate for visual servoing systems is associated with the image Jacobian that is an inherent concept in many visual servoing systems. EBFNNs incorporate the image Jacobian knowledge so that self-learning visual servoing manipulator systems based upon EBFNNs have an increased learning speed for new tasks using this inherent knowledge.

The proposed visual servoing systems can be used when a robot manipulator is required to perform several tasks for different applications during its lifespan. For example, this system can be employed in a manufacturing plant for parts assembly, which consists of picking up and placement tasks for different components. These self-learning systems do not require programming for each new task and reduce engineering design time and expense.

The proposed systems can also be applied in applications in which an accurate mathematical model for the robot and camera is not available. For example, in planetary exploration a rover-based manipulator works in an unknown environment that can affect the robot and camera parameters. These variations can cause a positioning error in

traditional visual servoing systems but the proposed system is able to learn by experimenting in the new environment and eliminating this positioning error.

1.2 Research Contributions

The major contributions of this research are as follows:

- A new self-learning visual servoing system for robot manipulators based on the neural networks implementation of the Q-learning algorithm is proposed. The Q target or evaluation neural networks are used instead of the Q tables in the Q-learning Reinforcement algorithm. These target neural networks are trained using on-line learning by random actions in learning episodes. When Q-learning is implemented using look-up tables there is no effort in its performance for generalization. This generalization means that the estimation of the Q value for unseen state-actions is derived from samples that have been seen. Using neural networks prevents rote learning and improves generalization of Q-learning [11]. In the traditional Q-learning algorithm based on look-up tables, a continuous input work-space is divided into a limited number of regions. All of the input state points in one region are presented by one value in one table entry cell that generates a quantization error. Using Back Propagation neural networks instead of Q tables also makes it possible to work on a continuous state space and prevents quantization errors due to the discretization of input states. To our knowledge, this is the first research to integrate Q-learning and artificial neural networks for the implementation of a highly non-linear robot manipulator visual servoing system.

- Another contribution of this research is the use of an Adaptive Neuro Fuzzy Interface System (ANFIS) for function approximation in conjunction with the Q-learning algorithm. Q-evaluation or target networks map the image features to the evaluation function Q for each basic action. This relation is highly non-linear. The ANFIS can achieve highly non-linear mappings with high performance. The ANFIS requires fewer parameters than other network architectures, such as multilayer feed forward neural networks. This can reduce the number of training iterations and the training time. Implementation of the Q-learning algorithm using ANFIS causes the parameters of the algorithm to converge quickly to values that can accurately estimate the next optimum action.

- A self-learning visual servoing system based on the hybrid induction and analytical learning methodologies is proposed in this thesis. The concept of Explanation-Based Fuzzy Neural Network (EBFNN) is introduced, and its effects in reducing the on-line learning time and increasing the training accuracy are investigated. Trained action EBFNNs store the changes in extracted image features (states) with the changes in robot joints (actions). These changes are partial derivatives (slopes) of the image features with respect to the robot joints that build the image Jacobian. The image Jacobian contains the robot and camera model information. These trained EBFNNs are used to learn the new tasks during the on-line training. They transfer the image Jacobian knowledge stored in these networks to increase the speed of the learning process. It can be concluded that EBFNN's training constructs the image Jacobian and comprises the inherent knowledge of a visual servoing system. This knowledge is stored in EBFNNs and

can be used throughout the life of the robot. The results demonstrate that the use of EBL can improve the learning process and the real-time performance of the system. Two statistical tests: T-test and Wilcoxon-Mann-Whitney U test are applied to show the statistical significance of the results.

1.3 Thesis Organization

The thesis is organized as follows. Chapter Two introduces visual servoing approaches, vision-based control categorization, camera model in machine vision systems, camera configurations in visual servoing systems; image features extraction methods and robot manipulator kinematics and dynamics to provide some background information on these topics. The focus then moves onto the problem of definition and limitations for visual servoing systems in Chapter Three. A literature survey is prepared in Chapter Four. The proposed visual servoing systems are presented in Chapters Five and Six. In Chapter Seven a statistical analysis is presented. The conclusions of this thesis are then summarized in Chapter Eight.

CHAPTER TWO

BACKGROUND

This chapter will discuss the different visual servoing approaches, categories, methods of image acquisition, and image features extraction methods. Manipulator kinematics and dynamics are then presented as essential topics for such systems.

2.1 Visual Servoing Approaches

Visual servoing systems can be categorized according to the available information about the target object and the parameters of the camera. A calibrated method can be applied when the camera parameters are known. If there is a lack of information about the camera's parameters an un-calibrated approach must be used.

A model-based visual servoing can be employed if a 3D model of the target is on hand. A model-free system is used when there is no access to the 3D model.

2.1.1 Model-based Visual Servoing

The 3D model of the target is accurately defined in this case. The desired and current camera poses can be calculated by projecting the coordinates of a minimum of four available points on the object. These coordinates are normally determined with respect to the coordinate frame attached to the target.

The manipulator’s end-effector can be moved to a reference grasping pose using this information. A prerequisite to using this system is an accurate knowledge of the camera's parameters. This type of system is known as calibrated visual servoing.

2.1.2 Model-free Visual Servoing

The robot can still perform the positioning task when a model of the target is not available. This can be done using training while the robot is servoed to the grasping position. In the first step the camera is moved to a desired target position and a reference image of the object is taken and stored. The camera's position with respect to the target is called the reference position. When this desired goal position has been learned and the camera or object moves to a new position an error control vector is calculated using the two images of the target. If the positioning error reaches zero it means the robot end-effector is at the desired position with a pre-defined accuracy.

2.2 Vision-based Control Categorization

Vision-based robot control systems can be divided into four categories. This classification is based on the measurement of error for the control law calculation. These four groups are: position-based, image-based, hybrid, and motion-based visual servoing systems. Error computation in the position-based and the image-based visual servoing systems is performed in the 3D and 2D Cartesian spaces respectively. Hybrid or 2 ½ D visual servoing [14] is a system in which the error is partially measured in both the 3D Cartesian space and the 2D image plane. The last class is a motion-based visual servoing system [62]. In this type of system, the error is calculated by comparing a desired reference optical flow and the optical flow measured in the image.

2.2.1 Position-based Visual Servoing (3D)

Position-based or 3D visual servoing calculates the error in its control system by using the position of the camera as depicted in Figure 2.1 [58]. The model of the object is used

to calculate the position error. It depends on the features which are available in the target image. This type of visual servoing system has the advantage that the camera trajectory is directly controlled in Cartesian space.

In this system there is no control to limit the movements of the image features inside the image boundaries used for the pose calculation. This problem is more probable if the robot or camera calibration is not accurate. In case of errors in the 3D model of the object or in camera calibration, the reference and the current camera poses cannot be estimated precisely and visual servoing fails.

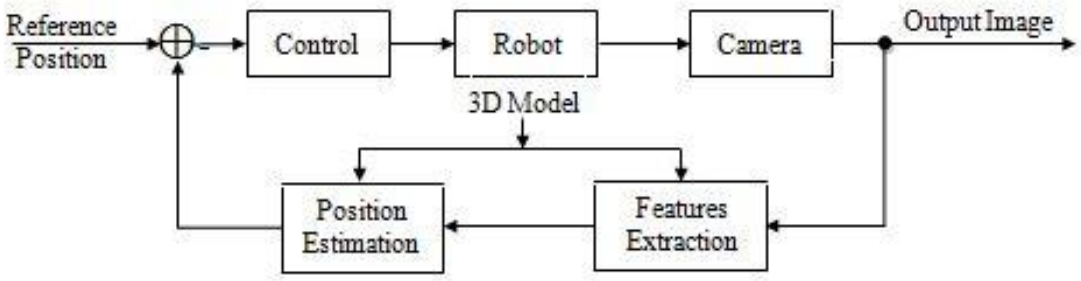


Figure 2.1: Model-based 3D visual servoing

2.2.2 Image-based Visual Servoing (2D)

The image-based or 2D visual servoing is a model-free control system which does not need a 3D model of the target. The control law is based on the calculation of error in a 2D image plane as illustrated in Figure 2.2 [58].

Image-based visual servoing is robust against robot and camera calibration errors. The convergence of the system in an area around the reference position is guaranteed in theory. A general stability analysis against calibration errors is not possible due to the nonlinear and coupled system.

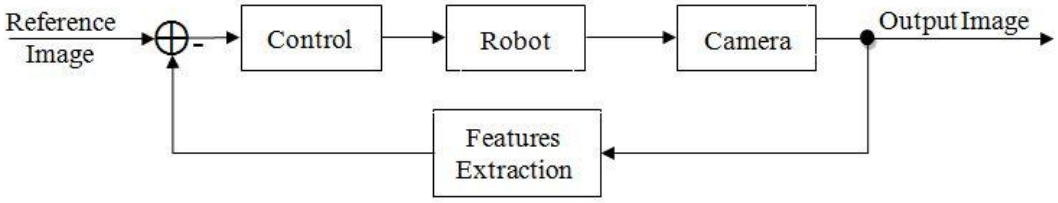


Figure 2.2: Image-based 2D visual servoing

2.2.3 Hybrid Visual Servoing (2 ½ D)

The main disadvantage of model-based visual servoing systems is that the target object may fall outside the camera's field of view since there is no control of the image. In this system a model of the target is needed to calculate the pose of the camera. Image-based visual servoing does not require the target model but a depth computation is unavoidable in its control law calculation. The major problem of the image-based visual servoing is that convergence can be only guaranteed in an area close to the goal position. This bottleneck can be avoided by using a hybrid visual servoing system [14]. The 2 ½ D visual servoing system does not require a 3D geometric of the target. This system can also guarantee the convergence of the control law in robot task space.

In each iteration of the control law in a 2 ½ D visual servoing, the partial camera translation between the camera reference and the existing poses is computed [14]. A decoupled control law can be schemed using visual characteristics generated from the partial translations. This system is illustrated in Figure 2.3 [58].

The robustness of this system against calibration errors has also been investigated. This analysis demonstrates an improvement of the convergence and stability in eye-in-hand hybrid visual serving when compared to position-based and image-based visual servoing systems [58].

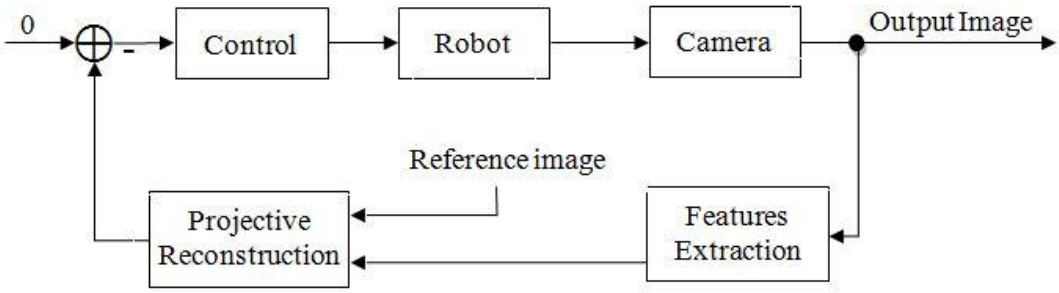


Figure 2.3: Hybrid 2 ½ D visual servoing

2.2.4 Motion-based Visual Servoing (d2D/dt)

Figure 2.4 [58] shows a motion-based visual servoing system [62]. The system's design is established using the measurement of optical flow in the image. This measurement can be done without former knowledge of the object. It can concluded that motion-based visual servoing is a model-free approach. The reference and actual motion fields in the image are compared, and an error signal is applied to the controller in an eye-in-hand motion-based visual servoing system. This system is able to perform other tasks such as docking, camera self-orientation and tracking. The main limitation of this system is the servoing rate, which enforces low robot speed. Improvements in motion estimation methods will overcome this drawback to build systems with a faster control loop.

Recent research on visual servoing can be classified into two major categories: position-based visual servoing (PBVS) and image-based visual servoing (IBVS). Detailed research on PBVS and IBVS and their differences is performed in [15]. Image-based visual servoing removes calibration and modeling errors, eliminates the need of image explanation and decreases the computational delay in the control loop. Researchers have used this system more frequently in recent years.

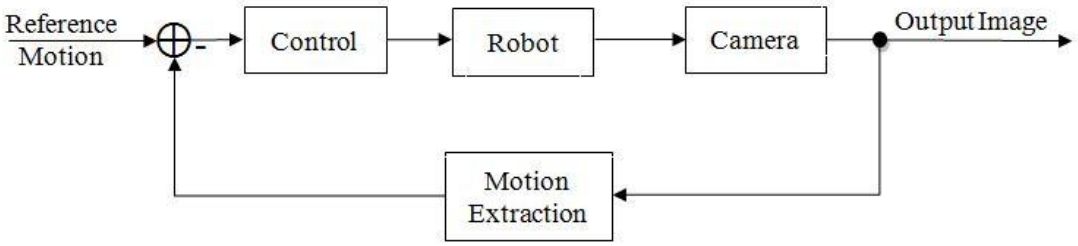


Figure 2.4: Motion-based d2D/dt visual servoing

2.3 Camera Model in Machine Vision Systems

The main task of a pinhole camera is perspective projection of a point in 3D coordinates to the image plane. It can be assumed that the camera image plane is like a matrix. The matrix is built of cells that are sensitive to the light. The size of the image plane matrix determines the resolution of the image. Each cell is named a pixel. The camera measures the intensity of the light for each pixel with coordinates (u, v) in an image plane. Figure 2.5 [58] illustrates a 3D point with homogeneous coordinate's $\boldsymbol{\psi} = (X, Y, Z, 1)$ that is projected to a point in an image plane with homogeneous coordinates P = (u, v, 1).

$$\boldsymbol{P} \propto \begin{bmatrix} \boldsymbol{K} & 0 \end{bmatrix} \boldsymbol{\psi} \tag{2.1}$$

The camera intrinsic matrix $\boldsymbol{K}$ is defined by the following Equation [58]:

$$\boldsymbol{K} = \begin{pmatrix} \lambda\kappa_u & \lambda\kappa_u.\cot(\varphi) & u_0 \\ 0 & \lambda\kappa_v/\sin(\varphi) & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

In this matrix the pixels coordinates of the principle point is shown by u_0 and v_0. The camera's focal length is λ. The u and v axes scaling parameters are specified by k_u and k_v with the pixels/meters unit. The angle between the u and v axes is φ. It can be

assumed that the square pixels for the majority of commercial cameras are $\varphi = \pi/2$ and $k_u = k_v$.

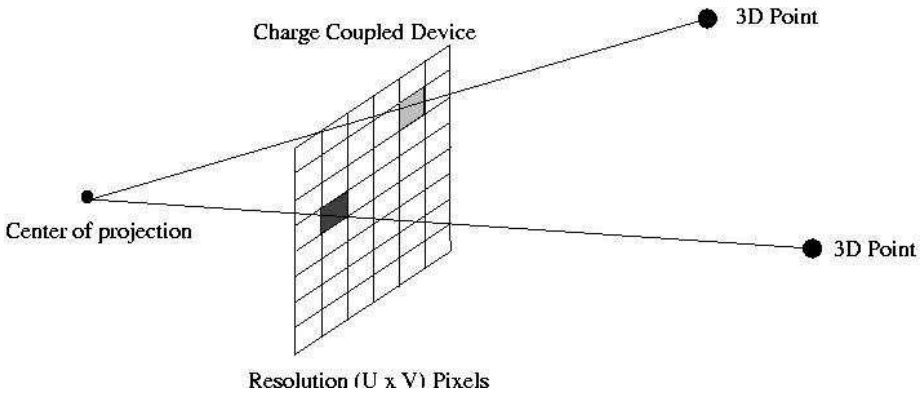


Figure 2.5: Camera model

The relationship between the image plane and object coordinates for a simplified perspective projection vision system can be expressed as [2]:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{\lambda}{z}\begin{pmatrix} x \\ y \end{pmatrix} \tag{2.3}$$

In this equation all coordinates are in the frame of the camera's coordinates. The values of the intrinsic parameters are seldom precisely known. Accurate calibration of the camera parameters is a tedious task which needs a special calibration grid [16].

The preference is to estimate the camera's parameters without using a model of the seen object. It is feasible to apply a self-calibration algorithm [17] to find the intrinsic parameters when numerous images of any inflexible object are available.

2.4 Camera Configurations in Visual Servoing Systems

The focus of this section is on camera configurations and image processing techniques in visual servoing systems. Visual servoing systems apply visual feedback in their control loop. The error measurement can be done in the camera image plane. This includes image differencing and optical flow methods. In another approach the camera parameters and some prior knowledge about the observed image features are used to estimate the pose of the object. Camera configuration, the number of cameras in the system, the required calibration accuracy and prior information about the work-space determine which method is more convenient. The following sections explain the visual servoing system's categories related to the number of cameras and their configuration.

2.4.1 Monocular Vision

Monocular vision systems utilize one camera either in stand-alone or as eye-in-hand configuration. The model-based visual servoing approach is often selected to smooth the process of depth estimation. In the stand-alone configuration the camera is the global sensor. In this case the complete pose of the target can regularly be estimated using a geometric model of the object. Feature-based and window-based tracking methods are more popular than the eye-in-hand configuration. The advantage of the monocular vision system is that a single camera minimizes the required processing time to extract visual characteristics. The disadvantage of this system is the lack of depth information, which restricts the types of executable servoing functions. It also requires that the design of the control system is more sophisticated. The eye-in-hand is one of the most common configurations. In this configuration the camera is rigidly attached to the end-effector. Primary knowledge about the transformation between the camera and the end-effector

coordinate frames is usually required. The main function is to move the camera so that the image features match a set of desired reference image features. The system can monitor both the current and the desired image features in each iteration. A common task in stand-alone vision systems is to estimate the pose of the object relative to the camera or the robot frame. This type of system requires an accurate camera, as well as calibration between the camera and the manipulator. The stand-alone configuration provides a wider field of view when compared with eye-in-hand vision systems.

2.4.2 Binocular Vision

Two cameras are used in a binocular vision system. This system is a stereo configuration of two cameras and is able to extract comprehensive 3D information about the scene. Calculation of disparity is a basic technique for depth estimation. The analogous features of two or more images are matched to estimate the disparity. This method applies to matching images of related areas or features such as edges and corners. Unlike monocular systems, this system does not need explicit models for depth. The disadvantage of this system is that its computational time will double for each iteration. Many visual servoing systems use this technique regardless of this drawback. Both stand-alone and eye-in-hand configurations can be used with this type of system. The eye-in-hand arrangement is rarely applied in visual servoing. This system may facilitate the depth estimation but the reconstruction accuracy can be reduced due to the restricted baseline. This technique is widely used to build a wireframe pattern of the object. The stand-alone configuration is generally used in binocular vision systems. The baseline can be extended enough to achieve a high accuracy in depth estimation. Another advantage of this system is a wide field of view. It is possible to observe the robot and the object at the same time. A linear

model of camera can be selected if the camera views the work-space from a large distance. In 1989 Andersson [63] proposed one of the initial stereo visual servoing systems. The specific application that he focused on in his research was a ping-pong playing robot. The system was position-based and was precisely calibrated. He utilized image color segmentation and a dynamic model for detecting the ball and extracting its trajectory respectively.

2.4.3 Redundant Camera Systems

Redundant vision systems employ multiple cameras. They generate supplementary information when compared with the monocular and binocular vision systems [59]. Visual servoing systems that use more than two cameras in their structure are rare because the computational time to match multiple camera sights is too high.

2.5 Image Feature Extraction Methods

Image feature extraction techniques are divided into two main categories: local feature extraction and global feature extraction. Local feature extraction is typically based on the detection of basic geometric characteristics, such as points, corners, edges, lines, regions or artificial marks on the object [64]. This approach is limited to the shape, texture and occlusion of the object. Environmental conditions such as lighting and noise can affect the performance of this system. Researchers have gradually paid more attention to global feature extraction methods in recent years. Global feature extraction includes Fourier descriptors, geometric moment, stochastic transform, optic flow and Eigenspace algorithms [5], [64]. Visual servoing systems which apply global features extraction, do not require calibration or previous information about the robot and the camera. The

reports show that this type of system is less accurate in comparison with visual servoing systems based on local feature extraction. The application environment is one of the major factors considered when selecting between a local or global feature extraction approaches.

2.5.1 Local Image Features Extraction

Local geometric characteristics are typically selected as image features. These characteristics consist of the points, lines or other geometric features in an image. One of the most common image features that can be selected for image feature extraction in vision-based control systems are points. Points in the image correspond to corners, holes, region centers or particularly designed points on the object. A Harris Detector [18] is a powerful algorithm for extracting the desired points from an image. This algorithm is also used for extracting image features in the proposed systems. Other image features, such as straight lines, ellipses and contours can be extracted from the object image to use in the control system. Canny [19] has proposed a more common algorithm to detect contours in the image. An essential problem in machine vision is matching features. Model-based visual servoing requires a match between the current image features and the reference model features. Image features are matched between recent and reference views in model-free techniques. Matching images with different resolutions is also compulsory when the camera is zooming.

2.5.2 Global Image Features Extraction

Often local geometric characteristics of an image cannot be extracted consistently due to variations in factors such as illumination and surface reflectance. One solution can be the use of artificial marks, but this is not always practical in the real world. Several global

image features extraction methods are presented here, such as Fourier descriptors, eigenspace methods, geometric moments, optic flow, stochastic transform, and appearance from subspace methods. Eigenspace methods based on the Principle Component Analysis (PCA) [65] are used for image compression and have been applied more often in recent years than other technique for global image features extraction.

2.6 Robot Manipulator Kinematics and Dynamics

The dynamics of an n DOF robot manipulator is explained by the following equation that is a set of highly non-linear and strongly coupled second order differential equations [20]:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \boldsymbol{G}(\boldsymbol{\theta}) \tag{2.4}$$

Where $\boldsymbol{M}(\boldsymbol{\theta})$ is the *n* x *n* mass matrix of the manipulator, $\boldsymbol{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is an *n* x 1 vector of centrifugal and coriolis terms and $\boldsymbol{G}(\boldsymbol{\theta})$ is an *n* x 1 vector of gravity term. These matrices are very complicated functions of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$.

Vectors $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$ are *n* x 1 for joint angular position, velocity and acceleration respectively, and $\boldsymbol{\tau}$ is *n* x 1 for a joint torque vector. Two essential problems in robot kinematics are forward and inverse kinematics. In forward kinematics, a set of robot joint angles is given and the problem is to find the corresponding location of the manipulator's end-effector. This desired location includes the end-effector's position and orientation. Forward kinematics can be interpreted as a one-to-one mapping from the robot's joint space to the Cartesian coordinate space. The forward kinematics problem can be solved by the 4x4 homogeneous transformation matrices. The Denavit and Hartenbergh model is often used for this purpose [20]. In an inverse kinematics problem, the location of the

end-effector in Cartesian space is known and the corresponding robot joint angles are calculated. Inverse kinematics has a multi-mapping feature and is a more difficult problem when compared with forward kinematics. Geometric, algebraic and numerical iterative methods are common approaches to solve an inverse kinematics problem. Some of these solutions are based on the inverse Jacobian matrix calculation, which demonstrates a mapping between the robot's joint and the task space.

The numerical iteration method can be applied to invert the forward kinematic Jacobian matrix, but this algorithm is not guaranteed to generate all of the possible inverse kinematic answers. This method involves substantial computation. When the manipulator geometry is not accurately determined the traditional methods become difficult. Visual servoing of manipulators is one of these cases.

2.7 Chapter Summary

The most important stage in designing visual servoing systems is selecting the proper type, configuration and components. This chapter discussed three main approaches to visual servoing including position-based, image-based and hybrid systems. Different camera configurations were introduced and local and global image features extraction methods were explained. Finally manipulator kinematics and dynamics as well as forward and inverse kinematics calculation were described. It was shown that decisions about the required types, methods and components of the visual servoing system can be made by considering the specific application, work-space environment and desired specifications.

CHAPTER THREE

VISUAL SERVOING PROBLEM DEFINITION

This chapter presents the problem definition for different categories of visual servoing systems. Closed-loop position control for the end-effector of a robot manipulator can be established using machine vision systems. This mechanism is called visual servoing. Information from the visual sensors is used to control the pose of the end-effector with respect to the target or its features. Two main camera configurations in these systems are generally eye-in-hand and stand-alone. The notations e, t, o and c will be used for coordinate frames attached to the robot end-effector, target, robot base and camera respectively. The relative pose of the camera with respect to the end-effector pose can be expressed by ${}^{e}T_{c}$. This transformation includes the relative position and orientation of the camera and the end-effector coordinate frames. It can be specified by the corresponding rotation and translation matrices [2]. The relative pose of the target object with respect to the camera coordinates system is symbolized by ${}^{c}T_{t}$. In a stand-alone arrangement the camera is fixed in robot's work-space. The pose of the camera with regards to the coordinate system of the manipulator is ${}^{o}T_{c}$. The transformation ${}^{c}T_{t}$ is the relative pose of the target with respect to the camera frame. The constructed target image in the camera is independent of the manipulator motion. The relationship between these poses for two camera configurations is shown in Figure 3.1 [2]. The calibration procedure must be executed before performing the visual servo task. This procedure estimates the camera's intrinsic parameters. These parameters consist of the principle point, the focal length and the pixel pitch. In a stand-alone configuration, a fixed camera's pose ${}^{o}T_{c}$ with respect to

the world coordinate system must be setup as the camera's extrinsic parameters. The hand-eye calibration is typically performed for eye-in-hand configurations to find the camera's relative pose with respect to the end-effector ${}^{e}T_{c}$.

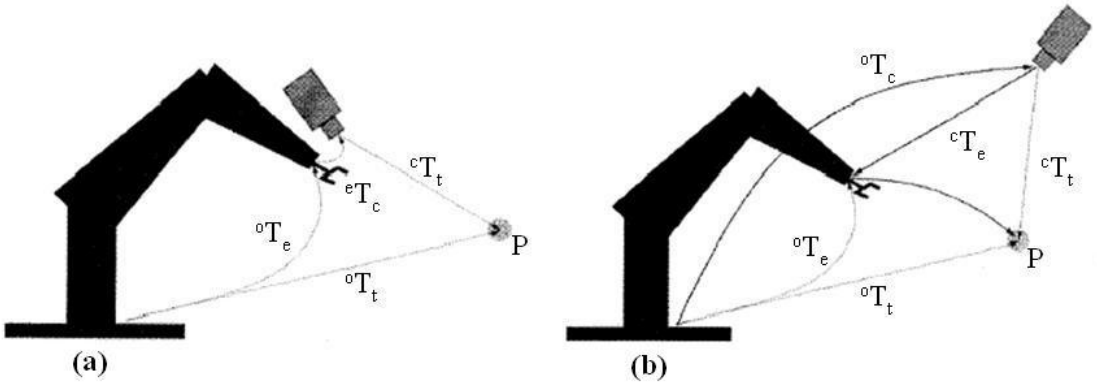


Figure 3.1: Relevant coordinate frames for the eye-in-hand and stand-alone configurations.

There has been extensive research on the calibration problem in machine vision systems. Different solutions can be found for this issue in several studies such as [21-23], and [64]. Visual servoing systems may also be categorized by position-based and image-based controls. In position-based visual servoing, the pose of the goal is estimated using a geometric model of the object and extracted image features as illustrated in Figure 3.1. Closed-loop control systems can reduce the error in pose space. The control unit executes the Cartesian control law and includes the joint controller and power amplifiers. In image-based servoing, control values are directly computed on the basis of image features. The control unit includes the feature space control law, joint controllers, and power amplifiers. The following sections will discuss these two main categories of visual

servoing systems in detail to obtain a better understanding of the problems, limitations, and potential improvement points in visual servoing systems.

