ADAPTIVE METHODS FOR REALISTIC AND INTUITIVE HUMAN-ROBOT INTERACTION

by

JARTUWAT RAJRUANGRABIN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2010

To my mother Chiaranai, my father Somkiat and my brother Jaratvit.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Dan Popa for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. I would like to also thank Dr. Harry Stephanou, Dr. Frank Lewis, Dr. Kai-Shing Yeung, Dr. Fillia Makedon and Dr. Vasillis Athitsos for their interest in my research and for taking time to serve in my dissertation committee.

I would also like to extend my appreciation to the Royal Thai Government for providing financial support for my graduate studies. I wish to thank my colleages Aditya Das, Rakesh Murthy, M Rashid Pac as well as MS and undergraduate students I have worked with for their help and support. I am grateful to Dr. Harry Stephanou for giving me opportunutiy to work at ARRI, and also Dr. Fillia Makedon for the opportunity to work at the Heracleia Lab.

I would like to thank Dr. Suthee Phoojaruenchanachai for encouraging and inspiring me to pursue graduate studies.

Finally, I would like to express my deep gratitude to my parents who have encouraged and inspired me and sponsored my undergraduate studies. I am also grateful to my mother, father and brother for their sacrifice, encouragement and patience. I also thank several of my friends who have helped me throughout my career.

November 17, 2010

ABSTRACT

ADAPTIVE METHODS FOR REALISTIC AND INTUITIVE HUMAN-ROBOT INTERACTION

JARTUWAT RAJRUANGRABIN, Ph.D.

The University of Texas at Arlington, 2010

Supervising Professor: Dan Popa

Due to the advancement in robotics and computer technology, having access to sophisticated robot hardware is becoming common. An increase in the number of robots allows one to achieve different kind of tasks by exploiting cooperation, such as: robot swarms, object manipulation using multiple degrees of freedom of different robots. It is quite challenging for the human operator to coordinate multiple robots to achieve a task in an efficient manner. The channels available to receive feedback information from the robot system can vary from simple encoder reading to video stream acquisition of the robot system performing tasks. It is important for us to make use of and coordinate as many sensing modalities as it is required to perform necessary tasks. On the other hand, we would want to keep the number of modalities utilized optimal just to maintain usage intuitiveness for human operators and efficiency.

In this dissertation, we have investigated 3 different modalities of interaction with robots. Physical sensing: we present a novel approach to enhance human-robot interactivity through the use of artificial skin and the Extended Kalman filter. Visual sensing: we present a novel approach to enhance interaction of humanoid robot actor through the use of pose estimation and visual servoing. Interface devices: we present work on combining dynamic gestures based commands from an interface device to improve the intuitiveness of control / planning of multiple degrees of freedom robot system through reinforcement learning.

We propose an efficient way to coordinate multiple modalities sensing as generalized interface for multiple robots. Performance metrics are proposed so that we have the quantitative way to identify our interaction efficiency. Eventually, the outcome of this research will be a reconfigurable multiple interface system that can be used with multiple robot systems in a way that it is easy and intuitive for the human operator to operate.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Humanoid robotics is an emerging research area that requires interdisciplinary expertise. With current advancement in sensors and computer hardware, more sophisticated technologies can be integrated into a humanoid robot to make it even more realistic. Realism in both form and function is not necessarily the only aspect of humanoid robot research since there are endless possibilities for the utilization of humanoid robots. To efficiently make use of humanoid robots, we need to have an optimized way to balance the robot's autonomy versus user control. A humanoid robot can be viewed as a system composed of an array of sensors actuators, and such a system requires art and science to coordinate the arrays so that it can accomplish meaningful tasks in an efficient manner.

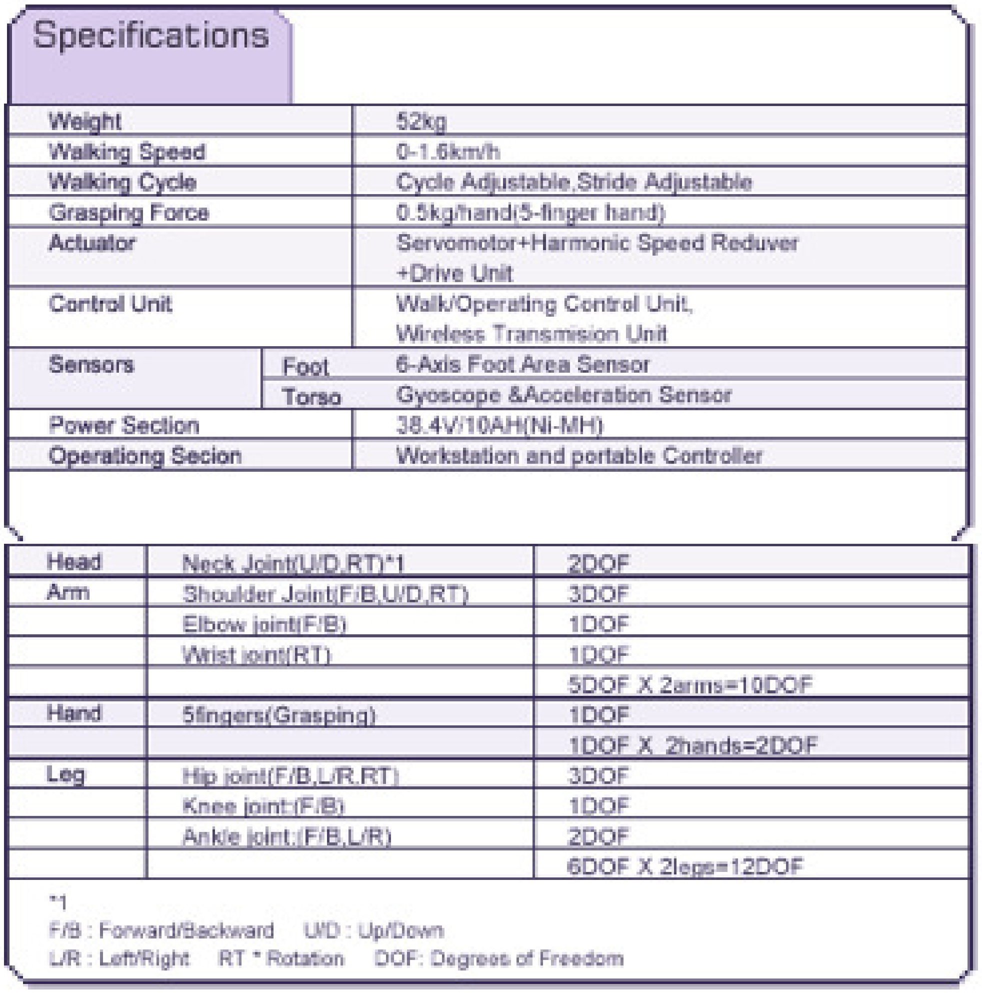
The major sensory input channels to humanoid robots are tactile and visual sensory inputs. By improving the interaction through these sensor modalities, we progress toward having humanoid robots that resemble human in terms of function. Machines that are capable of imitating the complicated behavior of humans, can be used in applications, such as: manufacturing (assembly line), entertainment, patient simulation. It is even more of a challenge to control robots with large arrays of sensors and actuators in order to accomplish meaningful tasks efficiently, without the operator having to put too much effort into learning to control such systems. An example of such a scenario would be a patient with motor skill impairment learning to operate assistive robots that would help him / her perform day to day activities. Such assistive robots can come in different shapes and forms and also with different control interfaces. In terms of manipulating the robots efficiently, it is apparent that there are many challenging aspects to the aforementioned scenario as the most efficient interface for a patient with motor skill impairment is not intuitive.

1.1.1 Characteristics of the Human-Robot Interaction Systems

A human-robot interaction system can refer to something rudimentary such as a simple motor control system or it can refer to something as complicated as a full-fledged humanoid robot. When we use the word "Human-Robot Interaction", it usually refers to an autonomous robot system being able to process human commands and take appropriate actions to achieve the human's desired outcome. To take a closer look at the "Human-Robot Interaction", the complete process consists of:

1. The human having desired outcome in mind
2. The human conveying his/her desired outcome through some forms
3. Robot system receiving human communication message
4. Robot system interpreting the message and taking actions
5. The human receives feedback from the robot system and adjusts the communication message so that desired outcome is achieved (learning)

We can see that in order to have an efficient "Human-Robot Interaction" system, it involves many different factors. The communication may take any form as the technology for interface devices develop. Hence, step 4 is a critical step to determine the overall efficiency of the interaction. In this proposal, we develop a framework that helps improve efficiency of human-robot interaction systems by improving the efficiency and effectiveness of step 4, and at the same time reduces the need for human user to do a lot of step 5.



Specifications

Weight		52kg
Walking Speed		0-1.6km/h
Walking Cycle		Cycle Adjustable,Stride Adjustable
Grasping Force		0.5kg/hand(5-finger hand)
Actuator		Servomotor+Harmonic Speed Reduver +Drive Unit
Control Unit		Walk/Operating Control Unit, Wireless Transmision Unit
Sensors	Foot	6-Axis Foot Area Sensor
	Torso	Gyoscope &Acceleration Sensor
Power Section		38.4V/10AH(Ni-MH)
Operationg Secion		Workstation and portable Controller

Head	Neck Joint(U/D,RT)*1	2DOF
Arm	Shoulder Joint(F/B,U/D,RT)	3DOF
	Elbow joint(F/B)	1DOF
	Wrist joint(RT)	1DOF
		5DOF X 2arms=10DOF
Hand	5fingers(Grasping)	1DOF
		1DOF X 2hands=2DOF
Leg	Hip joint(F/B,L/R,RT)	3DOF
	Knee joint:(F/B)	1DOF
	Ankle joint:(F/B,L/R)	2DOF
		6DOF X 2legs=12DOF

*1
F/B : Forward/Backward U/D : Up/Down
L/R : Left/Right RT * Rotation DOF: Degrees of Freedom

Figure 1.1. Asimo specifications (courtesy of: http://world.honda.com/ASIMO/technology/spec.html).

A real life example of an advanced humanoid robot is the Honda ASIMO. It is obvious that in order for the human operator to be able to interact with the robot, he/she would still have to go through all the steps outlined above. The total of 24 degrees of freedom, as shown in Fig. 1.1, will have to be efficiently coordinated so that the robot can perform meaningful actions.

Generally, the robot is operated in autonomous mode that does not require a human operator supervising commands. However, the development team at Honda

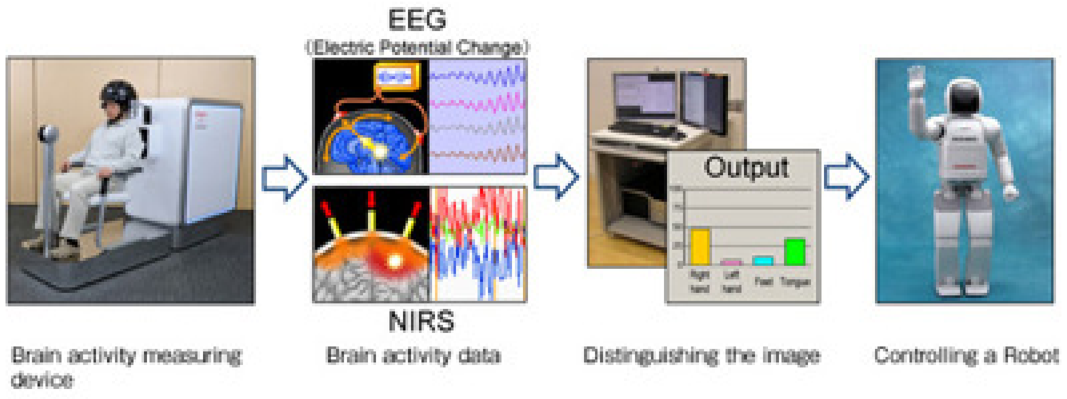


Figure 1.2. Brain machine interface technology for ASIMO robot control (courtesy of: http://world.honda.com/news/2009/c090331Brain-Machine-Interface-Technology).

does see the importance of developing intuitive / easy to use interface. The Honda research team developed brain machine interface (BMI) technology that uses electroencephalography (EEG) and near-infrared spectroscopy (NIRS) along with newly developed information extraction technology to enable control of a robot by human thought alone. The EEG and NIRS sensors are placed on the head of the human operator. The brain activity data get processed and then one of the four predefined body parts will be moved according to the human operator's mental image of the selected body part. The diagram of the overall process is show in Fig. 1.2.

The non-invasive BMI technology that is used in this device has certain limitations as oppose to invasive BMI technology, which involves electrode arrays implanted in the human's head. Even though the concept of using non-contact control for robot operations is novel, the technologies involved are not well-developed yet, so this non-contact control for man-machine interface is still under research.

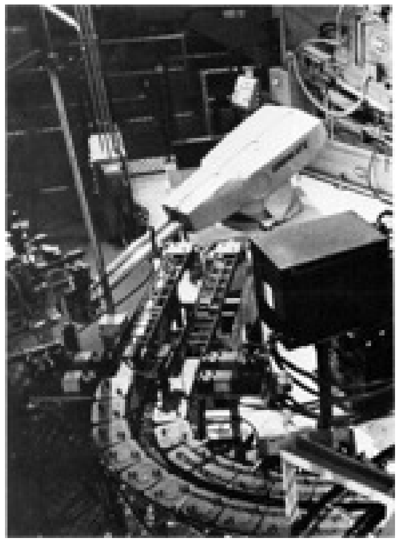


Figure 1.3. 1961 UNIMATE robot arm in the TV picture tubes manufacturing plant (courtesy of: http://www.prsrobots.com/1961.html).

1.1.2 Emerging Technologies for Human-Robot Interaction Systems

User interfaces for man-machine control have a long history dated back to the creation of the first robot arm. In Fig. 1.3, the early version of the UNIMATE arm was operated by storing step-by-step commands on a magnetic drum.

We can see that the interface device started out as something primitive where the operators were not even able to control the robot in real time. After robot manipulators became widely available in the manufacturing industry, a form of tethered control pendant became standard devices for robot operation. A picture of the early UNIMATE robot model that uses control pendant is shown below in Fig. 1.4.

A wide variety of control pendants became the standard for robot control. Basic operations for these control pendants include robot calibration, Cartesian space movement of the end-effector, individual joint movement and open/close gripper. Generally, the control pendant sends commands to a robot arm in an open loop fashion, which means that the control pendant itself is incapable of receiving feedback

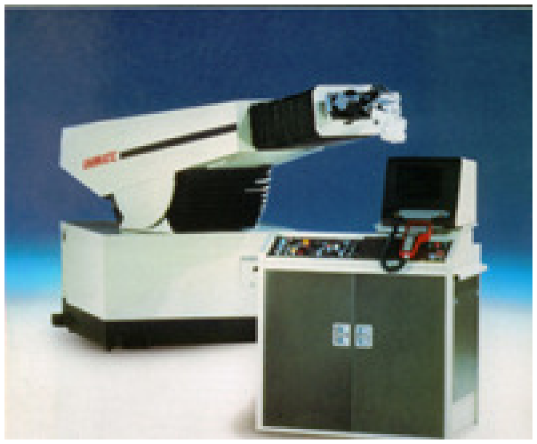


Figure 1.4. Early UNIMATE robot arm model with control box and control pendant (courtesy of: http://www.prsrobots.com/unimate.html).

from the robot arm. However, some of the modern control pendants for robot control may have more functionality. Fig. 1.5 shows the modern control pendant that is used with SCARA robots.

Control pendants are standard form of control for articulated robot arms. But for other types of robots such as mobile robots, humanoid robots and so on, control pendants might not be the best option for robot control interfaces. As other types of robots require more than just positioning the end-effector to a desired position, there is no standard form of control device, the design of the interface device depends heavily on the application and usually designed on an ad-hoc basis.

The gaming industry is also an area in which there is a need to develop interface devices. What has happened in the gaming industry lately is worth noting, since the nature of games is getting more complicated and the requirements of game manipulations are similar to robot control. Interfaces such as joysticks for gaming are also intuitive to some of the robot control applications. Fig. 1.6 shows a generic gaming joystick that can be used for robot control.

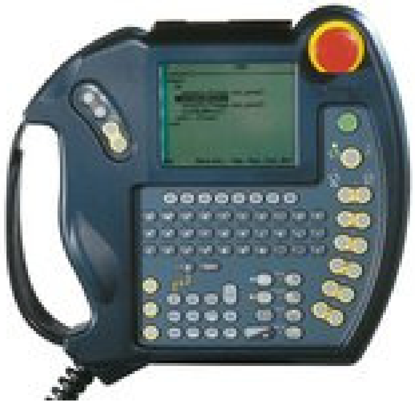


Figure 1.5. Modern control teach pendant for robot arms (courtesy of: http://www.staubli.com/en/robotics/products/robot-controller/robot-control-pendant/).

Figure 1.6. A gaming joystick that can also be used for robot control.

Modern gaming controllers have embedded accelerometers to detect object orientation as well as acceleration; similar devices can be used for robot control. Fig. 1.7 shows a picture of a modern game controller with accelerometer sensors.

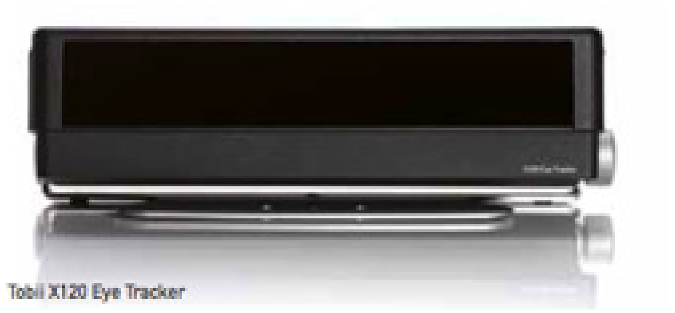
Other advanced interface devices include brain activity sensors, eye trackers etc. These devices are primarily designed for entertainment purposes; however, they have a lot of potential to be used as interface devices for human-robot interaction. Fig. 1.8 shows a commercially available brain activity sensor interface. It has electrodes that read signals from a user's scalp, the signals are processed to classify emotion levels, brain activities and facial gestures. Fig. 1.9 shows the eye tracker device. This device

Figure 1.7. A Nintendo Wii nunchuk controller (courtesy of: http://www.nintendo.com/wii).

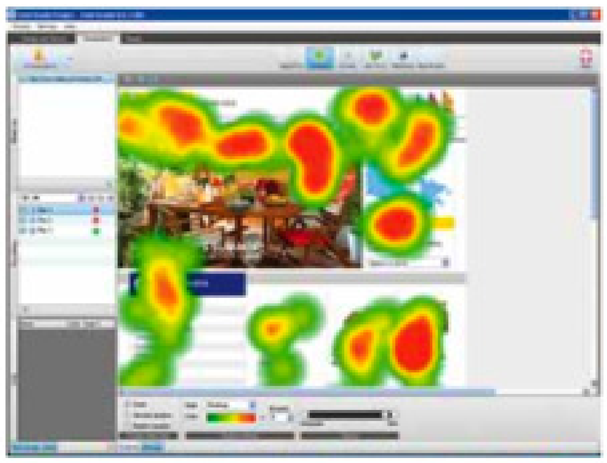
Figure 1.8. Emotiv brain activity sensor as gaming device (courtesy of: http://www.emotiv.com).

tracks and records the movement of the user's pupils with respect to a screen so that the data can later be used for analysis of eye movement patterns.

An example of an advanced interface / actuators system is the exoskeleton system. The operator physically interacts with the system and at the same time controls the system in real-time. This is a good example of a system that has an intuitive interface, however, the physical interaction with the system is the challenging aspect. As the system needs the ability to sense physical commands from the operator



(a)


Tobii Studio Heat map visualization, showing several peoples' views of a print ad.

(b)

Figure 1.9. (a) Tobii eye tracker system (b) Heat map generated from user eye activity (courtesy of: http://www.tobii.com).

so that it can perform appropriate tasks. Fig. 1.10 shows the exoskeleton system developed by team at UC Berkeley [1].