3.1 Visual Servoing Problem Definition in Position-based Control Systems

The notion of a positioning task in position-based visual servoing can be formalized as follows [2]:

Definition: A positioning task is defined by mapping $E: \Psi \rightarrow R^m$ where Ψ is the task space of the robot manipulator. It consists of the positions and orientations set that the robot end-effector can reach. m is the manipulator's degree of freedom. The task space is the configuration space of the robot end-effector. When the tool is a single inflexible body that moves in a three dimensional work-space, it can be assumed that $\Psi = SE^3 = R^3 \times SO^3$ and $m = 6$. R^3 is translational space, and SO^3 is the rotational space. This mapping is termed as the kinematic error function. The end-effector pose is expressed by ${}^{o}T_{e}$. When $E({}^{0}T_{e}) = 0$, a positioning task is accomplished. A regulator in the control system can be established when an appropriate kinematics error function has been formulated and its parameters are extracted from visual information. This regulator is able to decrease the estimated rate of the kinematic error function to zero. This regulator also generates the desired end-effector velocities screw $u_c \in R^6$ at every iteration applied to the control system. In the first step, the position-based systems are considered. The desired task is to bring a point with coordinates ${}^{e}\boldsymbol{P}$ on the end-effector to a fixed stationing location $\boldsymbol{S}$. This point is observable in the work-space as is shown in Figure 3.1(b). This task is recognized as point to point positioning. The kinematic error function

calculation in the robot base coordinate can be formulated for the fixed camera configuration as [2]:

$$\mathbf{E}_{pp}({}^{o}T_{e};{}^{o}\mathbf{S},{}^{e}\mathbf{P}) = {}^{o}\mathrm{T}_{e}({}^{e}\mathbf{P}) - {}^{o}\mathbf{S} \tag{3.1}$$

Control of ${}^{o}T_{e}$ is desired. Other terms after the semicolon are parameters of the positioning task. The error function $\boldsymbol{E}_{pp}$ determines the three degrees of freedom kinematic restriction on the end-effector position. When the workplace is limited to $\Psi = R^3$, the positioning task can be assumed to be a rigid link. This entirely constrains the pose of the end-effector with respect to the target. The camera is assumed to be calibrated to the manipulator base frame. In this case ${}^{c}\hat{\boldsymbol{S}}$ is an estimation of the stationing point coordinates relative to the camera's frame. An estimation of the stationing point to the robot base frame will be ${}^{o}\hat{\boldsymbol{S}}={}^{o}\hat{T}_{c}({}^{c}\hat{\boldsymbol{S}})$, where ${}^{o}\hat{T}_{c}$ is the estimation of the camera pose in the manipulator base coordinates from an off-line calibration. The desired end-effector translational velocity in $\Psi = R^3$ is the control input $\boldsymbol{u}_{c3}$ that is needed to compute in control law. Equation (3.1) is linear with respect to the ${}^{o}T_{e}$. When there is no outside disturbance the proportional control law can be expressed as:

$$\boldsymbol{u}_{c3} = k\boldsymbol{E}_{pp}({}^{o}\hat{T}_{e};{}^{o}\hat{T}_{c}({}^{c}\hat{\boldsymbol{S}}),{}^{e}\boldsymbol{P}) = -k({}^{o}\hat{T}_{e}({}^{e}\boldsymbol{P}) - {}^{o}\hat{T}_{c}({}^{c}\hat{\boldsymbol{S}})) \tag{3.2}$$

This control law brings the system to an equilibrium state. The value of the error function at that point will be zero. The right hand side of Equation (3.2) consists of the estimated values ${}^{o}\hat{T}_{e}$, ${}^{o}\hat{T}_{c}$ or ${}^{c}\hat{\boldsymbol{S}}$. These are the manipulator kinematics, the camera calibration with respect to the robot base, and the target visual reconstruction

respectively. It can be concluded that an error in each of these components can create a positioning error of the end-effector. The translation and rotation transformation between the target and the end-effector for this case will be based on the following calculation of the homogeneous transformation matrices:

$$ {}^{e}T_{S} = ({}^{e}T_{o})({}^{o}T_{c})({}^{o}T_{S}) = ({}^{o}T^{-1}{}_{e})({}^{o}T_{c})({}^{o}T_{S}) \tag{3.3} $$

The focus is on the eye-in-hand visual servoing systems. The camera is installed on the manipulator and is calibrated relative to its end-effector as illustrated in Figure 3.1(a). Equation (3.1) can be rewritten with respect to the end-effector coordinates system:

$$ {}^{e}\boldsymbol{E}_{pp}({}^{o}T_{e};{}^{o}\boldsymbol{S};{}^{e}\boldsymbol{P}) = {}^{e}\boldsymbol{P} - ({}^{e}T_{o})({}^{o}\boldsymbol{S}) \tag{3.4} $$

The related proportional control law will be computed as:

$$ {}^{e}\boldsymbol{u}_{c3} = -k(({}^{e}\boldsymbol{P}) - {}^{e}\hat{T}_{c}({}^{c}\hat{\boldsymbol{S}})) \tag{3.5} $$

The ${}^{o}\hat{T}_{e}$ element is eliminated in the most recent equation. Thus Equation (3.5) demonstrates that the positioning accuracy is independent of the robot kinematics accuracy. The translation and rotation transformation between the target and the end-effector for this case will be:

$$ {}^{e}T_{S} = ({}^{e}T_{c})({}^{o}T_{S}) \tag{3.6} $$

The visual servoing task in position-based control systems can be described as the Cartesian pose. This is the main benefit of these systems. The feedback in position-based systems is calculated using estimated values that are functions of the calibration parameters. This is the major disadvantage of position-based control that makes these systems very sensitive to calibration error in some situations.

3.2 Visual Servoing Problem Definition in Image-based Control Systems

The positioning error in the image-based visual servoing is described as the image feature parameters. The following definition is referred to this case.

Definition: An image-based visual servoing task is expressed using an image error function $E{:}F \to R^l$, where $l \le k$. F is the image feature space and the dimension of the image feature space is symbolized by k.

Image-based visual servoing systems may employ either a stand-alone or an eye-in-hand configuration. End-effector movement generates variations in the image observed by the camera in both cases. A proper error function, E, must be defined such that when the goal is reached, $E = 0$. The vector $\boldsymbol{f}_d$ expresses the desired image features. These are image features that indicate when the end-effector has reached the goal position. The error E is a function of both the end-effector and the object poses when the task is determined relative to the moving target. The robot control input is normally defined either in task space coordinates or in joint coordinates even though the error E is stated on the image features space. It is essential to relate the variations in the image features to the changes in the position of the end-effector. The image Jacobian includes these correlations. If $\boldsymbol{r}$ and $\dot{\boldsymbol{r}}$ represent the coordinates and the velocity screw of the end-effector in the task space Ψ respectively, then:

$$\dot{\boldsymbol{r}} = \begin{pmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.7}$$

$\boldsymbol{f}$ and $\dot{\boldsymbol{f}}$ are vectors of the image features and the image features rate of change respectively. $\boldsymbol{J}_v$ is the image Jacobian matrix that represents a linear transformation from the tangent space of Ψ at coordinates $\boldsymbol{r}$ to the tangent space of F at image features $\boldsymbol{f}$. It can be written as:

$$\dot{\boldsymbol{f}} = \boldsymbol{J}_v(\boldsymbol{r})\dot{\boldsymbol{r}} \tag{3.8}$$

where $\boldsymbol{J}_v \in R^{k\times m}$ and:

$$\boldsymbol{J}_v(\boldsymbol{r}) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{r}}\right) = \begin{pmatrix} \frac{\partial f_1(r)}{\partial r_1} \cdots & \frac{\partial f_1(r)}{\partial r_m} \\ \vdots & \vdots \\ \frac{\partial f_k(r)}{\partial r_1} \cdots & \frac{\partial f_k(r)}{\partial r_m} \end{pmatrix} \tag{3.9}$$

The dimension of the task space Ψ is *m*. The number of columns in the image Jacobian matrix is determined by the robot's task. Weiss et al. [24] initially presented the concept of the image Jacobian as the Feature Sensitivity Matrix. The image Jacobian is also referred to as the B Matrix and the Interaction Matrix. The above equation illustrates how image features change with respect to variations in the end-effector pose. There is interest in determining the manipulator velocity $\dot{\boldsymbol{r}}$ required to achieve some desired value of $\dot{\boldsymbol{f}}$ in visual servoing. This requires solving the system given by Equation (3.8).

The control law in visual servoing applications normally computes the velocity screw of the end-effector where the image features rate of change $\dot{\boldsymbol{f}}$ is given as input. The following three cases can be assumed: $k = m, k < m$ and $k > m$. If $k = m$, the $\boldsymbol{J}_v$ matrix is nonsingular and the inverse Jacobian $\boldsymbol{J}_v^{-1}$ can be calculated. In this case it can

be concluded that $\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{-1}\dot{\boldsymbol{f}}$. When $k \neq m$, the inverse Jacobian $\boldsymbol{J}_v^{-1}$ does not exist. If it is assumed that the Jacobian matrix is full rank or: $rank(\boldsymbol{J}_v) = \min(k,m)$, then a least squares solution can be calculated by:

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{+}\dot{\boldsymbol{f}} + (\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v)\boldsymbol{b} \tag{3.10}$$

where $\boldsymbol{b}$ is an arbitrary vector of the proper dimension and $\boldsymbol{J}_v^{+}$ is a suitable pseudo-inverse for the Jacobian matrix $\boldsymbol{J}_v$. This solution provides a value for the velocity screw $\dot{\boldsymbol{r}}$ that minimizes the norm $\left\| \dot{\boldsymbol{f}} - \boldsymbol{J}_v\dot{\boldsymbol{r}} \right\|$. When $k > m$ the number of image feature parameters is higher than the task space degrees of freedom. The coordinates $f_{m+1}, \ldots, f_k$ can be represented as smooth functions of $f_1, \ldots, f_m$ if, according to the implicit function theorem, in some neighborhood of coordinates $\boldsymbol{r}$, $m \leq k$ and $rank(\boldsymbol{J}_v) = m$ that means the Jacobian matrix $\boldsymbol{J}_v$ is full rank. The remaining *k*-*m* visual features are redundant and lead to a set of inconsistent equations. The *k* visual features will be acquired from a machine vision system and are probably noisy. The suitable pseudo-inverse is given by:

$$\boldsymbol{J}_v^{+} = (\boldsymbol{J}_v^{T}\boldsymbol{J}_v)^{-1}\boldsymbol{J}_v^{T} \tag{3.11}$$

In this case: $(\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v) = 0$, $m = rank(\boldsymbol{J}_v)$ and the rank of the null space of $\boldsymbol{J}_v$ is 0. The solution can be rewritten more briefly as:

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{+}\dot{\boldsymbol{f}} \tag{3.12}$$

The visual servoing system is under-constrained if $k < m$. It indicates that the sufficient image features are not observed to uniquely estimate the target motion $\dot{\boldsymbol{r}}$ and

that some specific elements of the target motion cannot be observed. The suitable pseudo-inverse of the Jacobian matrix can be expressed by:

$$\boldsymbol{J}_v^{+} = \boldsymbol{J}_v^{T} (\boldsymbol{J}_v \boldsymbol{J}_v^{T})^{-1} \tag{3.13}$$

The following example can be assumed for the image Jacobian. The camera system is in stand-alone configuration. The end-effector angular and translational velocities are ${}^{c}\boldsymbol{\Omega}_e = [\omega_x, \omega_y, \omega_z]$ and ${}^{c}V_e = [V_x, V_y, V_z]$ respectively, and they are both relative to the camera frame. Suppose that ${}^{c}\boldsymbol{P} = [x, y, z]^T$ is a point on the end-effector. The camera image features vector is denoted by $f = [u,v]^T$ .The velocity of a point in the image plane can be expressed based on the velocity with respect to the camera by the following image Jacobian equation and using Equation (3.8):

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} \lambda/z & 0 & -u/z & -uv/\lambda & \dfrac{\lambda^2 + u^2}{\lambda} & -v \\ 0 & \lambda/z & -v/z & \dfrac{-\lambda^2 - u^2}{\lambda} & \dfrac{uv}{\lambda} & u \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.14}$$

λ is the focal length of the camera.

This result can be extended to the general case. The Jacobian matrix can be stacked for each pair of image point coordinates using $k/2$ image points. The Jacobian matrix in the Equation (3.14) is a function of *z*. This is the distance from the camera to the target point. In a stand-alone configuration when the target is the end-effector this distance can be calculated using the camera calibration data and the manipulator forward kinematics. The accuracy of the image-based visual servoing is independent of the calibration. These

systems also have a computational advantage. One disadvantage of the image-based technique is the existence of singularities in the feature mapping which cause unstable points in the inverse Jacobian control law. These instabilities are usually less dominant in position-based systems. Another limitation in the Image-Based Visual Servoing algorithm is the need to compute the image Jacobian on-line because it fundamentally depends on the distance from the camera to the object. This distance is especially difficult to compute in a monocular system. Many visual servoing applications apply a constant image Jacobian. This method is computationally efficient but valid only over a small neighborhood of the goal in the robot work-space.

3.3 Limitations and Problems in Visual Servoing Systems Implementation

The limitations and problems in implementing visual servoing systems is investigated in this section. There are several restrictions in visual servoing systems. Examples of these limitations include image processing algorithms with high computational times, the existence of non-linearity in camera systems, the size of the training samples dataset, and inaccurate models and sensor information. These bottlenecks can be classified to the following categories:

- Accuracy in Positioning (Grasping): A positioning error of the end-effector is always a function of the estimated quantities ${}^{o}\hat{T}_{e}$, ${}^{o}\hat{T}_{c}$ and ${}^{c}\hat{\boldsymbol{S}}$ that are manipulator kinematics, camera calibration with respect to the robot base frame, and target image reconstruction, respectively. Errors in these estimated quantities can generate a positioning error of the end-effector.

- On-line Adaption for Real-time Applications: A set of adaptive control schemes can be defined to compensate the effects of the uncertainty and errors in existing visual servoing systems. These adaptive controllers are required to overcome uncertainties in the robot dynamics and errors in the camera calibration parameters. Design of the adaptive controllers is a challenging task because the unknown parameters related to the machine vision system generate nonlinearly that can affect on the overall system dynamics.

- Work-space Area: The image Jacobian is a function of the distance from the camera to the object. This distance is especially difficult to estimate in a monocular system. Many visual servoing systems employ a constant image Jacobian that is computationally efficient but is accurate only over a small neighborhood of the target in the robot work-space. There are other solutions such as adaptive depth estimation and partial pose estimation. These methods introduce extra computations and extensively increase the complexity of the control system design.

- Robustness: Variations in camera position and adjustments, environment conditions such as lighting and illumination, object color or soiled target and background can have an effect on visual servoing performance. It is desirable to design a visual servo control system to be robust against such variations.

- Structured Work-space: The robots work in a structured work-space and execute a set of repetitive actions in manufacturing applications. The system's performance and stability cannot be guaranteed when the work-space is not well structured. Visual servoing systems are the appropriate solution to overcome this problem. A

robot using vision-based control is able to work in an unstructured environment. Most research on visual servoing of robot manipulators has been focused on free motion control in an un-structured work-space and therefore the applications are limited. There might be static or moving obstacles in an unstructured work-space, hence planning obstacle avoidance or a collision free path is an essential need. These systems can be implemented by using visual feedback in control criteria.

- Processing Speed: Visual servoing is a closed-loop discrete-time system. The frame rate of the camera can be a constraint on the sampling rate of the system. Computational capacity can also limit the processing speed of the system. The time delay in the closed-loop control system can be due to the following factors:

  - Camera charge integration time,

  - Serial pixel transfer from the camera to the image processing unit, and

  - Computational time for image features extraction.

  It is possible that visual servoing system utilizes a moderately low bandwidth communications connection between the machine vision system and the robot controller, which enters more delay. A number of robot controllers work with a sample interval that is not compatible with the sample rate of the machine vision system. This can also introduce further delay [25]. It is possible for a closed-loop control system with delay to become unstable when its loop gain is increased. Many visual servoing systems are adjusted empirically. In this method the loop gain is increased until the response time of the system becomes intolerable.

- Training Restrictions in Learning-Based Visual Servoing Systems: Visual servoing systems based on robot learning techniques need training samples. The available number of training samples needed to achieve the desired system performance is a major bottleneck for these types of systems. Learning in the minimum amount of time, and therefore the minimum amount of initial knowledge or training samples, especially for self-learning robotics systems is desirable. When the robot's duties are more complex and there is less knowledge about the problem then more training data is required for successful generalization and completion of the task. Chapters 5 and 6 propose novel visual servoing systems with a higher degree of accuracy, flexibility and adaptability in comparison with traditional and existing systems. The proposed systems are unsupervised learning-based systems in which learning is performed in on-line mode without human intervention and with a reduced number of training episodes.

3.4 Chapter Summary

In this chapter the visual servoing problem definition for robot manipulators is presented. The control criteria for two position-based and image-based system types are explained and the effect of robot kinematics, object image reconstruction and camera calibration errors on the positioning accuracy for each approach is described. Finally the problems and limitations in existing systems are investigated.

CHAPTER FOUR

LITERATURE SURVEY IN VISUAL SERVOING

4.1 Research in Visual Servoing

A literature survey of visual servoing systems is presented in this chapter. Visual servoing systems with the following features are desirable in this research:

- The main task is accurate positioning.
- The machine vision system configuration is eye-in-hand.
- The system is image-based visual servoing.
- Local feature extraction methods are used by the image processing unit.
- Self-learning based on fuzzy neural networks is used.
- The research focuses on improving the training process in the learning system.

The main task in visual servoing systems is to achieve accurate positioning. Errors in the manipulator model and kinematics, camera model and calibration, and target image reconstruction can create an end-effector positioning error. This affects the performance of the control system. The camera-in-hand configuration is preferable because it eliminates the effect of kinematics on positioning accuracy. Camera-in-hand is the most common configuration found in research from early work such as [26-27], to more recent research [28]. Visual servoing accuracy can be independent of camera calibration through the use of image-based methods. Research is referred to that employs image-based approaches for this purpose, such as [4], [5], [10], [29], and [30]. Another important issue involves the manipulator work-space region. Expanding the working area and increasing

the accuracy of the manipulator are goals that interact and can interfere with each other. There are researchers [31-32] that use both eye-in-hand and eye-to-hand cameras as complementary parts in order to gain the benefits from both types of systems, and to reach a larger work-space with a higher positioning accuracy. The eye-in-hand configuration presents a partial but accurate sight of the scene, while the stand-alone vision system is less precise but provides a global view of the work-space.

There are other parameters that affect the positioning accuracy of a visual servoing system, such as image features extraction techniques, mapping algorithms, and the manipulator controller. The positioning accuracy depends on all of these variable quantities, and upon other design and implementation parameters. Most successful applications in robotics utilize neural networks to implement some kind of signal transformation that may not be easily computed by other means. This can be due to a lack of knowledge about the underlying process or because the conventional approach would be too complicated and computationally expensive. Neural networks have been applied in the following problems in the field of robotics [33]: to solve the inverse kinematic problem, to map the non-linear relationships in dynamics as an inverse dynamics controller, to solve trajectory planning problems, to map sensory information for control and in task planning and intelligent control.

Research that has used neural networks to establish a mapping between points in the real world and the corresponding points in the image plane is presented in [8]. The industrial application of this work is automatic welding and incising using a robot manipulator. The robot works in a structured environment. The machine vision system consists of two CCD cameras looking at a work table. The goal of this application is to

control the motion of a tool mounted on the end-effector along a curve as precisely as possible. A feed forward neural network with three layers is utilized to approximate the relationships between the image coordinates and the world coordinates. This method is employed instead of using an image Jacobian calculation. Image and real world coordinates are used to train the neural network. The curve tracking system captures an image and performs image preprocessing on it. In the next step the curve is thinned and its position on the image is stored. A trained neural network estimates the position of the curve in the world frame using the input image. This system can find a curve with a different form in the camera's field of view. A Motoman UP6 robot with 6 rotational joints is used for this research. The image coordinates of the left and right cameras are $(u_1,v_1)^T$ and $(u_2,v_2)^T$ respectively for an object point in the world coordinate system with $(x_\omega, y_\omega, z_\omega)^T$ coordinates. Image coordinates vector $\boldsymbol{x} = (u_1,v_1,u_2,v_2)^T = (x_1,x_2,x_3,x_4)^T$ of two cameras are applied to the neural network as inputs. The output quantities are arranged in the $y = (x_\omega,y_\omega,z_\omega)^T = (y_1,y_2,y_3)^T$ vector. $f(t) = (1 - e(t))/(1 + e(t))$ is the activation function of the neural network hidden layer. Five hidden units are used in the hidden layer. The network architecture is Back Propagation and performs the Levenberg-Marquardt algorithm [60] for training. The curve is drawn on a plane with a measured size of about 220mm x 150mm. Training and testing samples consist of eighty-eight groups of points. Eighty groups are dedicated to train the neural network. The remaining eight groups are used for testing. The mean square error results are presented in Table 4.1, where $RMS = \sqrt{(\Delta x^2 + \Delta y^2)/2}$. In this research the RMS unit is in millimeters. The root mean-square error of each experimental point is less than 1mm. This approach does

not need accurate models and former knowledge of the camera's parameters. This method has an appropriate level of accuracy and flexibility because the neural network can operate with different nonlinear and distortions features.

Table 4.1: The mean square error result

Targets(mm):x_w	Targets(mm):y_w	Outputs(mm):x_w	Outputs(mm):y_w	RMS
63.6	0.0	63.9	-0.2	0.25
127.2	21.1	127.2	20.9	0.14
0.0	42.2	0.2	42.3	0.16
190.8	42.2	190.9	42.7	0.36
84.8	63.3	84.8	63.2	0.07
148.4	84.4	148.5	84.4	0.07
42.4	105.5	42.4	106.4	0.64
106.0	126.6	105.9	125.8	0.57

In another work [34], a visual positioning control system with an eye-in-hand configuration is proposed. In this work a feed forward neural network is used instead of a proportional controller scheme for a robot positioning task. The visual data input is transferred to the world actuator domain. Simulation results in Figure 4.1 demonstrate that this technique can rapidly decrease the positioning error to zero and retains a proper dynamic response in comparison with the proportional controller.

The experimental results are promising but could be improved in the following points: a) feed-forward networks have problems learning high dimensional non-linear mapping, b) input space dimension can be decreased to make the problem simpler, and c) development of an on-line learning network.

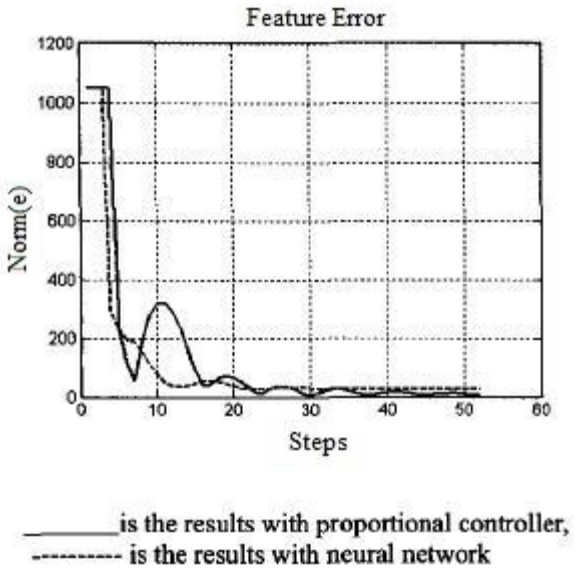


Figure 4.1: Visual positioning features error for neural network and proportional controller methods [34].

On-line adaptive visual servoing systems can respond to variations in parameters to modify their performance in real-time applications. These variations are caused by the manipulator model, vision system, and work-space environment parameters.

Another researcher employs a neural controller using the Back Propagation Learning algorithm [10]. This system utilizes vision data feedback, and the neural network maps information directly from an eye-in-hand camera to the arm control commands. This neural network controller is designed to work with an industrial robot arm. The system uses an on-line learning methodology which does not need an accurate model of the system. A series of target points that are uniformly distributed over the robot work-space are used to train the neural network. A set of experiments was implemented. Some system parameters were intentionally changed after training the neural network. The camera was turned by 0.1π in the first step and 0.5π in the second trial after

stabilizing the performance of the neural network controller. The results are depicted in Figures 4.2 (a) and (b).

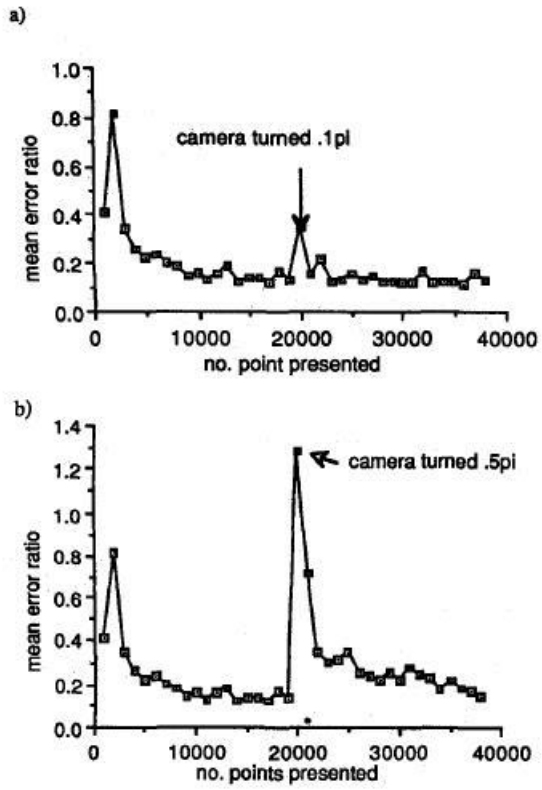


Figure 4.2: Relearning of the system after turning the camera (a) 0.1π, (b) 0.5π [10].

The system response convergence is achieved between 1000 and 3000 points when the camera angle deviation is small. The neural network requires more retraining in the case of larger changes in camera angle. These results demonstrate that the system is capable of learning the new mapping even with large deviations in camera angle. This paper shows that the on-line learning of a neural network can establish an adaptive controller for a robot manipulator. The researchers concluded that the system has two problems that prevent its industrial application. The first drawback is that the neural network controller needs a large number of learning samples which will require a long

on-line learning time. The neural network controller has the adaptability necessary to learn moderately small changes in the system parameters in on-line mode. There are several algorithms that can be employed to accelerate the Back Propagation Learning and can be implemented in order to achieve a faster convergence. A second point of concern is the inaccuracy of the system. This inaccuracy seems to be due to the use of such a network with the Back Propagation Learning algorithm. A neural network with a feed-forward topology is used to achieve these results. The potential improvement points are to increase the image sampling rate and to use a system structure with a neural network controller in the feedback path.

It is desirable to design a visual servo control system to be robust against variations in camera position and adjustments, environment conditions, object color, and background. There are a number of papers [35-36] that use global image feature extraction methods such as PCA and neural network in their system structure. These systems always lead to robust visual servoing systems for these types of variations. In [5], a *Takagi-Sugeno* Fuzzy Neural Networks Controller (TS-FNNC) [37] is presented using the image-based visual servoing technique. In this case Eigenspace based image compression is used as the global image feature transformation approach. No artificial marks are used and no previous knowledge of the camera calibration and the robot model is required. The manipulator is Motoman UP6 with 6DOF, and the camera configuration is eye-in-hand. A Gaussian membership function is selected for the TS-NNC. In this case only the area close to the fixed input point has a large membership quantity. Regions further from the fixed point have very low membership values that can be ignored. The TS-NNC is categorized using a local approximation neural network, such as a Wavelet

Neural Network or a Radial Basis Function Network. The learning algorithm for both premise and consequence parameters is Back Propagation. The visual servoing task is to transfer the end-effector to a desired pose using visual features. The target object is a metallic cylinder with a black radius on the surface. This research has the potential to create improvements in the following areas:

- The TS-FNNC stability when applied to visual servoing applications.
- Using Fuzzy Segmentation algorithm to improve the performance of TS-FNNC.
- The utilization of a global feature extraction approach with more robustness and effective computational operations.
- Implementation of a filtering technique to guarantee the robustness of the controller performance.
- Adding a Background Subtraction algorithm to improve the feature extraction operations efficiency.

Research in [3] has implemented an appearance-based visual learning system for fine-positioning applications. This system utilizes an adaptive non-linear controller for this purpose. The vision system has an eye-in-hand configuration which observes a region of about 11cm x 9cm in the x-y-plane. The Principal Component Analysis algorithm is used for image compression. This technique reduces the dimension of the primary camera images from 10,000 pixels to lower dimensions vectors that can be applied as inputs to the fuzzy neural network controller. It is demonstrated that this method builds a robust system that is stable against variations in environmental conditions. This approach does not require camera calibration and is used for three

degrees of the freedom visual servoing task. A B-spline model is selected to use in the construction of the fuzzy controllers. Off-line training and on-line evaluation are the system's two modes of operations. In the off-line mode a sequence is prepared which contains between 10 and 100 training image samples that show the object in different poses. The end-effector position in the plane is stored along with its rotation about the z-axis with respect to the optimal grasping position for each image. In the online phase the camera image is transferred into the Eigenspace processing unit and then is applied to the fuzzy controller. The output of the neuro-fuzzy controller is the amount of the end-effector's position and the angle correction (Figures 4.3, 4.4). A set of different objects, including a blue cube, a yellow cube, a partly covered yellow cube, a yellow screw head and a ledge, is selected to test the performance of the system. The ledge is a rectangular plate which has three holes along its length. The lighting conditions for all training image samples were optimal. A particular controller was trained for each object. Only for the three cubes was the training limited to the yellow one. When the target object is the ledge, different training images for y and z were used. The three largest eigenvalues are determined and their related eigenvectors were applied as inputs to the fuzzy controller.

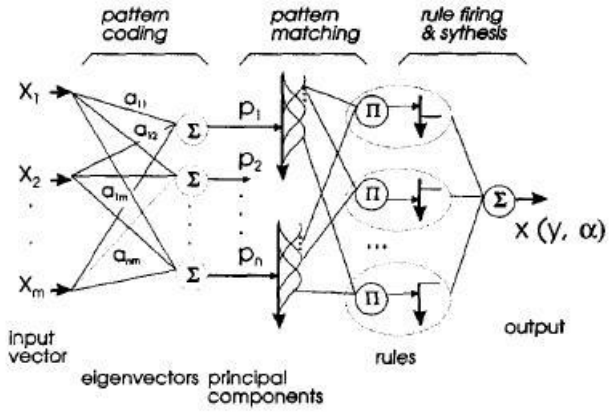


Figure 4.3: Neuro-fuzzy model for the task based mapping [3].

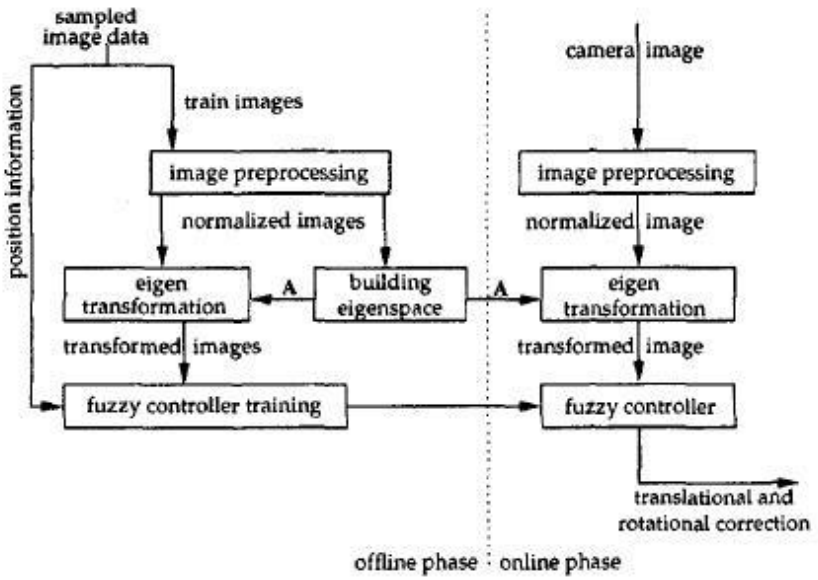


Figure 4.4: PCA neuro-fuzzy controller in off-line and on-line phases [3].

Different sets of experiments were done after training the fuzzy controller. In these experiments the five objects were positioned from the most remote start points and under optimal, worse, and poor illumination conditions. The average error of 50 positioning tasks for each experiment is used to calculate the accuracy of the control system. Tables 4.2 to 4.6 show the RMS error for x, y and the rotation angle for the positioning task of the different objects. The positioning task was apparently successful even for the blue cube with the fuzzy controller trained using only the yellow one. It can be seen that there is a slight difference in the translation if the illumination conditions are optimal or less optimal. The controller's performance gets worse in poor lighting conditions but the positioning task is still successful. The rotation is more affected by the illumination conditions, especially with the blue cube, because the vertical edges of the cube are hard to detect.

Table 4.2: RMS-error for a completely visible yellow cube under different lighting conditions

Illumination	x(mm)	y(mm)	α (degree)
Optimal	0.399	0.665	0.608
Worse	0.595	1.525	2.606
Poor	3.126	1.038	6.059

Table 4.3: RMS-error for a 20 percent visible yellow cube under different lighting conditions

Illumination	x(mm)	y(mm)	α (degree)
Optimal	0.832	1.093	0.997
Worse	0.524	2.373	1.141
Poor	6.395	4.728	19.786

Table 4.4: RMS-error for a blue cube under different lighting conditions

Illumination	x(mm)	y(mm)	α (degree)
Optimal	1.658	0.946	1.481
Worse	0.494	2.020	1.979
Poor	1.006	0.928	10.803

Table 4.5: RMS-error for a screw head under different lighting conditions

Illumination	x(mm)	y(mm)	α (degree)
Optimal	0.630	0.535	1.850
Worse	0.323	0.851	1.897
Poor	0.610	0.751	1.281

Table 4.6: RMS-error for a ledge with 3 holes under different lighting conditions

Illumination	x(mm)	y(mm)	α (degree)
Optimal	0.272	0.728	0.452
Worse	0.940	0.704	0.386
Poor	1.198	0.612	0.404

This technique has the following benefits over traditional methods:

- The vision system does not need calibration.
- This approach does not require computationally high cost algorithms for edge detection, region growing, etc.
- The B-spline interpolation and the eigenspace projection can be executed approximately in real-time.
- Object recognition procedure does not need a model and it is not required to execute particular algorithms for each object.
- The appearance-based method is robust against the changes in the object and the camera conditions. For example, if the target object is soiled or the camera focus is not properly adjusted, the system still works.

An alternative approach is to utilize a Genetic Algorithm instead of a neural network as the learning algorithm. This is presented in [38] which involves a position-based approach for visual servoing and pose estimation of an unknown target. The manipulator works in look-and-move mode to reach the object. This approach implements a Step Genetic Algorithm (Step-GA) and a pattern matching method to search for the target in the task space. The system consists of two main units. The GA

based controller and the machine vision system. The machine vision system uses the task space image and the GA-Pattern Matching algorithm. A PD controller based on the GA controls the manipulator. Each individual in the GA algorithm is encoded by means of a binary string that represents the position and orientation of the end-effector. This characterizes a candidate solution for the position and orientation of a target object in the image.

The GA uses Step-GA evolution. Every generation of the GA's operation is applied to a new input image and supplies the manipulator with the momentarily desired joint angles q_d. The path planned using GA can be performed in real time when a command vector is applied in every generation. A PD controller employs the temporary GA search results, which are computed in a short time. The end-effector can then follow the path to reach a goal, whose position and orientation are not formerly known without waiting until the end of the search procedure. The recognition system has the capability to compensate the translation and rotation errors. The GA acts as a path planner in this system. The proof of stability analysis for the controller is achieved by the Lyapumov theorem.

The population in the GA consists of 60 individuals. The selection rate is 0.4, the mutation rate is 0.01, and the length of an individual is 23 bits. Simulation results show that the manipulator tracks a path towards the object while the GA is executing the search. The termination condition for the control criteria is acquired when $q_d - q < 0.01$ *rad*. These results show that the system does not require a long time when executing the control law to move the end-effector to the position of an unknown block. This control approach may be appropriate for the real-time control of a manipulator. There are several

papers [39-41] that used GA neuro and GA Neuro-fuzzy algorithms in robotic control for visual servoing systems.

The research in visual servoing with specifications close to that of the proposed system's framework is summarized in this section. Different Machine Learning algorithms such as Neural Networks, Fuzzy Neural Networks and Genetic algorithms are used in these works. The learning process for these systems is performed off-line. A visual servoing system with a self-learning methodology would have a great advantage over these existing approaches. In the next chapters, new self-learning visual servoing systems are introduced with learning that is performed in on-line episodes without human supervision.

4.2 Machine Learning in Robotics

For robots to be really flexible they require the capability to adapt to partially known workplaces or dynamic environments. These robots need to learn new tasks and to compensate for sensors and end-effector deficiencies. The problem of robot learning is basically to employ the robots to carry out tasks without the need for explicitly programming them. It can be assumed that robot learning is a particular case of the general problem of machine learning. Machine learning is a subfield of Artificial Intelligence (AI). The ultimate objective of machine learning is to substitute the explicit programming by teaching. Teaching is usually less difficult and more efficient than programming. There are two categories of learning: supervised and unsupervised [11]. In supervised learning a teacher evaluates the output of the learner, while in unsupervised learning the learner is provided with slight or no feedback regarding the learning job. At

least three kinds of knowledge can be determined that would be valuable for robots to learn. These are Control Knowledge, Environments Models, and Sensors-effector Models. From the different sorts of knowledge it seems clear that supervised learning alone is not sufficient. For example, when a robot works in an unknown environment or has to learn a new task, it must operate in an exploratory trial and error mode. Therefore robot learning needs both supervised and unsupervised approaches.

Robot learning is a difficult machine learning problem. The following explanations are presented to clarify why robot learning is a challenging problem:

- Unreliable Sensors: There are some devices in robotic systems that do not perform in a reliable manner. This problem is regardless of the price. Sonar and laser scanners are examples of these components. It is possible for these sensors and transducers to fail to detect the target, or that there is error in their distance measurement. This drawback can affect the whole performance of the robotic system.

- Real time Performance: The robot must be able to reply to unanticipated conditions in its environment. This is particularly important for robot real time response.

- Stochastic behavior: The nature of the actions and their complexity in the real world may seldom seem to be deterministic. Sensing inaccuracies can also be another source of stochastic behavior in robotic systems.

- Learning in On-line mode: The training samples needed to teach a robot might not be accessible in the off-line mode. It is desirable for a robot to explore its

environment and to perform experiments to collect necessary samples. Incremental learning is an appropriate feature for any robotic learning approach since training data will be attained over time.

- Restricted training time: Training time for a robot cannot be too long in the real world. An extended training time is acceptable in simulations. A reasonable training time for robotic systems in the real world can be a few hours or less.

- Environment representations: A robot can perceive its work-space mainly through its sensors. A learning methodology must be capable of working with the limitations of the utilized sensors.

There are four major machine learning paradigms that can be applied to a robotic learning problem:

1. Inductive learning: This is the most classical paradigm in machine learning that includes several samples of the visual servoing systems described in the previous sections. The most well known Inductive Concept Learning algorithms are Neural Networks and Decision Trees. The principle bottleneck of inductive learning is that it needs an experienced teacher who can provide a sufficiently dissimilar set of training instances. It may be impractical to collect this training data in some applications. Another limitation of inductive learning methodology is the need for thousands of training samples for function approximators such as neural networks. One critical point is how to accelerate the speed of learning by integrating some kind of bias.

2. Reinforcement Learning: This approach investigates the problem of inducing by trial and error to find a policy that maximizes a desired performance measure or reward

[13], [42]. Reinforcement learning is an unsupervised methodology in which training samples are not categorized by a teacher. The distribution of instances is affected by the robot's actions because the robot experiences the states and rewards pertaining to the actions it performs. The robot encounters a challenging temporal credit assignment problem. This is the evaluation of the optimality for available states and actions using a scalar reinforcement feedback. This approach has several interesting features. As opposed to some kinds of Inductive Learning algorithms, it can be implemented using on-line learning. The second property is that it can be used in many applications by providing the robot with appropriate reward functions. Reinforcement learning has a number of constraints. It can be very slow and needs a very large number of iterations to converge. It is also difficult to integrate domain knowledge to expedite the learning process.

Existing research, such as [48-50], demonstrates the control of mobile robots by using neural networks in the implementation of Q-learning. In one study [48], the Reinforcement Learning algorithm is investigated to make an agent that learns from its own experience. An artificial neural network is used to assist the agent to learn in an unknown environment. A Team AmigoBot^{TM} mobile robot with 8 sonar sensors is used to locate the shortest path from the fixed starting point to the goal without hitting any obstacle. Each state corresponds to a vector of 10 parameters, which included 8 sensor readings, the relative distance from the mobile robot's current center of mass position to the goal, and the relative angle from the robot's current forward moving axis to the goal. At each time step, the robot is required to choose one of the three available actions: (1) move forward 100 mm, (2) turn 15° in clockwise direction, or (3) turn 15° in a counter-clockwise direction. The learning process is divided into two stages. In the initial stage

the agent will map the environment by collecting state-action information according to the Q-learning procedure. The second training process involves neural network training which will utilize the state-action information gathered in the earlier phase as a training samples. At the end of the training process, the robot showed a certain degree of understanding about its environment. In cases where a goal was sought from a random position, the robot also managed to reach the goal.

In another study [49], Q-learning and a multi-layer neural network are applied to integrate learning in a behavior-based autonomous mobile robot. The simulated mobile robot used in this research is similar to the Pioneer 3. The task was to find a path from various initial points to the target without colliding with any obstacles. This was to be accomplished after a period of learning by itself in an unknown environment. Six sonar sensors and one camera were used to detect the robot's situation in the environment. In goal-directed obstacle avoiding, three main behaviors are included: turn left/right and move forward. Turn left/right can be divided into several sub behaviors. The neural network used here has one input layer with 7 neurons for establishing the distance of sonar sensors and the angle between the current direction and the target. It also has one hidden layer with 18 neurons, and one output layer with 7 neurons for 7 behaviors. After hundreds of learning epochs, the robot selects the best action according to its current state. Consequently the robot can arrive at the target without colliding with obstacles, and its trajectory is very smooth.

Researchers in [50] proposed a system for a Robot Soccer simulator. The task of the simulation was to train the agent to shoot the goal using reinforcement Q-learning method with a Back Propagation neural network. Ten dimensional environmental

variables are the inputs of neural networks and three actions are available. One hundred simulation experiments were conducted and the results were compared for both the neural network implementation of Q-learning and the traditional Q-learning methods in terms of shooting success. The results showed that the proposed method is more stable and effective for strategy selection.

3. Evolutionary Learning: This is an unsupervised paradigm that includes Genetic algorithms. The capability to start with an appropriate set of policies is a significant strength of this approach. The evolutionary learning methodology permits the designer to initiate the system in a primed state which assists the speed of learning. Another benefit of this approach is that it is not limited to stationary policies and is able to learn arbitrary ones. The convergence is uncertain for general policies. A major limitation of the evolutionary approach is that it does not easily permit on-line learning. A training phase must be performed using a simulator and then the learned policy can be executed on a real robot. This restricts the method to those applications where a proper simulation is obtainable.

4. Explanation-based Learning: It seems humans have the capability to generalize from a few examples. Humans make use of a significant amount of background knowledge to a learning task. Explanation-Based Learning (EBL) [43] investigates how domain knowledge can be employed to accelerate the learning process. An EBNN algorithm is an example of the EBL approach that is proposed by Mitchell and Thrun [12]. Domain knowledge is built as an approximate model of each available action and is expressed by a neural network in this method. These networks are trained using examples collected from an earlier learning task. The initial experiments applied an EBNN to the

problem of learning target object recognition for a mobile robot in [44]. A mobile robot moves through a corridor and learns to recognize the distant doors in this research. The previously learned knowledge includes a neural network that detects the adjacent doors and another network that foresees the state of the environment after moving forward in the corridor. The results confirm that the EBNN is capable of applying this previous knowledge to considerably decrease the required number of training samples. The principal benefit of the EBL is that it makes a solution for incorporating domain knowledge to speed-up the learning process. The reduction in the number of training samples required to learn the policy can be fairly considerable. The agent is required to have previously learned some domain knowledge.

4.3 Chapter Summary

The available research regarding visual servoing systems related to our proposed system was explored. Achieving a sufficient number of training samples for all learning-based visual servoing systems, such as those investigated in this research, is an important stage in the design and implementation of the system. When a robot encounters a set of different visual servoing tasks, new training with a new dataset is required. These training stages require extra time and engineering design expenses. If there is a methodology to reduce the required training dataset size and to transfer the learned knowledge between the different visual servoing tasks it can reduce the major bottleneck effect during the learning process of such systems. On-line learning with a minimum learning time, and therefore a minimum number of training samples, is especially desirable for the self-learning visual servoing systems.

Reinforcement learning is an appropriate approach for self-learning systems among other machine learning paradigms, but its convergence can be very slow. EBL can be a proper solution to increase the speed of learning using previous domain knowledge. If a methodology is created which incorporates the Reinforcement learning and EBL, it can benefit from the strengths of both of these approaches, and speed up the convergence of the Reinforcement learning algorithm.

The focus has been on improving the training sample numbers, and consequently reducing the learning time in self-learning visual servoing systems using Reinforcement learning and EBL. New self-learning visual servoing systems are proposed in the next chapters to reduce the size of the training dataset, and therefore shorten training time to achieve a specific positioning accuracy.

CHAPTER FIVE

SELF-LEARNING VISUAL SERVOING OF A ROBOT MANIPULATOR USING Q-LEARNING ALGORITHM AND FUZZY NEURAL NETWORKS

5.1 Introduction

A new self-learning visual servoing system for a robot manipulator is proposed in this chapter. The system uses reinforcement learning, and Q-learning, to find the optimal policy. This policy is applied by the robot to reach a predetermined object that has been randomly placed in the environment. The Q-learning algorithm is implemented using fuzzy neural networks to estimate the Q-evaluation function for each robot action. Each network is trained using the input state and the Q-value for the basic actions in on-line training episodes. The input to the system consists of extracted image features. A camera mounted on the robot's end-effector captures the target image in each iteration and sends it to a feature extraction unit. This system learns the optimal policy in order to select the best action that maximizes the cumulative reward at each time step. This learning approach does not use robot or camera models, or require calibration.

Designing robotic systems that are able to learn for executing complicated real-world tasks is still a challenging problem in the area of robotics and machine learning. Most robots utilize particular controllers that have been accurately designed using broad domain knowledge. Designing the robot controller needs previous information about the robot, its kinematics, dynamics, working environment and the desired tasks. Such designs include writing the code for the hardware by hand. Generating the sensors and environment models require a large amount of programming effort. The complexity of programming the robot's tasks increases with the complexity of the hardware and sensors,

and also when the robot must work in a less predictable environment. Reinforcement learning can overcome some of these limitations by enabling a robot to experiment and learn from its environment. Reinforcement learning responds to the problem of how an autonomous agent that senses and performs tasks in its environment can learn to select the optimal actions required to reach its goal.

The Q-learning algorithm represented by Watkins and Dayan [45] is extensively employed to implement reinforcement learning. The Q-learning algorithm is selected from among other available algorithms to implement the on-line learning procedure. This algorithm has the benefit that it can be utilized when the agent has no previous knowledge of how its actions affect its environment. In Reinforcement learning, the framework is based on the Markov Decision Process. There are other Reinforcement learning algorithms such as Temporal Difference Learning and Monte Carlo methods. Temporal Difference algorithms learn by reducing discrepancies between estimates made by the agent at different times. Q-learning reduces the difference between the estimates of a state and its immediate successors. In comparison, Temporal Difference algorithms reduce discrepancies between estimates about this state and one that is more distant ahead. This makes the algorithm more complex and increases the computational cost. The Monte Carlo methods learn from simulated experience and solve a reinforcement learning problem based on averaging sample returns. In the Monte Carlo methods, convergence is always slow and only works in the small and finite Markov Decision Process. In Q-learning the environment consists of a set of states for the agent and a number of actions which the agent can carry out. A learning episode is a sequence of randomly selected states and actions. Each learning episode starts from a random point in

the robot's work-space. The agent initially observes the current state at every time step and then randomly picks an available action to perform. The agent receives an immediate reward or penalty based on its new state which reflects the desirability of the executed action. The old state is substituted with the new state and this process continues during training and builds a learning episode.

The agent learns from the feedback and builds an evaluation function based on the states and actions. This evaluation function is determined as the maximum discounted cumulative reward the agent can attain. If the agent reaches the target during the learning episode then the evaluation function will be equal to its maximum value. If the agent fails to reach the target the evaluation function will be equal to its minimum value. With image-based visual servoing (IBVS) the dynamics of the system are usually expressed as a Jacobian matrix that relates the variations in the image features to the changes in the robot's joint angles.

Various stable and robust controllers have been investigated for use with this task. A wide range of Machine Learning algorithms such as Neural Networks, Radial Basis Functions, Fuzzy Neural Networks and Genetic Algorithms are employed in the controller design of robot manipulators [5]. The learning process for these systems is performed off-line. During this training process the robot is moved to different positions near the desired target point. An image is the captured and saved with the corresponding robot joint angles for each end-effector. Image processing is then executed to extract the desired features from the stored images. The extracted image features and the corresponding robot joint angles are then used as the training samples for the neural network.

The proposed system is an IBVS which does not require Jacobian matrix calculations. On-line training is applied which differs from many of the existing systems mentioned above. The robot executes a random basic action to move its end-effector and the camera to a new position at every time step. Each basic action affects one of the robot's joint rotations by a determined angle. For example, the joint J1 rotation with an angle of 5º can be defined as a basic action. The camera then captures an image which has features extracted from it. The Q-evaluation value for each basic action is then calculated. This value includes both an immediate and a delayed reward. The image features and related Q-values for each basic action are combined to create a training dataset of input and output pairs which are used for each Q-evaluation fuzzy neural network.

The evaluation function in the traditional Q-learning algorithm is expressed as an explicit look-up table with a table entry for each separate input value. This is the main constraint with Q-learning the algorithm will not perform generalizations. It means the algorithm does not try to estimate the Q-value for an unseen state-action pair by those that have previously been observed. A second limitation in Q-learning is that the work-space must be divided into a finite number of discrete regions for use as input states. When a random input state is converted to one of these discrete states it introduces a quantization error to the system. In this research the implementation of the Q-learning algorithm uses neural networks instead of look-up tables. Using a neural network in this way prevents rote learning and allows for generalization of previously unseen state-action pairs. Using a neural network in this manner can also eliminate the effects of quantization error. There is research [46-47] on visual servoing which has used a table implementation

of Q-learning. Previous works that have applied neural networks with Q-learning for mobile robots are detailed in papers such as [48-50], but to the author's knowledge, this is the first work to integrate Q-learning and artificial neural networks for implementation of a highly non-linear robot manipulator visual servoing system.

5.2 Visual Servoing System Architecture

The architecture of the proposed visual servoing system is illustrated in Figure 5.1. The main parts of the system consist of the Q-learning evaluation unit, two sets of ANFIS for centering and reaching actions, the robot arm and robot controller, and the CCD camera and image feature extraction unit.

The Q-learning evaluation unit is used in the learning phase. This unit is divided into two units that work independently and calculate the Q_C and Q_R evaluation values for centering and reaching behaviors.

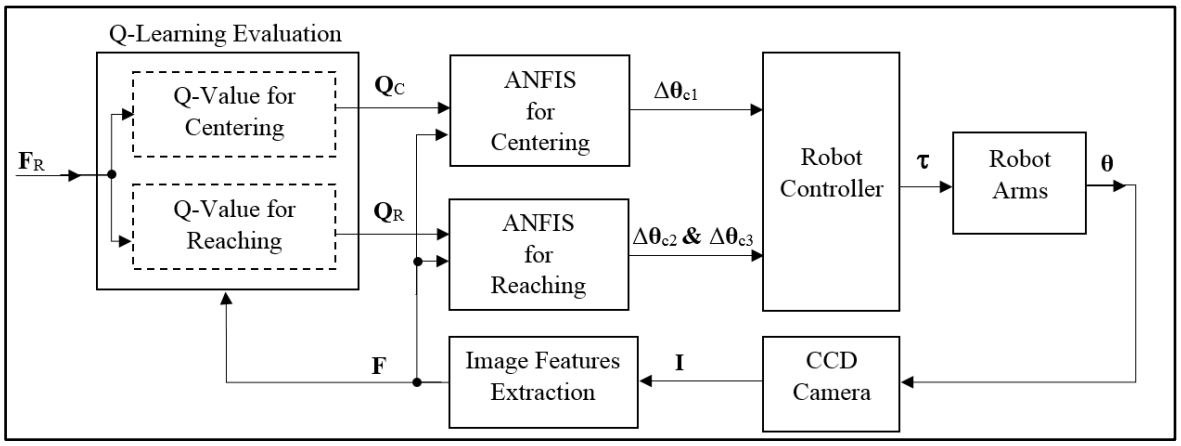


Figure 5.1: Architecture of the proposed visual servoing system

The inputs to this unit are the reference state and the current state of the agent. These are the target image features and current image features respectively. The target image features are captured by the camera-in-hand when the end-effector is at the desired

grasping position. The Q-learning evaluation unit estimates the immediate and delayed rewards at each time step based on the target and the agent's current state. The Q_C and Q_R values at each time step are calculated and sent to the centering and reaching neuro fuzzy inference systems as training output samples.

The image feature extraction unit sends the related training input samples to the neural network. The outputs of the centering and reaching neural networks, $\Delta\theta_{c1}$, $\Delta\theta_{c2}$, and $\Delta\theta_{c3}$, are the command signals, which are sent to the robot's joints 1, 2, and 3. The robot controller receives these centering and reaching command signals and generates the proper torque vector $\boldsymbol{\tau}$ to control the relevant robot joints at each time instance.

In the real-time visual servoing phase, the current image features are applied to the set of trained neuro fuzzy networks for the centering and reaching action. The basic action with the highest ANFIS Q-value output is selected and executed which transfers the end-effector to the new state. This sequence will continue until the goal state is reached.

5.3 Proposed Learning Algorithm

Learning occurs on-line and is performed using random episodes. After every instance that the robot performs a basic action, the extracted image features and calculated Q-value are sent to the corresponding ANFIS as input/output training samples. The ANFIS on-line learning is based on the Recursive Least-Squares Estimator (LSE) algorithm [37]. The components of the learning algorithm are explained in the following sections.

5.3.1 Q-learning Algorithm

The task of the agent in Q-learning is to learn a policy, π: $S \rightarrow A$, for selecting its next action a_i based on the current observed state $\boldsymbol{s}_i$; that is, $\pi(\boldsymbol{s}_i)=a_i$. This optimal policy produces the greatest possible cumulative reward for the robot over time. If the discounted cumulative reward function is defined from the state $\boldsymbol{s}_i$ as follows [11]:

$$V^{\pi}(\boldsymbol{s}_i) = \sum_{k=0}^{\infty} \gamma^k r_{i+k} \tag{5.1}$$

then the agent is required to learn the policy π that maximizes $V^{\pi}(s)$ for all states $\boldsymbol{s}$. The γ is the discount rate parameter and $0 \leq \gamma < 1$. This optimal policy is denoted by π^* [11]:

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(\boldsymbol{s}), (\text{All } \boldsymbol{s}) \tag{5.2}$$

The evaluation function $Q(\boldsymbol{s},a)$ is specified in order that its value is the maximum discounted cumulative reward that can be reached starting from state $\boldsymbol{s}$ and performing action a as the first action. Equation (5.2) can be rewritten in terms of $Q(\boldsymbol{s},a)$ as [11]:

$$\pi^*(\boldsymbol{s}) \equiv \arg\max_{a} Q(\boldsymbol{s}, a) \tag{5.3}$$

The Q-evaluation value update is based on the Bellman Equation [51] as follows:

$$Q^{\pi}(\boldsymbol{s}_i, a_i) \longleftarrow (1-\lambda)\, Q^{\pi}(\boldsymbol{s}_i, a_i) + \lambda(r_i + \gamma \max_{a} Q^{\pi}(\boldsymbol{s}_{i+1}, a)) \tag{5.4}$$

where λ is the learning rate and $0 \leq \lambda < 1$. A separate Q-learning process is executed for each of the centering and reaching behaviors. The first joint J1 rotation undertakes the centering behavior. The purpose of centering is to move the robot's arms by rotating J1 until the center of target image lies inside a boundary close to the camera's Y axis. Six basic actions are dedicated for centering. These are J1 rotation with: $\pm 1^{o}$, $\pm 5^{o}$ and $\pm 10^{o}$. The second and third joints, J2 and J3, apply the reaching behavior. There are a total of

eight reaching basic actions which include J2 and J3 rotations with: ±1º and ±5º that are selected empirically. The aim of these actions is to move the end-effector and its attached gripper closer to the target. The reaching task is successful when the gripper is in position to grasp the object, and when the maximum error between the captured image features and the desired target image features is less than a predefined limit. This predefined limit can be determined based on the desired positioning accuracy. In this work a successful accuracy for centering and approaching the target is set to 0.01 as seen in Equations (5.8) and (5.9). This limit is equal to 10 pixels or a 0.1mm error in the camera image plane. When the goal position is reached the gripper mechanism will be actuated to grasp the target object. The three waist, shoulder and elbow joints of the robot are used in this research for the positioning task, but the proposed methodologies are not limited to three degrees of freedom and can be extended to higher number.

For each iteration in a learning episode, one basic action is randomly selected using the Boltzmann distribution [52]. The probability of selecting an action a_i at state $\boldsymbol{s_i}$ is determined by the following formula:

$$p(a=a_i|\boldsymbol{s}=\boldsymbol{s_i})=\frac{e^{Q(\boldsymbol{s_i},a_i)/T}}{\sum_a e^{Q(\boldsymbol{s_i},a)/T}} \tag{5.5}$$

where T is the Boltzmann constant that can specify the degree of exploration and exploitation of the algorithm. Smaller values of T cause basic actions with higher Q-values to be selected which leads to more exploitation. Larger T values permit the algorithm to choose basic actions with smaller Q-values which results in more exploration.

The immediate reward $r_i=r(\boldsymbol{s_i},a_i)$ in the above formula helps the algorithm find the optimal solution in a smaller numbers of time steps. This is because the agent receives direct information about the error variations in the image plane. If $\boldsymbol{F_t}$, $\boldsymbol{F_i}$ and $\boldsymbol{F_{i+1}}$ represent the target, the current, and the next image feature vectors respectively, then the immediate reward functions for the centering and reaching actions r_{ci} and r_{ri} at ith state can be calculated from the following formula:

Centering immediate reward:

$$Err_{i+1}=\text{avg}(\boldsymbol{F_{i+1}})-\text{avg}(\boldsymbol{F_t}) \tag{5.6}$$

$$Err_i=\text{avg}(\boldsymbol{F_i})-\text{avg}(\boldsymbol{F_t})$$

$$K_{i+1}=100/(1+c_1|\,Err_{i+1}|)$$

$$K_i=100/(1+c_1|\,Err_i|)$$

$$r_{ci}=c_2(K_{i+1}-K_i)$$

Reaching immediate reward:

$$\boldsymbol{Err_{i+1}=F_{i+1}-F_t} \tag{5.7}$$

$$\boldsymbol{Err_i=F_i-F_t}$$

$$K_{i+1}=100/(1+c_1\sum_{1}^{f}|\,\boldsymbol{Err_{i+1}}\,|)$$

$$K_i=100/(1+c_1\sum_{1}^{f}|\,\boldsymbol{Err_i}\,|)$$

$$r_{ri}=c_2(K_{i+1}-K_i)$$

where c_1 and c_2 are constants. The values for c_1 and c_2 in the centering and reaching equations are c_1=10, c_2=1.6 and c_1=2, c_2=3 respectively. f is the number of extracted

image features. The criteria to determine the Q-evaluation values at each time step in the centering and reaching learning episodes are summarized in Equations (5.8) and (5.9).

$$Q_c(\boldsymbol{s_i}, a_i) = \begin{cases} 100, & |Err_{i+1}| \le 0.01 & \text{Success} \\ -100, & \min(F_{i+1}) \le 0 \text{ or } \max(F_{i+1}) \ge 1 & \text{Fail} \\ \text{Equation (5.4)} & & \text{Otherwise} \end{cases} \quad (5.8)$$

$$Q_r(\boldsymbol{s_i}, a_i) = \begin{cases} 100, & \max|\boldsymbol{Err_{i+1}}| \le 0.01 & \text{Success} \\ -100, & \min(\boldsymbol{F_{i+1}}) \le 0 \text{ or } \max(\boldsymbol{F_{i+1}}) \ge 1 & \text{Fail} \\ \text{Equation (5.4)} & & \text{Otherwise} \end{cases} \quad (5.9)$$

5.3.2 Adaptive Neuro Fuzzy Inference Systems

In this research Adaptive Neuro Fuzzy Inference Systems (ANFIS) are used to implement the Q-learning algorithm. Q-evaluation or target networks map the image features to the evaluation function Q for each basic action. This relation is highly non-linear.