In some cases the interface device might not be available for interactions with robots. Vision is the another important sensing form that is widely used in robots. Humanoid robots usually has the vision sensing capability, and it is difficult to interact with the robot through vision. Fig. 1.11 shows an example of a humanoid robot that has the ability to generate different facial expressions [2]. The challenging aspect would be to make the robot interact with humans through vision in real-time in a

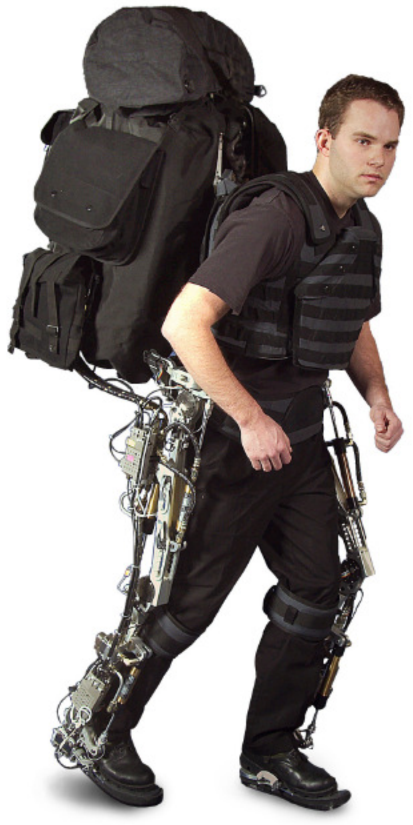


Figure 1.10. The Berkeley Lower Extremity Exoskeleton (BLEEX) [1].

realistic manner. Realistic looking humanoid robots are being developed at many research institutes. Some of the examples of these robots are: MIT robots [3] shown in Fig. 1.12 and Albert HUBO robot by David Hanson [4] shown in Fig. 1.13. Although these robots appear realistic, it is a challenging tasks to add realistic real-time interactions to these robots.

A good example of a very intuitive interface device for humanoid robot controls is the marionette, a small replica of the actual robot to be controlled that has the position sensing capability. Fig. 1.14 shows the marionette used for humanoid robot control [5]. This is the most intuitive interface device for the of controlling the life size humanoid robot as the marionette provides one-to-one mapping of all the joints. A

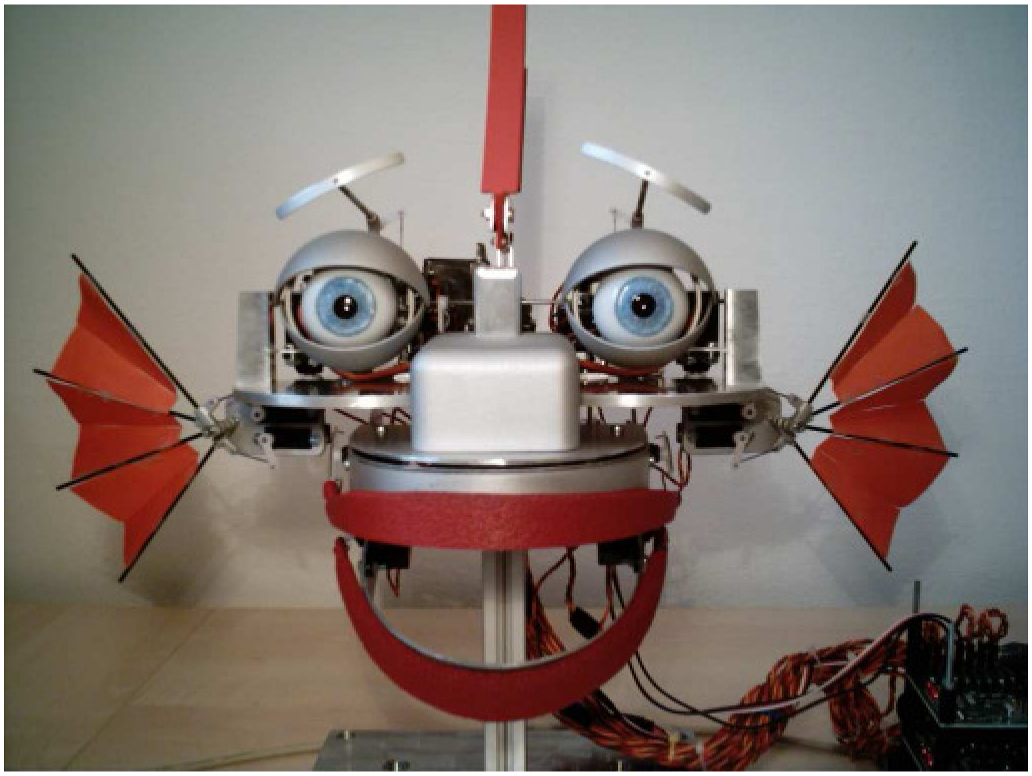
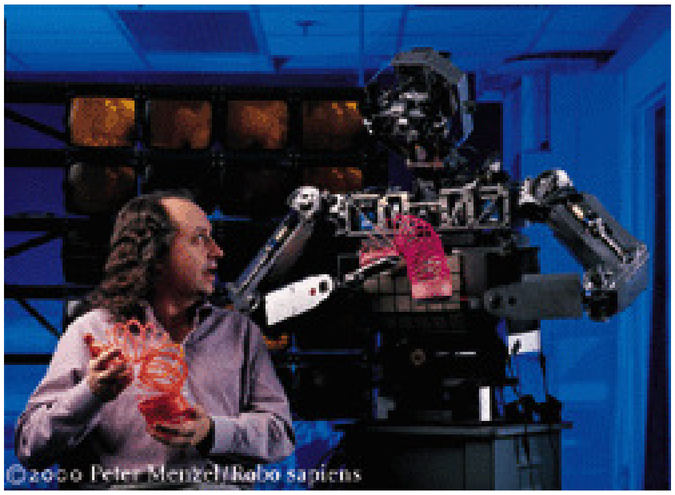


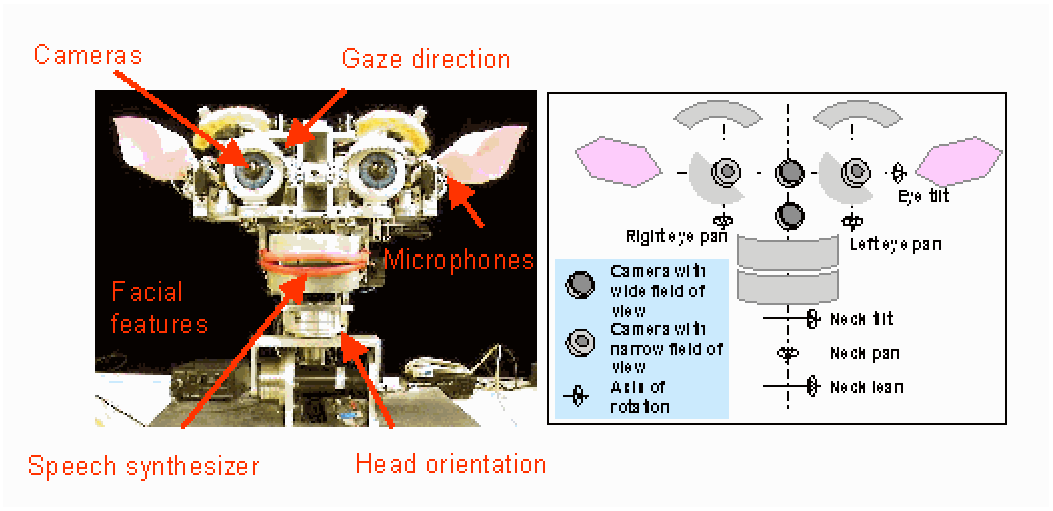
Figure 1.11. Emotion-Display EDDIE humanoid robot [2].

similar interface device that provide intuitive one-to-one mapping might not always be available for arbitrary robot systems. It is challenging to come up with an interface that is as intuitive as this marionette device where the one-to-one mapping of the interface device to the actual robot system is difficult to achieve.

With current advancement in sensors and computer hardware, more sophisticated technologies are becoming commonplace. A good example would be a high-speed high-resolution camera with fast image processing capabilities is becoming more accessible. This means that the sky is the limit for the possibility of the robot sensing capability. Classical and modern control feedback loop make use of physical variables such as position, angular position, velocity, angular velocity, acceleration, gyro rate, temperature, flow rate, elevation and so on. Recent technologies enable us to have ac-



(a)



(b)

Figure 1.12. (a) MIT Cog robot (b) MIT Kismet robot [3].

cess to the listed physical variables through smaller and smaller devices. The sensing capability is becoming more reliable, accurate and smaller in sizes. Thanks to sensing technologies, the robot systems will be able to make use of better sensing capability for lower cost. With precise, fast and reliable sensing capability plus the advancement in computer technology, we are able to run complex algorithm processing multiple sensing data at faster speed. We are now able to achieve faster real time response

Figure 1.13. Albert HUBO humanoid robot [4].

and having better offline processing power for tasks like clustering, learning, detection and so on.

1.1.3 Potential Areas of Application for Adaptive Human-Robot Interaction Control Systems

As long as there is an autonomous machine that human operates, human-robot interaction is always a relevant topic. Systems with multiple interface inputs can take advantage of the adaptive human-robot interaction control method. The application can range from a simple wheel-based mobile robot with joystick control to multiple degrees assembly station. Also, with adaptive human-robot interaction control system, a simple interface can be used as an input device for complex multiple degrees

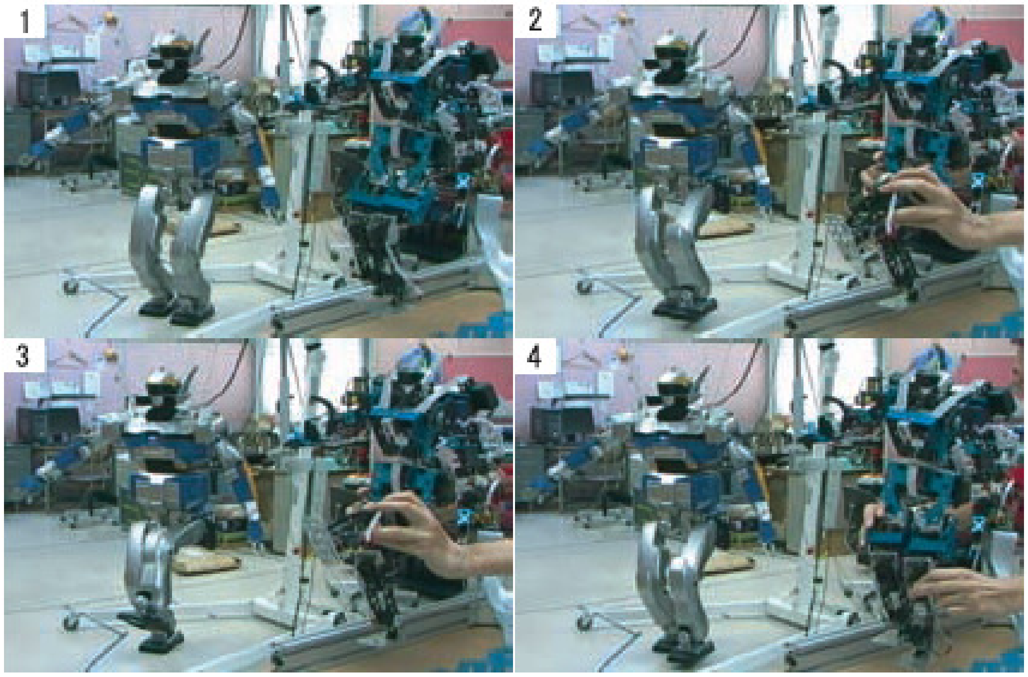


Figure 1.14. Marionette device and humanoid robot control [5].

of freedom robot systems. Simplification of interface device may have tremendous impact on robot control methodologies. We envision that areas that might benefit from adaptive human-robot interaction control systems are listed as follow, this is by no mean a complete list potential areas; unmanned vehicles operations, robot teleoperation, humanoid robots, industrial robots, robot swarms and etc.

1.1.4 Human-Robot Efficiency and Usability Improvement

Nowadays, devices such as high-tech gadgets are around us all the time. It is important that users of such devices have the capability to give commands to the devices in an efficient manner. The same principle applies to all the man-machine interaction systems that require users to operate the system to achieve desired actions / outcomes. More efficiency means that we can get more tasks done for given time.

1.1.5 Coordinating Multiple Sensing Inputs and Multiple Actuators for Well-Defined Tasks

Robots are usually made up of multiple actuators, multiple sensors and decision-making capability. It takes extensive implementation of processing, planning and control to make a robot do meaningful actions. From users point of view, they do not care about low-level mechanism such as motor control, path planning and so on. The users only care about final outcomes, actions or behaviors of the robot systems. In this proposal, we suggest that coordinating multiple sensing inputs and multiple actuators in order to accomplish well-defined tasks is an important matter.

1.2 Contributions of this work

In the past, the intuitiveness and realism of man-machine interaction have been mainly determined qualitatively on ad-hoc basis that varies for different systems. Both quantitative intuitiveness and realism of man-machine interaction have rarely been used as an online parameters adaptation rule, but rather as post evaluation of system performance that can hardly be translated into useful quantities / measurements. In this thesis, we develop an adaptive framework for human-robot interaction that make use of the online intuitiveness and realism quantity feedback from the user as a way to enhance the overall intuitiveness and realism of the interaction.

This dissertation makes the following contributions to the research in man-machine interaction in order to improve the existing schemes and offer novel solutions to the problem.

1. Physical interactivity and safety enhancement scheme for robot manipulators
   - A force estimation algorithm using Kalman filter has been formulated. The scheme is able to accurately estimate force in 3D space using only a sensor that provides 1 dimensional reading of normal physical interaction force.

The proposed force estimation scheme makes use of the known kinematics of the robot and online reading of robot joints position to recursively solve for an estimation of real time interaction force in 3D space. An impedance control scheme using the estimated force from Kalman filter has been formulated. A real time implementation of the impedance control for assistive robot / robot safety is used in conjunction with real time force estimation using Kalman filter. Conventionally, force in Cartesian space has to be measured in order to implement impedance control for an actuator system that is able to move freely in 3D space. With the proposed impedance control scheme, the requirement of sensing 3D force is eliminated [6].

2. Realistic human mimicking for social robots
   - A human head pose estimation / human head tracking algorithm using extended Kalman filter has been formulated. A nonintrusive way of extracting human head pose information from 2D visual sensing (a single camera) has been developed. The method is able to estimate human head pose in real time by processing a sequence of images obtained from a single camera. The method does require 4 non-colinear points to be able to accurately estimate the human head pose. The head pose information obtained from the method can be further utilized for real-time realistic interaction with humanoid robot head [7].
   - A realistic neck-eye motion distribution algorithm for humanoid robot head has been developed. Generally the mechanical / muscular structure of humanoid robot head is not similar to the structure of an actual human head. Under object tracking motion, the motion distribution between neck and eyes is governed by Listing's Law [8]. In order for the humanoid robot head to behave the same way as the human head, we have developed an

optimization approach leading to a realistic neck-eye motion distribution. The approach involves solving recursive optimization problem in real time. We have confirmed experimentally that the approach yields realistic neck-eye motion distribution for a humanoid robot head [9, 10].

- A visual feedback for human head tracking robustness enhancement has been formulated. The exact detailed kinematics of human head structure as well as humanoid robot head structure is often difficult to obtain. Most of the time an approximated kinematics models is used. Due to the discrepancy between actual model and the approximated model, the solution to the optimization problem of the neck-eye motion distribution has some offset error in it. A real time visual feedback algorithm for online error correction and tracking robustness enhancement has been proposed [9, 10]. Exponential convergence of the algorithm was proven.

3. Framework for intuitive interaction of humans with multi-DOF robots through multi-modal input devices

   - An intuitive, easy to learn interface mapping methodology to interact with a multiple degrees of freedom robot has been proposed. The dynamic model of the interface mapping is updated using reinforcement learning in conjunction with a reward function that ties to user input. The reward function can be set so that the interface mapping is updated according to certain performance metrics, which in turn adapt the mapping of the interface to yield a more intuitive and easier to use interface from the perspective of the current user [11].

1.3 Dissertation Organization

In chapter 3, a robot physical interaction scheme for force estimation using Kalman filter is introduced. In this chapter, we focus on physical interaction with robots. We use an articulated robot arm as a device we would like to physically interact with. The objective is to be able to manually guide the robot arm to a desired position and at the same time being able to specify the impedance of the robot arm. The impedance control is used so that we can have an assistive robot as well as a safety mechanism for relatively large robots. The challenge for this chapter is that we can only measure force in one direction (one degree of freedom). This makes it difficult to implement impedance control of the robot arm in 3D. Therefore, we proposed a 3D force estimation scheme using an Extended Kalman filter, the experimental work in this chapter we use a CRS465 robot arm and a Omni Haptics device. Based on the detailed dynamical model of the robot [12]., we developed a force estimation scheme and impedance control scheme. Simulation results of force estimation using Kalman filter of the CRS465 robot based on detailed dynamical model are presented in this chapter. Also, experiment of simultaneous implementation of force estimation and impedance control is also conducted using the detailed simulation model as well as the actual robot.

Chapter 4 presents a robot interaction scheme through a camera system. In this chapter, we focus on realism of interaction with a humanoid robot actor. The objective is to implement a motion control scheme on the Lilly and Zeno robot actors so that it mimics motions of human head for the application of interest - conversational robotics. The Lilly robot actor's main sensing modality is vision. We want the robot to be able to exhibit a realistic motion in a manner that is similar to humans, suppose that the robot actor is capable of carrying a conversation with humans. To achieve the objective, the robot needs the ability to indentify human head position and orientation so that the robot is able to follow the human head in a realistic manner. Also, for the neck-eye structure of the robot, they have to be coordinated so that they look realistic during the following of the human head. In this chapter, a pose estimation scheme using the extended Kalman filter is presented as well as an optimization approach for realistic neck-eye motion distribution. The pose estimation using the extended Kalman filter is able to estimate 3D pose of the object through a video stream of an object of interest in real time by tracking four unique points on the object. An optimization approach for realistic neck-eye motion distribution is implemented on the Lilly robot actor, tracking discrepancy is an issue using this scheme. We also address this issue by implementing visual feedback to enhance tracking robustness of the robot actor.

In chapter 5, interaction with multi degree of freedom robots is discussed. In this chapter, we focus on interacting with multiple robots or robots with multiple degrees of freedom through simple interfaces. We are interested in developing an interface mapping that is intuitive and easy to learn for the human operators. In this chapter, we discuss the definition of intuitive interface mapping for multiple agents / multiple DOFs robot control as well as performance metrics. We propose the use of reinforcement learning approach for adapting the interface mapping. The reinforcement-learning scheme has been implemented with a simulated mobile robot motion control system through the use of a 3D stylus as an interface device. Also, the scheme has been implemented on the actual mobile robot system for position control using the brain activity sensor. A reward function is used for learning of the interface mapping. We present results for different reward functions used.

Finally chapter 6 provides discussion on summary of the dissertation as well as listing the future work.

CHAPTER 2

BACKGROUND

2.1 Physical Interaction with Robots

According to the Robotics Industries Association (RIA), sales of industrial robots have risen by more than 20% annually through during the last two years. Even though a large portion (75%) of these robots is currently concentrated in operations such as welding, painting, and material handling, it is expected that this proportion will decrease significantly in the future [13]. During the last few years, among notable trends in industrial robotics has been the introduction of “smart assist devices” and “safe robots” [14]. These devices augment the dexterity and power output of a human operator by amplifying his or her motion through a force-sensitive pendant, or they can sense the presence of humans inside work cells and thus avoid injuries. Robotic skin is one type of heteroceptive sensor that could eventually enable “cobots” (e.g. cooperative robots) to share their workspace with humans [15]. In [16], presents a robot equipped with differential elastic actuators that are backdrivable and torque controlled, capable of being force-guided. A coordinated haptic training architecture useful for transferring expertise in teleoperation-based manipulation between two human users has been proposed in [17]. In [18], discuss a human-machine interface using Hill-based muscle model to control the isometric force of a robotic thumb. Surface electromyogram from the skin surface is measured and converted to muscle activation information.

For the last several years, researchers in Japan, the US, and Europe have been working on creating artificial robotic skin. Lumelsky, Shur, and Wagner [19] were the first who proposed the idea of using a large-area, flexible arrays of sensors with data processing capabilities, which is called "sensitive skin" and can be used to cover the entire surface of the machine. Sensors to be used with robotic skin include IR sensors by Vladimir Lumelsky, organic FET arrays at University of Tokyo [20], wireless RF sensors at Tokyo University of Agriculture [21], PVDF tactile sensors at Tohoku University [22], work at Univ. Nebraska [23], and force, temperature and electric field sensors at MIT Media Lab [24].

We have recently introduced a new robotic skin patch ("Quickskin") [25] based on piezoelectric transducers embedded into a soft elastomer base, and we propose its use to enhance human-robot interactivity. Physical interaction with through pushing or pulling on the robot arm can be used to guide through a desired motion. The characteristic or feeling of the interaction can be defined by the impedance of the robot arm. Since its introduction in the 1980s, [26, 27, 28, 29] impedance control has been used to control robot interaction in a variety of applications, including Intelligent Assist Devices [26, 30, 31, 32], or in medicine [33, 34].