Hybrid Neuro-Fuzzy Systems (NFS) merge artificial neural networks and fuzzy systems in a synergetic approach. Fuzzy systems build a framework to symbolize inaccurate information and to reason with this sort of knowledge, while neural networks improve fuzzy systems with the ability to learn from input-output examples. Learning methods are applied to adjust the parameters of the fuzzy system. NFSs have become very popular in the last decades mainly due to their capabilities as universal function approximators.

There are many kinds of neuro-fuzzy network architectures. The Adaptive Network based on Fuzzy Inference System (ANFIS) architecture is considered to demonstrate the hardware implementation of neuro-fuzzy systems. Jyh-Shing Roger Jang [37] has

presented the architecture and learning procedure of ANFIS. The Explanation-Based Fuzzy Neural Networks (EBFNN) have been introduced in this research. The reasons that using the EBFNN can lead to a higher performance of the proposed self-learning visual servoing systems when compared to the EBNN are:

- The ANFIS can achieve a highly non-linear mapping. The relationship between the extracted image features and the robot's joints space is highly non-linear and the EBFNN can establish this non-linear mapping with greater accuracy.
- The ANFIS always requires fewer adjustable parameters in comparison to other networks such as MLP. This can reduce the required training iterations and time. Fuzzy neural networks are also used for target networks that map the captured image features onto the evaluation function Q. This relation is also non-linear and these networks are trained in on-line mode during the Q-Learning process in order to minimize the training time and the number of training samples.
- Although not according to prior knowledge, the primary parameters of the ANFIS are intuitively rational and all the input space is covered. This leads to a fast convergence with proper parameter values that capture the fundamental dynamics and is suitable for this application.
- The ANFIS is comprised of fuzzy rules which are local mappings or local experts as opposed to the global experts. These local mappings support the minimal disturbance principle [61]. This rule states that the adaption should not only decrease the output error for the present training sample but also

minimize disturbance to the response to what has been previously learned. This is mainly important in on-line learning. Using the least-squares to estimate the output of each local mapping is particularly significant. The typical learning time would have been 5 to 10 times longer without the use of LSE.

The ANFIS is therefore a good choice for Q-evaluation networks. The first-order Sugeno fuzzy model is utilized for this work. The structure of the neuro fuzzy model is shown in Figure 5.2. The figure shows an ANFIS with four inputs and two membership functions. The network used for this work has five membership functions and will have five nodes attached to each input node. The selection of this number of membership functions is a compromise between the mapping accuracy and the required processing time for the application. The first-order Sugeno neuro fuzzy network consists of five layers, as seen in Figure 5.2. These five layers include fuzzy membership functions, rules firing strength calculations, normalization, rule outputs, and overall network output. Each ANFIS has four inputs that are extracted from image features. These inputs are extracted point coordinates from the object in the image plane. The ANFIS output is the Q-evaluation function value. There is one ANFIS for each of the centering and reaching actions.

The total number of centering and reaching ANFIS is six and eight respectively. There are two sets of network parameters that must be estimated during learning. The first group is the premise parameters that define input membership functions. A bell-shaped membership function with three parameters is used. These parameters control the centre, width, and slope of the membership function. This type of membership function has a smooth shape with a low number of parameters and is useful for highly

non-linear mappings. The second group is the consequent parameters. These are four layer parameters that have a linear relationship with the network output.

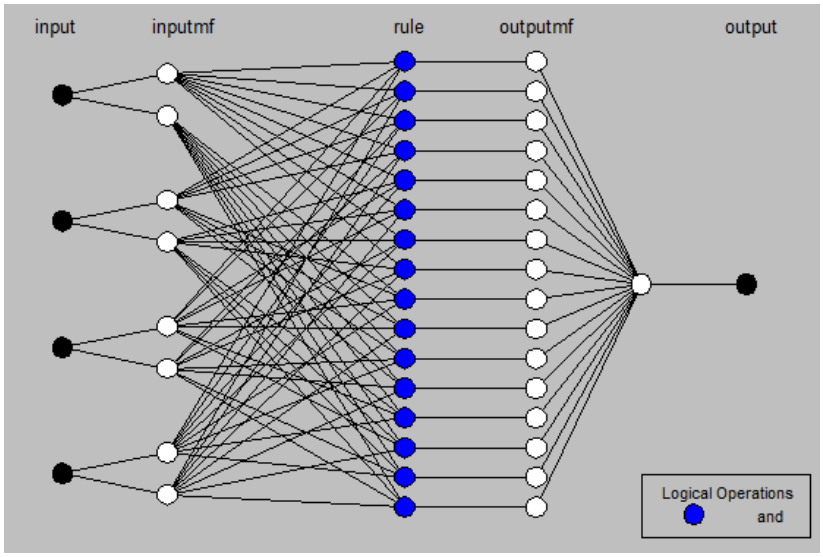


Figure 5.2: First-order Sugeno fuzzy model with 4 inputs and 2 membership functions.

A hybrid learning approach is used which combines a Steepest Descent (SD) and a Least-Squares Estimator for a fast estimation of the parameters. The learning is performed in on-line episodes and ANFIS parameters are updated after each sample is presented. Each training iteration is comprised of two forward and one backward pass. The Least-Squares Estimator algorithm is applied in the forward pass to estimate the linear consequent parameters and Back Propagation is executed in the backward pass to update the non-linear premise parameters. The premise and consequent parameters are assumed to be fixed in the forward and backward passes respectively. The ANFIS hybrid training in the forward pass, as seen in Equation (5.10), is the general form of the LSE problem [37].

$$\boldsymbol{A\theta} = \boldsymbol{y} \tag{5.10}$$

In this equation $\boldsymbol{\theta}$ is the consequent parameters vector. $\boldsymbol{A}$ is the design matrix that is constructed by the input training patterns and the normalized firing strengths, and $\boldsymbol{y}$ is the output training vector. For the (k+1)th new sample: $\boldsymbol{a}^T_{k+1}\,\boldsymbol{\theta}_{k+1} = \boldsymbol{y}_{k+1}$. The Least-Square Estimator solution can be expressed by the recursive Equations (5.11) and (5.12) [37].

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k \boldsymbol{a}_{k+1} \boldsymbol{a}^T_{k+1} \boldsymbol{P}_k}{1 + \boldsymbol{a}^T_{k+1} \boldsymbol{P}_k \boldsymbol{a}_{k+1}} \qquad k=0,1,\ldots,P-1 \qquad (5.11)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{P}_{k+1}\,\boldsymbol{a}_{k+1}\,(\boldsymbol{y}_{k+1} - \boldsymbol{a}^T_{k+1}\,\boldsymbol{\theta}_k) \qquad (5.12)$$

In these equations: $\boldsymbol{P}_k = (\boldsymbol{A}^T\,\boldsymbol{A})^{-1}$ and $(\boldsymbol{a}^T_k, \boldsymbol{y}_k)$ is the kth row of $\boldsymbol{A}$ that is the kth data pair of the training data.

5.4 Simulation Results and Discussion

The proposed system is implemented using MATLAB. The robot manipulator used in the research simulations is a Puma 560 (Programmable Universal Manipulator for Assembly). This robot's structure is shown in Figure 5.3 [20] and includes 6-axis revolute joints. The Puma 560 is an enormously popular robot manipulator that features an anthropomorphic (human-like) design, electric motors, and a spherical wrist. The Puma 560 robot was designed to have the approximate dimensions and reach of a human worker. It also has a spherical joint at the wrist just like a human. This robot is a very common laboratory robot and has been described as the "white rat" of robotics research. The Denavit and Hartenbergh link parameters of Puma 560 are summarized in Table 5.1.

In this table:

i: Joint axis number,

α_{i-1}: Link twist,

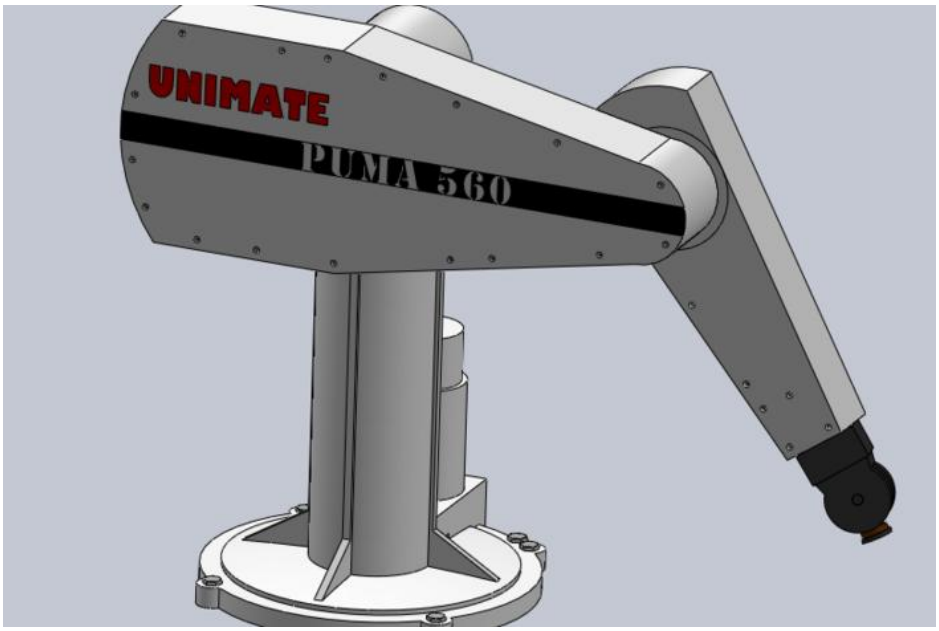


Figure 5.3: Puma 560 robot [67]

Table 5.1: Link Parameters of Puma 560

i	α_{i-1} (deg)	θ_i	a_{i-1}(m)	d_i(m)
1	0	θ_1	0	0
2	-90	θ_2	0	0.2435
3	0	θ_3	0.4318	-0.0934
4	90	θ_4	-0.0203	0.4331
5	-90	θ_5	0	0
6	90	θ_6	0	0

θ_i: Joint angle,

α_{i-1}: Link length, and

$d_{i:}$ Link offset.

The camera has a resolution of 1024 x 1024, a focal length of 8mm and 10μm pixels. The camera configuration is eye-in-hand. There is no transition between the robot and camera in our experiments. The offset between the height of the camera and the end-effector can be determined by a simple translation in the *z* axis direction if required. The target object is a cube with 20mm sides.

The initial object position is (0.75m, -0.15m, -0.6m) with respect to the manipulator coordinate system. The Harris Corner Detector algorithm is applied for feature extraction to detect the cube's vertexes. The coordinates for the cube's two top points in the image plane are the image features that will be normalized and sent to the ANFIS as training input samples. In the first step the end-effector is moved to the desired grasping position. The camera then captures the object image and stores it as the target image. This reference image is shown in Figure 5.5(a). The robot starts on-line learning episodes using a random initial orientation. The starting orientation for centering is within the region around the target with a ±60º deviation in the 3 first joint angles θ_1, θ_2, and θ_3. The region for reaching behavior is determined by a ±30º deviation in joints angles θ_2, and θ_3.

The total number of training episodes for centering and reaching are 300 and 200 respectively. Each learning episode starts from a random point in the pre-defined region. A learning episode is terminated if the end-effector reaches the goal state or if the camera loses sight of the target object. If the camera loses sight of the object then the maximum penalty is assigned to the Q-value. This prevents losing the target image in the visual servoing phase. The values for the Q-learning parameters are the learning rate λ=0.7 and the discount factor γ=0.9.

5.4.1 ANFIS Training and Validation Results

The training input/output samples for six centering and eight reaching ANFIS networks are generated during the on-line episodes. The camera captures the image of the object in every time step of a learning episode. The image processing unit then extracts the corners of the object and the X-Y coordinates of these points in the image plane that makes a sample input training vector $\boldsymbol{s_i}$. The output training sample for each input vector is a Q-value calculated using Equation (5.4). Another set of input/output samples are collected for centering and reaching ANFIS validation. These samples are from separate on-line random visual servoing episodes and are used to validate the performance of the fuzzy neural networks. The method for gathering validation samples is the same as for gathering training samples.

Tables 5.2 and 5.3 include training and validation sample numbers and the Mean Square Error (MSE) for the centering and reaching behaviors. In these tables the MSE is calculated based on the normalized values and cannot be related back to real world dimensions. Each column of these tables consists of a number of training and validation samples and the MSE for an ANFIS related to one basic action. The last columns indicate the total number of samples and the MSE. As can be seen, the training and validation errors for all ANFIS are quite similar and are small. These results show that all ANFIS respond to both input training and validation instances with a high degree of accuracy. The ability of the ANFIS to generate accurate results for previously unseen inputs demonstrates its ability to generalize.

Sample ANFIS membership functions for inputs 1 to 4 for the centering and reaching actions are shown in Figure 5.4. This figure illustrates how centre, width, and

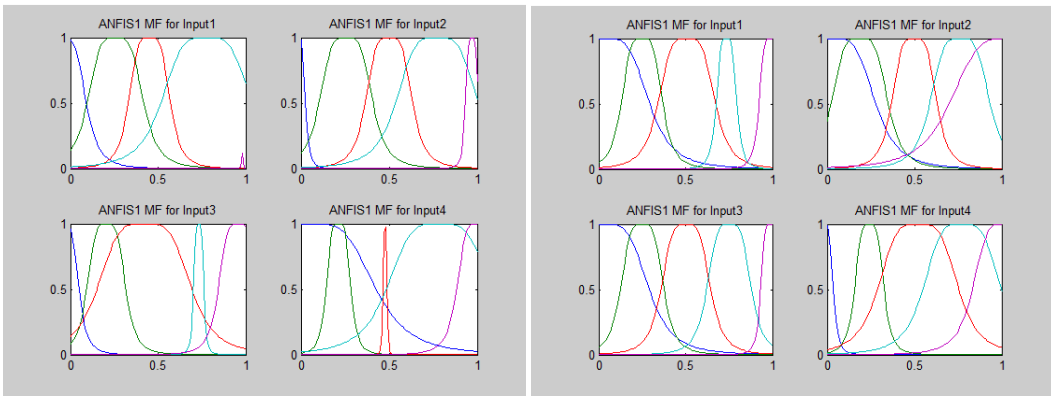
slope of the sample membership functions are adjusted after on-line training of the ANFIS. In these membership functions, tuning is performed using Back Propagation of error during ANFIS training.

Table 5.2: Centering training and validation samples and MSE

	B.A.1 q_1:+1°	B.A.2 q_1:-1°	B.A.3 q_1:+5°	B.A.4 q_1:-5°	B.A.5 q_1:+10°	B.A.6 q_1:-10°	Total
Training MSE	0.0089	0.0087	0.0067	0.0054	0.0055	0.0058	2.6147
Training Samples	965	976	858	875	837	846	5357
Validation MSE	0.0091	0.0083	0.0094	0.0056	0.0067	0.0062	2.7425
Validation Samples	929	965	835	836	766	799	5130

Table 5.3: Reaching training and validation samples and MSE

	B.A.1 q_2:+1°	B.A.2 q_2:-1°	B.A.3 q_2:+5°	B.A.4 q_2:-5°	B.A.5 q_3:+1°	B.A.6 q_3:-1°	B.A.7 q_3:+5°	B.A.8 q_3:-5°	Total
Training MSE	0.0018	0.0014	0.0023	0.0016	0.0052	0.0019	0.0011	0.0015	1.0980
Training Samples	1305	1245	1171	1178	1325	1248	1237	1218	9927
Validation MSE	0.0019	0.0013	0.0021	0.0016	0.0053	0.0018	0.0011	0.0013	1.0847
Validation Samples	1212	1310	1229	1167	1280	1304	1225	1225	9952



(a) (b)

Figure 5.4: Sample ANFIS final membership functions (a) centering, (b) reaching.

The learning rate for the best performance of ANFIS is η=0.9 which is found empirically.

5.4.2 Real-time Visual Servoing Results

The simulation results are shown in Figures 5.5 to 5.9. These results are provided for three randomly selected near, far and extremely far real-time start points from the target. The joint angle deviations ($\Delta\theta_1$, $\Delta\theta_2$, $\Delta\theta_3$) from the robot joint angles at the desired grasping position are (5º, 25º,-45º), (15º, 60º, -100º) and (-25º, 100º, -140º) for near, far and extremely far start points respectively.

The corresponding Cartesian coordinates of near, far and extremely far start points with respect to the manipulator coordinate system are (0.6712, -0.0919, -0.4969), (0.5580, -0.0059, -0.3038), and (0.2706, -0.2916, -0.0618) respectively. These visual servoing start points are out of the learning episodes regions especially for the far and the extremely far start points. Several images captured by the camera are shown in Figure 5.5 to illustrate the actual view from different positions. The final camera image is shown in Figure 5.5(a) where the camera and the reference target image completely match. Figure 5.5 (b), (c) and (d) show the images captured by the camera at near, far and extremely far start points respectively. As can be seen from Figure 5.5(b) one corner of the target cube is out of the camera's field of view but the object recognition and visual servoing tasks are successful. This is due to the capability of trained Q-evaluation fuzzy neural networks that generate proper Q-values when some of inputs are not present.

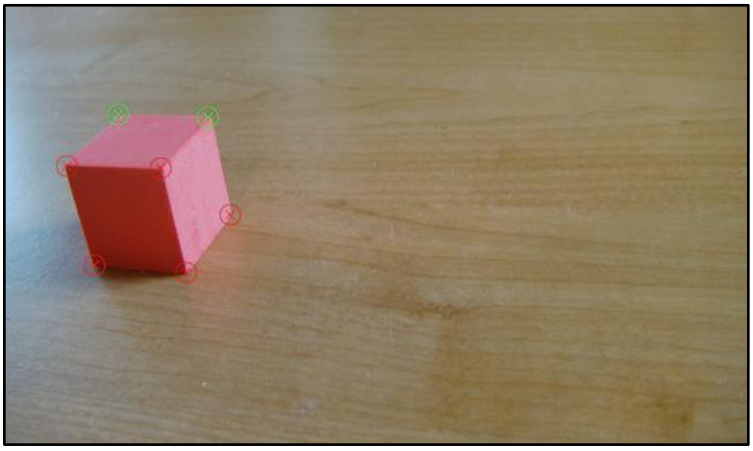
Figure 5.6 shows the manipulator when it is initialized close to the end location and when it is initialized at an extremely far starting location. The proposed system is an image-based visual servoing system. The control problem for this type of visual servoing

system can be expressed in terms of the image coordinates as was discussed in Chapter 3. The centering positioning error is defined in the camera image plane using the average image features error of Equation (5.6). This error is the difference in average image features at the next state and the target image. The average image features error curves are depicted in Figure 5.7 for the centering operation. The centering average features error for the near start point is shown in Figure 5.7 (a). The robot needs to perform only one basic action to center its end-effector relative to the target object position. The required centering basic action numbers for far and extremely far start points are 2 and 9 respectively as illustrated in Figures 5.7 (b) and (c).

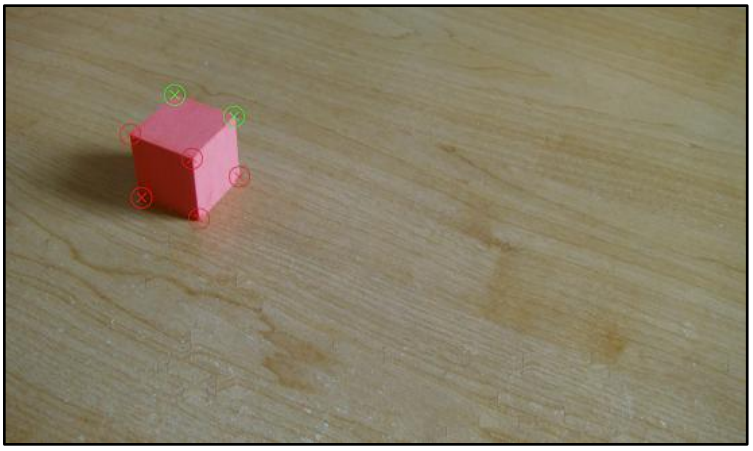
The reaching positioning error is defined in the camera image plane using the image features error of Equation (5.7). This error is the difference between the image features vectors at the next state and the target image. The error curves for the reaching task, which start from three different points, are illustrated in Figure 5.8. The required number of basic actions for the end-effector to reach the target object position from the near, far and extremely far start points are 18, 108 and 199 respectively as seen in Figures 5.8 (a), (b) and (c). When the normalized average image features error for centering and the maximum image features error for reaching is less than 0.1mm (equal to 10 pixels) in the image plane then the end-effector is considered to have reached the target and the visual servoing is recognized as successful.

The successful visual servoing results from various randomly selected starting points demonstrate the performance of the learning algorithm and its generalization capability. Figure 5.9 illustrates the trends for immediate rewards for the centering and reaching actions with an extremely far starting point.

(a)



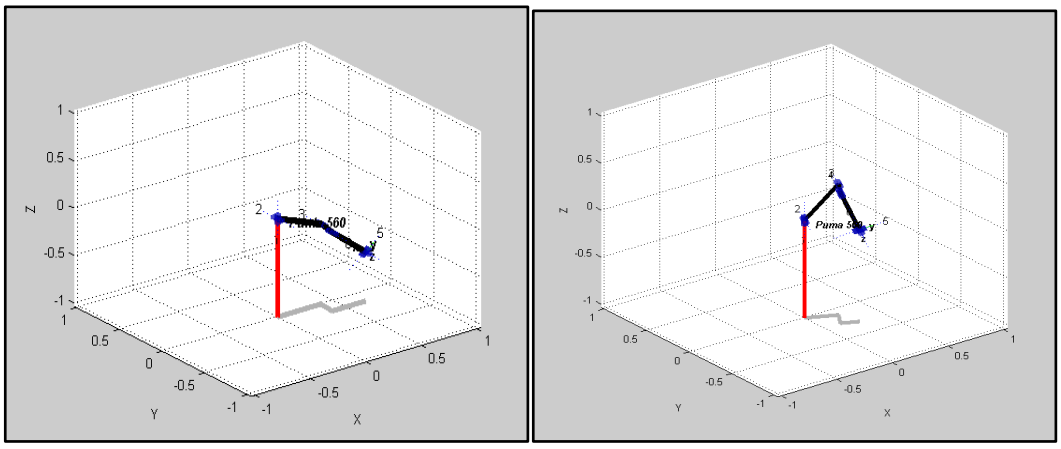
(b)



(c)

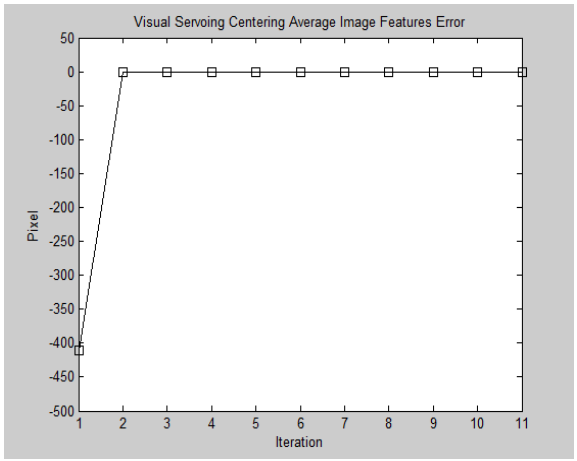
(d)

Figure 5.5: Target object images for different end-effector positions (a) reference image at desired grasping position, (b) near start point, (c) far start point, and (d) extremely far start point.

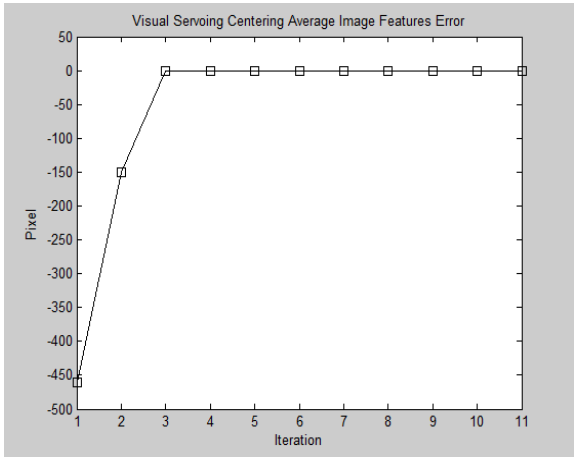


(a) (b)

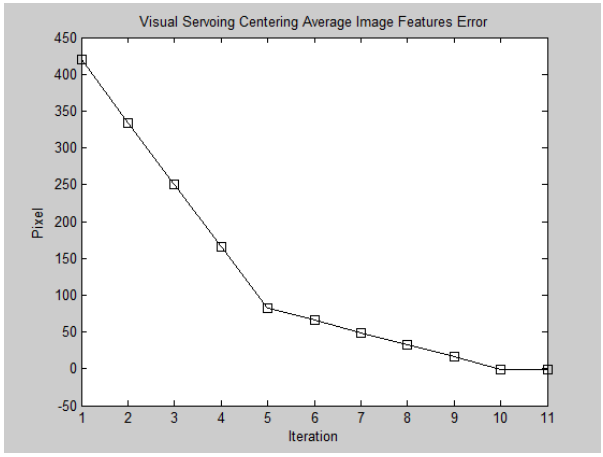
Figure 5.6: Robot manipulator positions (a) desired grasping position, (b) extremely far position.



(a)

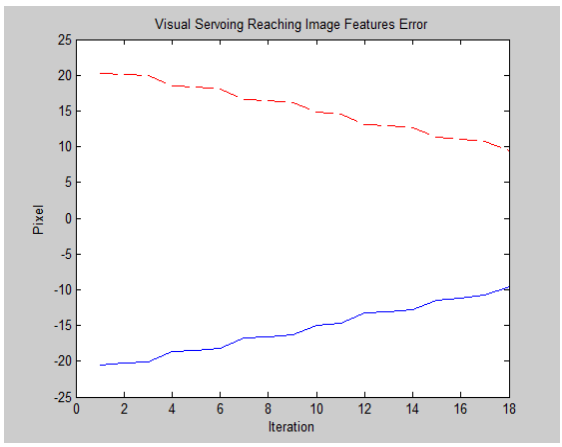


(b)

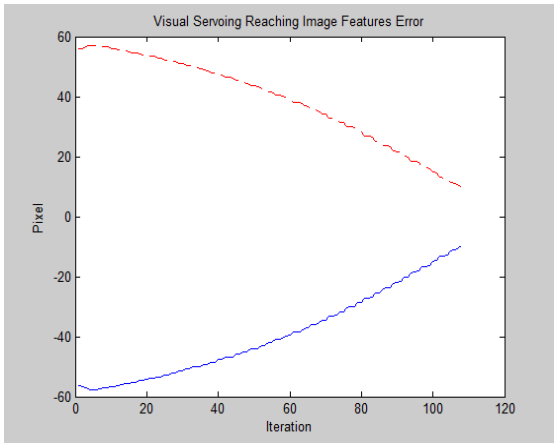


(c)

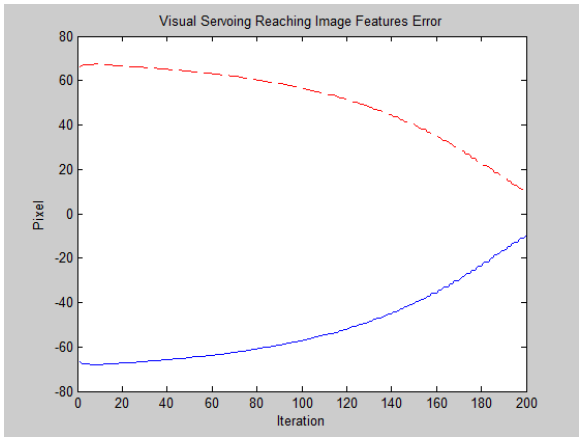
Figure 5.7: Visual servoing centering average image features error (a) near start point, (b) far start point, and (c) extremely far start point.