The typical physical interaction between human and robot ("HRI") is achieved using a force-torque sensor mounted on the robot wrist. However, these sensors are expensive, not always available, and they do not provide direct measurement of the interaction between the robot chain or the payload with the environment. Several researches have worked on estimation of interaction force from other proprioceptive sensors. Kobayashi, Muis and Ohnishi [35] proposed a way to interact with a robot manipulator without using any sensor by means of Extended Kalman Filtering. And in [32] crane pushing force is estimated from cable angle measurements. Drawbacks of this approach include reduced bandwidth of the impedance controller because of force filtering and also dead-zone of force estimation due to friction or ill-posed numerical conditioning. Kazerooni et al. [36] have developed an instrumented glove for robotic and human-assisted material handling manipulators. The sensory glove measures the force the wearer imposes on any part of the material handling robot or the object being maneuvered by the material handling robot and appropriate force is generated by the actuators to assist the wearer in material handling.

[37] proposed an alternative method by combining impedance control with direct control scheme for control of robot manipulators. The combined impedance / direct control scheme presented in [37] is shown to be more robust and high in disturbance rejection. A computer simulation has performed to support the proposed method of combined impedance / direct control.

[38] emphasizes the study of haptic cooperation between two people and between people and machines. Based on the idea that in order to achieve a particular task i.e. moving and object, it would be more efficient to have two people (or one person and machines) performing the task. Examples of physical interaction include; when two people exchange an object, when two people jointly move a big object, when one individual teach the other manual skill and when two people dance. Although, it may perceived by a person that the other person helping causes hindrance but in fact it is more efficient. In order to efficiently achieve the task with two people working together, there must be some kind of communications between the two. One important aspect of the communications is haptic communication. Even though the two people might not aware of that kind of communication it is the most important communication that helps getting the task done. The goal of [38] is to study physical cooperation of two people working together on the same task and possibly replace one person with a machine that replicated a human interaction behavior. They set up an experiment for two people to perform a moving of an object to a certain points and recorded forces acting on the object that were produced by each participant. The forces applied to an object by both participants were individually measured and

the total force on the object was also measured. The resulted data were analyzed and they concluded that it is more efficient, faster to have two people perform on the experiment. The other experiment was they had one person come in to move the object with another person pretended to help him but in fact the first person was assisted by a moving mechanism. The results from those two experiments were similar. And they proposed to conduct more study in order to thorough understand the haptic interaction between two individual helping each other to accomplish the same task.

In chapter 3, we combine the estimation of the human-robot interaction force using EKF, with direct force measurement along one direction (perpendicular to the robot arm). We perform realistic simulations with a well identified CRS A465 robot model, and compare the HRI performance when robot skin is added to the system. Pushing forces can be applied anywhere on the robot arm as is consistent with the availability of robot skin sensors. Simulation results show that the addition of robot skin greatly improves the HRI force estimates, and thus enhances the responsiveness and safety of the robot. Measurement noise covariances in our simulation are based on force measurements from “Quickskin”, and experimental work with the skin patch and an actual CRS robot arm and Phantom Omni haptic device is also discussed in chapter 3.

2.2 Interaction with Robots through Vision

The role of vision guided control is important and in a wide range of applications such as humanoid robots, manufacturing environments, satellite and missile control and creating augmented reality motion. Recently, due to the increased availability of camera systems, robotic cameras are routinely used in surveillance [39], on movie sets [40], or in industrial robotics. One of the main issues related to the vision guided control system is the evaluation of the position and orientation vector of an object from the camera field of view, the so-called pose estimation problem. Visual servoing for following the object is typically achieved using an image based Jacobian, or a combination of position and image [41, 42] based methods. Certain applications, such as automated space-station docking, or the interaction between humans and humanoid robots require the estimation of an object pose in 6D, and visual tracking of the object based on that pose. Even more challenging, applications can sometimes provide a single "eye-in-hand" camera view, as opposed to stereo vision, leading to the so-called 2 1/2 D vision servoing problem [43]. The corresponding pose determination problem is to calculate the three-dimensional (3D+3D) position and orientation of an object from a set of feature points captured by two-dimensional (2D) images. In [44], stereo-based head tracking framework for robust head motion estimation under varying lighting conditions has been proposed.

Considerable research has already been conducted and various approaches have been proposed to determine the pose of an object using robot cameras. In [45], an approach based on active appearance model (AAM) is proposed, which is based on model matching. A neural network method to determine the pose of an object is proposed in [46], which requires offline training of the neural network and it fails if the object is cubical in shape. In [47], a geometric based approach is given, which uses 4 features points to obtain the pose parameters and an averaging method is used to deal with system noise. Another method based on hypothesis-testing logic is proposed in [48], which also uses more than 6 feature points.

Since visual measurements are highly effected by the system noise due to lens distortion, lighting and background inconsistency, inconsistency in image processing algorithms, etc, the pose estimation algorithm may produce large errors. This issue of system noise could be significantly avoided by the use of a Kalman filter, which improves the accuracy of the estimation process. Various approaches using Kalman filter for pose determination have been effectively adopted in [49, 50, 51, 52, 53] and [54]. In [49, 50], an approach based on Kalman filter is proposed, which uses 5 non-coplanar feature points with a single camera. Use of multiple cameras with Kalman filter has been proposed in [51, 52, 53] and [54]. These methods use more than 4 non-coplanar feature points for accurately determining the pose parameters.

Since the 2D camera measurements are related through a nonlinear projection relationship with the pose parameters, an Extended Kalman filter (EKF) is needed. It is well-known that such filters are sensitive to state vector and covariance matrix initial conditions. Inappropriately chosen parameters could produce large errors or even to the divergence of the estimates. For more on the divergence of the extended Kalman Filter see [55].

[56] discusses a human head detection algorithm that is able to detect a human head in a distance more than 2.5 meters. The algorithm was implemented on a robot with limitations of moving of a camera, orientation of a camera and unfixed illumination. The assumption is that a human head is an omega (Ω) shape contour. The proposed algorithm separates background from the omega contour shape with a combination of three features; Gray Module, Edge Module and Color Module. The proposed algorithm has been tested on several movie files. The results showed robustness of head scale, head orientation and moving camera. The detection has shown significant error when the head to be detected is not facing a camera (from the back).

[57] proposed approaches of developing a human-like robot photographer. In order for the robot to be able to behave like human photographer the robot requires following capabilities; mobility, wireless communication, ability to recognize human command and ability to follow pre-defined photographic composition rules. [57] uses

a mobile robot called ETRO which is a wheel-based mobile robot with vision camera. ETRO has 10 ultrasonic sound sensors, 8 infrared sensors, and wireless connectivity. The robot is programmed to follow photographic composition rules that are written in a book photographic composition by T. Grill and M. Scalon which is considered a very popular and widely used book.The authors summarized four major rules (according to the book) that were implemented in the experiment. The robot also has the capability to recognize a human caller. The robot recognizes a waving hand. Since the authors suggested that it is difficult to do voice recognition in an open space and face recognition is computationally expensive so they proposed a simpler way of communication between caller and the robot by recognize hand waving. In the experiment false detections occurred due to surrounding peoples movement. They tried to improve the caller finding by adding a face detection routine after the waving hand detected so that the robot can respond to the correct caller. The conclusion of the experiment was the robot took photographs well according to four simple position rules but lacking the artistic aspect of a human photographer. And the robot was unable to take a scenic picture since it cannot recognize objects. Due to the authors several aspects concerning intelligence of this robot photographer have to be improved in the future work so that the robot can behave more like a human photographer.

Visual servoing is an important aspect of our work. [58, 59], proposed a method of combining Position-based and Image-base visual servoing together. By using kinematics measurement, it provides robustness to visual distractions and the occurrence of the object going out of the field of view. Visual measurement, provide useful information to achieve pose estimation and to obtain online estimation of the jacobian. [58] improves robustness and performance of the visual servoing by using the proposed scheme. [60] proposed a method of using non-linear state feedback model to perform position-based visual servoing. [60] suggested that the main advantage of the proposed method is that camera translation and camera rotation are separately controlled due to the use of frames selection. [61] proposed a method of tracking a moving target by uncalibrated model independent visual servo control method namely dynamic quasi-Newton approach. The control problem is formulated as a nonlinear least squares optimization. It uses a time-varying objective function which is minimized using the Newtons method. [61] proved that the convegence of the servoing system using this method is guaranteed. [62] proposed a method of using laser pointers attached to the end-effector of the eye-in-hand Image-based visual servoing system. The feature extraction become just a problem of identifying the laser points on the object which reduce the computation load of the feature extraction part. [62] showed that by using the proposed method it produces a control scheme with nice properties such as decoupling, stability, and good camera trajectory.

A motivating application is visual tracking of a human using a humanoid robot. In this dissertation, tracking is performed through position based visual servoing, where the features to be tracked are projected through the camera model from 6D pose estimates. A PID controller with tuned gains will be used to implement the proposed servoing scheme. The overall estimation-servoing scheme is experimentally validated using a simple pan and tilt camera mechanism, LILLY and Zeno humanoid robot actors.

The advancement and cost-effectiveness of sensor/ actuator and computing technology has played an important role on the emergence of humanoid robotics. More recently, research has focused on realistic interaction between humans and humanoid robots in manners as intuitive and natural as human-to-human conversation. Hence, in “social robotics”, it now possible to construct humanoid hardware with human-like appearance: e.g. mouth and eyes. The human-like humanoid hardware has to be controlled so that the overall behaviors are similar to those of humans.

Mori [63, 64], was among the first to suggest that people are likely to become familiar with robots that exhibit human-like features, appearance and characteristics. However, if those characteristics are not apparent, an “uncanny valley” is created which leads to feeling of “repulsion” by humans. Bartneck [65] proposed the alternative model to the Moris “uncanny valley” model and Shimada [66] suggested that the “uncanny valley” model varies from individual to individual. The “uncanny valley” and other human-likeness theories continue to be attractive research topics. Studies done by Woods [67] and Goetz [68] suggested that human-like robots are more likely to be treated the same way people are treated in a social setting, if they do not have a mechanical-like appearance. Extensive studies of human imitation (realistic human-likeness) and sociable robot (human-robot interaction) have been conducted using MIT robots COG and Kismet. Breazeal [3] discussed the potential challenges regarding building robots that imitate humans and Scassellati [69] studied human social development models, the way social interaction skills are developed in humans, relating to human-like robot interaction. It is clear that creating both a realistic appearance, as well as realistic behaviors for humanoid robots will play an important role in the future of such robots. In [70], the method of using virtual mechanism approach to control humanoid robot head for object tracking has been proposed.

The work in [71] is based on the fact that the humanoid robot is currently widely used, studied and also the humanoid robot shares workspace with people. Therefore, it is important to study how people would react with the humanoid robot presented in the people’s workspace. In [71], the robot was placed in the office and repeatedly performing the interactions and record how human would respond to various type of interactions. The conclusion of [71] focuses on the social relationships of a robot among people. A variety of aspects of the social relationships have been issued from the experiments, and the authors have proposed ways to improve the appearance and intelligence of the robot, so that the robot would fit into daily life.

In [72], authors developed an artificial facial expression imitation system that can recognize and imitate humans facial expressions. They proposed a classification that was suitable to recognize human facial expression in real time. They also built a robot that can generate facial expressions in order to validate proposed methods of recognizing human facial expressions. The methods used in the facial expression recognition include; model-based method s which require high resolution images and are time-consuming to fit the data, therefore this approach is not good for real-time facial expression recognition and Principle Component Analysis (PCA), which distinguishes facial expression using an image intensity profile. [72] proposed a facial expression imitation system that consists of two parts; facial expression recognition and facial express generation through a robot. The facial expression part is able to classify basic facial expressions such as: neutral, happiness, sadness, anger, surprise, disgust and fear. The recognition method is called rectangle feature, and was implemented along with a learning algorithm called AdaBoost Learning Algorithm by using 1065 facial expression images from database. The facial expression generation part is based on facial action coding system (FACS). The experiment they conducted includes capturing a picture of a human, detecting/classifying facial expressions, and then playback of the captured facial expression on the robot.

Another motivating application for our work is to create a conversational robotic actor with realistic interactivity, in particular person recognition, identification, tracking, eye contact, and conversational interaction. Breazeal [73] proposed a vision system for social robots that allows them to interact with humans in a meaningful way, e.g., the intention of the human can be perceived and elicit a natural response from the robot. Gurbuz [74] proposed a novel approach to do human mouth mimicking on a humanoid robot head in real time. Facial features are tracked in real time through stereo vision and a learning algorithm is implemented to detect human mouth which in turn is used as a trajectory for the humanoid robot to follow.

The main idea of [75] is to generate a motion of humanoid robots that close to natural motions of humans. There is no solid boundary as what kind of motion qualify as natural as it is difficult to define “Natural Motion”. In [75], the term “Natural Motion” is lightly refers to the motion that is resemble human motion. It can be categorize into 3 types of works that are relating to generating a natural motion: motion capture method, optimization based method and control based method.

The motion capture method is widely used in computer animation, movie for which real actors enact a scene, and the motion of actors are captured through various methods of data acquisition, and then played back on animated characters, so that the computer animated characters will appear to behave naturally as if they were real humans. There are many researchers that are currently working on this topic.

Optimization-based methods are based on minimizing force/torque required to move the robot. It has been studied that by minimizing force/torque of the robot actuators it will result in having natural look. Motions of humans are minimized in terms of energy, a process we have developed through learning since we were children. So we might not aware that each motion we perform are optimal in terms of energy consumption. The method is based on the assumptions that by minimizing force/torque require to move robot joints the robot will appear to move naturally the same way human does. [75] focuses on this approach. The authors have done two simulations to show the strength of the proposed method. First, a the 1 DOF manipulator has been simulated to show the minimized force/torque motion. Second, the authors have worked on a simulation of a humanoid robot to generate a reaching motion. The result has shown that the proposed method is effective in order to generate a human-like motion.

In this dissertation, we present the basics of an optimization algorithm as well as experimental results showing a kinesiological sound movement of the humanoid robot head and eyes. Our work extends past results we employed for creating realistic interactions between a human and a virtual actor [76]. Unlike other published work in this area [77, 78, 79, 80], a straightforward, approximate kinematic model of humanoid robot head was used by our algorithm, rather than an accurate one. Moreover, the objective function in our algorithm is formed based on kinesiological properties of real human eyes (Listings Law) [81], so that the resulting motion of the humanoid robot actor is similar to humans head-eye motion. We also take into consideration that the actual humanoid hardware might not match the approximate kinematic model. The visual feedback scheme is used to minimize the error that may arise from model discrepancy and/or disturbances.

In chapter 4 we also propose and experimentally validate an efficient, EKF-based 6D pose estimation algorithm and combine it with object tracking with a robotic camera. Furthermore, pose estimation using the proposed approach is accomplished with only 3 non-collinear feature points. The combined algorithm was found to be robust to large initial condition errors and noise through the experiments we conducted. In addition, we are interested in generating realistic motion of the humanoid robot head and eyes using a combination of objective function optimization method and visual servoing method to create head-eye motion profile based on human kinesiology and keep track of the human target in the presence of head-eye modeling error and hardware imperfections. The visual feedback method does not require knowledge of an exact kinematic model and joint axis coordinates for either robot or human. We study the interaction with conversational robot through vision

[7], based on extraction of human-head 3D poses from video streaming at 30Hz frame rates. The experiments were conducted using Lilly, a human-like facially animated robot actor, currently being developed in our lab. We make use of the location information of a person as detected by the robot, and we augment it with a head-eye motion coordination scheme that enables the robot to achieve a realistic gaze during tracking.

2.3 Interaction with Multiple Degrees of Freedom Robots

It is a challenging task to control a system of multiple robots and/or a robot with multiple degrees of freedoms by a single operator in an efficient manner. The most intuitive way for a human operator to manipulate a robot is to use hands to position a robot or a small replica of a robot manually (master / slave operation). However, such an interface is not always available.

The objectives of building an intuitive interface mapping for multiple agents/ multiple DOFs robot control can be summarized as follow:

- Build a simple reconfigurable interface system that allows a single operator to manipulate a robot with multiple degrees of freedom and/or multiple robots.
- The proposed interface system should be intuitive, easy to use and can be learned quickly by an operator.
- The interface system should be able to use with different robot configurations.

The supervisory control of multiple agents/ multiple DOFs robots is a very demanding application. There are many tasks that can be achieved using multiple robots, however, the main benefit of using multiple multiple agents/ multiple DOFs robots is to get task done more efficiently in scenarios where it is considered to be difficult for humans. It is challenging to evaluate performance of such scenarios due to several restricting factors. Nevertheless, performance evaluation of supervisory multiple multiple agents/ multiple DOFs robots control is vital, since it is almost impossible to improve the control scheme without it. Hence, as a foundation for development of better control scheme for multiple multiple agents/ multiple DOFs robots, we must first be able to evaluate the control performance of the existing control schemes. In this section, we are discussing the conventional human factors evaluation that have been in use in the past, and then propose a set of performance metrics that are suitable to multiple robots control based on common principles.

Research on the psychology of interaction between a human and a robot have been widely studied. Most of the research in this area are focus on the study of force, motion, planning but in [82], it concentrated on the study of physiological effects of a robot on a human.Emotional stress, reduces productivity, can be caused by working with a robot for a long time. Sensor measurements taken during experiments include: heart rate (electrocardiograph), respiration, perspiration, pulse wave and blood pressure. [82] used the mentioned data to analyze the emotional state of the human participated in the experiment. They also used a 12 DOFs serial mechanism to capture a human motion so that they could incorporate motion with the data from the sensor measurements. The experiment was performed by a human wearing the sensors in the working environment of a robot, and they interpreted all the data from measurements to indicate the participants stress level. They recorded certain actions that significantly increase or decrease the stress level of the human. Through the experiment, they managed to decrease the subject's stress by performing certain robot's motions.

In [83], proposed a reinforcement learning method for robots to learn to solve a task based on brain activity recorded by an EEG-based BCI system as reward signals. In [84], reinforcement learning approach for parameterized control policies based on the framework of stochastic optimal control with path integrals has been proposed.

In [85], reinforcement learning with Decision Trees (RL-DT) for learning of agent and environment model by generalizing the relative effect of actions across states is used for humanoid robot control.

In order to be able to quantify whether or not an arbitrary human-robot task is completed satisfactorily some performance metrics have to be used. [86] lists general human-robot performance metrics, while in [87], a set of definitions that form a framework for describing the types of awareness that humans have of robot activities, and the knowledge that robots have of the commands given them by humans has been introduced. [88] discusses supervisory control of multiple robots performance metrics, and in [89], a method for identifying sets of metric classes by decomposing a human-robot team consisting of a single human and multiple robots has been proposed. Human supervisory control metric classes and subclasses and performance metric evaluation criteria have also been proposed in [90]. [91] proposed generalizable metric classes for human-multiple robots supervisory tasks. In [92], an objective function for different levels of human-robot collaboration in a target recognition task has been developed to assess human-robot collaboration performance in many conditions. In [93], the method of improving robot operator performance by using augmented reality has been proposed. In [94], the authors have compiled a set of seven principles for efficient interaction for the designing efficient interfaces. In [95], common metrics for task-oriented human-robot interaction have been identified by using a fuzzy temporal model to evaluate the human trust in automation while interacting with robots and machines to complete some tasks.

Most of the previous work is qualitative, and based on repeated experimentation with human subject. The interfaces evaluated are not adaptable to the results or to individual users. In chapter 5, we propose learning components to be incorporated in interfaces between a human and multi DOF robots.

CHAPTER 3

PHYSICAL INTERACTION WITH ROBOTS

Industrial Robots nowadays are considered hazardous machinery. Since operation of robots can consist of moving objects in space with considerable high speed, human operators sharing the same workspace with the robots might get seriously injured because robots are not “aware” of the presence of human operators. It is desirable that human operators should be able to share workspace and work safely with robots. The idea is to make robots “smarter” so that robots can detect a presence of human operators and avoid physical contact with them.

Physical interaction force sensing is an important aspect for good force control of the robot manipulator. Suppose that we want to be able to physically interact with the robot manipulator any point along the manipulator chain, we would need to place force sensors all over the robot, which is difficult. Sensing interaction forces in 3D is important. However, in this chapter we develop an extended Kalman filter 3D force estimation scheme, this allows us to obtain estimated force in the unmeasured directions (we assume that the sensor can only measure normal force). With the need to have 3D force sensing eliminated, the ultimate goal is to have some kind of artificial skin force sensor cover the entire robot manipulator so that we can physically interact with the robot manipulator any point on its chain. In this chapter, we will show the effectiveness of the extended Kalman filter force estimation scheme by putting a force sensor at a certain location on the robot manipulator. We will also make use of the estimated force information to implement impedance control.