(a)

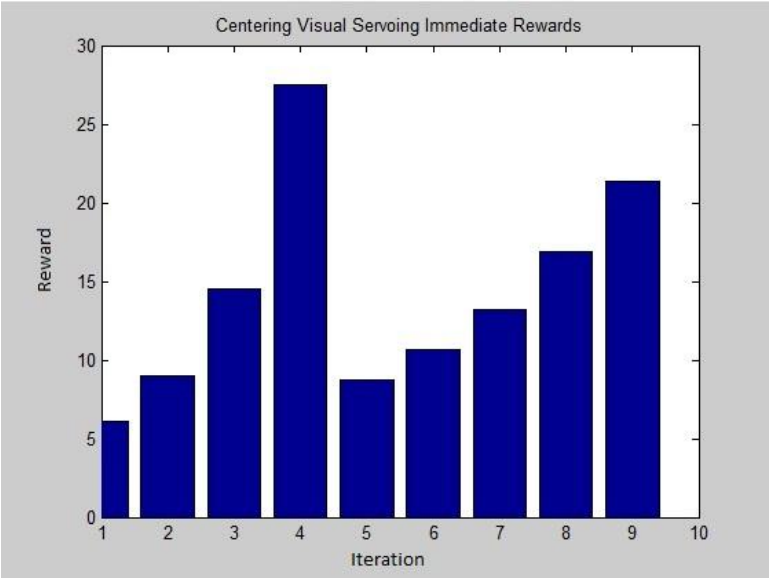


(b)

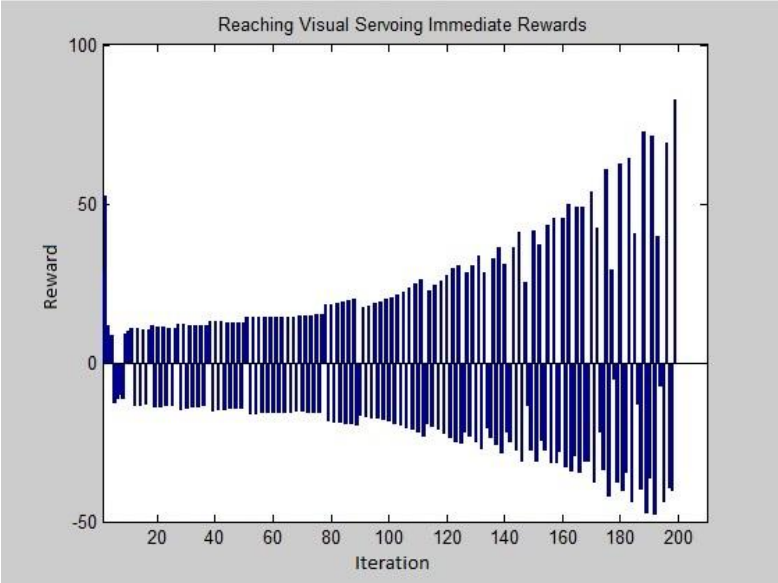


(c)

Figure 5.8: Visual servoing reaching average image features error (a) near start point, (b) far start point, and (c) extremely far start point.



(a)



(b)

Figure 5.9: Visual servoing immediate reward (a) centering, (b) reaching.

The immediate reward functions for the centering and reaching actions r_{ci} and r_{ri} at the *i*th state are calculated based on variations in the image features error between two

adjacent states as depicted in Equations (5.6) and (5.7). These immediate reward values are used in the main Q-learning Equation (5.4) to calculate the Q-evaluation values for the centering and reaching tasks in each iteration. Other terms in this formula are an effect of the cumulative rewards that have a significant role in the Q-values update. These figures however show that the immediate centering and reaching rewards are direct feedback from the environment to the agent, but the main criteria for the visual servoing performance is determined by the Q-evaluation values for these behaviors.

5.5 Chapter Summary

A self-learning system based on a neuro fuzzy implementation of the Q-learning algorithm for visual servoing is proposed in this chapter. The simulation results illustrate the successful performance of the learning system in a reasonable number of trial episodes. The robot is also capable of generalizing about its work-space. This is due to the generalization capability of ANFIS to respond to unseen input states with acceptable accuracy after training. The intelligent behavior of the robot prevents it from losing track of the target object during real-time operation.

The proposed system is image-based and does not require camera calibration. The control algorithm of this system is self-learning and the robot performs experiments in its work-space to extract input/output samples for the Q-evaluation FNN during on-line training. This type of system does not require a robot and camera model for its controller design. A limitation of existing image-based visual servoing systems is their use of the Jacobian matrix. The image Jacobian is a function of the distance between the camera and the object. This is hard to compute continuously and requires an inverse Jacobian that

makes the system unstable at its singular points. The proposed system does not require a Jacobian matrix calculation and therefore does not have this limitation. The results also demonstrate that using ANFIS is a successful solution for the implementation of Q-learning in a continuous and highly non-linear work-space.

The disadvantage of the proposed system is the increase in the number of on-line training iterations and the time required with both the number of input feature states and the number of ANFIS membership functions. The number of fuzzy rules and of ANFIS networks is determined by these two parameters. Higher numbers of input feature states and membership functions increases the number of connections between layers and the number of fuzzy rules in ANFIS. This leads to more calculations and therefore a longer processing time in on-line training. In applications where larger numbers of robot actions are needed, the number of ANFIS networks will increase. In this case more training samples and processing time are required to complete the on-line learning.

In self-learning systems a smaller number of training samples and a shorter training time are desirable. Learning in the proposed system is performed in on-line episodes. If an approach was found to reduce the number of these training episodes, it would improve the system significantly. In the next chapter a methodology is proposed to reduce the number of on-line learning episodes, and therefore the total number of training samples for this system. The new system inherits the characteristics of the system discussed in this chapter and accelerates the on-line learning process by adding an analytical learning component to the existing inductive training. The new system is more complex and has a higher computational cost but it will reduce the total convergence time of the learning algorithm and improve the real-time visual servoing performance.

CHAPTER SIX

SELF-LEARNING VISUAL SERVOING OF A ROBOT MANIPULATOR USING EXPLANATION-BASED FUZZY NEURAL NETWORKS AND Q-LEARNING

6.1 Introduction

A new self-learning visual servoing system for robot manipulators is discussed in this chapter. This system includes two main properties, on-line training and lifelong learning that are implemented using the Q-learning algorithm and Explanation-Based Fuzzy Neural Networks (EBFNN). It will be demonstrated that by combining the EBFNN and a Q-learning algorithm, the number of training samples, and therefore the training time needed for a specific robot positioning task, will be reduced. The system uses Q-learning to find the optimal policy based on Reinforcement learning. This policy is applied by the robot to reach a predetermined object that has been randomly placed in the work-space. Background knowledge about the robot and its environment is transferred to the agent during the learning process using a set of neural networks which have been previously trained. This system learns the optimal policy in order to select the best basic action that maximizes the cumulative reward received at each time step. This learning approach does not use robot or camera models, or require calibration. Simulation results demonstrate the effectiveness of this methodology to improve learning performance.

Learning is an essential feature for the automatic design of autonomous robots. In the area of robotic learning there are normally a limited number of training samples available. If the task is more complicated and there is little knowledge about the work-space, then more training instances are necessary for the system to accomplish the desired goal. If the set of tasks which a robot must perform over its lifetime is considered, then it

would be desired to simplify learning, so new tasks can be learned more quickly. The hardware is assumed to remain constant in such a scenario. Lifelong learning allows for the transfer of background information between tasks. Complex tasks, which could need larger training sets, can be trained more quickly by using this method. Explanation-Based Neural Networks (EBNN) learning is proposed [12] as an approach for reducing the amount of training data necessary for reliable generalizations. The EBNN is based on combining of inductive and analytical learning to apply current training data and previously learned knowledge respectively. Through this training the robot may learn the invariants of the individual tasks and of the workspace. This task-independent knowledge builds Domain Theory, which can be learned by EBNNs and used to bias generalization when learning. This approach can decrease the necessity for real-world experimentation.

In self-learning robot control systems the training is on-line. A smaller number of training samples and a shorter training time are desirable in these systems. In this work a set of neural networks are initially trained off-line as action models for the basic tasks.

On-line learning is performed by Q-learning implemented using fuzzy neural networks as discussed in the previous chapter. The environment consists of a set of states for the agent and a number of actions which the agent can perform. A learning episode is a sequence of randomly selected states and actions. Each learning episode starts from a random point in the robot's work-space. The agent initially observes the current state at every time step and then randomly picks an available action to perform. The agent receives an immediate reward or penalty based on its new state which reflects the desirability of the executed action. The old state is replaced by the new state, and this process continues during training to build a learning episode. The agent learns from the

feedback and constructs an evaluation function based upon the states and actions. The evaluation function is determined as the maximum discounted cumulative reward that the agent can reach. If the agent achieves the target during the learning episode then the evaluation function will be equal to its maximum value. If the agent fails to reach the target the evaluation function will be set to its minimum value.

The background knowledge of the robot and its environment are transferred from previously trained action model neural networks to the Q-learning process. The key point in understanding why EBFNNs can improve the performance of the Q-learning algorithm is due to the image Jacobian that is an inherent concept in visual servoing systems. In Image-Based Visual Servoing (IBVS) the dynamics of the system are usually expressed as an image Jacobian. A limitation of existing image based visual servoing methods is their use of the Jacobian matrix as a function of the distance between the camera and the target. It is difficult to calculate an inverse Jacobian continuously and this can make the system unstable at singular points. The proposed system does not require the Jacobian matrix to be calculated and therefore does not have this limitation. Trained action model neural networks store the changes in extracted image features (states) with the changes in robot joints (actions). These changes are partial derivatives (slopes) of the image features with respect to the robot joints that build the image Jacobian. The image Jacobian contains the robot and the camera model information. The trained action model neural networks include image Jacobian knowledge, and the visual servoing learning system will speed up the on-line learning of new robot tasks using this inherent knowledge. This knowledge is stored in action model neural networks which can be used throughout the life of the robot.

On-line training is used in the proposed visual servoing system. This differs from many of the existing neural network-based systems [3-5, 10, 34-36] in which the learning process is performed off-line. The Q-learning agent for this work is a robot manipulator with an attached camera. The agent performs tasks in its environment and provides the input and output training samples used in learning. The robot executes a random basic action to move its end-effector and camera to a new position at every time step. The camera captures an image which then has features extracted from it. The Q-evaluation value for each basic action is then calculated. This value includes both an immediate and a delayed reward. The image features and related Q-values for each basic action are combined to create a training dataset of input and output pairs which are used for each Q-evaluation or target for a fuzzy neural network. Research such as [48-50] demonstrates visual servoing of mobile robots which use neural networks in the implementation of Q-learning.

Researchers in [66] presented a visual servo controller for a robot manipulator using neural network Reinforcement learning. It is, however, a hybrid approach that involves switching control between traditional image-based visual servoing and Reinforcement learning to enable the approaching behavior of the manipulator. In the traditional visual servoing phase of that research, calculation of the image Jacobian and depth estimation are necessary.

Research involving robot learning using the EBNN is rare. The initial research on lifelong robot learning was performed by Sebastian Thrun and Tom M. Mitchell [53]. That research was limited to mobile robots. To the best of the author's knowledge, this research is the first work that integrates Q-learning and EBNN for visual servoing of a

robot manipulator. Another contribution of this research is the use of an Adaptive Neuro Fuzzy Interface System (ANFIS) in the EBNN structure in conjunction with the Q-learning algorithm.

6.2 Q-learning EBFNN Visual Servoing System Architecture

The architecture of the proposed visual servoing system is illustrated in Figure 6.1. The main components of the system consist of the Q-learning evaluation unit, two sets of Explanation-Based Adaptive Neuro Fuzzy Inference Systems (EB-ANFIS) and two sets of action model neural networks for centering and reaching behavior, the robot arm and robot controller, and the CCD camera and image feature extraction unit.

The Q-learning evaluation unit is used during the learning phase. This unit is divided into two units that work independently and calculate the Q_C and Q_R evaluation values for centering and reaching behaviors. The inputs to this unit are the reference state and the current state of the agent. These are the target image features and current image features respectively. The target image features are captured by the camera-in-hand when the end-effector is at the desired grasping position. The Q-learning evaluation unit estimates the immediate and delayed rewards at each time step based on the target and the agent's current state.

The final Q_C and Q_R values at each time step are calculated and sent to the centering and reaching EB-ANFIS as training output samples. The image feature extraction unit sends the related training input samples to both sets of EB-ANFIS and action model neural networks. The derivative of centering and reaching action model outputs with

respect to the input image features, $\Delta \boldsymbol{B_C}/\Delta \boldsymbol{F}$ and $\Delta \boldsymbol{B_R}/\Delta \boldsymbol{F}$ is also calculated at each time step of online learning and is sent to the centering and reaching EB-ANFIS.

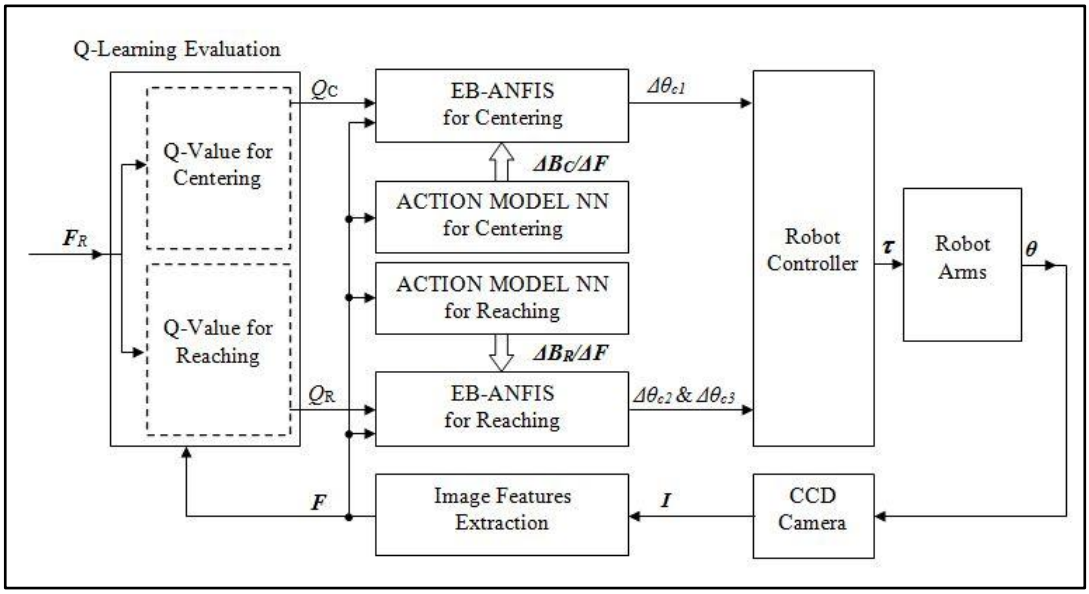


Figure 6.1: Architecture of the Q-learning EBFNN visual servoing system

The outputs of the centering and reaching EB-ANFIS, $\Delta\theta_{c1}$, $\Delta\theta_{c2}$, and $\Delta\theta_{c3}$ are the command signals which are sent to the robot's joints 1, 2, and 3. The robot controller receives these centering and reaching command signals and generates a proper torque vector $\boldsymbol{\tau}$ to control the relevant robot joints at each time instance. In the real-time visual servoing phase, the current image features are applied to the set of trained EB-ANFIS for the centering and reaching action. The basic action with the highest EB-ANFIS Q-value output is selected and executed to transfer the robot end-effector to the new state. This sequence will continue until the robot achieves the goal state.

6.3 Learning Methodology Based on Inductive and Analytical Learning Algorithms

EBFNN is based on a hybrid of inductive and analytical learning methodologies. After each time instance during a learning episode in which the robot performs a basic action,

the extracted image features and calculated Q-value are sent to the corresponding EB-ANFIS as input/output training samples. This part of learning is based on inductive learning. The EB-ANFIS is provided with an additional set of inputs for use in analytical learning. The outputs of the action model neural networks are used to construct the domain theory when combined with the extracted image features. The Tangent-Prop algorithm is used to analyze the training examples to extract useful information which can then be used to refine the target networks. The components of the learning algorithm are explained in the following sections.

6.3.1 The Q-learning Algorithm

The Q-learning algorithm is the core of on-line learning. The same Q-learning algorithm that is discussed in section 5.3.1 is used here. EB-ANFIS is employed for this Q-learning algorithm implementation.

6.3.2 Basic Action Model Neural Networks

An action model neural network for each defined basic task is built and trained off-line. Each network can estimate the next state (image feature) based on the given present state. The training samples for each basic task model neural network are in the format of (s_i, s_{i+1}). Data collection for action neural networks is done during training. Each training episode starts from a random point in the robot work-space. A camera is mounted on the end-effector which then captures an image. The image features are extracted and stored as the present state s_i in each time step. The robot then performs a random basic action and a new image is captured when this task is completed. The new image features are extracted and will be stored as next state s_{i+1}. This process continues by executing one random basic action per episode. The related input/output samples for each basic action are

collected in a training dataset.

The required number of action model neural networks for centering and reaching behavior is 6 and 8 respectively. The network's outputs derivative with respect to the current state is calculated and sent to the EB-ANFIS at each time step that will be used for Q-evaluation training by the Tangent-Prop algorithm.

6.3.3 Explanation-based Fuzzy Neural Networks

The EBFNN are similar to the EBNN in that analytical and inductive learning are combined. The fundamental difference between inductive and analytical learning is that they suppose two different criteria for the learning problem:

- In inductive learning a hypothesis space H is given to the agent, which then must choose an output hypothesis and a set of training instances $D = (x_l, f(x_1));...; (x_n, f(x_n))$ where $f(x_i)$ is the target value for the sample x_i. The desired output of the learner is a hypothesis h from H that is consistent with these training samples.
- In analytical learning the agent has access to the same hypothesis space H and training instances D as for inductive learning. The agent is supplied with a supplementary input. This is a domain theory B which includes the background knowledge that can be used to describe observed training samples. The desired output of the agent is a hypothesis h from H that is consistent with both the training instances D and the domain theory B.

Explanation-based learning is a method for using approximate prior knowledge to improve the learner's ability to generalize from limited training data. Prior knowledge

consists of a collection of previously learned neural networks (the domain theory networks), with the function to be learned represented by an additional neural network (the target network). The EBNN algorithm uses the domain theory networks to analyze each training example, in order to extract information about the relevance of the different features of the example. This relevant information is then used, together with standard neural network induction, to constrain the weights of the target network. The target network is constrained both inductively by the training data, and analytically by the knowledge implicit in the domain theory.

The key point is to analyze the training samples so that information is extracted in a form useful for refining the target network using the Tangent-Prop algorithm. This algorithm is an extension of Back Propagation that is able to adjust network weights to minimize the error in both the values and the derivatives of the function computed by the target network.

Consider some target function F and a training example X consisting a vector of components x_i. The Tangent-Prop algorithm iteratively adjusts the weights of the target network *NET* to minimize an error measure containing two terms. The first error term is the difference between the target function value $F(X)$ and the value approximated by the network $NET(X)$. The second error phrase is the difference between the partial derivatives of the target function $dF(X)/dx_i$ and the partial derivatives of the function represented by the target network $dNET(X)/dx_i$. The EBNN obtains estimates of target values for $dF(X)/dx_i$ by using its domain theory to analyze training examples.

The summary of the EBNN algorithm is as follows:

For each training example:

(a) Explain how the training example satisfies the target function. Predict the value of the target function for the training sample using the domain theory networks.

(b) Analyze this explanation to establish the relevance of the characteristics. Extract the partial derivatives of the explained target function value with respect to each training sample feature by checking the weights and activations of the domain theory networks in the above explanation.

(c) Perform refining of the target function network. Update the weights of the target function to fit both the observed target function value (inductive component) and the target derivatives extracted from the explanation (analytical component).

The domain knowledge in our proposed system is the robot and the invariants in the environment. This is represented by real-valued neural networks. There is an action network for each defined basic task. The agent learns the domain knowledge by training these action networks in a primary step. Whenever the robot encounters a new learning task, these previously learned action models can be used to bias the generalization in order to reduce the volume of the required training experience and time. This background knowledge can be transferred between the robot's tasks during its life using the EBFNN. If the robot is changed then it is required to perform the action models training for that particular robot hardware.

ANFIS is used for the Q-evaluation fuzzy neural network. Q-evaluation or target networks map the image features to the evaluation function Q for each basic action. This relation is highly non-linear. The ANFIS can achieve highly non-linear mappings with a high performance. There is one ANFIS for each centering and reaching basic actions. Six centering and eight reaching robot basic actions have been defined; therefore the total

number of centering and reaching ANFIS is six and eight respectively. The core algorithm for ANFIS on-line learning is based on the Recursive Least-Squares Estimator. The Tangent-Prop algorithm is used for EB-ANFIS training. This algorithm is an extension of the Back Propagation algorithm that is able to adjust network parameters to minimize the error in both the values and the derivatives of the Q-evaluation function computed by the target networks.

The first-order Sugeno fuzzy model with four inputs and five membership functions is used for this application. The selection of the number of membership functions is a compromise between the mapping accuracy and the required processing time for this particular application. The first-order Sugeno neuro fuzzy network [37] consists of five layers. These five layers include fuzzy membership functions, rule firing strength calculation, normalization, rules outputs, and overall network output. Each ANFIS has four inputs that are extracted from the image features. These inputs are coordinates of the extracted object points in the image plane. The ANFIS output is the Q-evaluation function value.

There are two sets of network parameters that must be estimated during learning. These are premise and consequent parameters, as explained in the previous chapter. A hybrid learning approach is used which combines Tangent-Prop and the Least-Squares Estimator for fast identification of the parameters. Learning is in on-line episodes and the EB-ANFIS parameters are updated after each data presentation. A training iteration is comprised of two forward and backward passes. The Least-Squares Estimator algorithm is used in the forward pass to estimate the linear consequent parameters, and Tangent-Prop is executed in the backward pass to update the non-linear premise parameters. The

premise and consequent parameters are assumed to be fixed in the forward and the backward passes respectively. The EB-ANFIS hybrid training in the forward pass, as seen in Equation (6.1), is the general form of the LSE problem [37] discussed in Chapter 5.

$$\boldsymbol{A\theta = y} \tag{6.1}$$

In this equation $\boldsymbol{\theta}$ is the consequent parameter vector, $\boldsymbol{A}$ is the design matrix that is constructed by the input training patterns and the normalized firing strengths, and $\boldsymbol{y}$ is the output training vector. For the (k+1)th new sample: $\boldsymbol{a^T_{k+1}\,\theta_{k+1} = y_{k+1}}$. The least-square estimator solution can be expressed by the recursive Equations (6.2) and (6.3) [37].

$$\boldsymbol{P_{k+1}} = \boldsymbol{P_k} - \frac{\boldsymbol{P_k a_{k+1} a^T_{k+1} P_k}}{1 + \boldsymbol{a^T_{k+1} P_k a_{k+1}}} \qquad k=0,1,\ldots,P\text{-}1 \tag{6.2}$$

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} + \boldsymbol{P_{k+1}\, a_{k+1}\, (y_{k+1} - a^T_{k+1}\, \theta_k)} \tag{6.3}$$

In these equations: $\boldsymbol{P_k} = (\boldsymbol{A^T A})^{-1}$ and $(\boldsymbol{a^T_k, y_k})$ is the kth row of $\boldsymbol{A}$ that is the kth data pair in the training data.

The target value for a Q-value derivative in the current state $\boldsymbol{s_i}$ is calculated using the following equation:

$$\nabla_{s_i} Q^{\pi}(\boldsymbol{s_i}, a_i) = \lambda \frac{\partial r_i}{\partial \boldsymbol{s}}\Big|_{s_i} + \lambda\gamma \frac{\partial Q(\boldsymbol{s}, a_{i+1})}{\partial \boldsymbol{s}}\Big|_{s_{i+1}} \cdot \frac{\partial \boldsymbol{B}(\boldsymbol{s}, a_i)}{\partial \boldsymbol{s}}\Big|_{s_i} + (1-\lambda) \frac{\partial Q^{\pi}(\boldsymbol{s_i}, a_i)}{\partial \boldsymbol{s}}\Big|_{s_i} \tag{6.4}$$

where $\frac{\partial r_i}{\partial \boldsymbol{s}}\Big|_{s_i}$ is the immediate reward derivative with respect to the current state, $\frac{\partial Q(\boldsymbol{s}, a_{i+1})}{\partial \boldsymbol{s}}\Big|_{s_{i+1}}$ is a derivative of the EB-ANFIS network output for the next action with a maximum Q-value at the next state with respect to the input state, $\frac{\partial \boldsymbol{B}(\boldsymbol{s}, a_i)}{\partial \boldsymbol{s}}\Big|_{s_i}$ is a

derivative of the basic action neural network output with respect to the current state, and $\frac{\partial Q^{\pi}(\boldsymbol{s_i}, a_i)}{\partial \boldsymbol{s}}|_{\boldsymbol{s_i}}$ is a derivative of the EB-ANFIS output for the current action with respect to the current state. The applied Tangent-Prop error formula [11] is:

$$E_i = (Q(\boldsymbol{s_i}, a_i) - Q^{target}(\boldsymbol{s_i}, a_i))^2 + \mu_i \sum_{j=1}^{f} (\nabla_j Q(\boldsymbol{s_i}, a_i) - \nabla_j Q^{target}(\boldsymbol{s_i}, a_i))^2 \qquad (6.5)$$

where: $\nabla Q^{target}(\boldsymbol{s_i}, a_i) = \nabla_{\boldsymbol{s_i}} Q^{\pi}(\boldsymbol{s_i}, a_i)$

$Q^{target}(\boldsymbol{s_i}, a_i)$ and $\nabla Q^{target}(\boldsymbol{s_i}, a_i)$ are the Q-evaluation EB-ANFIS output target and the output derivative target values respectively. The relative importance of the analytical and inductive learning components is determined in EBFNN by μ_i and defined in Equation (6.6):

$$\mu_i = 1 - \frac{(\sum_{j=1}^{f} (\boldsymbol{B}_j(\boldsymbol{s_i}, a_i) - \boldsymbol{s_{(i+1)j}})^2)^{1/2}}{\mu_c} \qquad (6.6)$$

In this equation $\boldsymbol{B_j}(\boldsymbol{s_i}, a_i)$ is the *j*th output of the basic action model neural network and $\boldsymbol{s_{(i+1)j}}$ is the *j*th element of the next state.

The constant μ_c is chosen 1 for $0 \leq \mu_i \leq 1$. The value of μ_i is calculated by the discrepancy between the domain theory forecast from the action model neural networks and the robot's actual next state. The weight μ_i is specified separately for each training sample, based on how accurately the action model neural networks estimate the next state of the robot. Consequently, the analytical component of learning is weighted more heavily for training instances in which the next states are predicted accurately by the action models and is suppressed for samples in which the next states have not been properly predicted.

6.4 Simulation Results and Discussion

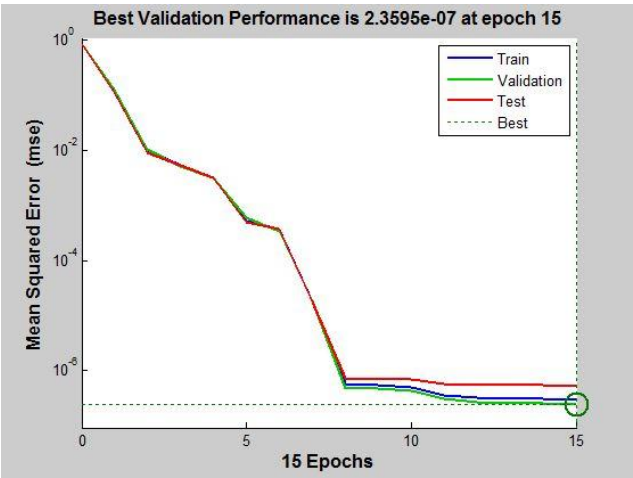
The proposed self-learning visual servoing system is implemented using the same robot and camera model and parameters explained in Chapter 5. The camera configuration is eye-in-hand. The target object is a cube with sides of 20mm. The initial object position is at (0.75m, -0.15m, -0.6m) with respect to the manipulator coordinate system. The Harris Corner Detector algorithm is applied for feature extraction to detect the cube vertexes. The coordinates of the cube's two top points' in the image plane are the desired image features that will be normalized and sent to the ANFIS as training input samples. In the first step, the end-effector is moved to the desired grasping position, and the camera captures the object's image, which is stored in the memory as the target image. The robot starts on-line learning episodes using random points in its work-space. The learning episodes' start points for centering are within the region around the target with ±60º deviation in the 3 first joint angles, θ_1, θ_2, and θ_3. The region for reaching behavior is determined by ±30º deviation in the joint angles, θ_2, and θ_3. The acceptable real-time visual servoing region is not limited to these boundaries. A learning episode is terminated if the end-effector reaches the goal state or if the camera loses sight of the target object. If the camera loses sight of the object, then the maximum penalty is assigned to the Q-value. This prevents the loss of the target image in the real-time visual servoing phase. The values for the Q-learning parameters are a learning rate of λ=0.9 and a discount factor of γ=0.9.

6.4.1 Action Model Neural Networks Training and Validation Results

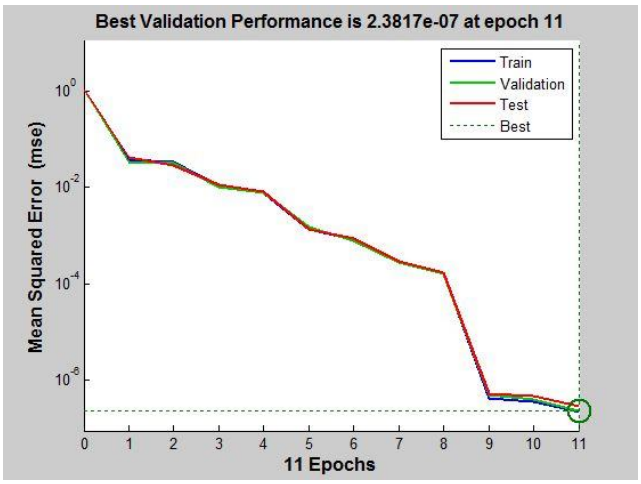
A total of 14 different neural networks with one hidden layer and 20 neurons are dedicated to the basic action models for centering and reaching. The action model neural

networks are trained offline. Training input/output samples are collected in random episodes and organized in a separate dataset for each basic action. The training algorithm that is used is the Levenberg-Marquardt Back Propagation algorithm, and logsig and purlin are used as activation functions for the hidden and the output layers respectively. Figures 6.2 and 6.3 are the training performance curves for the centering and the reaching neural networks respectively. Net1 to Net6, shown in Figures 6.2 (a) to (f), and Net1 to Net8, shown in Figures 6.3 (a) to (h), indicate the 6 centering and 8 reaching basic action networks respectively. In these figures, 1153 training, 247 testing and 247 validation samples were provided for each neural network. The number of training samples is 70% of the total samples.