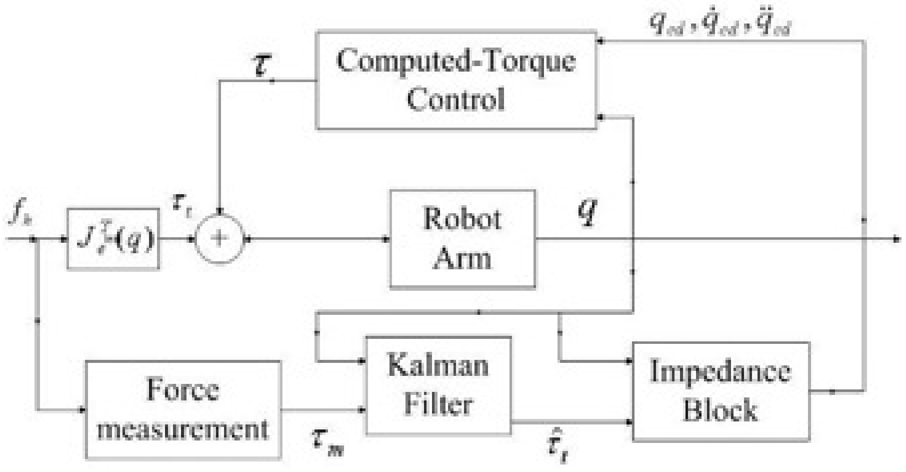


Figure 3.1. Overall control system diagram which includes Kalman filtering estimation, Impedance controller, and computed-torque control.

3.1 Force Estimation Using Kalman Filter

3.1.1 Modeling of the Human-Robot Interaction Scenario

In this chapter we will consider arbitrary robot manipulator for the EKF force estimation scheme. The kinematics and dynamical model of the manipulator is assumed to be well known. We assume that the robot can be equipped with a force-torque sensor, but here we pretend that none is available, and that robot skin is mounted at the end-effector of the robot. As a result, we will be primarily interested in the kinematics of the robot as we need to calculate torque acting on the joints according to the force acting on the end-effector. The overall EKF force estimation and impedance control scheme is shown in Fig. 3.1.

3.1.1.1 Kinematics

The point of interaction P can be anywhere in the chain, and we refer to this point as the “virtual end-effector”. Using the product of exponentials formula, the forward kinematics map from the robot base to a point P of application of the force is $g_{st} : Q \times \Re^3 \to SE(3)$, is given by

$$g_{st}(q,p) = (\prod_{i=1}^{j} e^{\xi_i q_i}) g_{st}(0,p) \tag{3.1}$$

where j is the robot link where the force is applied, q_i's are joint coordinates, and ξ_i's are the link twists given by:

$$\xi_i = \begin{bmatrix} -\omega_i \times \nu_i \\ \omega_i \end{bmatrix} \tag{3.2}$$

where $\omega_i \in \Re^3$ is a unit vector in the direction of the twist axis and $\nu_i \in \Re^3$ is a point on the axis. In general, the base to virtual end-effector transformation matrix is defined as follows

$$g_{st}(q,p) = \begin{bmatrix} {}_{j+1}^{0}R(q) & p(q) \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

At the same time, the relationship between Jacobian expressed in base frame and Jacobian expressed in virtual end-effector frame is as follows:

$$^{0}J(p,q) = \frac{\partial\, p(q_{1,\ldots,j})}{\partial\, q_{1,\ldots,j}} =_{j+1}^{0} R(q) J_e(p,q) \tag{3.4}$$

where j^{th} frame is the virtual end-effector frame. Therefore the Jacobian expressed in the virtual end-effector frame can be obtained from:

$$\begin{aligned} J_e(p,q) &= \ {}^{j+1}_{\ \ 0}R(q)\ \ {}^{0}J(q) \\ &= \left({}_{j+1}^{0}R(q)\right)^T\ {}^{0}J(q) \end{aligned} \tag{3.5}$$

where $j_e(q)$ is the Jacobian expressed in the virtual end-effector frame. Because forces acting on the manipulator are measured in the end-effector frame, we use $j_e(p,q)$ in the manipulator dynamics.

3.1.1.2 Dynamics

The dynamical model of an arbitrary manipulator is written conventionally as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + f_c(\dot{q}) + G(q) = \tau + J_e^T(p,q)f_h \tag{3.6}$$

where $q \in \Re^n$ is the vector of generalized joint coordinates, n is the number of joints, $M \in \Re^{n\times n}$ is the symmetric positive definite mass (inertia) matrix, $C(q,\dot{q})\dot{q} \in \Re^n$ is the vector of Coriolis and centripetal forces, $G(q) \in \Re^n$ is the vector of gravitational torques, $D \in \Re^{n\times n}$ is the positive semi definite diagonal matrix for joint viscous friction coefficient, $f_c(\dot{q}) \in \Re^n$ is the Coulomb friction term, is the vector of generalized torques acting at the joints, $J_e^T(p,q) \in \Re^{3\times n}$ is the conventional Jacobian to the virtual end-effector expressed in the end-effector frame and $f_h \in \Re^n$ is the human-robot interaction force at the virtual end-effector represented in its frame.

3.1.2 Force Estimation using Extended Kalman Filter

It is assumed that force measurement at the end-effector or the virtual end-effector (where an artificial skin patch is mounted) is available only in one dimension or less (no measurement at all). Rather than using an inverse dynamical model of the robot manipulator as in [35], which is prone to numerical instability and needs additional filtering, we use an Extended Kalman Filter as a way to estimate torques acting at the joints due to pushing force at the virtual end-effector. We write dynamical model of the manipulator from the base to the end-effector as follows:

$$M(q)\ddot{q} + N(q,\dot{q}) = \tau_m + \tau_u \tag{3.7}$$

where $q \in \Re^j$, $N(q,\dot{q}) = C(q,\dot{q})\dot{q} + D\dot{q} + f_c(\dot{q}) + G(q)$ and $\tau_m = \tau + J_e^T(q)f_m \in \Re^3$ is the known torque acting at the joints (either through direct measurement or from control input), $\tau_u = J_e^T(q)f_u \in \Re^3$ is the unknown torque to be estimated (due to pushing force in directions that cannot be measured). Since we use a force sensor skin element to measure the force perpendicular to the manipulator, we can assume that $f_m = \begin{bmatrix} f_{mx} & 0 & 0 \end{bmatrix}^T$, i.e, only force element in x direction can be measured.

The state-space model of (3.7) is:

$$\dot{z} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}[\tau_m + \tau_u - N(q,\dot{q})] \end{bmatrix} \tag{3.8}$$

To include the unknown torque vector into the estimation, we define the new extended state x including the unknown torque vector τ_u:

$$x = \begin{bmatrix} q^T & \dot{q}^T & \tau_u^T \end{bmatrix}^T \in \Re^{2j+3} \tag{3.9}$$

From (3.7), τ_u can be written as follows:

$$\tau_u = M(q)\ddot{q} + N(q,\dot{q}) - \tau_m \tag{3.10}$$

Since τ_u is a time-varying function, the first order time derivative $\dot{\tau}_u$ is given by:

$$\dot{\tau}_u = \frac{d}{dt}[M(q)\ddot{q} + N(q,\dot{q}) - \tau_m] \tag{3.11}$$

or

$$\dot{\tau}_u = \dot{M}\ddot{q} + M(q)\dddot{q} + \dot{N} - \dot{\tau}_m \tag{3.12}$$

The new augmented state-space model is written as:

$$\begin{aligned} \dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \\ \dot{\tau}_u \end{bmatrix} &= \begin{bmatrix} \dot{q} \\ M(q)^{-1}[\tau_m + \tau_u - N(q,\dot{q})] \\ \dot{M}\ddot{q} + M(q)\dddot{q} + \dot{N} - \dot{\tau}_m \end{bmatrix} \\ &= f(x,\tau_m) \end{aligned} \tag{3.13}$$

where $f(x,\tau_m)$ is a nonlinear representation of the new augmented state-space model,

$$\dot{M} = \frac{d}{dt}\big(M(q)\big) \tag{3.14}$$

$$\dot{N} = \left(\frac{d}{dt}C(q,\dot{q})\right)\dot{q} + C(q,\dot{q})\ddot{q} + D\ddot{q} + \frac{d}{dt}[f_c(\dot{q})] + \frac{d}{dt}[G(q)] \tag{3.15}$$

$$\dot{\tau}_m = \frac{d}{dt}\big(M(q)\big) \tag{3.16}$$

In order to convert the continuous augmented state-space model into a discrete model (3.13) needs to be discretized. The first order discretization of (3.13) with sampling time T leads to:

$$x_{k+1} = f_T(x_k, \tau_{mk}) + \begin{bmatrix} \nu_k \\ \eta_k \end{bmatrix} \tag{3.17}$$

where $f_T(x_k,\tau_{mk}) = x_k + \dot{x}(x_k,\tau_{mk})T$, $\nu_k \in \Re^{2j}$ is process white noise, and $\eta_k \in \Re^j$ is also process white noise associated with the unknown torque. Note that we used the process noise η_k to introduce the measurement noise from the force skin sensor element. The overall state covariance matrix is now:

$$Q = E\left(\begin{bmatrix} \nu_k \\ \eta_k \end{bmatrix}\begin{bmatrix} \nu_k \\ \eta_k \end{bmatrix}^T\right) \tag{3.18}$$

The observation equation from the robot joints encoders is given by:

$$y_k = Cx_k + w_k \tag{3.19}$$

where $C = \begin{bmatrix} I_{2j} & 0_{j\times j} \end{bmatrix}$, $w_k \in \Re^{2j}$ is the measurement white noise with covariance:

$$R = E\left(w_k\ w_k^T\right) \tag{3.20}$$

Define

$$\begin{aligned} F_x(x,\tau_m) &= \frac{\partial f_T\left(x_k,\tau_{mk}\right)}{\partial x} \in \Re^{(2j+3)\times(2j+3)} \\ &= \begin{bmatrix} \frac{\partial f_T(x_k,\tau_{mk})}{\partial q} & \frac{\partial f_T(x_k,\tau_{mk})}{\partial \dot{q}} & \frac{\partial f_T(x_k,\tau_{mk})}{\partial \tau_u} \end{bmatrix} \end{aligned} \tag{3.21}$$

to be the Jacobian matrix of $f_T\left(x_k, \tau_{mk}\right)$ with respect to $\left[q^T \dot{q}^T \tau_u^T\right]^T$. Instead of a signum function in the f_c Coulomb friction term, we use a smooth hyperbolic tangent function tanh(), so that the partial derivative (3.21) is well defined:

$$f_c\left(\dot{q}\right) = \mu \tanh\left(\alpha \dot{q}\right) \tag{3.22}$$

where μ is a friction coefficient, α is a design constant chosen to be larger than the desired HRI force bandwidth. An Extended Kalman Filter is now used to estimate the state of the linearized system in two steps:

1) Time Update:

$$\hat{x}_{k+1}^{-} = f\left(\hat{x}_k, \tau_{mk}\right) \tag{3.23}$$

$$P_{k+1}^{-} = F_x P_k F_x^T + Q \tag{3.24}$$

2) Measurement Update:

$$K_{k+1} = P_{k+1}^{-} C^T \left[C P_{k+1}^{-} C^T + R\right]^{-1} \tag{3.25}$$

$$P_{k+1} = P_{k+1}^{-} - K_{k+1} C P_{k+1}^{-} \tag{3.26}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K_{k+1}\left(y_{k+1} - C\hat{x}_{k+1}^{-}\right) \tag{3.27}$$

where $P_{k+1}^{-} \in \Re^{(2j+3)\times(2j+3)}$ is the covariance matrix of the prediction error, $P_{k+1} \in \Re^{(2j+3)\times(2j+3)}$ is the covariance matrix of the estimation error.

3.2 Impedance Control for Robot Safety

Assuming that the HRI pushing force can be estimated by the Kalman Filter, it is desirable to make use of it in order to guide the motion of the manipulator. We use an impedance control scheme to program the robot compliance. The goal is to specify the desired compliance characteristic of the robot manipulator so that when a human-operator applies force to the robot it feels as if the robot has certain mass and viscous coefficient. Define the desired impedance of the robot manipulator from its base to link j as:

$$M_d\ddot{q}_{cd} + B_d\dot{q}_{cd} + K_d q_{cd} = \tau_t \tag{3.28}$$

where $\tau_t \in \Re^j$ is the total torque due to pushing force. τ_t can be written as follows:

$$\tau_t = J_e^T(q)\, f_m + J_e^T(q)\, f_u \tag{3.29}$$

where J_e^T is defined in (3.5), f_m and f_u are defined in (3.7), $M_d \in \Re^{j\times j}$ is desired Mass (Inertia) matrix, $B_d \in \Re^{j\times j}$ is desired Damping matrix, $K_d \in \Re^{j\times j}$ is desired stiffness matrix, q_{cd} is the desired angular position of the manipulator.

The relationship between desired angular, position, velocity and acceleration and M_d, B_d and K_d can be written in matrix form as follows:

$$\begin{bmatrix} \dot{q}_{cd} \\ \ddot{q}_{cd} \end{bmatrix} = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ -M_d^{-1}K_d & -M_d^{-1}B_d \end{bmatrix} \begin{bmatrix} q_{cd} \\ \dot{q}_{cd} \end{bmatrix} + \begin{bmatrix} 0_{n\times 1} \\ M_d^{-1}\tau_t \end{bmatrix} \tag{3.30}$$

(3.30) can be solved numerically to find q_{cd}, $\dot{q}_{cd}$ and $\ddot{q}_{cd}$.

We can now use a straightforward Computed-Torque Controller to track the desired manipulator trajectory q_{cd} with an overall scheme shown in Fig. 3.1. From(3.7):

$$M(q)\ddot{q} + N(q,\dot{q}) = \tau + J_e^T(q)\, f_m + J_e^T(q)\, \hat{f}_u \tag{3.31}$$

where $\hat{f}_u$ is obtained from estimation of the unknown torque. The control signal becomes:

$$\tau = M\left(\ddot{q}_{cd} - u\right) + N - J_e^T\left(f_m + \hat{f}_u\right) \tag{3.32}$$

where

$$u = -K_v\left(\dot{q}_{cd} - \dot{q}\right) - K_p\left(q_{cd} - q\right) \tag{3.33}$$

and K_p and K_v are PD controller gains.

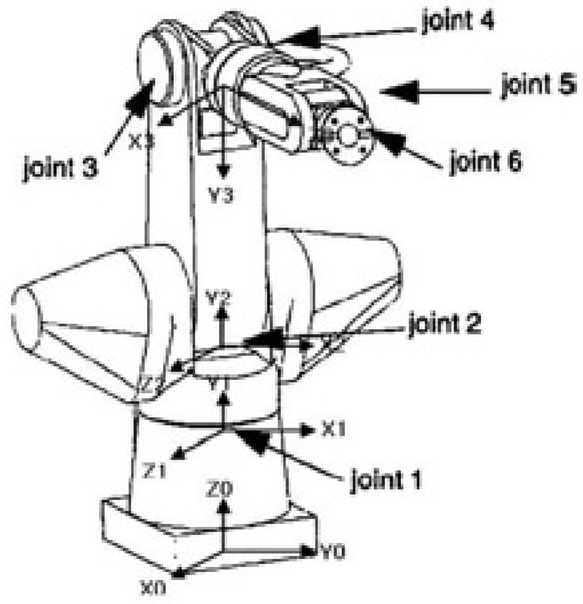


Figure 3.2. The links coordinate of the CRS A465 robot.

3.3 Simulation and Experiment Results

In our simulation, we use actual values of masses, lengths, friction coefficients, etc. from a well-identified CRS A465 robot [12]. The CRS A465 is depicted in Fig. 3.2. The A465 has six degrees of freedom, and can be equipped with a force-torque sensor, but here we assume that none is available, and that robot skin is mounted on the first three links. As a result, we will be primarily interested in the kinematics of the first three DOFs in the chain.

Fig. 3.2 shows the link coordinate system of the CRS A465 robot, its base coordinate system (X_0, Y_0, Z_0), and coordinate systems for the first three links (X_i, Y_i, Z_i), $i = 1 \ldots 6$. For the A465, the first three twists are given by:

$$\begin{aligned} \omega_1 &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \omega_2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \omega_3 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \\ \nu_1 &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \nu_2 = \begin{bmatrix} 0 & 0 & d_1 \end{bmatrix}^T \nu_3 = \begin{bmatrix} 0 & 0 & d_1 + d_2 \end{bmatrix}^T \end{aligned} \tag{3.34}$$

where d_i is the link length of the i^{th} joint.

The transformation between base and virtual end-effector frames at $q = 0$ is given by:

$$g_{st}(0,p) = \begin{bmatrix} I & \begin{pmatrix} 0 \\ d_3 \\ d_1 + d_2 \end{pmatrix} \\ 0 & 1 \end{bmatrix} \tag{3.35}$$

In this section we validate the estimator-controller for HRI through the use of a Simulink® model based on a well-identified CRS robot from [12]. Actual noise covariances based on skin sensor measurements and encoder resolution is used to create a realistic simulation model. A nominal model of the CRS A465 robot (3.7) is built together with the Kalman Filter equations (3.23)-(3.27), while (3.21) is obtained in symbolic form using Matlab®'s symbolic toolbox. We run simulations to verify that adding a skin patch greatly improves the performance of the HRI. Simulation results are presented in five different cases and two different cases for actual implementation on Phantom Omni haptics device (as shown in Fig. 3.5):

Experiment 1) Kalman filter parameters (3.13) are same as the robot dynamics (3.6) (known model), and we assume no force measurement.

Experiment 2) Same as experiment 1) but pushing force is lower than the robot arm static friction.

Experiment 3) The robot dynamics (3.6) and Kalman filter parameters (3.13) are slightly different (model is uncertain). Here we compare results between no measurement and 1 dimension force measurement at the pushing point p.

Experiment 4) Estimated pushing force from having 1 dimension force measurement (experiment 3) is used to control the motion of the robot arm through (3.32) and (3.33).

Experiment 5) Assuming that HRI interaction in all 3 dimensions is available; we want to compare the performance with experiment 4.

Experiment 6) EKF force estimation and impedance force control are implemented simultaneously on Phantom Omni haptics device without force measurement.

Experiment 7) EKF force estimation and impedance force control are implemented simultaneously on Phantom Omni haptics device with 1D force measurement through the Flexiforce sensor.

3.3.1 Experiment Testbed

In this subsection we describe an experimental setups that are used to validate HRI interaction algorithm depicted in Fig. 3.1. We apply our studies on two different experiment set ups. Firstly, Fig. 3.3 shows a planned experiment using a CRS A465 robot arm and an artificial skin patch used as a one-dimensional force sensor. It is assumed that the human operator can apply force at the point where the artificial skin patch is mounted. Secondly, we use an Phantom Omni haptics device, shown in Fig. 3.5, to test our proposed algorithm. In this set up we use a different kind of force sensor called Flexiforce, shown in Fig. 3.6. Detailed kinematics and dynamics of the Phantom Omni device are given in the Appendix A.

An actual signal reading from the piezo-based artificial skin patch ("Quickskin" from [25]) is shown in Fig. 3.4, from which the covariance corresponding to the signal was calculated to be 0.0591, and this value is used in the numerical simulation in the next section. The relationship between the voltage and force of the artificial skin patch is almost linear by a factor of 3.

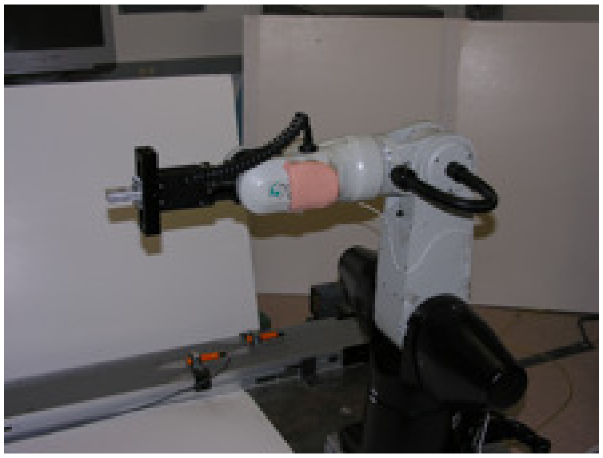


Figure 3.3. Photo of an artificial piezo-electric skin patch mounted on the CRS A465 and used as a one dimensional force sensor.

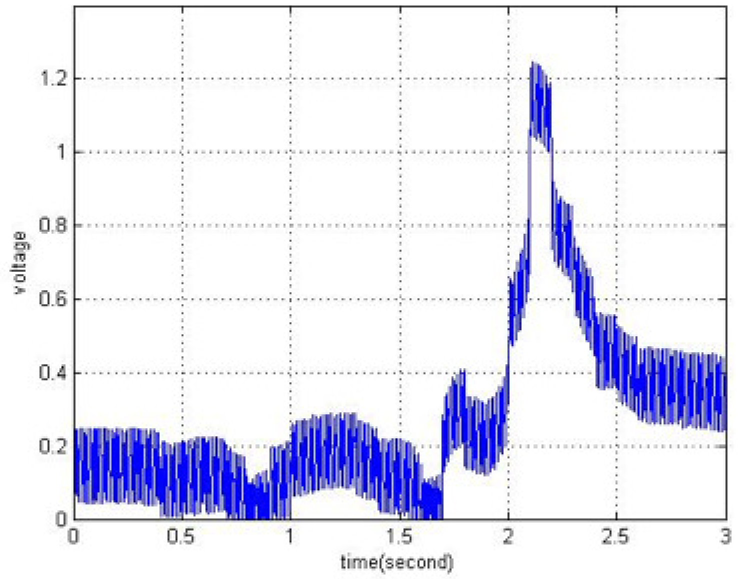


Figure 3.4. Actual signal measured from the artificial skin patch that is mounted on the CRS robot arm as shown in Fig. 3.3.