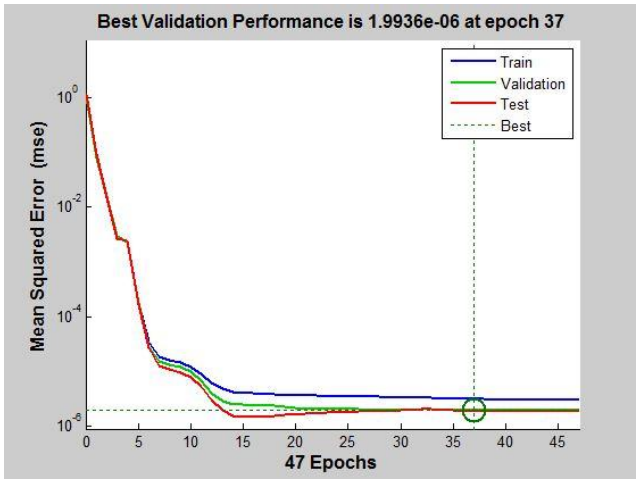
The remaining 30% of the total samples are divided equally between test and validation samples. These results show that the Mean Square Error Performance function reaches a very low limit in an acceptable number of training epochs. For each neural network a separate second validation data set is gathered. The number of validation samples for each neural network is 1647.



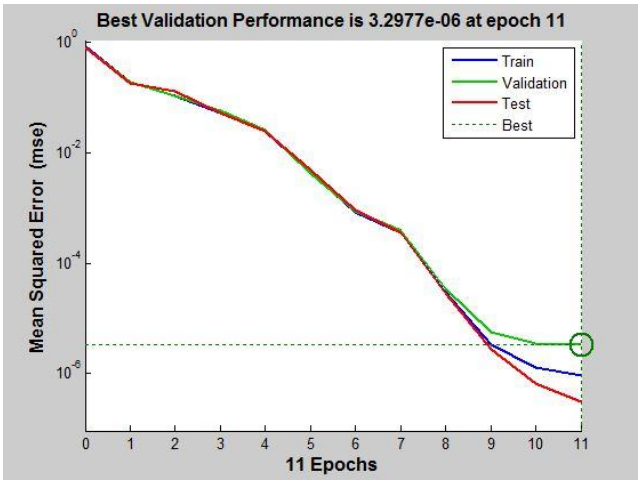
(a)


Best Validation Performance is 2.3817e-07 at epoch 11
Train
Validation
Test
Best
Mean Squared Error (mse)
11 Epochs

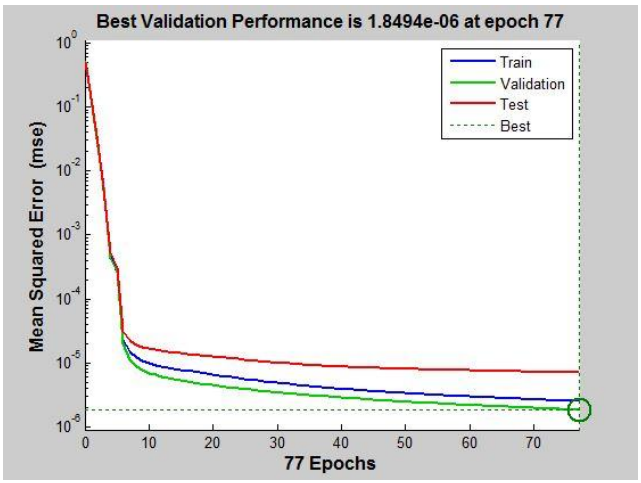
(b)


Best Validation Performance is 1.9936e-06 at epoch 37
Train
Validation
Test
Best
Mean Squared Error (mse)
47 Epochs

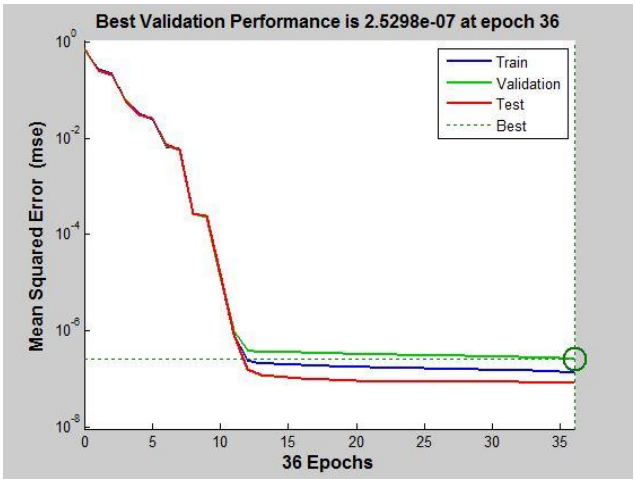
(c)


Best Validation Performance is 3.2977e-06 at epoch 11
Train
Validation
Test
Best
Mean Squared Error (mse)
11 Epochs

(d)

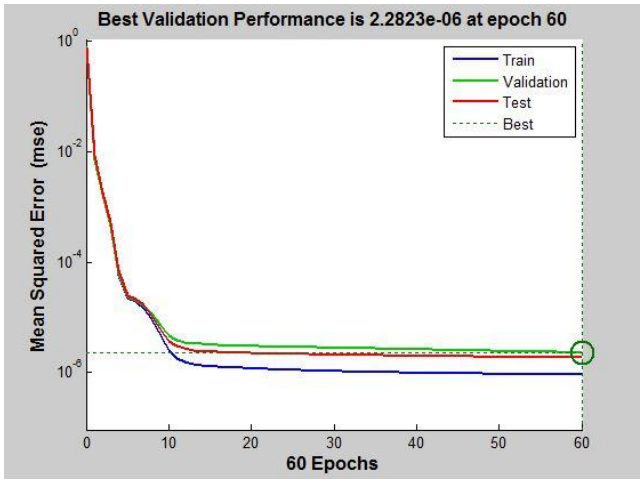


(e)

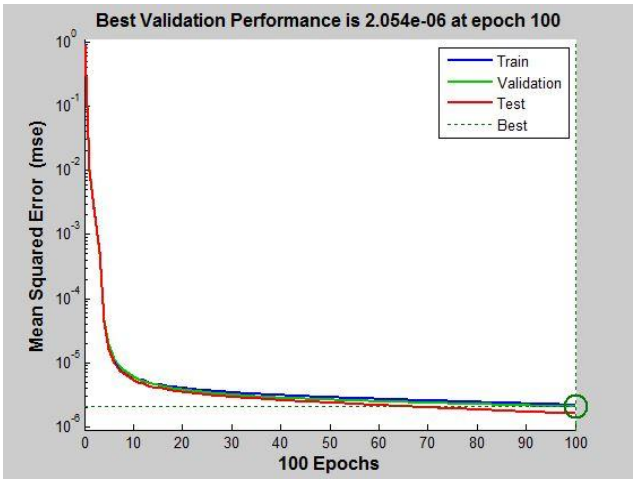


(f)

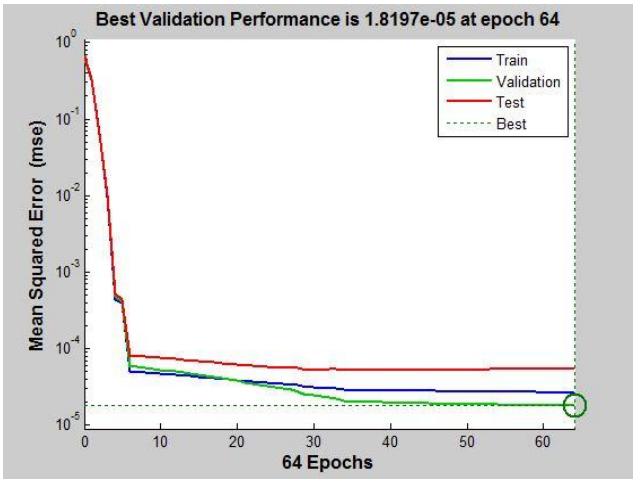
Figure 6.2: Training performance curves for the centering action model neural networks: (a) Net1, (b) Net2, (c) Net3, (d) Net4, (e) Net5, and (f) Net6.



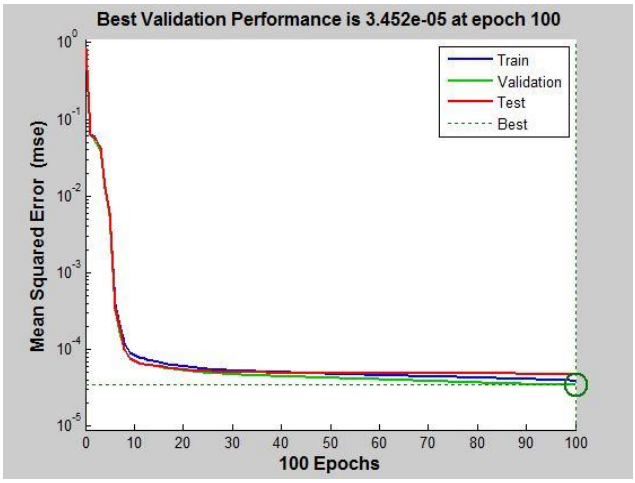
(a)


Best Validation Performance is 2.054e-06 at epoch 100
Train
Validation
Test
Best
Mean Squared Error (mse)
100 Epochs

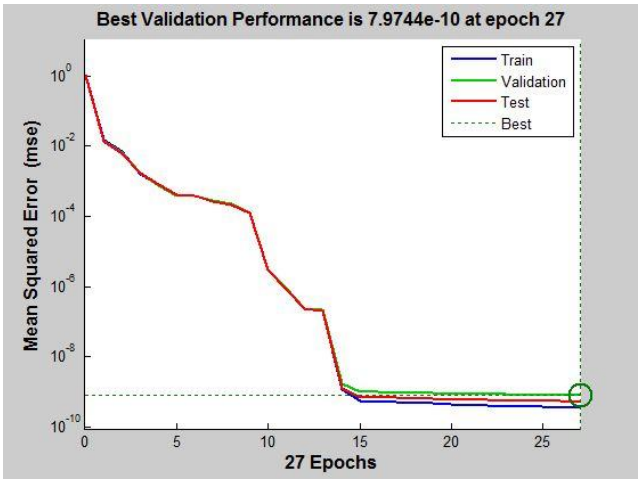
(b)


Best Validation Performance is 1.8197e-05 at epoch 64
Train
Validation
Test
Best
Mean Squared Error (mse)
64 Epochs

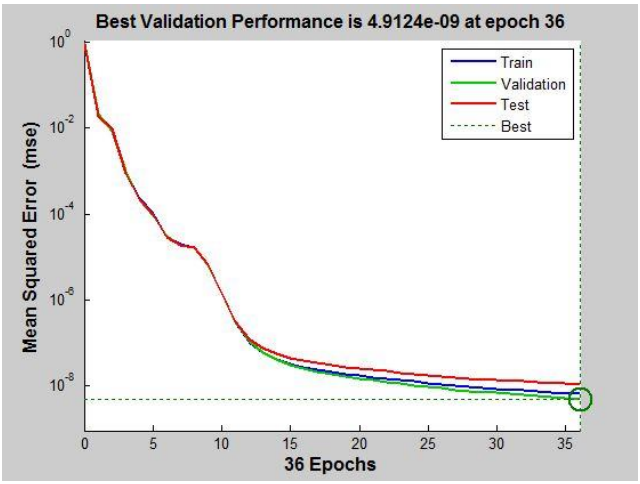
(c)


Best Validation Performance is 3.452e-05 at epoch 100
Train
Validation
Test
Best
Mean Squared Error (mse)
100 Epochs

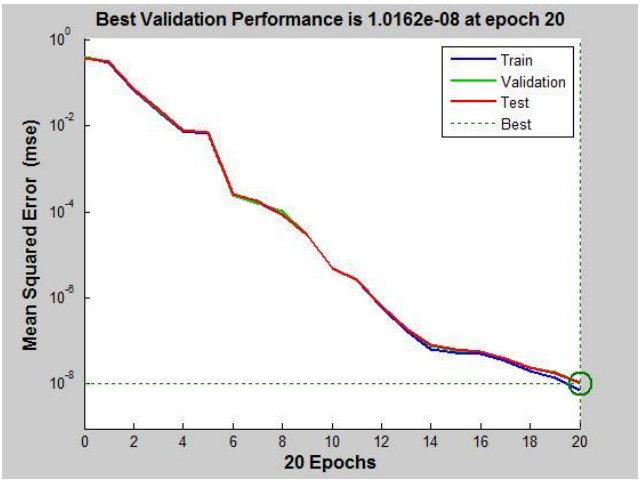
(d)


Best Validation Performance is 7.9744e-10 at epoch 27
Train
Validation
Test
Best
Mean Squared Error (mse)
27 Epochs

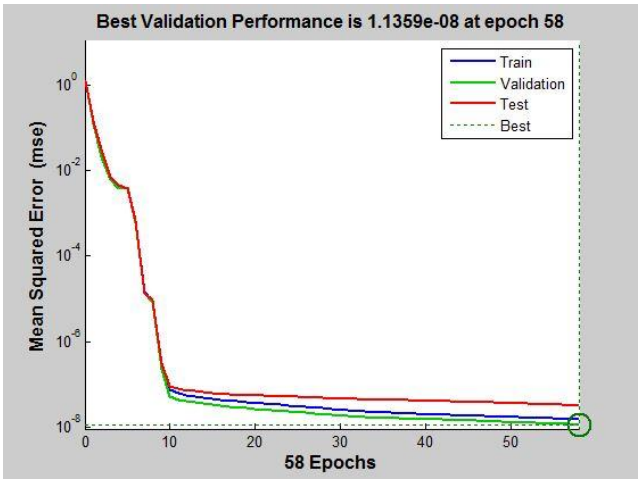
(e)


Best Validation Performance is 4.9124e-09 at epoch 36
Train
Validation
Test
Best
Mean Squared Error (mse)
36 Epochs

(f)


Best Validation Performance is 1.0162e-08 at epoch 20
Train
Validation
Test
Best
Mean Squared Error (mse)
20 Epochs

(g)



(h)

Figure 6.3: Training performance curves for the reaching action model neural networks: (a) Net1, (b) Net2, (c) Net3, (d) Net4, (e) Net5, (f) Net6, (g) Net7, and (h) Net8.

Table 6.1: Training and validation MSE for centering action models NNs

1.0e-03 *	Input	B.A.1 q_1:+1º	B.A.2 q_1:-1º	B.A.3 q_1:+5º	B.A.4 q_1:-5º	B.A.5 q_1:+10º	B.A.6 q_1:-10º
Training MSE	1	0.0038	0.0041	0.0175	0.0130	0.0373	0.0064
	2	0.0041	0.0037	0.0241	0.0126	0.0405	0.0055
	3	0.0177	0.0151	0.0513	0.0325	0.0445	0.0105
	4	0.0202	0.0176	0.0568	0.0384	0.0499	0.0130
Validation MSE	1	0.0047	0.0045	0.0249	0.0249	0.0367	0.0176
	2	0.0049	0.0033	0.0370	0.0198	0.0462	0.0159
	3	0.0184	0.0222	0.0918	0.0778	0.0678	0.0235
	4	0.0209	0.0258	0.1019	0.0917	0.0758	0.0288

Table 6.2: Training and validation MSE for reaching action models NNs

1.0e-03 *	Input	B.A.1 q_2:+1º	B.A.2 q_2:-1º	B.A.3 q_2:+5º	B.A.4 q_2:-5º	B.A.5 q_3:+1º	B.A.6 q_3:-1º	B.A.7 q_3:+5º	B.A.8 q_3:-5º
Training MSE	1	0.0191	0.0289	0.0628	0.0583	0.0003	0.0023	0.0014	0.0019
	2	0.0159	0.0313	0.0656	0.0751	0.0003	0.0018	0.0015	0.0015
	3	0.0344	0.0406	0.1781	0.2092	0.0007	0.0023	0.0028	0.0043
	4	0.0350	0.0397	0.1793	0.2091	0.0006	0.0016	0.0027	0.0042
Validation MSE	1	0.0212	0.0447	0.0922	0.0751	0.0003	0.0027	0.0015	0.0021
	2	0.0209	0.0399	0.0997	0.0840	0.0003	0.0020	0.0018	0.0016
	3	0.0447	0.0553	0.2386	0.2466	0.0012	0.0032	0.0047	0.0050
	4	0.0445	0.0618	0.2431	0.2471	0.0009	0.0023	0.0044	0.0046

The validation samples are generated using the same method that is used to create

the training samples. As depicted in these figures, all training convergence occurs in a reasonable number of epochs.

Each column in Tables 6.1 and 6.2 consists of the training and the validation MSE for an action neural network related to one basic action. The numbers shown in these tables are scaled by 1.0e+03. As can be seen from these results, the training and validation errors are very low, which demonstrates that the action model networks can predict the next state of the robot after performing each basic action with a high degree of accuracy.

6.4.2 Explanation-based Fuzzy Neural Networks Training and Validation Results

The input training vector $\boldsymbol{s_i}$ for the EB-ANFIS consists of the *X*-*Y* coordinates for the corners of the desired object in the camera's image plane. The output training sample for each input training vector is the calculated Q-value. Another set of input/output samples are collected for centering and reaching for the EB-ANFIS validation. These samples are from separate on-line random episodes and are used to validate the performance of the EBFNN. The method of gathering validation samples is the same as that used for the training samples.

Tables 6.3 and 6.4 include a number of training and validation samples and the MSE for EB-ANFIS centering and reaching behavior. For these results, the number of on-line learning episodes for centering and reaching is 300 and 50 respectively. Each column in these tables consist a number of training and validation samples, and the MSE for an EB-ANFIS related to one basic action. The last columns indicate the total sample numbers and the MSE. As can be seen, the training and validation errors for all EB-ANFIS are

quite similar and are very low. These results illustrate that all EB-ANFIS respond to both the input training and validation instances with a high degree of accuracy.

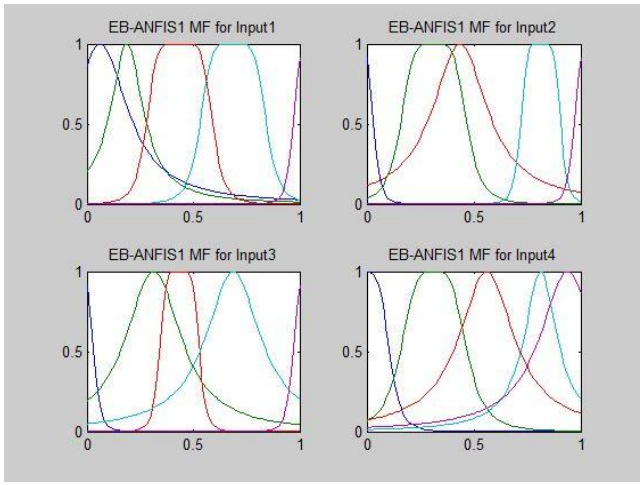
The high performance of EB-ANFIS in generating accurate outputs for unseen inputs demonstrates its generalization capability. Sample centering and reaching membership functions for EB-ANFIS inputs 1 to 4 are shown in Figure 6.4. In this figure each membership function is shown by a different color. This figure illustrates how the centre, the width, and the slope of the sample membership functions are adjusted after on-line training. Tuning of these membership functions is performed using the Tangent-Prop algorithm during EB-ANFIS training. The learning rate for the best performance of ANFIS is η=0.9, which is found empirically.

Table 6.3: Centering training and validation samples and MSE for EB-ANFIS

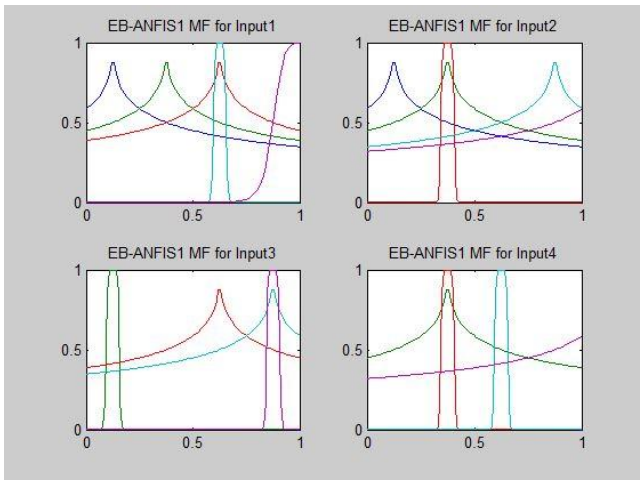
	B.A.1 q_1:+1º	B.A.2 q_1:-1º	B.A.3 q_1:+5º	B.A.4 q_1:-5º	B.A.5 q_1:+10º	B.A.6 q_1:-10º	Total
Training MSE	0.0074	0.0124	0.0084	0.0093	0.0110	0.0070	3.7241
Training Samples	1043	1019	1003	972	867	836	5740
Validation MSE	0.0100	0.0149	0.0106	0.0129	0.0166	0.0094	3.1147
Validation Samples	654	658	563	584	567	571	3597

Table 6.4: Reaching training and validation samples and MSE for EB-ANFIS

	B.A.1 q_2:+1º	B.A.2 q_2:-1º	B.A.3 q_2:+5º	B.A.4 q_2:-5º	B.A.5 q_3:+1º	B.A.6 q_3:-1º	B.A.7 q_3:+5º	B.A.8 q_3:-5º	Total
Training MSE	0.0098	0.0170	0.0129	0.0150	0.1036	0.0174	0.0657	0.0094	5.3042
Training Samples	344	324	312	277	352	318	288	310	2525
Validation MSE	0.0085	0.0134	0.0111	0.0110	0.0990	0.0136	0.0526	0.0074	6.9173
Validation Samples	537	532	484	500	475	530	486	500	4044



(a)



(b)

Figure 6.4: Sample EB-ANFIS final membership functions (a) centering, (b) reaching.

6.4.3 Q-learning EB-ANFIS Real-time Visual Servoing Results

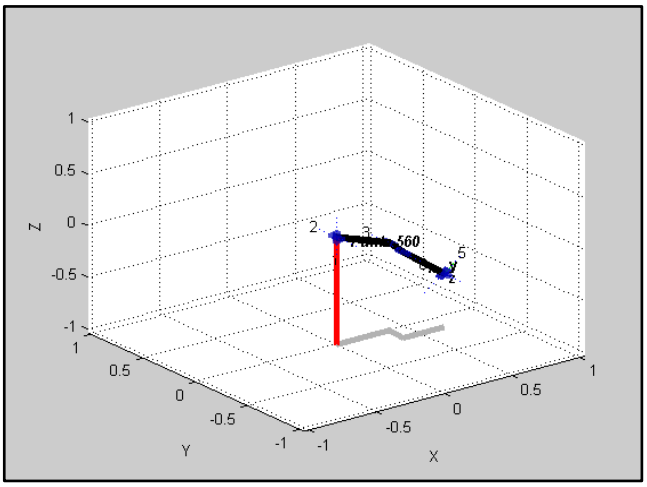
These results are provided for three randomly selected near, far and extremely far real-time start points. The joint angle deviations ($\Delta\theta_1$, $\Delta\theta_2$, $\Delta\theta_3$) at the desired grasping position are (5º, 25º,-45º), (15º, 60º, -100º) and (-25º, 100º, -140º) for near, far and extremely far start points respectively.

The corresponding Cartesian coordinates of near, far and extremely far start points with respect to the manipulator coordinate system are (0.6712, -0.0919, -0.4969),

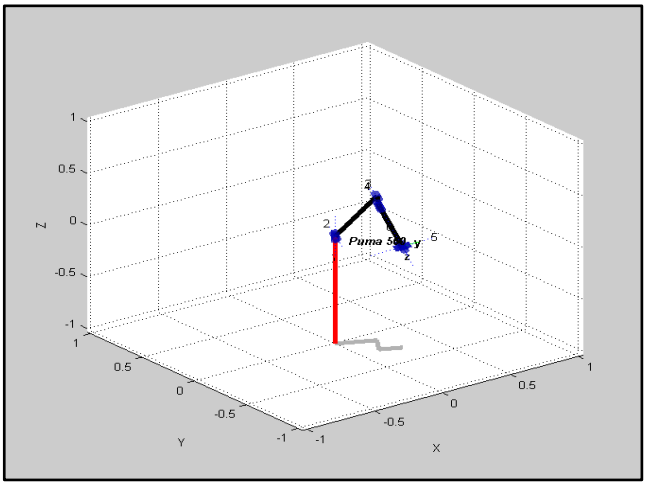
(0.5580, -0.0059, -0.3038), (0.2706, -0.2916, -0.0618). These visual servoing start points are not limited to the learning episode region. The real-time visual servoing start points for the far and extremely far start points are outside of the learning regions. This is done to investigate the generalization capability of the system. The successful visual servoing results from different randomly selected start points demonstrate the performance of the learning algorithm and its generalization capability. Figure 6.5 shows the manipulator at the desired target position and at an extremely far starting position. Visual servoing centering average image features error curves are depicted in Figure 6.6. These figures show that the robot end-effector's centering behavior is completed after performing 1, 2 and 5 basic actions for near, far and extremely far start points respectively. The average image features error is reduced after executing each basic action until this error falls in the acceptable boundary. Two sample error curves for reaching from different start points are also illustrated in Figure 6.7. When the normalized average image features error for centering and the maximum image features error for reaching is less than 0.1mm (equal to 10 pixels) in the image plane then the end-effector is close enough to the target and the success of the visual servoing is recognized.

Figure 6.8 illustrates the trends of centering and reaching immediate rewards for visual servoing with an extremely far episode start point. The Q-learning immediate reward functions for the centering and reaching actions r_{ci} and r_{ri} at the ith state, are calculated based on variations in the image features error between two adjacent states as depicted in Equations (5.6) and (5.7). These immediate reward values are used in the main Q-learning Equation (5.4) to update the Q-evaluation values. As can be seen from these figures however, the immediate reward is direct feedback from the environment to

the agent, but the Q-evaluation values have a final role in determining the visual servoing performance.

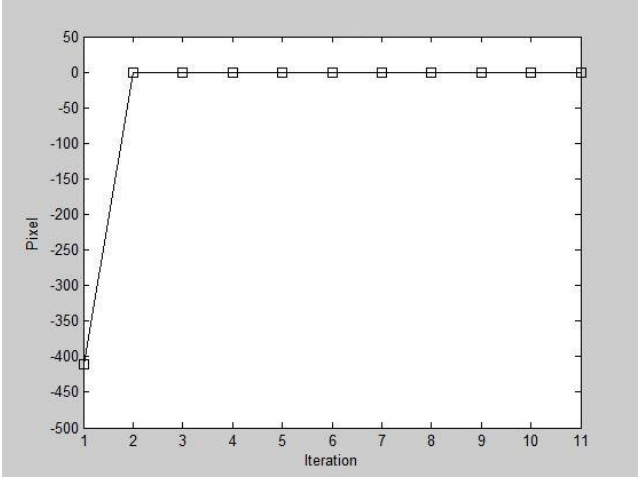


(a)

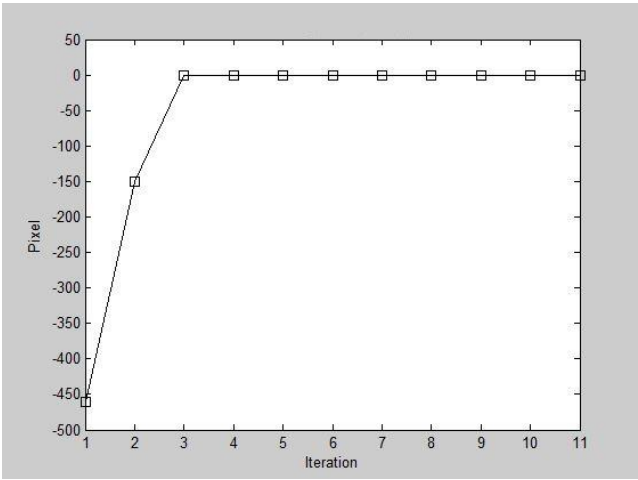


(b)

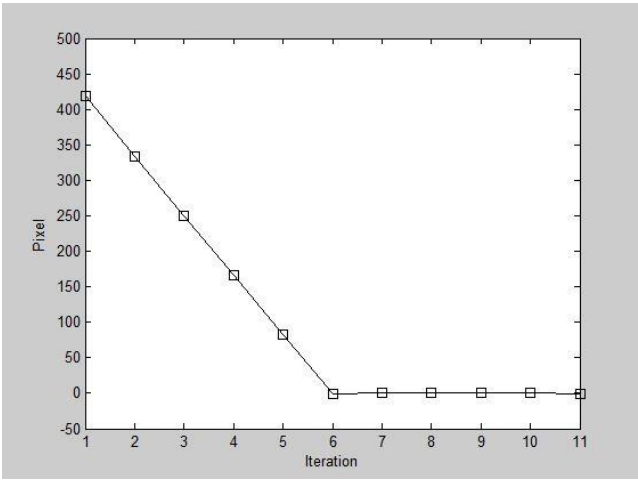
Figure 6.5: Robot manipulator positions (a) desired grasping position, (b) extremely far position.