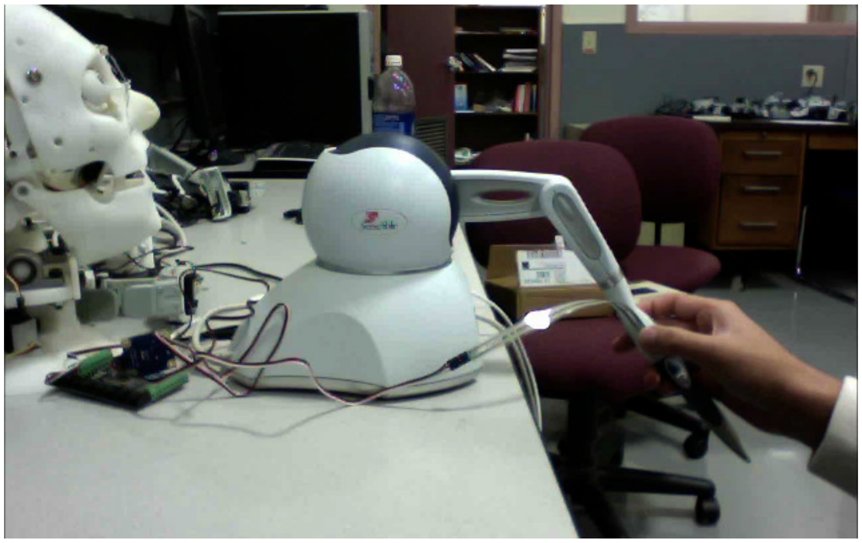


Figure 3.5. Phantom Omni haptics device that we used in experiment 6 and 7 to validate the proposed HRI algorithm.

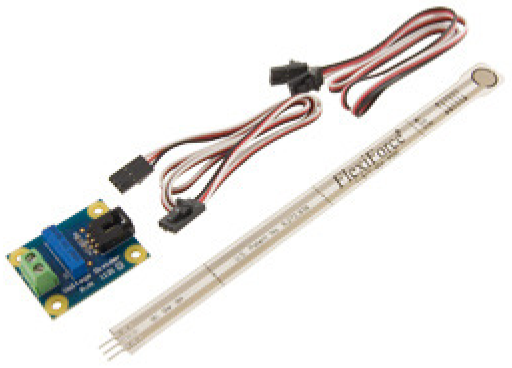


Figure 3.6. Flexiforce sensor used as a 1D force sensor.

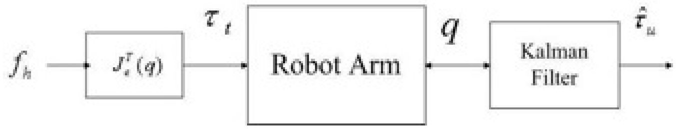


Figure 3.7. Diagram of the system without force measurement.

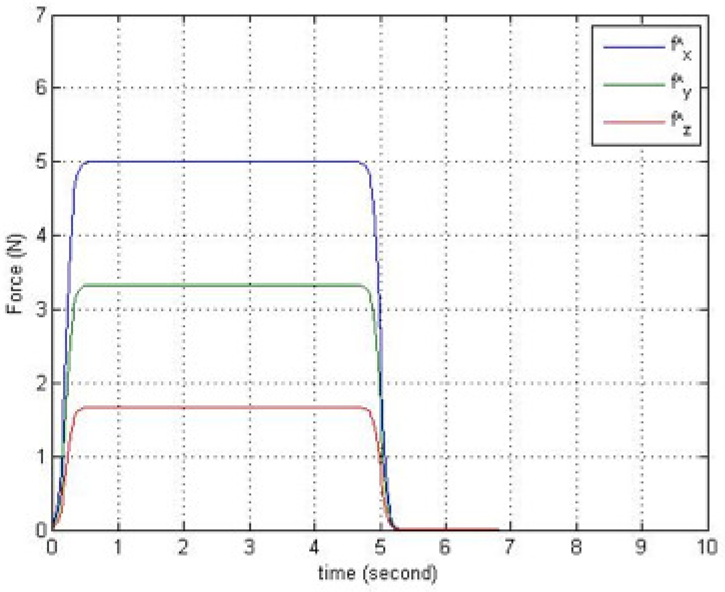


Figure 3.8. Simulated pushing force f_h is composed of force components in x, y and z direction in the local virtual end-effector frame.

3.3.2 Experiment 1

The diagram of the system in this case is shown in Fig. 3.7; input f_h is shown in Fig. 3.8. $M(q)$ and $N(q, \dot{q})$ [12] are exactly the same for both the robot arm model and the Kalman filter parameters (3.6). T sampling period equals 0.002 second, $\tau_m = 0$. The estimation error is shown in Fig. 3.9. $d_1 = 0.33$, $d_2 = 0.305$ and $d_3 = 0.25$ hence, Jacobian at the virtual end-effector (artificial patch mounting point) can be calculated using (3.4)-(3.5). Since the manipulator is in motion and the system is well identified, we expected that the estimation errors will be close to zero.

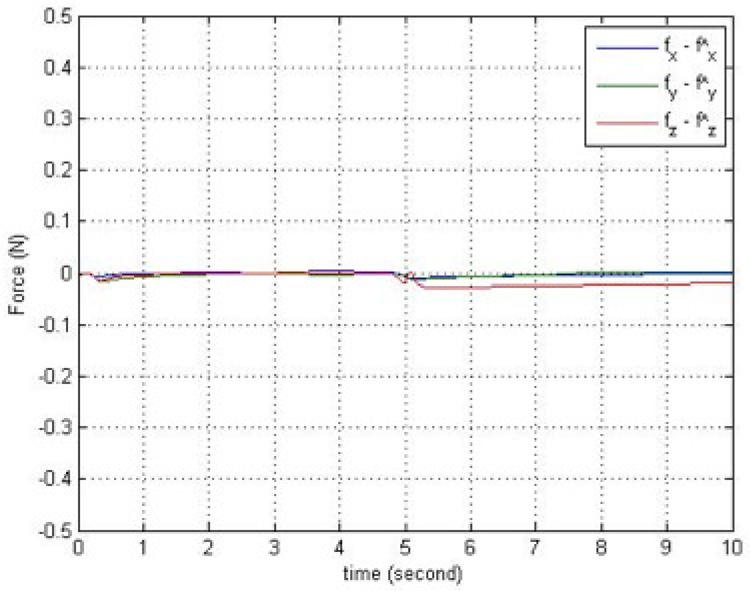


Figure 3.9. Experiment 1: HRI force estimation error if pushing force exceeds static friction.

3.3.3 Experiment 2

In this case everything is exactly the same as Experiment 1) except is smaller than in experiment 1 by a factor of ten. It is obvious that when pushing force is lower than the static friction of the robot arm, the arm does not move and as a result the Kalman Filter fails to estimate the unknown torque, as shown in Fig. 3.10.

3.3.4 Experiment 3

The diagram of the system in this case is shown in Fig. 3.11. In this case the robot arm model is assumed to be as in (3.6) but we introduce system uncertainty through $\tilde{N} = 1.5N$. In our simulation, this change is known to the robot arm dynamic model, but unknown to the EKF estimator. In addition we assume that $\alpha = 100$, joint encoder covariance noise (3.20) of 10^{-6}, e.g. $R = \begin{bmatrix} 10^{-6} & I_{2j} \end{bmatrix}$. The model error

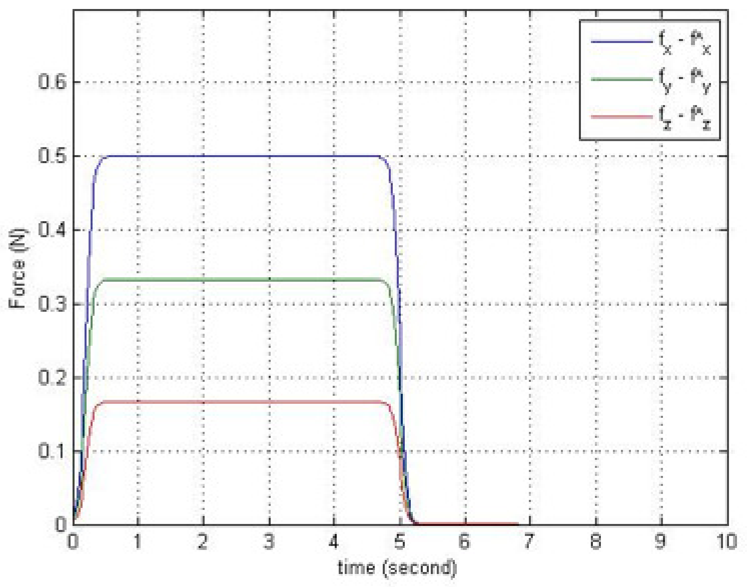


Figure 3.10. Pushing force is too low to move the arm. HRI force estimation errors do not converge to zero because they are based on encoder readings only.

is viewed as process noise by the Kalman Filter, and Q equals to $\begin{bmatrix} 0.1I_{2j} & 0 \\ 0 & 0.0591I_3 \end{bmatrix}$. The HRI pushing force input is the same as in experiment 1, e.g. the manipulator friction is overcome, but the system modeling error is large. Fig. 3.12 shows the estimation error without force measurement, Fig. 3.13 shows the estimation error of the system with one dimensional force measurement.

We conclude that adding a 1D force measurement greatly improves the estimation of the HRI interaction in the presence of noise and modeling uncertainties.

3.3.5 Experiment 4

The diagram of this case is shown in Fig. 3.1, same assumptions as in experiment 3 with one dimension force measurement. M_d is selected to be I_n, B_d is selected to be $0.7I_n$ and K_d equals to zero vector (3.28). PD gains of the controller (3.33) are

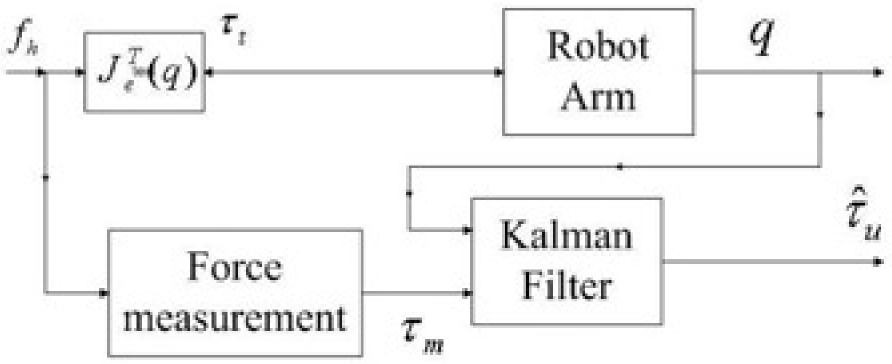


Figure 3.11. Diagram of the system with force measurement.

set as follow: $K_v = 10I_n$, $K_p = 25I_n$. The impedance control tracking error is shown in Fig. 3.14.

3.3.6 Experiment 5

An ideal system to measure the HRI interaction will measure force in all 3 dimensions to feed it into the impedance controller as pushing force shown in Fig. 3.11. The output difference between the ideal system and the system with only 1D force measurement from Experiment 4) is shown in Fig. 3.15. One can see that the difference in manipulator response is not significant.

Fig. 3.16 shows the output difference between the system using 3D measurement of HRI force and the system in Experiment 4) but with a small force input from Experiment 2) (small pushing force that does not overcome robot static friction). For the ideal system the robot will move regardless of the magnitude of the pushing force but the system in Experiment 4) the robot arms initial move is solely due to 1D force measurement and once the robot arm starts moving the other 2 force component can be estimated. This results in a small end position error as shown in Fig. 3.16. We conclude that a robot skin with 1D force measurement capability is not identical, but is sufficiently performant in creating desired small force HRI interactions.

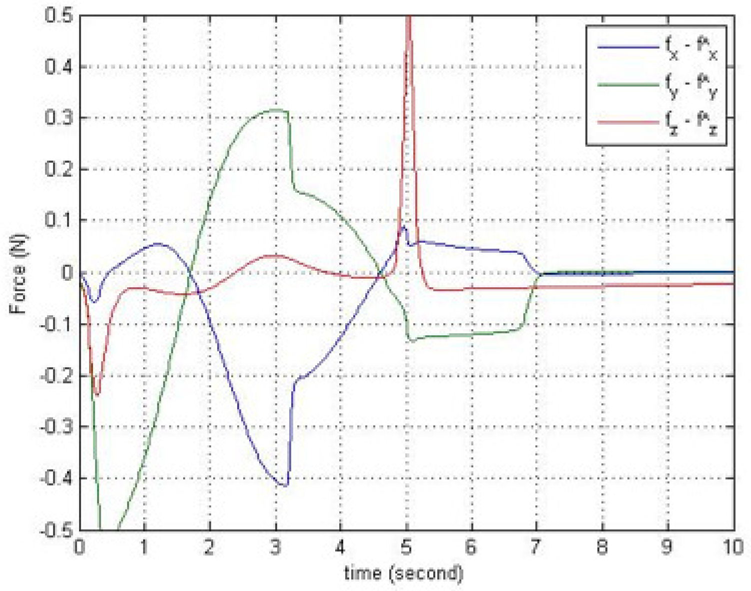


Figure 3.12. Due to modeling error, Kalman filter parameters are different from the robot dynamics model. Without force measurement, performance of the estimation is poor.

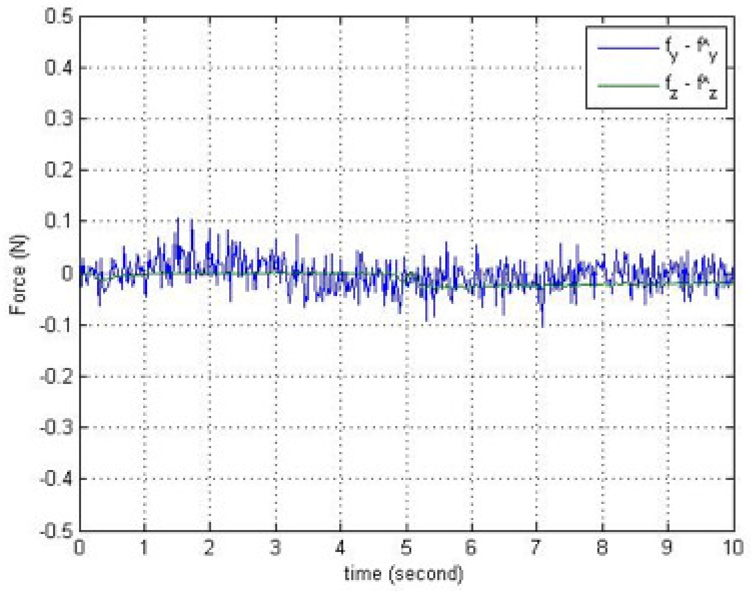


Figure 3.13. The estimation error is improved by adding 1 dimensional measurement to the system even though sensor noise is presented.

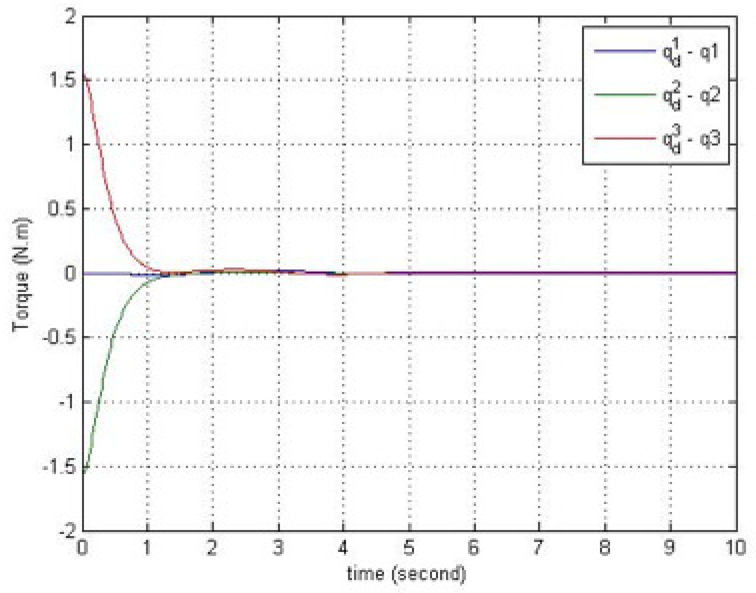


Figure 3.14. Tracking error of the system shown in Fig. 3.14.

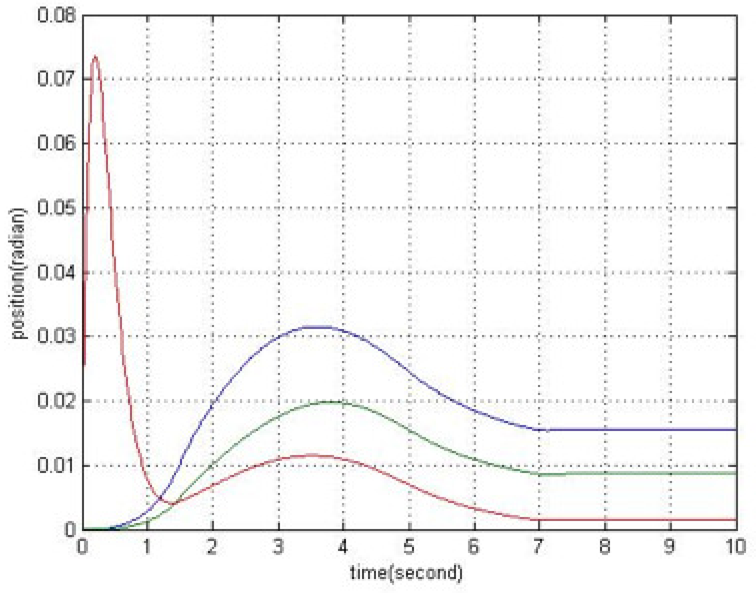


Figure 3.15. Joint output difference between the ideal system case and the system with only 1 dimension force measurement (Experiment 4) and the pushing force is low (as in Experiment 2) showing that HRI performance is similar.

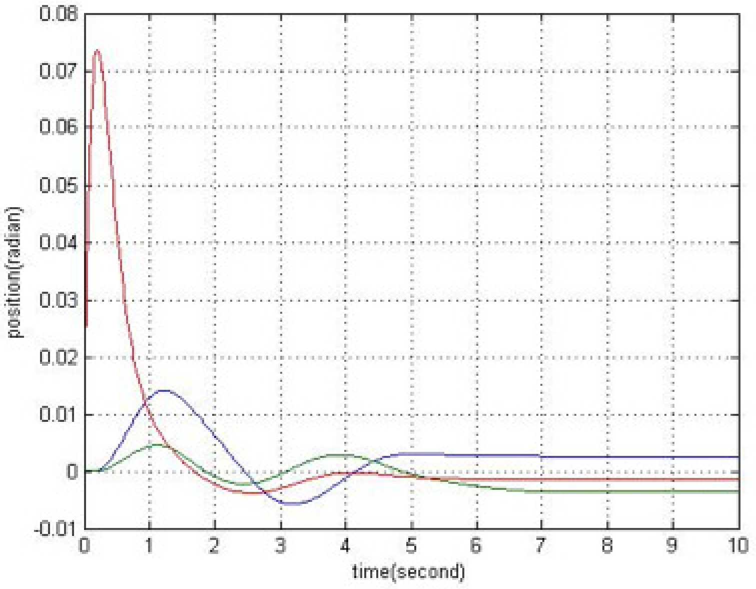


Figure 3.16. Joint output difference between the ideal system case and the system with only 1 dimension force measurement (Experiment 4). By adding more force sensing to the skin, HRI improvements are marginal.

3.3.7 Experiment 6

In this experiment, we implement the physical HRI system according to Fig. 3.1. There is no force measurement in this experiment. The parameters of the EKF estimator are the same as used in Experiment 3. For impedance force control, the control gains are the same as those used in Experiment 4 . The user interact with the device at the end-effector, resulting joint trajectory is shown in Fig. 3.17. Estimation of torque acting on the joints according to the interaction at the end-effector is shown in Fig. 3.18. Through the transformation we obtain estimation of the pushing force at the point of interaction, shown in Fig. 3.19. Fig. 3.21 shows the square error of the estimation, we can see from the results that without force measurement the performance of the estimation is poor similar to Experiment 3. Fig. 3.22 shows the trajectory of the end-effector according to interaction by the user.