(a)

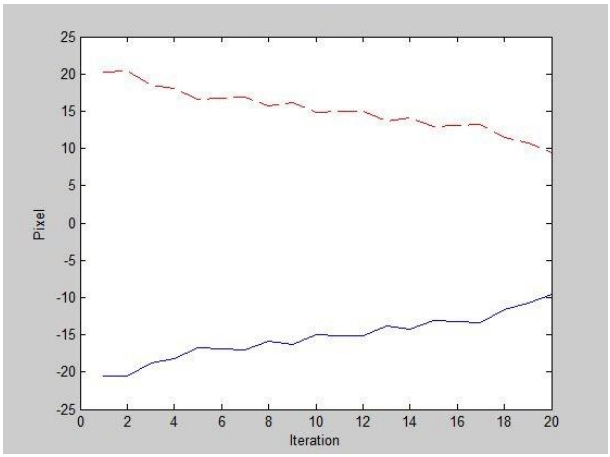


(b)

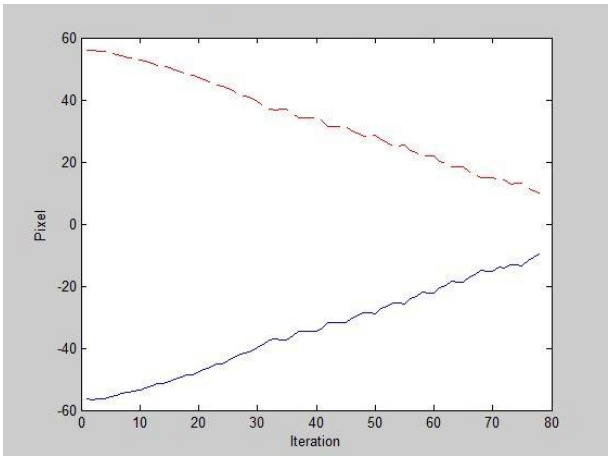


(c)

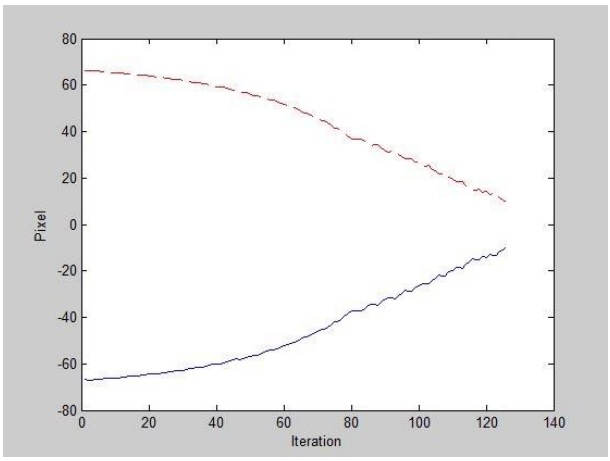
Figure 6.6: Q-learning EB-ANFIS visual servoing centering average image features error: (a) near start point, (b) far start point, and (c) extremely far start point.



(a)

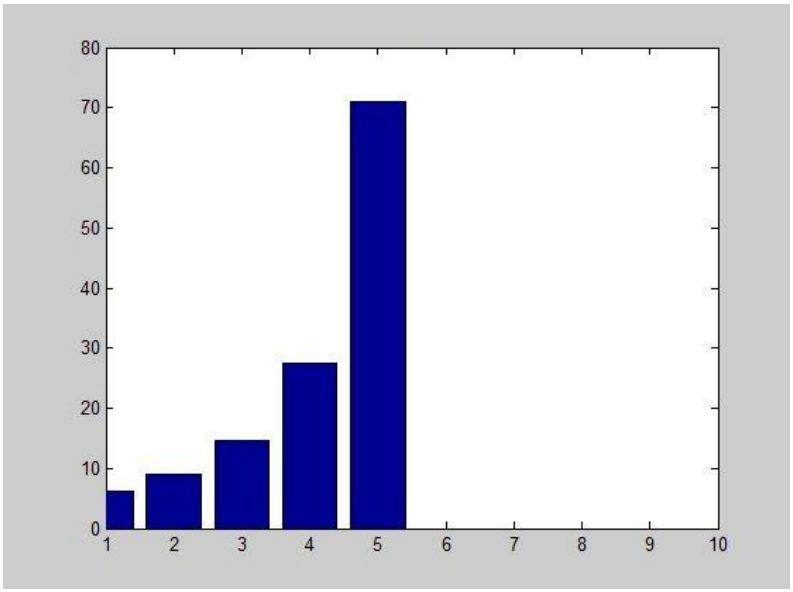


(b)

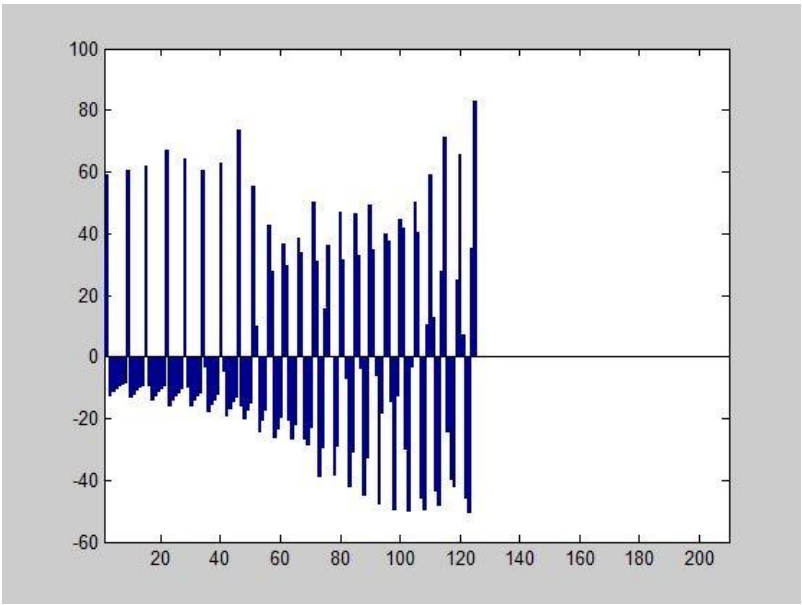


(c)

Figure 6.7: Q-learning EB-ANFIS visual servoing reaching average image features error: (a) near start point, (b) far start point, and (c) extremely far start point.



(a)



(b)

Figure 6.8: Q-learning EB-ANFIS visual servoing immediate reward (a) centering, (b) reaching.

6.4.4 Comparison of Q-learning EB-ANFIS with Q-learning ANFIS Visual Servoing Results

A set of experiments was performed that compared the learning and performance of a visual servoing system implemented using Q-learning EB-ANFIS and Q-learning ANFIS. The aim of these experiments is to investigate the effect of explanation-based analytical learning on real-time performance of the visual servoing system. Fourteen sets of online learning are performed for both systems. Between 25 to 300 random learning episodes are performed. Visual servoing tasks with random starting positions are executed for each system. The results of the real-time centering tests for systems with different numbers of learning episodes are illustrated in Figure 6.9 (a). This figure shows the centering success rates for different learning episodes. The table below Figure 6.9 (a) includes the number of learning episodes and the success rates for real-time visual servoing. The first row of this table consists of the on-line learning episode iteration numbers. The second and third rows depict the succeed rates for Q-learning EB-ANFIS and Q-learning ANFIS respectively. These results demonstrate that the system's performance is improved when using the EB-ANFIS. The positive effect of the EB-ANFIS is greater for smaller numbers of training samples. The EB-ANFIS success rate for 25 learning episodes is 69% which is 21% higher than when using the ANFIS. This rate reaches 91% for 300 learning episodes. The ANFIS success rate for 300 learning episodes is 88%. Higher EB-ANFIS centering success rates are also achievable if the acceptable centering error is increased. The reaching task may still be successful for higher ranges of centering error. The same set of experiments is performed for the reaching task. Figure 6.9 (b) shows the variations in reaching success rate with learning episode numbers. The table below Figure 6.9 (b) contains the same types of information as the table below Figure 6.9 (a) but for reaching

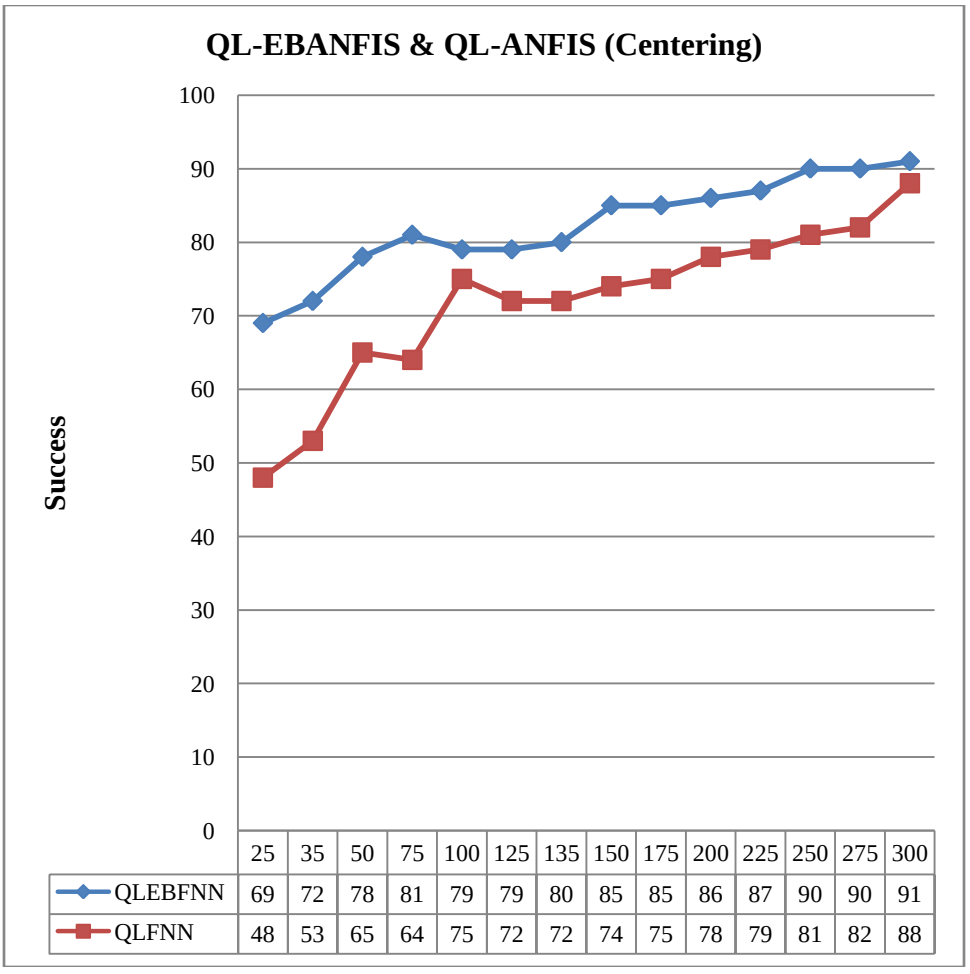
behavior. This shows that the EB-ANFIS can achieve a 100% success rate after a minimum of 50 number learning episodes, but the ANFIS requires a minimum of 100 learning episodes to achieve a 100% success rate.

Figure 6.10 illustrates the centering and reaching Q-learning ANFIS visual servoing average image features error for an extremely far start point. If the results in Figure 6.10 are compared with the same curves for Q-learning EB-ANFIS in Figure 6.6(c) and Figure 6.7(c), it is evident that the successful tasks are achieved using fewer basic actions in the case of Q-learning EB-ANFIS.

Centering basic action numbers for ANFIS and EB-ANFIS are 9 and 5 respectively, which demonstrates an 80% reduction in the number of required basic actions. The number of basic actions required for reaching using the ANFIS is 199. For the EB-ANFIS this number is 125, which shows a 59.2% reduction in the number of required basic actions to complete the task.

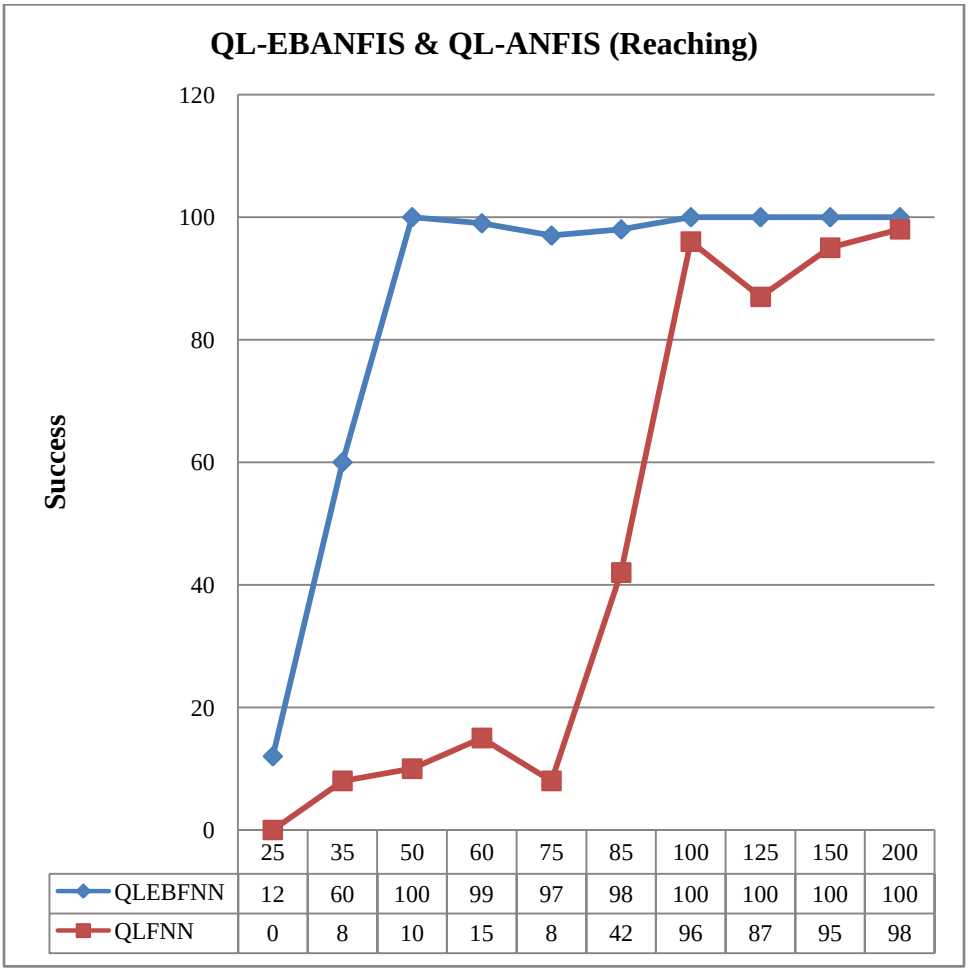
A comparison of the average number of basic actions for Q-learning EB-ANFIS and Q-learning ANFIS in the reaching task is summarized in Tables 6.5. The number of basic actions required to complete the reaching task for Q-learning EB-ANFIS and Q-learning ANFIS systems which were trained with 200 online episodes, are 87 and 127 respectively. This shows 40 basic actions or a 31.5% reduction in the case of EB-ANFIS.

These results demonstrate that explanation-based learning not only causes the visual servoing system to learn the determined task in considerably fewer training episodes, but that it also decreases the number of basic actions used in the real-time testing phase to complete the task. The statistical proof of these results is presented in the next chapter.



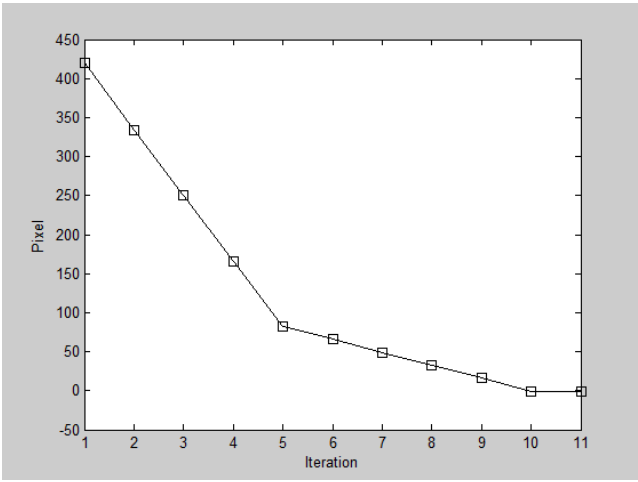
	25	35	50	75	100	125	135	150	175	200	225	250	275	300
QLEBFNN	69	72	78	81	79	79	80	85	85	86	87	90	90	91
QLFNN	48	53	65	64	75	72	72	74	75	78	79	81	82	88

(a)

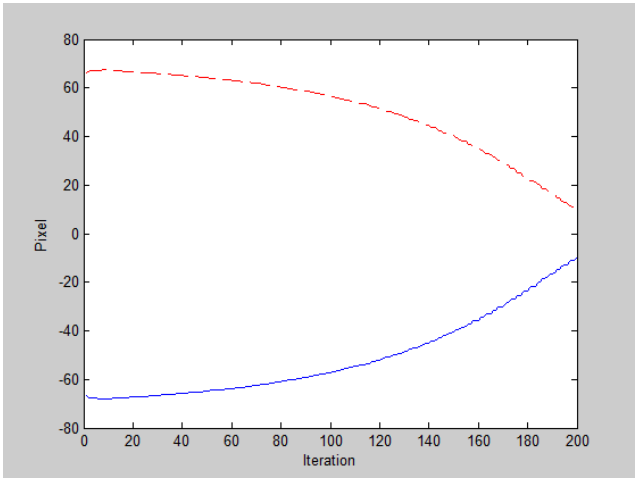


(b)

Figure 6.9: Comparison of Q-learning EB-ANFIS and Q-learning ANFIS real-time performance (a) centering, (b) reaching.



(a)



(b)

Figure 6.10: Q-learning ANFIS Visual servoing average image features error for an extremely far start point a) centering, (b) reaching.

Table 6.5: Comparison of average basic action numbers for Q-learning EB-ANFIS and Q-learning ANFIS in the reaching task

Episodes	25	50	75	100	200
QLEBFNN	-	115	85	92	87
QLFNN	-	-	-	122	127

6.5 Chapter Summary

A self-learning visual servoing system based on the hybrid induction and analytical learning methodologies is proposed in this chapter. Online learning is performed using a Q-learning algorithm which is implemented by the EBFNN for a highly non-linear robot manipulator system in a continuous work-space. The simulation results illustrate the successful performance of the self-learning system in a reasonable number of trial episodes. The robot is also capable of generalizing about its work-space and can perform the visual servoing task from episode start points outside of its training data set successfully. The intelligent behavior of the robot prevents it from losing track of the target object during real-time visual servoing.

The proposed visual servoing system does not require camera calibration because it does not need a robot and camera mathematical model for its controller design. Camera calibration is the procedure of estimating the camera's intrinsic and extrinsic parameters in relation to the world coordinate system. These parameters build the camera projection model and are unknown in practice. The camera's intrinsic parameters were discussed in matrix $\boldsymbol{K}$ of Equation (2.2). The camera pose at base coordinates ${}^{o}\hat{T}_{c}$ includes extrinsic parameters as introduced in Equation (3.2) of Chapter 3. The camera's principle point is not normally at the centre of the photosite array. The accuracy of the lens' focal length is only 4% of what it claims to be, and is only accurate if the lens is focused at infinity. If the lens is detached and reattached, or adjusted for focus or aperture, it is common for the intrinsic parameters to change. The only intrinsic parameters that may probably be acquired from the sensor manufacturer's data sheet are the photosite dimensions determined by scaling factors k_u and k_v along the u and v axes in Equation (2.2).

Estimation of camera's intrinsic and extrinsic parameters in the calibration process causes an error in locating the center point of the camera. This error can affect the performance of traditional visual servoing systems that need a camera projection model in the controller design. The calibration methods rely on sets of world points whose relative coordinates and also whose analogous image plane coordinates are known, and are always a tedious and time consuming task.

The proposed system also does not require a Jacobian matrix calculation, which is always needed for image-based visual servoing systems. The results also demonstrate that using explanation-based learning can improve the learning process and the real-time performance of the system. Q-learning with EB-ANFIS, in comparison to Q-learning with ANFIS, learns the visual servoing task in a considerably lesser number of online episodes. It also improves the real-time performance of the system by reducing the required numbers of basic actions to fulfill the visual servoing task.

The learning algorithms, which are discussed in this and the previous chapters, are methodologies to find the optimal policy for learning procedures and the real-time performance of the system, and will be on the top level of robot manipulator controller design. The stability of the total system depends on the specific design of the controller for the robot arm. The statistical hypothesis proof for the obtained results will be presented in the next chapter.

CHAPTER SEVEN

STATISTICAL HYPOTHESIS TESTS

7.1 Introduction

In this chapter statistical hypothesis tests are used to demonstrate the statistical significance of the results explained in the previous chapter.

Having set the null hypothesis, the probability of the observed data sample is calculated as if the null hypothesis were true. This probability is known as the p-value. Smaller p-values suggest that the null hypothesis is less likely to be true. If the p-value is small, then it is said that the data is unlikely to have happened if the null hypothesis were true. The null hypothesis is not disproved; instead, the sample is unlikely but not impossible. If the p-value is not small, then it is said that the data is consistent with the null hypothesis. There are many different types of statistical significant proof. When a p-value is low it is said that the observed sample value is significantly different from the hypothesized population value. The lower the p-value, the more significant it is said to be. If the p-value is very low, it is said the result is highly significant. It is a fairly common practice for p-values of less than 0.05 to be called 'significant', whilst those > 0.05 are said to be 'non-significant'.

The test that is appropriate for a given situation will depend on the type of outcome variable being studied (categorical or numerical) and the number of variables is considered. Significance tests yield a p-value which quantifies how likely a null hypothesis is to be true.

Two statistical tests are used: T-test and the Wilcoxon-Mann-Whitney U test that are parametric and non-parametric test types respectively. In parametric statistics [54] the

assumption is that the data has originated from a particular probability distribution and is composed of deductions about the distribution parameters. In non-parametric statistics [55] the populations do not have a specific distribution or parameters. Parametric methods make more assumptions than non-parametric approaches. Parametric methods can generate more precise estimates if those additional assumptions are correct. The parametric techniques can be misleading if those assumptions are wrong. For that reason two tests have been selected from two different categories: parametric and non-parametric.

7.1.1 T-Test

This test is used when a numerical variable is considered and the averages of two separate populations or groups are compared [56]. The T-test is normally employed if the samples have a normal distribution. The null hypothesis tends to be that there is no difference between the means of the two populations. The *t* value is calculated using the following Equation [56]:

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{(\frac{s_x^{\,2}}{n} + \frac{s_y^{\,2}}{m})}} \tag{7.1}$$

where $\bar{x}$ and $\bar{y}$ are the sample means, s_x and s_y are the sample standard deviations, and *m* and *n* are the sample sizes.

7.1.2 Wilcoxon-Mann-Whitney U Test

The Wilcoxon-Mann-Whitney U test [57] is a non-parametric test. This test compares two groups, treatments or conditions without the assumption that the values are normally distributed. It can be applied to both normal and non-normal distributions. The

requirements of this test are two random, independent samples with an ordinal, interval or ratio scale of measurement. The null hypothesis asserts that the medians of the samples are identical. The following steps are performed to find the test results for large sample numbers:

1. Organize all the samples into a single ranked series.
2. Calculate the rank of all data without considering the sample's source.
3. Add the ranks for the observations that are belong to sample 1 and 2 to find R_1 and R_2 respectively.
4. Calculate the U_1 and U_2 values using Equations (7.2) and (7.3) [57].
5. Compare the smaller value of U_1 and U_2 with the critical values in significance tables. These tables cannot be used for $n>20$, and p value can be computed using the normal distribution approximation.

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2} \tag{7.2}$$

$$U_2 = R_2 - \frac{n_2(n_2+1)}{2} \tag{7.3}$$

The T-test and the Wilcoxon-Mann-Whitney statistical test for QLEBFNN and QLFNN systems results are discussed in the following section.

7.2 Statistical Tests for QLEBFNN and QLFNN Visual Servoing Systems

A comparison of Q-learning EB-ANFIS with Q-learning ANFIS visual servoing results are described in the previous chapter. These results include:

1. The real-time visual servoing success rates for QLEBFNN and QLFNN systems for centering and reaching behaviours.

2. The real-time visual servoing basic action counts of QLEBFNN and QLFNN systems required for centering and reaching behaviour.

A test for normal distribution for each of the above samples is executed using the z-test function. The null hypothesis for this test is that the collected data are a random sample from a normal distribution with a specified mean *m* and a standard deviation, against the alternative hypothesis that the mean is not *m*. The result of the test points is to prove or reject the null hypothesis at the 5% significance level. The results show that all sample distributions for centering and reaching success rates, and basic action counts of QLEBFNN and QLFNN systems is approximately normal. The above outcomes show the reliability of the T-test results. However, two parametric and non-parametric tests are selected to demonstrate the significant importance of the results regardless of the assumption that samples are normally distributed.

The T-test and the Wilcoxon-Mann-Whitney statistical test are performed for each of the above results and are summarized in the following sections.

7.2.1 Statistical Tests for QLEBFNN and QLFNN Centering Success Rates

The results of real-time centering visual servoing for systems with different total numbers of learning episodes are illustrated in Figure 6.9 (a). These results are applied to perform the statistical tests:

1. T-test: Tests the null hypothesis that the samples come from populations with equal means. These are the mean success rates for centering using the QLEBFNN and QLFNN compared against the alternative that the means are not equal. The test rejects the null hypothesis at $\alpha = 0.01$ significance level, T-value = 3.0212, and p-value = 0.0064.

2. The Wilcoxon-Mann-Whitney test: Tests the null hypothesis that the samples come from populations with equal medians, against the alternative that the medians are unequal. The test rejects the null hypothesis at $\alpha = 0.01$ significance level. The p-value = 0.0081, and z-score = 2.6467. The U-value is 40. The critical value of U at $\alpha \leq 0.01$ is 42.

7.2.2 Statistical Tests for QLEBFNN and QLFNN Reaching Success Rates

The results of real-time reaching visual servoing for systems with different numbers of learning episodes are illustrated in Figure 6.9 (b). These results are used to perform the statistical tests:

1. T-test: Tests the null hypothesis that the samples come from populations with equal means (the means for reaching success rates using the QLEBFNN and QLFNN), against the alternative that the means are unequal. The test rejects the null hypothesis at $\alpha = 0.05$ significance level, T-value = 2.4859, and p-value = 0.0245.
2. Wilcoxon-Mann-Whitney test: Tests the null hypothesis that the samples come from populations with equal medians, against the alternative that the medians are unequal. The test rejects the null hypothesis at $\alpha = 0.01$ significance level. The p-value = 0.0038, and z-score = 2.8966. The U-value is 11.5. The critical value of U at $\alpha \leq 0.01$ is 16.

7.2.3 Statistical Tests for QLEBFNN and QLFNN Centering Basic Action Counts

The results of real-time centering visual servoing for systems using 300 learning episodes are illustrated in Figures 7.1 (a) and (b). The success counts are 91 and 88 for QLEBFNN and QLFNN respectively. In these figures the horizontal axis is the successful real-time

visual servoing experiment number, and the vertical axis is the required basic action counts that robot has performed for a successful visual servoing centering task. These results are used to perform the statistical tests:

1. T-test: Tests the null hypothesis that the samples come from populations with equal means (the means of the centering basic action counts for QLEBFNN and QLFNN), against the alternative that the means are unequal. The test rejects the null hypothesis at $\alpha = 0.01$ significance level, T-value = -4.9545, and p-value = 1.6860e-06.
2. Wilcoxon-Mann-Whitney test: Tests the null hypothesis that the samples come from populations with equal medians, against the alternative that the medians are unequal. The test rejects the null hypothesis at $\alpha = 0.01$ significance level. The p-value = 5.2811e-09, and z-score = 5.8381. The U-value is 2011. The distribution is approximately normal. Therefore, the above Z-value can be used.

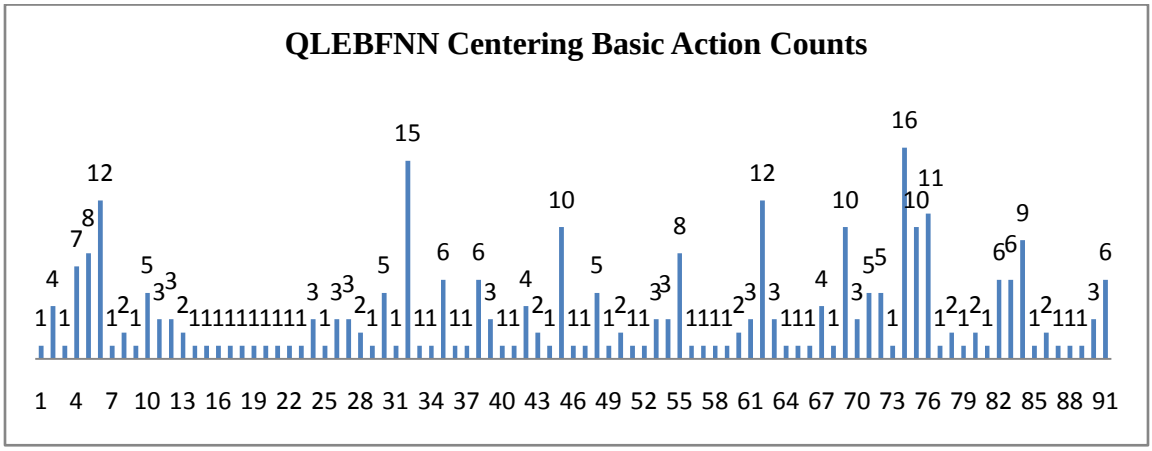
7.2.4 Statistical Tests for QLEBFNN and QLFNN Reaching Basic Action Counts

The results of real-time reaching visual servoing for systems using 200 learning episodes are illustrated in Figure 7.2 (a) and (b). The success counts are 73 and 71 for QLEBFNN and QLFNN respectively. In these figures the horizontal axis is the successful real-time visual servoing experiment number, and the vertical axis is the required basic action counts that robot has performed for a successful visual servoing reaching task. These results are used to perform the statistical tests:

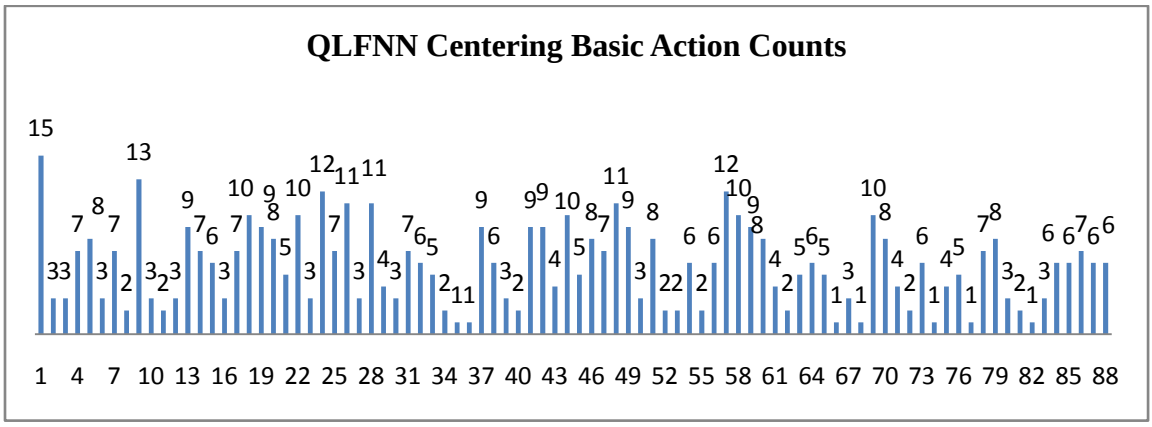
1. T-test: Tests the null hypothesis that the samples come from populations with equal means (the mean of the reaching basic action counts for QLEBFNN and QLFNN), against the alternative that the means are unequal. The test rejects the

null hypothesis at α = 0.01 significance level, T-value = -5.6061, and p-value = 1.5175e-07.

2. Wilcoxon-Mann-Whitney test: Tests the null hypothesis that the samples come from populations with equal medians, against the alternative that the medians are unequal. The test rejects the null hypothesis at α = 0.01 significance level. The p-value = 1.0849e-08, and z-score = 5.7169. The U-value is 1160.5. The distribution is approximately normal. Therefore, the Z-value can be used.

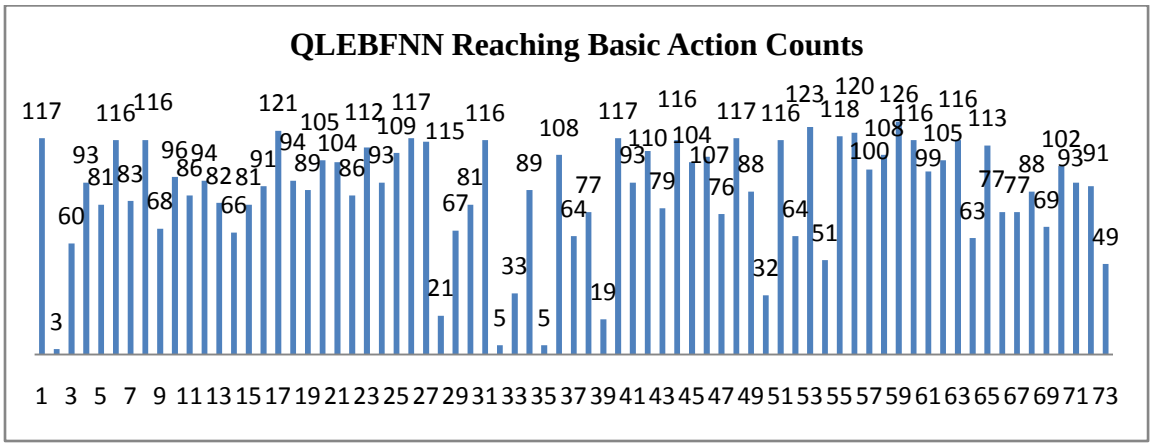


(a)

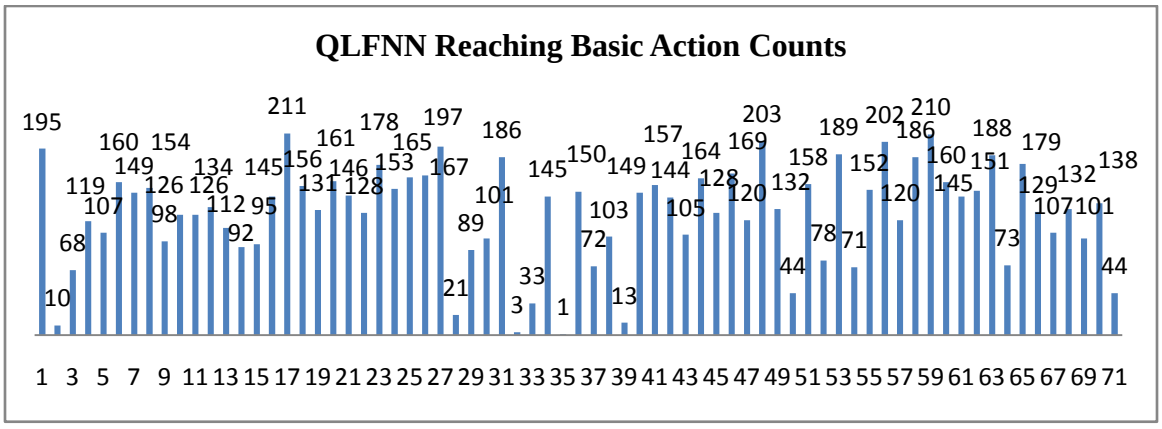


(b)

Figure 7.1: Centering basic action counts (a) QLEBFNN and (b) QLFNN.



(a)



(b)

Figure 7.2: Reaching basic action counts (a) QLEBFNN and (b) QLFNN

These analyses demonstrate the statistical significance of the results for QLEBFNN and QLFNN. It has therefore been statistically proven that explanation-based learning allows the system to learn the task in considerably fewer training episodes and also decreases the required amount of basic actions used to complete the task. The explained statistical test results for QLEBFNN and QLFNN are summarized in Table 7.1.

Table 7.1: Statistical test results for QLEBFNN and QLFNN

Data Sample	Test	Result	Significance Level (α)	Test Parameters	Comment
Centering Success Rate	T-test	Rejects the null hypothesis	0.01	T-value = 3.0212, p-value = 0.0064	
	W.M.W. test	Rejects the null hypothesis	0.01	p-value = 0.0081, z-score = 2.6467	U-value=40 U(Critical)=42 at $\alpha \leq 0.01$
Reaching Success Rate	T-test	Rejects the null hypothesis	0.05	T-value = 2.4859, p-value = 0.0245	
	W.M.W. test	Rejects the null hypothesis	0.01	p-value = 0.0038, z-score = 2.8966	U-value=11.5 U(Critical)=16 at $\alpha \leq 0.01$
Centering Basic Action Counts	T-test	Rejects the null hypothesis	0.01	T-value = -4.9545, p-value = 1.6860e-06	
	W.M.W. test	Rejects the null hypothesis	0.01	p-value = 5.2811e-09, z-score = 5.8381	U-value=2011
Reaching Basic Action Counts	T-test	Rejects the null hypothesis	0.01	T-value = -5.6061, p-value = 1.5175e-07	
	W.M.W. test	Rejects the null hypothesis	0.01	p-value = 1.0849e-08, z-score = 5.7169	U-value= 1160.5

7.3 Chapter Summary

In this chapter two statistical hypothesis tests are applied to prove the statistical significance of the results obtained for the proposed visual servoing systems. The real-time visual servoing success rates and the basic action counts of the two systems are collected into separate groups. Then T-test and U-test are implemented on groups of samples for the QLEBFNN and QLFNN systems. The analysis proves the statistical significance of the results.

CHAPTER EIGHT

CONCLUSIONS AND RESEARCH FUTURE DIRECTIONS

8.1 Conclusions

Two new self-learning visual servoing robotic manipulator systems are proposed in this research. These employ Q-learning to find the optimal policy based on reinforcement learning. This policy is utilized by the robot to reach a predetermined object that has been randomly placed in the environment.

Q-learning implementation requires the definition of states and actions pairs. The input state consists of extracted image features. A camera mounted on the robot end-effector captures the target image in each iteration and sends it to a feature extraction unit. The Harris Corner Detector algorithm is selected for feature extraction to detect the object vertexes. Each basic action is one of the robot joint rotations with a determined angle. For example, the J1 rotation with an angle of 5^{o} can be defined as a basic action.

The Q-learning algorithm is implemented using fuzzy neural networks to estimate the Q-evaluation function for each robot action. Each fuzzy neural network is trained using the input state and the Q-value for the basic action in on-line training episodes. These self-learning systems learn the optimal policy in order to select the best basic action that maximizes the cumulative reward received at each time step. The proposed systems are image-based visual servoing systems and do not require camera calibration. This type of visual servoing system does not need a robot and camera mathematical model for its controller design. The proposed systems also do not require a Jacobian matrix calculation which is always needed for image-based visual servoing systems.

In the first system the Q-learning algorithm is implemented using ANFIS to estimate the Q-evaluation function for each action. This system is based on pure inductive learning and is described in Chapter 5. The simulation results illustrate the successful performance of the learning system in a reasonable number of trial episodes. The robot is also capable of making generalization about its work-space.

The main contributions of this research for the first system are as follows:

- The Q-learning algorithm is implemented using neural networks for a highly non-linear robot manipulator visual servoing system. The architecture of the proposed system is illustrated in Figure 5.1 that shows the Q-learning implementation for centering and reaching behaviour using neural networks. The original Q-learning algorithm uses look-up tables instead of neural networks. Using neural networks in this way prevents rote learning and improves generalization of Q-learning with previously unseen state-action pairs. Using neural networks in this manner can also eliminate the effects of quantization error.

- Another contribution is the use of the Adaptive Neuro Fuzzy Interface System (ANFIS) for function approximation in conjunction with the Q-learning algorithm. The ANFIS can achieve highly non-linear mappings with a high performance level. The ANFIS requires fewer parameters than other network architectures such as multilayer feed forward neural networks. This can decrease the necessary number of training iterations and the training time. The use of ANFIS leads to a fast convergence of parameters that accurately estimate the fundamental dynamics. The ANFIS includes a set of fuzzy rules, which are local mappings, and is particularly important in on-line learning. The results of ANFIS

training in on-line episodes are summarized in Tables 5.2 and 5.3. These tables include training and validation sample numbers and Mean Square Error (MSE) for centering and reaching behavior. These results show that all ANFIS respond to the both the input training and the validation instances with a high degree of accuracy. The high performance level of ANFIS in generating accurate outputs for the unseen inputs illustrates its generalization capability. Real-time visual servoing centering and reaching average image features error curves are depicted in Figure 5.7 and 5.8, which demonstrate the successful performance of the learning system in a reasonable number of trial episodes.

In the second system explained in Chapter 6 an analytical learning component is added. This system includes two main properties, on-line self training and lifelong learning which are implemented by the Q-learning algorithm and EBFNN respectively.

Reducing the required training samples and therefore the learning time, is desirable for a self-learning system. The target for such a system is to reduce the size of the training dataset and still be able to achieve a determined accuracy. Q-learning and EBFNN are used to implement on-line learning with a reduced number of training samples, and therefore training time. Using the EBFNN makes transferring the knowledge between different tasks during the system's lifetime possible.

For example, this system could be used in a flexible manufacturing plant where a robot manipulator is installed for parts assembly consisting of pick up and place tasks. This robot could perform several different tasks during its life. It is necessary to program and train the robot for each new task, which will require more engineering design time and expense. The proposed systems can also be used in applications in which an accurate

mathematical model for the robot and camera are not available, such as planetary exploration. The proposed visual servoing system is based on self-learning that does not need off-line training for new tasks. Off-line initial training is executed for basic tasks to build the domain knowledge in action neural networks after it has been installed in the environment.

On-line learning is established using Q-learning for each new task. Background knowledge is stored in action neural networks and will be transferred to new tasks during on-line learning to reduce the required training samples and time. Trained action neural networks store the changes in extracted image features (states) with the changes in robot joints (actions). These changes are partial derivatives (slopes) of the image features with respect to the robot joints that build the image Jacobian. The image Jacobian contains the robot and camera model information. These trained neural networks are used to learn the new tasks during on-line training. They transfer the image Jacobian knowledge stored in these networks to increase the speed of the learning process. It can be concluded that the trained EBFNNs take the place of the image Jacobian and contain the inherent knowledge of a visual servoing system. This knowledge is stored in EBFNNs and can be used throughout the life of the robot.

The main contributions of the research for the second system are as follows:

- This work integrates Q-learning and explanation-based neural networks for visual servoing of a robot manipulator. The proposed system architecture is shown in Figure 6.1, which illustrates the integration of Q-learning and explanation-based ANFIS for centering and reaching behaviour.

- A contribution of this research is the use of an ANFIS with an EBNN structure in

conjunction with a Q-learning algorithm. Tables 6.3 and 6.4 include training and validation sample numbers and the MSE for EB-ANFIS centering and reaching behaviour. As can be seen, the training and validation errors for all EB-ANFIS are quite close together and have low values. These results show that all EB-ANFIS respond to both the input training and the validation instances with a high degree of accuracy. Figures 6.6 to 6.10 include the results of the Q-learning EB-ANFIS and its comparison with the Q-learning ANFIS. The results demonstrate that the use of explanation-based learning improves the learning process and the real-time performance of the system. Q-learning with EB-ANFIS, in comparison to Q-learning with ANFIS, learns the task in considerably fewer online episodes. It also improves the real-time performance of the system by reducing the required count of basic actions needed to fulfill the task.

A statistical proof of the above results is presented in Chapter 7. Two statistical tests, T-test and the Wilcoxon-Mann-Whitney U test, are used to prove the statistical significance of the results. The T-test analysis tests the null hypothesis that the samples come from populations with equal means (the centering and reaching success means for QLEBFNN and QLFNN), against the alternative that the means are unequal. The Wilcoxon-Mann-Whitney U analysis tests the null hypothesis that the samples come from populations with equal medians (centering and reaching success medians for QLEBFNN and QLFNN), against the alternative that the means are unequal. The same tests are performed for the required count of basic actions to complete the centering and reaching tasks.

8.2 Research Future Directions

Implementation of the proposed self-learning visual servoing systems using a real robot manipulator can be a future research area. For this purpose some key points should be considered for system integration. The image frame grabber hardware and software is needed to provide an interface between a camera mounted on the robot and a computer. The frame grabber captures an individual digital image from a video stream and sends it to the computer. The image features extraction algorithm, on-line learning, and Real-time Visual Servoing algorithms are executed by the computer. An interface program is required between the computer and the robot manipulator controller to send the robot's joints command signals. The robot controller receives the centering and reaching signals and generates the proper torque vector to control the relevant robot joints at each time instance.

***

LIST OF PAPERS

1. **Self-learning Visual Servoing of Robot Manipulator Using Explanation-Based Fuzzy Neural Networks and Q-learning.** Published by "Journal of Intelligent & Robotic Systems", Springer, (Copyright), Nov. 01, 2014, (DOI 10.1007/s10846-014-0151-5).

2. **Self-learning Visual Servoing of Robot Manipulator Using Q-learning Algorithm and Fuzzy Neural Networks.** Submitted to "International Journal of Robotics and Automation", ACTA Press.

REFERENCES CITED

[1] J. Hill & W.T. Park, Real Time Control of a Robot with a Mobile Camera, *in Proc. 9th ISIR*, Washington, D.C., Mar. 1979, 233-246.

[2] S. Hutchinson, G.D. Hager & P.I. Corke, A Tutorial on Visual Servo Control, *IEEE Transactions on Robotics and Automation, 12*(5), 1996, 651-670.

[3] Jianwei Zhang, Ralf Schmidt & Alois Knoll, Appearance-Based Visual Learning in Neuro-Fuzzy Model for Fine-Positioning of Manipulator, *Proceeding of the IEEE International Conference on Robotics and Automation*, *2*, May 1999 , 1164-1169.

[4] Qingjie Zhao, Zengqi Sun, Fuchun Sun & Jihong Zhu, Appearance-based Robot Visual Servo via a Wavelet Neural Network, *International Journal of Control, Automation and Systems*, *6*(4), Aug. 2008, 607-612.

[5] Miao Hao, Zengqi Sun & Masakazu Fuji, Image Based Visual Servoing Using Takagi-Sugeno Fuzzy Neural Network Controller, *22nd IEEE International Symposium on Intelligent Control*, Oct. 2007, 53-58.

[6] Hideki Hashimoto, Takashi Kubota, Moonhong Baeg & Fumio Harashima, A scheme for Visual Tracking of Robot Manipulator Using Neural Network, *IEEE International Joint Conference on Neural Networks, 2*, 1991, 1073-1078.

[7] D. Kuhn, J.L. Buessler & J.P. Urban, Neural Approach to Visual Servoing for Robotics Hand Eye Coordination, *IEEE International Conference on Neural Networks*, *5*, 1995, 2364-2369.

[8] Qingjie Zhao, Fasheng Wang & Zengqi Sun, Using Neural Network Technique in Vision-based Robot Curve Tracking, *International Conference on Intelligent Robots and Systems*, 9-15 Oct. 2006, 3817-3822.

[9] Q.M. Jonathan Wu & Kevin Stanley, Modular Neural-Visual Servoing Using a Neural-Fuzzy Decision Network, *Proceeding of the IEEE International Conference on Robotics and Automation*, *4*, April 1997, 3238-3243.

[10] Ben J.A. Krose, Michiel J. Van. der Krose & Frans C.A. Groen, Learning Strategies for a Vision Based Neural Controller for a Robot Arm, *Proceedings of International IEEE Workshop on Intelligent Motion Control*, *1*, 1990, 199-203.

[11] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[12] T. Mitchell & S. Thrun, Explanation-based learning, A comparison of symbolic and neural net approaches, *In Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993, 197-204.

[13] R. Sutton, Integrated architectures for learning, planning and reacting based on approximating dynamic programming, *In Proceeding of the seventh International Conference of Machine Learning*, Morgan Kaufmann, 1990, 216-224.

[14] E. Malis, F. Chaumette & S. Boudet, 2 1/2d Visual Servoing, *IEEE Trans. on Robotics and Automation*, April 1999, 234-246.

[15] Lingfeng Deng, Comparison of Image-Based and Position-Based Robot Visual Servoing Methods and Improvements, Ph.D. Dissertation, Waterloo University, Canada, 2003.

[16] Peter Corke, *Robotics, Vision and Control*, Springer, 2011.

[17] O. Faugeras, Q.T. Luong & S.J. Maybank, Camera Self Calibration Theory and Experiments, In G. Sandini Editor, Computer Vision - ECCV' 92, vol. *588* of Lecture Notes in Computer Science, Springer-Verlag, 1992, 321-334.

[18] C. Harris & M. Stephen, A Combined Corner and Edge Detector, *In 4th Alvey Cision Conference*, 1988, 147-151.

[19] J.F. Canny, A Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1986, 679-698.

[20] John J. Craig, Introduction to Robotics Mechanics and Control, Pearson Education Inc., 3rd Edition, 2005.

[21] I.E. Sutherland, Three-dimensional Data Input by Tablet, *Proc. IEEE, 62,* April 1974, 453-461.

[22] R. Tsai & R. Lenz, A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration, *IEEE Trans. Robot. Automat., 5,* June 1989, 345-358.

[23] R. Tsai, A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-The-Shelf TV Cameras and Lenses, *IEEE Trans. Robot. Automat., 3,* Aug. 1987, 323-344.

[24] A.C. Sanderson, L.E.Weiss & C.P. Neuman, Dynamic Sensor-Based Control of Robots with Visual Feedback, *IEEE Trans. Robot. Automat., R.A-3*, Oct. 1987, 404-417.

[25] P.I. Corke, High-Performance Visual Closed-Loop Robot Control, Ph.D. Dissertation, University of Melbourne, Dept. Mechanical and Manufacturing Engineering, July 1994.

[26] P. Liang, J.F Lee & S. Hackwood, A General Framework for Robot Hand-Eye Coordination, *IEEE International Conference on Robotics and Automation, 2,* 1988, 1082-1087.

[27] W. Jang & Z. Bien, Feature-based Visual Servoing of an Eye-In-Hand Robot with Improved Tracking Performance, *Proceedings of the IEEE International Conference on Robotics and Automation, 3,* April 1991, 2254-2260.

[28] M.I. de la Fuente, J. Echanobe, I. del Campo, L. Susperregui & I. Maurtua, Hardware implementation of a Neural-Network Recognition module for Visual Servoing in a Mobile Robot, *Workshops on Database and Expert Systems Applications*, 2010, 226-232.

[29] I. Siradjuddin, L. Behera, T.M. McGinnity & S. Coleman, Image Based Visual Servoing of a 7 DOF Robot Manipulator Using a Distributed Fuzzy Proportional Controller, *IEEE International Conference on Fuzzy Systems*, 2010, 1-8.

[30] Yan-Xi Yang, Ding Liu & Han Liu, Robot Self-Learning Visual Servoing Algorithm Using Neural Networks, *Proceedings of the First International Conference on Machine Learning and Cybernetics*, Beijing, *2*, 4-5 Nov. 2002, 739-742.

[31] G. Flandin, F. Chaumette & E. Marchand, Eye-in-hand / Eye-to-hand Cooperation for Visual Servoing, *Proceedings of the IEEE International Conference on Robotics and Automation, 3,* 2000, 2741-2746.

[32] W. Bing & L. Xiang, A Simulation Research on 3D Visual Servoing Robot Tracking and Grasping a Moving Object, *15th International conference on Mechatronics and Machine Vision in Practice*, Dec. 2008, 362-367.

[33] D. Ramachandram & M. Rajeswari, A Short Review of Neural Network Techniques in Visual Servoing of Robotic Manipulators, School of Computer Science, University Sains Malaysia, 11800 Minden, Penang, Malaysia.

[34] Yan-Xi Yang, Ding Liu & Han Liu, Robot End-Effector 2D Visual Positioning Using Neural Networks, *Proceeding of the second International Conference on Machine Learning and Cybernetics, 5*, Nov. 2003, 2940-2943.

[35] S.K. Nayar, H. Murase & S.A. Nene, Learning, Positioning, and Tracking Visual Appearance, *Proceedings of the IEEE International Conference on Robotics and Automation, 4*, 1994, 3237-3244.

[36] Bach H. Dinh, Matthew W. Dunnigan & Donald S. Reay, Position Control of a Robotic Manipulator Using a Radial Basis Function Network and Simple Vision System, *IEEE International Symposium on Industrial Electronics*, 2008, 1371-1376.

[37] Jyh-Shing & Roger Jang, ANFIS, Adaptive-Network-Based Fuzzy Inference System, *IEEE Transaction on Systems, Man and Cybernetics, 23*, May/June 1993.

[38] Han Liu, Ding Liu & Yan-Xi Yang, Research of Real Time Robot Visual Servoing Based on Genetic Algorithm, *Proceeding of the First International Conference on Machine Learning and Cybernetics*, Beijing, *1*, 4-5 Nov. 2002, 87-90.

[39] Gyula Mester, Neuro-Fuzzy-Genetic Controller Design for Robot Manipulator, *Proceedings of the IEEE IECON 21st International Conference on Industrial Electronics, Control and Instrumentation*, *1*, 1995, 87-92.

[40] M.N.H. Siddique & M.O. Tokhi, GA-based Neuro-Fuzzy Controller for Flexible-Link Manipulator, *Proceedings of the IEEE International Conference on Control Applications*, *1*, Sep. 2002, 471-476.

[41] Sufian Ashraf Mazhari & Surendra Kumar, Hybrid GA Tuned RBF Based Neuro-Fuzzy Controller for Robotics Manipulator, *International Journal of Electrical, Computer and Systems Engineering*, Fall 2008.

[42] L. Kaelbling, Learning in Embedded Systems, MIT Press, 1993.

[43] T. Mitchell, R. Keller & S. Kedar-Cabelli, Explanation-Based Generalization, A unifying view, *Machine Learning*, 1986.

[44] T. M. Mitchell, Joseph O'Sullivan & Sebastian Thrun, Explanation-Based learning for Mobile Robot Perception, *In Workshop on Robot Learning, Eleventh Conference on Machine Learning*, 1996.

[45] Christopher J.C.H. Watkins & Peter Dayan, "Q-Learning", *Machine Learning*, *8*, 1992, 279-292.

[46] Ying Wang, Haoxiang Lang & Clarence W. de Silva, A Robust Mobile Robot Manipulation System Using a Switch-Based Visual-Servo Controller, *International Conference on Industrial Mechatronics and Automation*, Chengdu, 15-16 May 2009, 364-367.

[47] Ying Wang, Haoxiang Lang & Clarence W. de Silva, A Hybrid Visual Servo Controller for Robust Grasping by Wheeled Mobile Robots, *IEEE/ASME Transactions on Mechatronics*, *15*(5), Oct. 2010, 757-769.

[48] Velappa Ganapathy, Soh Chin Yun & Halim Kusama Joe, Neural Q-Learning Controller for Mobile Robot, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, July 14-17, 2009, 863-868.

[49] Junfei Qiao, Zhanjun Hou & Xiaogang Ruan, Q-learning Based on Neural Network in Learning Action Selection of Mobile Robot, *Proceedings of the IEEE, International Conference on Automation and Logistics*, 18 - 21 August 2007, 263-267.

[50] Shi-chao Wang, Zheng-xi Song, Hao Ding & Hao-bin Shi, An Improved Reinforcement Q-Learning Method with BP Neural Networks In Robot Soccer, *Fourth International Symposium on Computational Intelligence and Design*,2011, 177-180.

[51] B. Huang, G. Cao & M. Guo, Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance, *Proceedings of the 4th International Conference on Machine Learning and Cybernetics*, Guangzhou, 2005.

[52] Cervera. E & A. P. del. Pobil, Sensor-based Learning for Practical Planning of Fine Motions in Robotics, *Information Sciences, 145,* 2002, 147-168.

[53] Sebastian Thrun & Tom M. Mitchell, Lifelong robot learning. The biology and technology of intelligent autonomous agents, *NATO ASI Series,* 144, 1995, 165-196.

[54] Geisser S. & Johnson W.M., *Modes of Parametric Statistical Inference,* John Wiley & Sons, 2006.

[55] Gibbons Jean Dickinson & Chakraborti Subhabrata, *Nonparametric Statistical Inference,* 4th Edition, CRC, 2003.

[56] John A. Rice, *Mathematical Statistical and Data Analysis*, Third Edition, Duxbury Advances, 2006.

[57] Myles Hollander & Douglas A. Wolfe, *Nonparametric Statistical Methods*, Second Edition, Wiley-Interscience, 1999.

[58] Ezio Malis, Survey of Vision-based Robot Control, *European Naval Ship Design, Captain Computer IV Forum*, ENSIETA, Brest, France, April, 2002.

[59] Richard Hartley & Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[60] K. Madsen, H.B. Nielsen & O. Tingleff, *Methods for Non-linear Least Squares Problems*, Second Edition, Technical University of Denmark, April 2004.

[61] B. Widrow & M.A. Lehr, 30 Years of Adaptive Neural Networks Perception, Madaline, and Backpropagation, *Proceedings of IEEE, 9*(78), Sep. 1990, 1415-1442.

[62] A. Cretual & F. Chaumette, Positioning a Camera Parallel to a Plane Using Dynamic Visual Servoing, *In IEEE Int. Conf. on Intelligent Robots and Systems,* 1, Grenoble, France, Sep. 1997, 43-48.

[63] R. Andersson, Dynamic Sensing in Ping-Pong Playing Robot, *IEEE Transactions on Robotics and Automation, 5*(6), 1989, 728-739.

[64] David A. Forsyth & Jean Ponce, *Computer Vision- A Modern Approach*, Pearson Prentice Hall, 2009.

[65] Rafael C. Gonzalez & Richard E. Woods, *Digital Image Processing*, Pearson Prentice Hall, Third Edition, 2008.

[66] Z. Miljkovic, M. Mitic, M. Lazarevic & B. Babic, Neural Network Reinforcement Learning for Visual Control of Robot Manipulators, *Expert Systems with Applications*, Elsevier,*40*, 2013, 1721-1736.

[67] http://www.antenen.com.