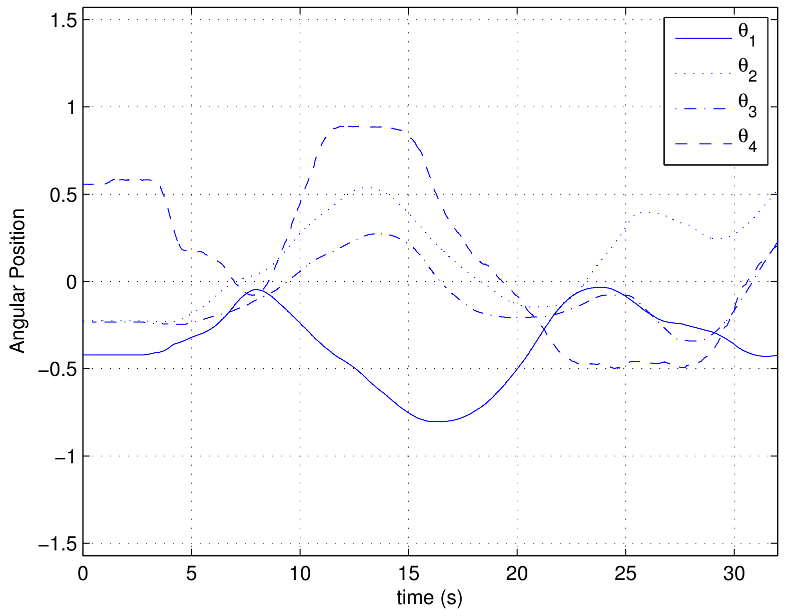


Figure 3.17. Experiment 6 - joints angular position due to interaction at the end-effector by the user.

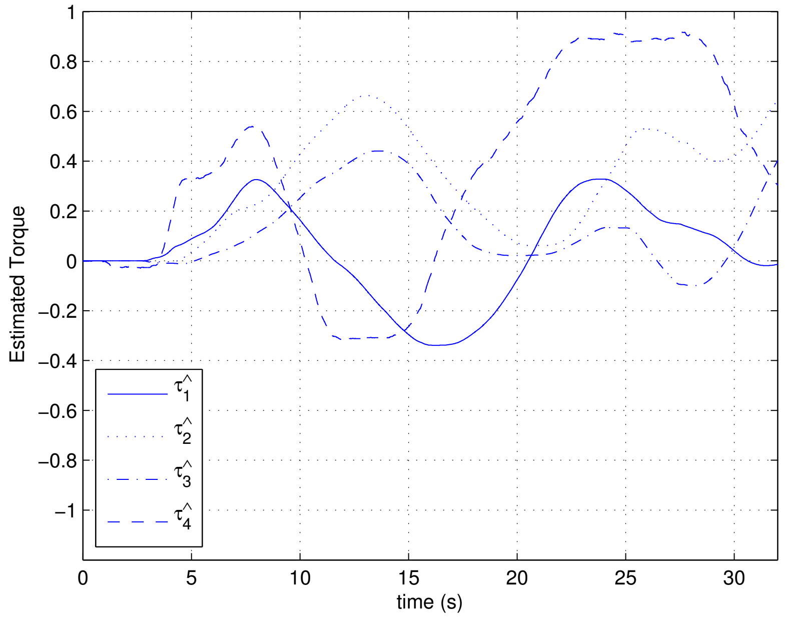


Figure 3.18. Experiment 6 - estimation of torque acting on individual joints according to the interaction at the end-effector.

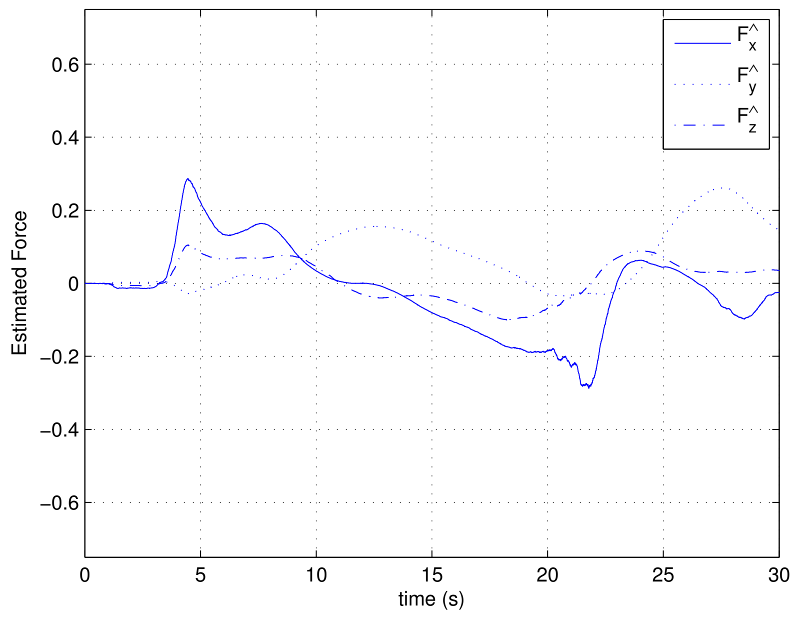


Figure 3.19. Experiment 6 - estimation of interaction force being applied at the end-effector by the user.

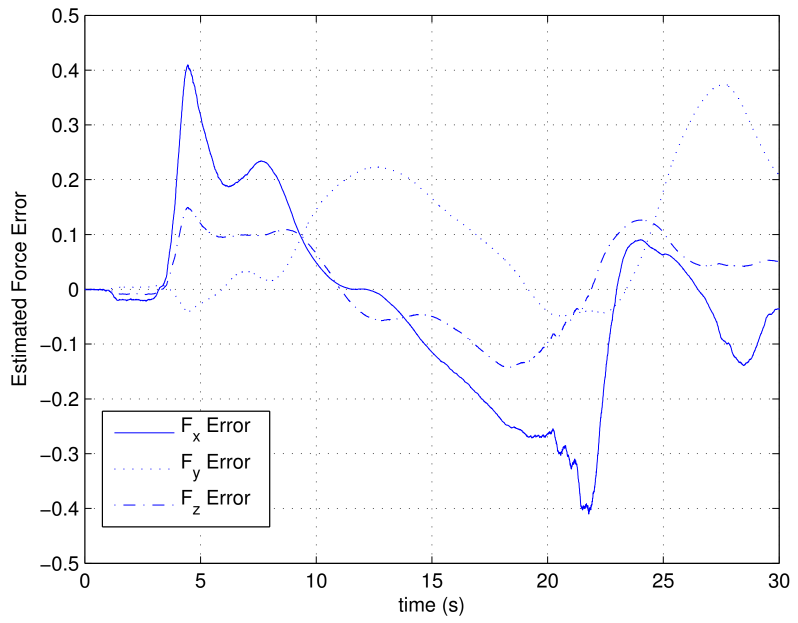


Figure 3.20. Experiment 6 - estimation error.

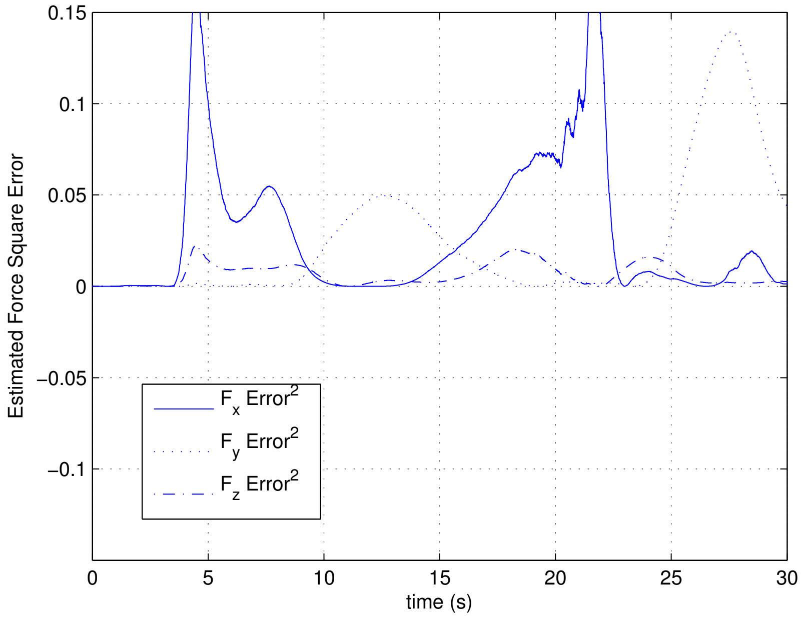


Figure 3.21. Experiment 6 - estimation error square.

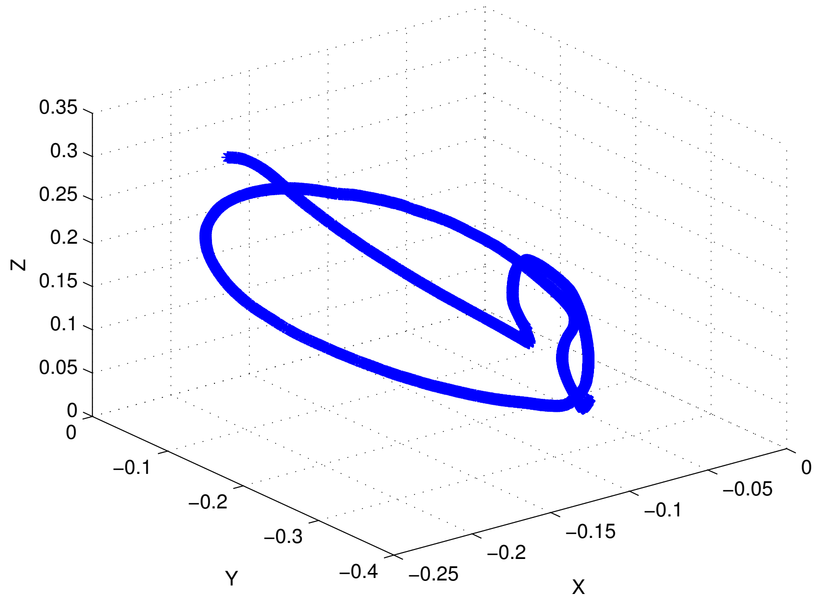


Figure 3.22. Experiment 6 - the trajectory of the end-effector according to interaction by the user.

3.3.8 Experiment 7

In this experiment, we implement the physical HRI system according to Fig. 3.1. The 1D force estimation is used in this experiment, the Flexiforce sensor is attached to the Omni haptics device as shown in Fig. 3.5. The parameters of the EKF estimator are the same as used in Experiment 3. For impedance force control, the control gains are the same as those used in Experiment 4 . The user interacts with the device at the end-effector, and the resulting joint trajectory is shown in Fig. 3.23. The estimation of torque acting on the joints according to the interaction at the end-effector is shown in Fig. 3.24. Torque acting on the joints resulting from 1D force reading is shown in Fig. 3.25. 1D force measurement is shown in Fig. 3.26. Through the transformation we obtain estimation of the pushing force at the point of interaction, shown in Fig. 3.27. The difference between the measured force by 1D sensor and estimated force obtained from EKF estimation is shown in Fig. 3.28. The squared error of the force estimation is shown in Fig. 3.29, by adding a 1D force measurement the force estimation improves significantly. Fig. 3.30 shows the trajectory of the end-effector according to interaction by the user.

The same conclusion as Experiment 3 can be drawn from Experiment 6 and 7 that adding a 1D force measurement greatly improves the estimation of the HRI interaction in the presence of noise and modeling uncertainties.

3.4 Summary

3.4.1 Summary

In this chapter we proposed a Human-Robot Interaction (HRI) algorithm based on EKF estimation and measurement of physical interaction force through robotic skin (1D force sensor). Simulation and experiment results confirm that the proposed

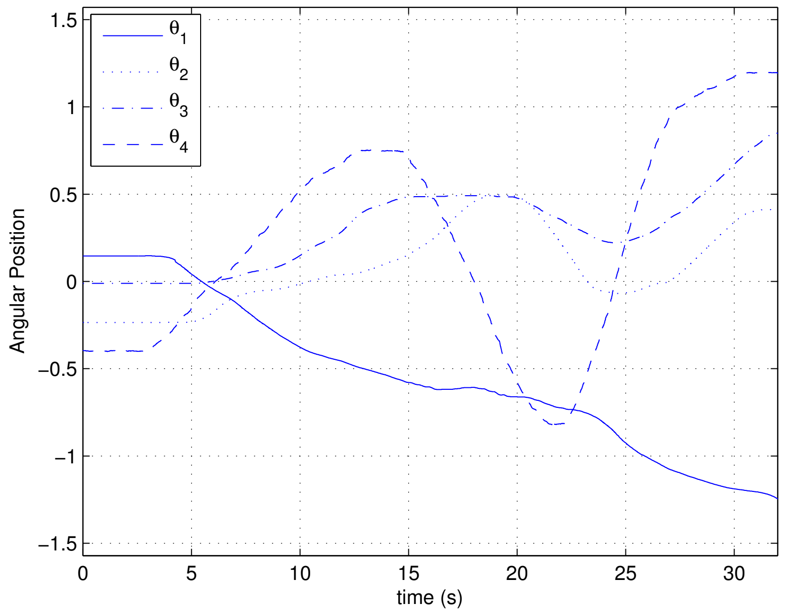


Figure 3.23. Experiment 7 - joints angular position due to interaction at the end-effector by the user.

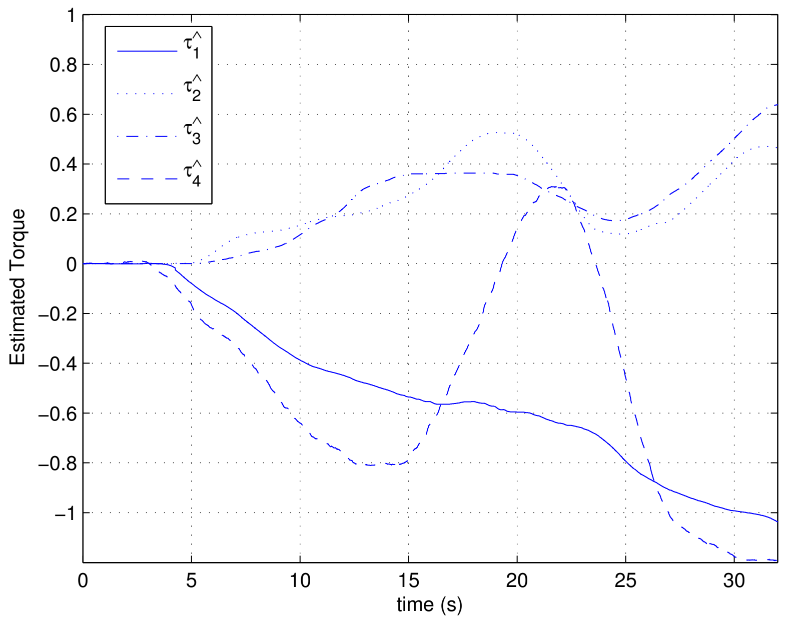


Figure 3.24. Experiment 7 - estimation of torque acting on individual joints according to the interaction at the end-effector.

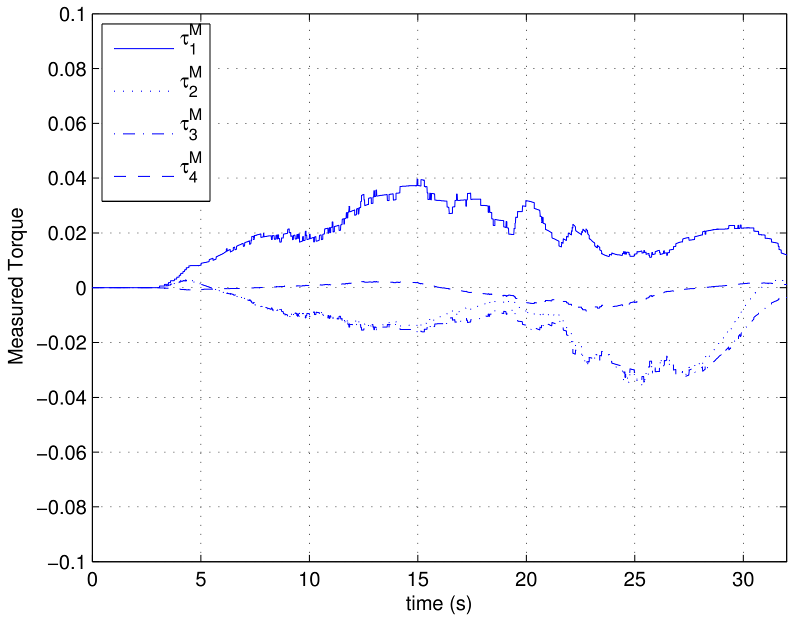


Figure 3.25. Experiment 7 - torque acting on the joints obtained through Jacobian transformation of the 1D measured force.

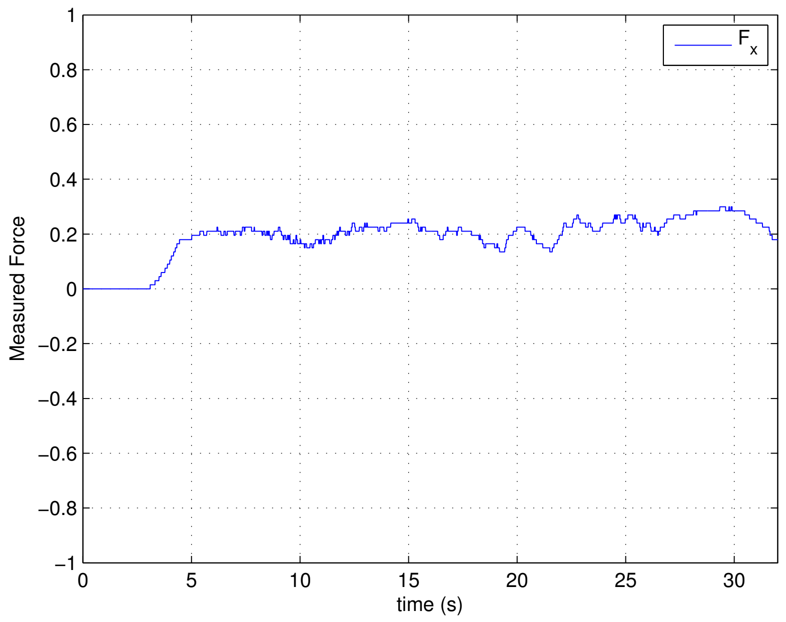


Figure 3.26. Experiment 7 - 1D force measurement of the force acting at the end-effector by the user.

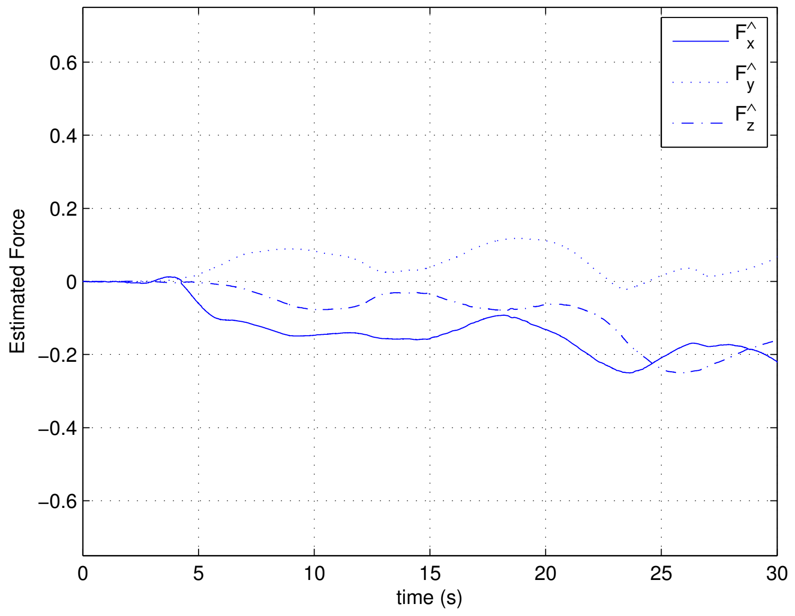


Figure 3.27. Experiment 7 - estimation of interaction force being applied at the end-effector by the user.

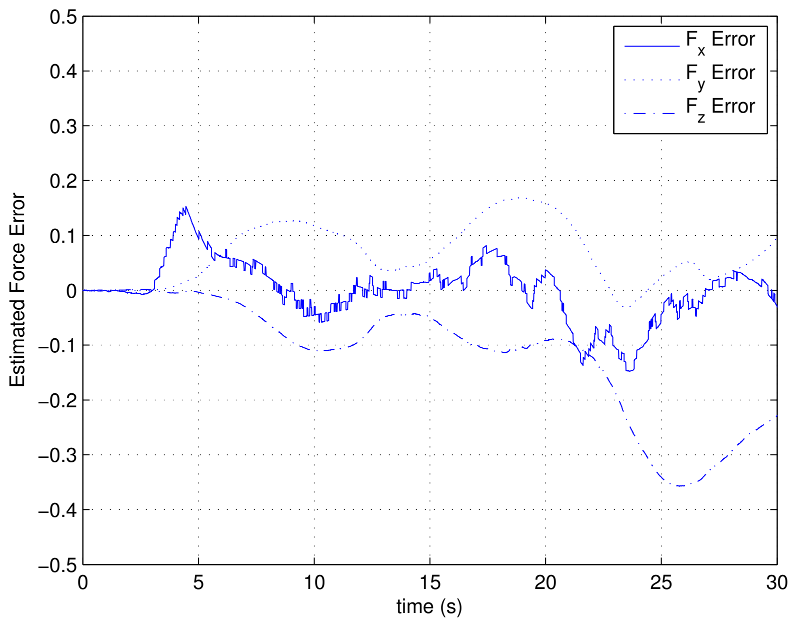


Figure 3.28. Experiment 7 - difference between the measured force by 1D sensor and estimated force obtained from EKF estimation.

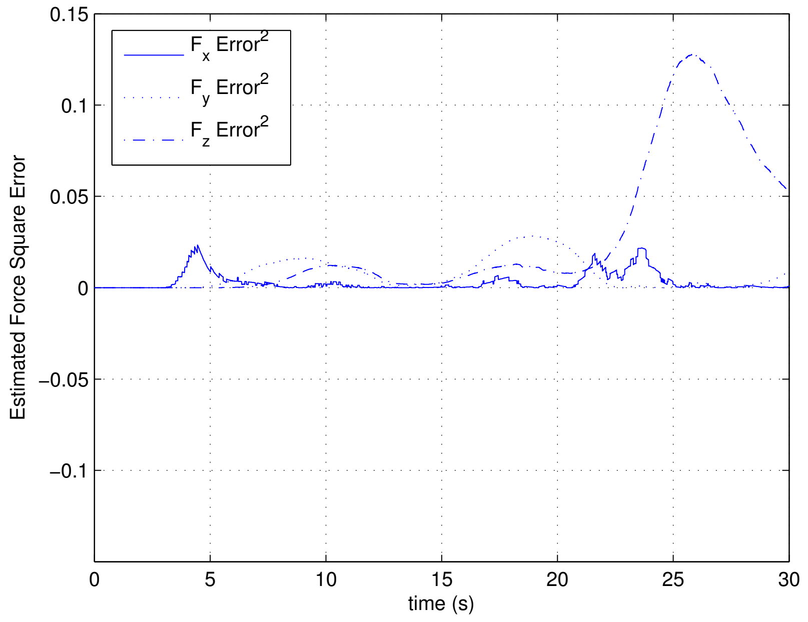


Figure 3.29. Experiment 7 - square error of the measured force by 1D sensor and estimated force obtained from EKF estimation.

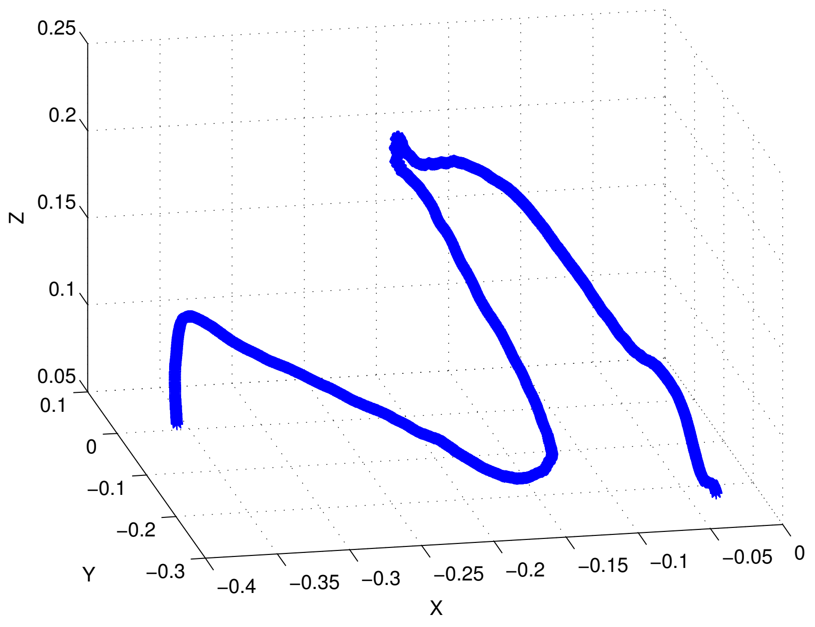


Figure 3.30. Experiment 7 - the trajectory of the end-effector according to interaction by the user.

method is efficient even though not all components of the contact force are available for measurement. The proposed method is able to estimate joint torque due to pushing force even when no measurement is available. However, if no force measurements are available, adequate HRI interaction requires an accurate robot dynamic model and a pushing force large enough to overcome joint static friction. By adding a 1D force sensor as a robot skin, we can greatly improve the HRI interaction. The simulation and experiement results also show that marginally better estimation is achieved by adding force measurements in more directions.

CHAPTER 4

HUMAN-ROBOT INTERACTION THROUGH VISION

4.1 Humanoid Robotic Actors: LILLY and Zeno

A humanoid actor with facial expressions, LILLY, is being designed at ARRIs Humanoid Lab in collaboration with Hanson Robotics Inc. LILLY exhibits facial expressions and enhanced interactivity, including eye movement, eye contact, speech synthesis, human expression mimicking, human emotion synthesis. LILLY is a second generation humanoid, similar in design with the head of Einstein Hubo [4], as shown in Fig. 4.1. One of LILLYs subsystems includes tracking of a human in its field of view for purposes of conversational interaction and mimicking. To enhance the realism of interaction, this subsystem requires both 6D pose estimation of the human, as well as tracking of the human in a natural, direct eye contact pose.

Zeno is a humanoid robot head developed by Hanson Robotics. David Hanson provided Heracleia lab at UTA with a copy of Zeno head prototype for experimental testing and algorithm development. The picture of the humanoid robotic actor Zeno is shown in Fig. 4.2

A set of feature points on the images acquired by LILLYs vision system are used to estimate the pose of the human. The feature points selected for this purpose are left eye, right eye and the location of the mouth. These feature points are extracted using standard image processing techniques such as horizontal and vertical gradient methods, circular Hough transform and probability distribution of skin color [96], [97] and [98]. The relative coordinates of the feature points (left eye, right eye, mouth) are

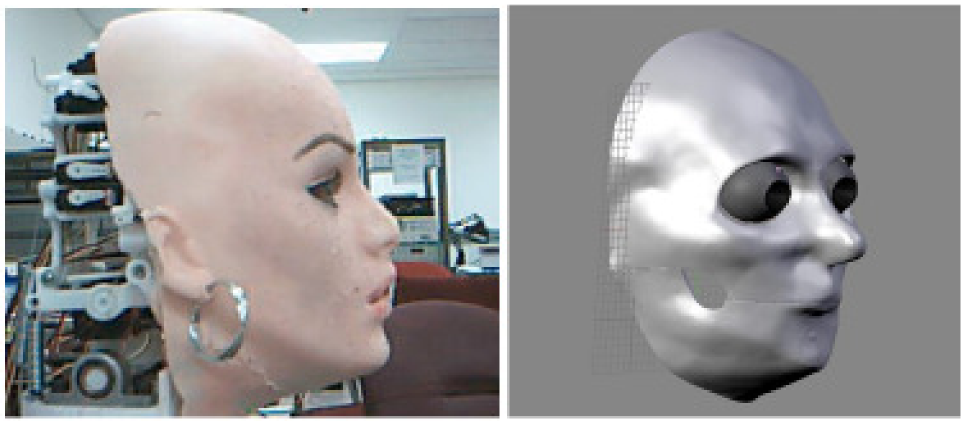


Figure 4.1. (a) Humanoid actor, LILLY, at ARRIs Humanoid Lab. This model uses RC servos to accomplish facial expressions. We are currently embedding muscle actuators directly in the robot skin (b) Solidworks model of Lillys skull and eyes animated via the tracking controller.

nominally known a priori. In [99], they presented an efficient two stage neural-network based approach for feature extraction and expression classification.

4.2 Pose Estimation and Object Tracking Problem

In this chapter we test a target pose-estimation and tracking algorithm separately from face/feature recognition. For this purpose, we use a simple 3D object as target instead of a human. Much simpler image processing is performed on colored markers placed on the 3D object. Fig 3.2 shows the geometry of the system where the camera and the rigid object coordinate frames is represented. The feature point coordinates are related to the image frame (row, column) through a standard transformation written as:

$$\begin{bmatrix} c \\ r \end{bmatrix} = \frac{1}{Z^C} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} x^C \\ y^C \end{bmatrix} + \begin{bmatrix} c_0 \\ r_0 \end{bmatrix} \tag{4.1}$$

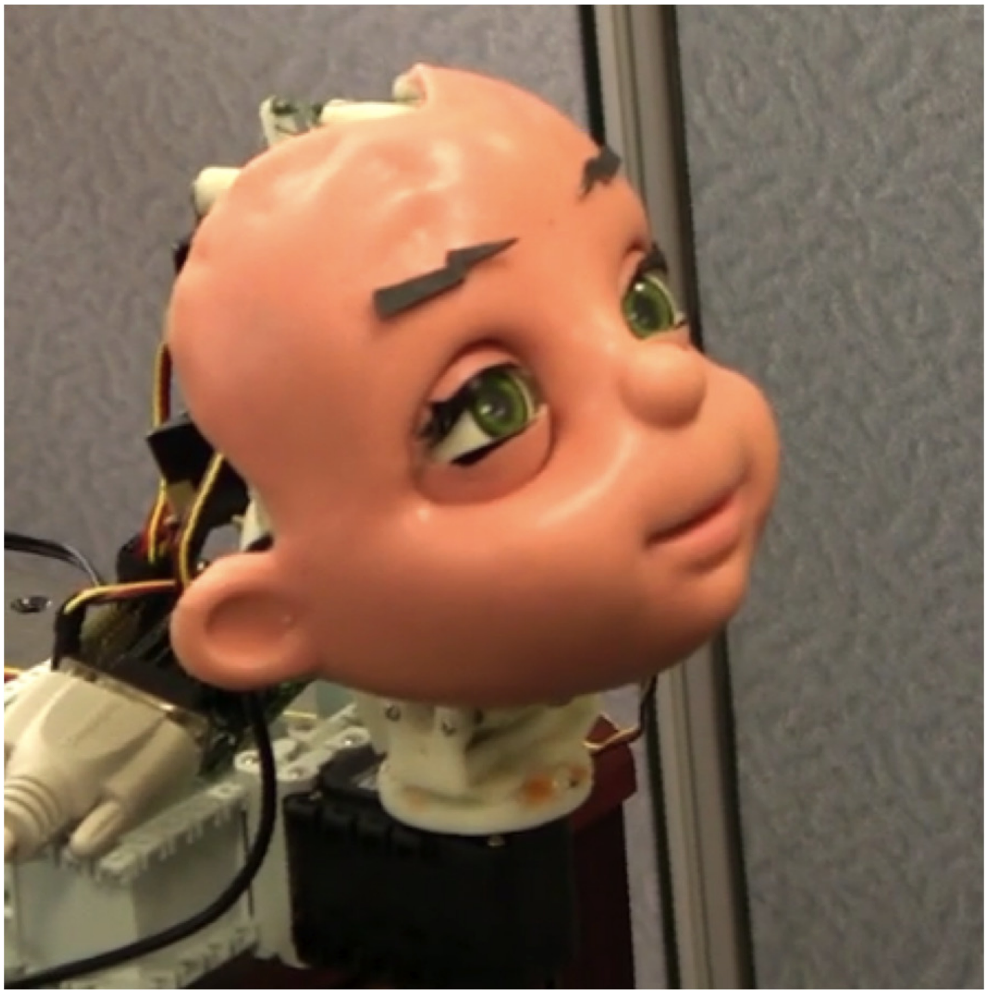


Figure 4.2. Side view of the Zeno head doing human tracking.

where $[c, r]^T$ are the image coordinates of the feature point, $[x^C, y^C, z^C]^T$ are the feature point coordinates in the camera frame, (f_x, f_y) are the intrinsic camera parameters defining the focal length and pixel dimensions and $[c_0, r_0]^T$ is the principal point of the camera frame in the image plane.

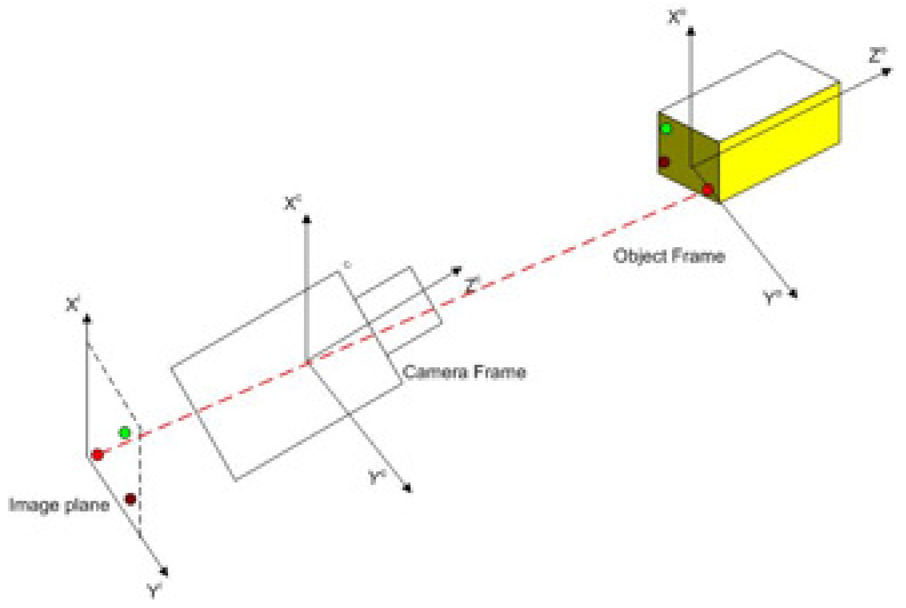


Figure 4.3. Geometric model of the object-camera-image reference frame.

The feature point coordinates are determined in the camera frame using the transformation, given as:

$$\begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} = R \begin{bmatrix} x^0 \\ y^0 \\ z^0 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.2}$$

where $[x^0, y^0, z^0]^T$ is the known coordinate vector of the feature points in the object frame, $[x, y, z]^T$ is the unknown translation vector of the object and R is the rotation matrix of unknown orientation angles (roll(ϕ), pitch(θ), yaw(φ)) of the object, given by:

$$R = \begin{bmatrix} C_\phi C_\theta & C_\phi S_\theta S_\varphi - S_\phi C_\varphi & C_\phi S_\theta C_\varphi + S_\phi S_\varphi \\ S_\phi C_\theta & S_\phi S_\theta S_\varphi + C_\phi C_\varphi & S_\phi S_\theta C_\varphi - C_\phi S_\varphi \\ -S_\theta & C_\theta S_\varphi & C_\theta C_\varphi \end{bmatrix} \tag{4.3}$$

where $C_\phi = \cos\phi$, $S_\theta = \cos\theta$ and $C_\varphi = \cos\varphi$.

The transformation given in (4.2) and (4.3) forms the basis to estimate the position ($[x, y, z]^T$) and orientation (roll(ϕ), pitch(θ), yaw(φ)) vectors. These unknown parameters are estimated using a semi-decoupled EKF method proposed in the following section with respect to the camera frame. These estimated parameters are then used for object tracking in the next section.

4.3 Pose Estimation Using the Extended Kalman Filter

The features used are image measurement points for the Kalman filter; we use only three non-collinear points, instead of 4, 5 or more used by others. It has been shown in [100] and [101] that with three feature points, a given 3D object with known dimensions can be uniquely located in a 3D space with at most 4 distinct ambiguous solutions. In our case this ambiguity does not exist if the initial triangle pose estimate is close to the correct pose, or if the initial triangle pose estimate is roughly perpendicular to the camera line of sight. This is obvious because of the EKF state update and propagation from close to a correct estimate would yield the correct pose while the object is undergoing motion.

For example, consider the triangle shown in Fig. 4.4, with three vertices as the selected feature points. Points B' and B'' are produced by the rotation along AC - axis by an angle φ. Even though these two points would yield same pixel measurements, the estimation process yields the incremental rotations the object has undergone from B' to B''. The whole trajectory is still ambiguous, but if we can distinguish B' to B'' at the beginning of the estimation process, we never lose track of the right pose.

The experimental results presented in the later subsection also demonstrate that we can track three non-collinear feature points through the proposed EKF pose estimation approach with high performance and convergence.

The Kalman Filter requires a dynamical model and a measurement model. For the dynamical model, it is assumed that the object moves with a constant velocity

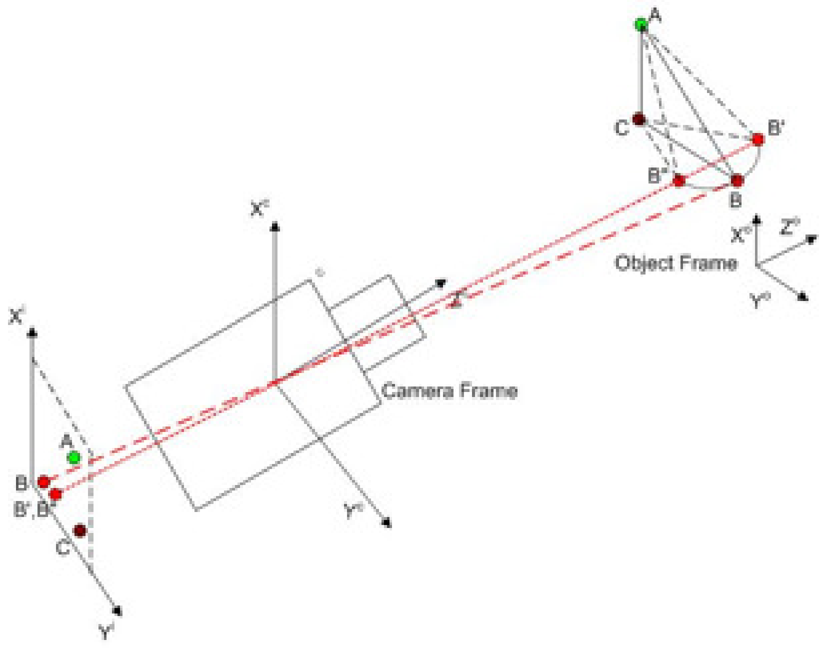


Figure 4.4. Illustration of three non-collinear points producing different pixel measurements as the object moves between ambiguous poses.

over a defined sampling time, which is certainly the case at low speeds, typical of natural human motions during conversations with a robot. Even when the constant velocity assumption does not hold, the error in dynamical model will be compensated by properly tuning the process covariance matrix Q. The measurement model for the EKF is given by (4.2) and (4.3), as detailed in the following subsections.

4.3.1 Position and Orientation Vector Estimation

For the estimation of position and orientation vector, the dynamical model is defined as follow:

$$\ddot{S} = 0 \tag{4.4}$$

where $S = \begin{bmatrix} X & \dot{X} & \Phi & \dot{\Phi} \end{bmatrix}^T$, $X = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and $\Phi = \begin{bmatrix} \phi & \theta & \varphi \end{bmatrix}^T$. Converting the continuous model in (4.4) to the discrete time model yields

$$S_{k+1} = AS_k + \gamma_k \tag{4.5}$$

where γ_k is the process noise of the discrete time dynamical model described as Gaussian with zero mean and a noise covariance of Q and the plant matrix is given as

$$A = \begin{bmatrix} O_3 & T \times I_3 \\ O_3 & O_3 \end{bmatrix} \tag{4.6}$$

where O_3 and I_3 are the 3x3 zero and identity matrices, respectively, and T is the sampling time. The measurement model for estimation of position and orientation vector is defined by (4.2) and (4.3). The Extended Kalman filter is defined by the following equations:

Model:

$$S_{k+1} = AS_k + \gamma_k \tag{4.7}$$

$$B_k = h_k\left(S_k\right) + \vartheta_k \tag{4.8}$$

where $B_k = \begin{bmatrix} c & r \end{bmatrix}^T$, $h_k = \frac{1}{Z^C} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} x^C \\ y^C \end{bmatrix}$ and ϑ_k is the measurement noise with the covariance given by R_k.

Gain:

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k \right]^{-1} \tag{4.9}$$

where $H_k = \frac{\partial h_k}{\partial S_k}|_{S_k^-}$,

Propagation:

$$\hat{S}_{k+1}^- = D\hat{S}_k^- \tag{4.10}$$

$$P_{k+1}^- = AP_k^+ A^T + Q_k \tag{4.11}$$

Update:

$$P_{k+1}^{+} = \left[I - K_k H_k\right] P_k^{-} \tag{4.12}$$

$$\hat{S}_k^{+} = \hat{S}_k^{-} + K_k \left[B_k - h_k\left(\hat{S}_k^{-}\right)\right] \tag{4.13}$$

(4.13) is the estimated position and orientation vector. This position and orientation vector estimation from this Kalman filter is the estimated position and orientation of the object of interest.

4.3.2 Object Tracking Using Visual Servoing

Since pose estimation for object using a single fixed camera is limited by the reduced camera field of view, we add an object tracker using visual servoing using a motorized robotic camera. This is equivalent to a humanoid robot head tracking an object by motions of its neck and eyes in order to increase its field of view. The visual servoing scheme presented in this section uses the position and orientation estimates from the EKF pose estimation filters. Besides interactivity, the obvious advantage of using a tracker is the fact that the EKF can recover feature points when they go out of the field of view. With only 2 feature points in the field of view the estimates will diverge, therefore object tracking is essential in actual applications. In this section, object tracking is performed through a simple 2 degrees-of-freedom (DOF) pan and tilt which represents neck mechanism of LILLY, where the point to be tracked is obtained by converting the estimated position vector into the image coordinate using (4.1). The equation in Laplace domain of image-based visual servoing scheme is given by:

$$v_c\left(s\right) = -\lambda\left(s\right)\hat{L}^{+} e\left(s\right) \tag{4.14}$$

where v_c is the camera spatial velocity, λ is control gain,$\hat{L}^+$ is the image Jacobian and e is the difference between desired feature image and current feature image, defined as

$$e = s^* - s \tag{4.15}$$

where s^* is the desired feature, in this case the center of the image frame and s is the vector of image coordinates obtained from the position estimates using (4.1). The overall estimation-servoing scheme is shown in Fig 4.4. A simple 2 DOF pan and tilt camera was used to validate the scheme, and therefore, $\hat{L}^+$ can be taken to be the identity matrix and λ is chosen as a PID controller with tuned gains.

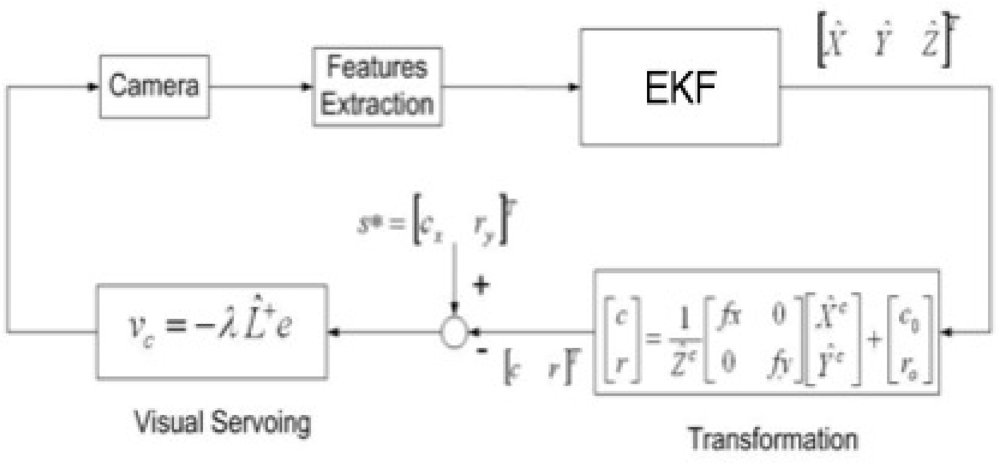
The resulting controller is simply:

$$\begin{bmatrix} v_{x_c} \\ v_{y_c} \end{bmatrix} = -\left(K_p + \frac{1}{s}K_i + sK_d\right) \left[\begin{bmatrix} c \\ r \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix}\right] \tag{4.16}$$

where $s^* = \begin{bmatrix} c_x & r_y \end{bmatrix}^T$ is the center of the image frame, v_{x_c} and v_{y_c} are the motor control signals and $\begin{bmatrix} c & r \end{bmatrix}^T$ is the projected image coordinates of the estimated pose.

4.4 Realistic Neck-Eye Motion Distribution

This section describes the theoretical approach on how to track an object using a robot head with head-eye coordination given that the position of object is known. The analysis of head-eye kinematics in this section is based on past work with simulated conversational animated figures [76].



Camera
Features Extraction
EKF
$[\hat{\dot{X}} \ \hat{\dot{Y}} \ \hat{\dot{Z}}]^T$
$s^* = [c_x \ r_y]^T$
+
-
$v_c = -\lambda \hat{L}^+ e$
$[c \ r]^T$
$\begin{bmatrix} c \\ r \end{bmatrix} = \frac{1}{\hat{Z}^c} \begin{bmatrix} fx & 0 \\ 0 & fy \end{bmatrix} \begin{bmatrix} \hat{X}^c \\ \hat{Y}^c \end{bmatrix} + \begin{bmatrix} c_0 \\ r_0 \end{bmatrix}$
Visual Servoing
Transformation

Figure 4.5. A diagram representing the overall estimation-tracking system.

4.4.1 Robot Actor Tracking With Head Eye Coordination

A diagram describing the robot head, along with its coordinate frame notation while tracking an object is shown in Fig. 4.6.

The distance between the center of the skull and the object is given by:

$$I = \sqrt{(x - r_x)^2 + (y - r_y)^2 + (z - r_z)^2} \tag{4.17}$$

where r_x, r_y and r_z epresent the location of the center of the skull with respect to a reference frame and x, y, z represent the location of the object of interest. Furthermore, the trigonometric roll-pitch-yaw (RPY) angles of the skull pointing toward the object can be represented through the following equations:

$$k_x = \frac{x - r_x}{l}, \qquad k_y = \frac{y - r_y}{l}, \qquad k_z = \frac{z - r_z}{l} \tag{4.18}$$

$$\cos\alpha = \frac{k_x}{\sqrt{k_x^2 + k_y^2}} \tag{4.19}$$

$$\beta = \arcsin k_z - \arcsin\frac{h}{l} \tag{4.20}$$

Furthermore, these angles can also be obtained from the standard inverse kinematics:

$$\gamma_d = \arctan 2\,(r_{32},\ r_{33}) \tag{4.21}$$

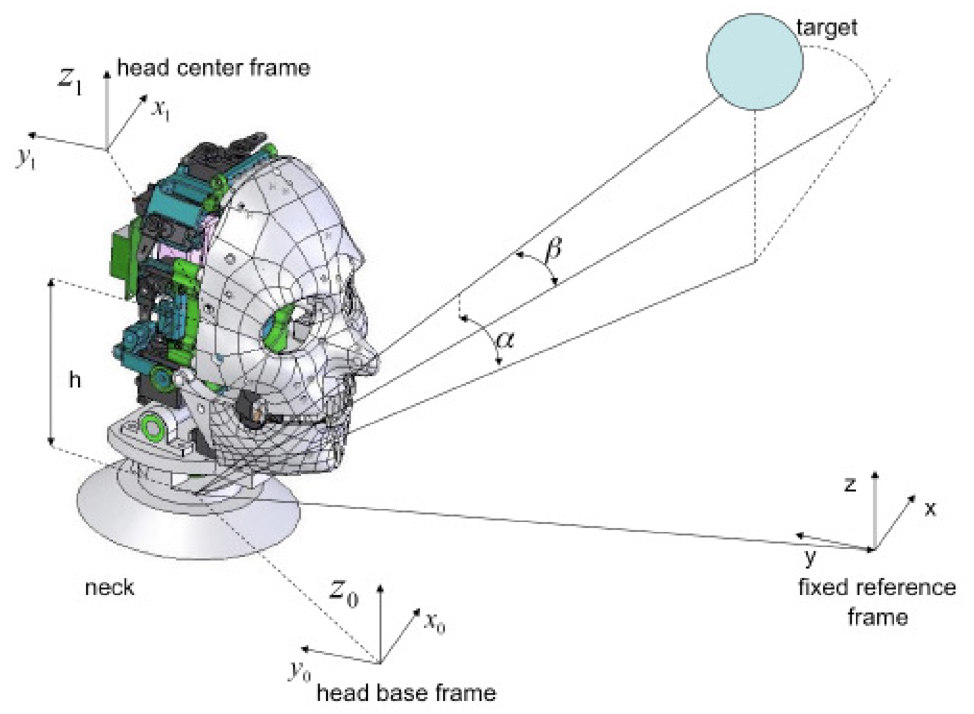


Figure 4.6. A robot head with respect to a fixed reference frame and a target object to track.

$$\beta_d = \arctan 2\left(r_{31},\ \sqrt{r_{11}^2 + r_{21}^2}\right) \tag{4.22}$$

$$\alpha_d = \arctan 2\left(r_{21},\ r_{11}\right) \tag{4.23}$$

where $r_x y$ is the element of the transformation matrix.

The neck must rotate to thsese angles if the skull must point directly to the object, however, if eyes can also pan and tilt, we can distribute the required rotational motion between the skull and the eyes. In [76], the partial motion of the neck, represented by angles α_n, β_n and γ_n are obtained using a percentage function but in this paper we propose a new approach to calculate them, as detailed in Section 4.4.2.

After the neck rotation angles α_n, β_n and γ_n are obtained, the head rotation matrix is:

$$R = \begin{bmatrix} C\alpha_n\, C\beta_n & -S\alpha_n\, C\gamma_n + C\alpha_n\, S\beta_n\, S\gamma_n & S\alpha_n\, S\gamma_n + C\alpha_n\, S\beta_n\, S\gamma_n \\ S\alpha_n\, C\beta_n & C\alpha_n\, C\gamma_n + S\alpha_n\, S\beta_n\, S\gamma_n & -C\alpha_n\, S\gamma_n + S\alpha_n\, S\beta_n\, C\gamma_n \\ -S\beta_n & C\beta_n\, S\gamma_n & C\beta_n\, C\gamma_n \end{bmatrix} \tag{4.24}$$

The new location of the eyes can be calculated by

$$e_l = R\, e_{l0} \tag{4.25}$$

$$e_r = R\, e_{r0} \tag{4.26}$$

where e_l, e_r are the new location of left and right eyes and e_{l0}, e_{r0} are the current location of left and right eyes. Once the new location of the eyes are know, pan and tilt angles of both eyes can be obtained

$$\theta_l = \arcsin\left(\frac{x - e_l^x}{\sqrt{(x - e_l^x)^2 + (y - e_l^y)^2}}\right) \tag{4.27}$$

$$\psi_l = \arcsin\left(\frac{z - e_l^z}{\sqrt{(x - e_l^x)^2 + (y - e_l^y)^2 + + (z - e_l^z)^2}}\right) \tag{4.28}$$

where θ_l is the pan angle of the left eye and ψ_l is the tilt angle of the left eye.

$$\theta_r = \arcsin\left(\frac{x - e_r^x}{\sqrt{(x - e_r^x)^2 + (y - e_r^y)^2}}\right) \tag{4.29}$$

$$\psi_r = \arcsin\left(\frac{z - e_r^z}{\sqrt{(x - e_r^x)^2 + (y - e_r^y)^2 + + (z - e_r^z)^2}}\right) \tag{4.30}$$

where θ_r is the pan angle of the rightt eye and ψ_r is the tilt angle of the right eye.

4.4.2 Optimization Approach for Realistic Neck-Eye Motion Distribution

In order for the motion distribution of the head and eyes to look realistic, we apply the following qualitative rules, similar to those in humans [76, 81]:

- The robot head is attached to a torso hence the head position is considered to be mobile.
- With respect to the torso, the rotation at the neck cannot exceed certain limits.
- When the object is close to the head only saccadic movement occurs (eyes movement only).
- Both eyes point to the object of interest independently.
- Head rotation velocity should not exceed certain maximum values (allowable by the hardware).
- Eyes angular displacement should gradually converge to zero to reduce the strain on the eyes if the object is stationary.

In [76], the motion of the head and eyes was coordinated by the use of a percentage function a way to distribute neck and eye motion according to a predefined weighting function, which imposes turning angles at the neck and eyes which are dependent on each other. In this paper, the dynamical characteristics of the head-eye motion are further factored into the distribution scheme via the addition of an angular velocity term into an objective function to minimize. This optimization has a similar effect to a quick eye movement (saccadic movement) during the transient response period, and to a minimal eye displacement as the object motion approach steady-state.

Consider the quadratic objective function:

$$J = \dot{\Theta}^T Q \dot{\Theta} + \Phi^T R \Phi \tag{4.31}$$

where $\dot{\Theta}$ is the head angular velocity and $\Theta = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T$, Φ is the eye angular displacement and $\Phi = \begin{bmatrix} \theta_l & \psi_l & \theta_r & \psi_r \end{bmatrix}^T$, Q and R are weighting matrices. Since the head and eye angles are coordinated to point toward an object, we can express the eyes angular displacement as a function of head angular displacement through a kinematic function $f(.)$:

$$\Phi = f(\Theta) \tag{4.32}$$

Factoring in a forward time-step discretization, the head angular velocity can be approximated by:

$$\dot{\Theta} \approx \frac{1}{T}(\Theta_t - \Theta_{t-1}) \tag{4.33}$$

where Θ_{t-1} is angular position of the previous step and T is a sampling period. (4.31) can be rewritten as follows:

$$J = \left(\frac{1}{T^2}(\Theta_t - \Theta_{t-1})^T Q (\Theta_t - \Theta_{t-1}) \right) + f(\Theta)^T R f(\Theta) \tag{4.34}$$

The neck pose Θ that minimizes (4.34) will result in a motion that satisfies all the qualitative requirements above:

$$\Theta^* = \underset{\Theta \in [-\Theta_{max} \;\; \Theta_{max}]}{\arg\min} \left[\frac{1}{T^2}(\Theta_t - \Theta_{t-1})^T Q (\Theta_t - \Theta_{t-1}) + f(\Theta)^T R f(\Theta) \right] \tag{4.35}$$

The Q and R weight matrices are arbitrary positive definite matrices, however, they should be selected so that the cost function increases significantly for large head angular displacement and small eye angular velocity.

The solution to (4.35) Θ^* can be used with (4.24) to compute the appropriate rotation angles of the eyes:

$$\alpha = \alpha^*, \qquad \beta = \beta^* \qquad \mathit{and} \qquad \gamma = \gamma^* \tag{4.36}$$

Experimental results will be presented in support of concluding that the proposed optimization scheme can accomplish saccadic motion without requiring the hardware to incorporate designs taken from eyes kinesiology studies [81]. The solution for equation (4.35) can be obtained numerically, for instance using MATLAB optimization toolbox simplex search.

4.5 Enhancement of Tracking Speed and Robustness to Modeling Errors

The derivations of the tracking model in the previous section are based on a kinematic head-eye model in which dimensional parameters are known. In an actual hardware implementation, however, we usually encounter significant discrepancies between the actual robot head dimension and the kinematic model entries, and unmodeled kinematics of the head mechanisms. In addition, the optimization (4.35) must be run at every single sampling instance.

In practice, a robot head-eye pose at the desired optimal angles (* values), will exhibit a tracking error as depicted in Fig. 4.7. The effect of this error will be apparent to both the human engaged in eye contact with the robot, and to the robot through its eye camera images. We propose using two enhancements to minimize tracking error and eliminate the need for solving the optimization(4.35) at every time step. The first enhancement is the use of visual feedback from the tracked object, and the second one is the Actor-Critic reinforcement learning method as an combined alternative to finding the optimal solution of (4.35), and visual tracking. The addition of visual feedback to be used with method detailed in section 4.4.2 is discussed in subsection 4.5.1 and the Actor-Critic method is discussed in subsection 4.5.2.

4.5.1 Visual Feedback Approach

To correct for the gaze discrepancy, we use a dynamical visual feedback scheme. The goal is to correct the tracking error on-the-fly while still maintaining the realism of head-eye coordination. Assuming that an image tracking error is available in pixel

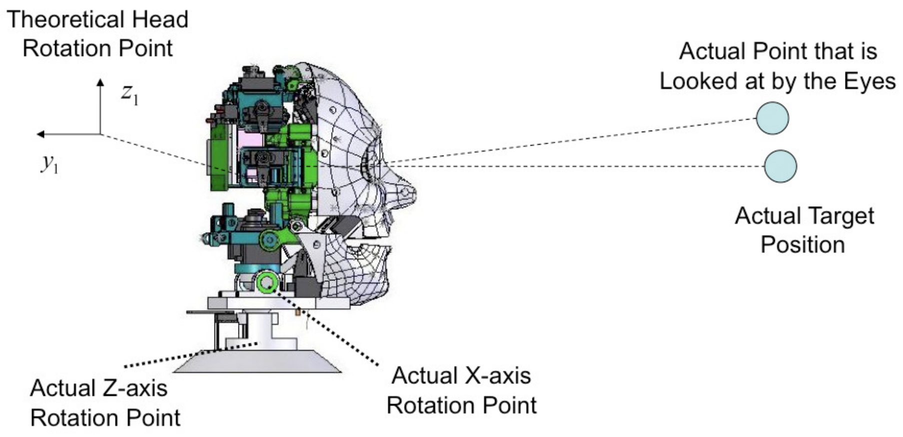

Theoretical Head Rotation Point
z_1
y_1
Actual Point that is Looked at by the Eyes
Actual Target Position
Actual Z-axis Rotation Point
Actual X-axis Rotation Point

Figure 4.7. A diagram showing the error of the actual hardware.

coordinates from a camera placed in the robot eyes, image-based visual servoing can be used to track this object. The difference between the desired object feature (for instance the center of a human face at the center of the robot image) and the actual feature is defined in (4.15), where s^* is the desired feature position, in this case the center of the image frame and $s = [s_x \;\; s_y]^T$ is the vector of image coordinates, in this case the center of the face of the human subject. To stabilize this error to zero, an image-based visual servoing feedback law can be employed:

$$\dot{\phi} = -\lambda \hat{L}^{+} e \tag{4.37}$$

where $\dot{\phi}$ is the combined head/eye angular velocity, λ is control gain, $\hat{L}^{+}$ is the approximation of the pseudo inverse of the image Jacobian.

Since in this section we consider the effect of modelling errors, e.g. the exact kinematics of the robot is uncertain, a direct calculation of the image Jacobian is not be the best option. Furthermore, the direct computation of the pseudo inverse of the Jacobian (4.37) of a redundant head-eye robot does not necessarily maintain the natural head-eye realistic relationships establish in the previous section.

Hence, we compensate the optimal values obtained in the previous section using a correction term proportional to the feature tracking errors, without the use of the Jacobian map. From (4.36), α corresponds to a pan angle of the neck. A new α with an error compensation term will be calculated as:

$$\alpha = \alpha^* + \alpha_e \tag{4.38}$$

where $\dot{\alpha_e}$ is defined as:

$$\dot{\alpha_e} = \frac{\lambda \alpha^* e_x}{\alpha_d |y|} \tag{4.39}$$

where e_x is the tracking error along x-axis of pixel coordinate, α_d is defined in (4.23) and$y$ is a displacement along y - axis. α_d is calculated from inverse kinematics of the head without considering eye motion. Similarly, β corresponds to a tilt angle of the neck. A new β with an error compensation term will be given by:

$$\beta = \beta^* + \beta_e \tag{4.40}$$

$$\dot{\beta_e} = \frac{\lambda \beta^* e_y}{\beta_d |y|} \tag{4.41}$$

where e_y is the tracking error along y - axis of pixel coordinate, β_d is defined in (4.22). For error compensation of the eyes, (4.27) is redefined as:

$$\theta_l = \theta_l + \theta_e \tag{4.42}$$

where $\dot{\theta}_e$ is:

$$\dot{\theta}_e = \frac{\lambda \left(sgn\left(\alpha_d\right) \left| \, \alpha - \alpha^* \right| \right) \, e_x}{\alpha_d |y|} \tag{4.43}$$

Furthermore (4.28) is redefined as:

$$\psi_l = \psi_l + \psi_e \tag{4.44}$$

where $\dot{\psi}_e$ is:

$$\dot{\psi}_e = \frac{\lambda \left(sgn\left(\beta_d\right) \left| \, \beta - \beta^* \right| \right) \, e_y}{\beta_d |y|} \tag{4.45}$$

Equations (4.29) and (4.30) can also be redefined similar to equations (4.42) - (4.45).The choices in equations (4.39) and (4.41) are chosen so that the proportional neck feedback gains vary according to the distance of the object to be tracked. This is done because the closer an object is to the humanoid robot (camera) the more rotation angle is required in order to cover the same vertical/horizontal displacement. Similarly, the terms in equations (4.43) and (4.45) are chosen so that the eye feedback gains also vary according to the distance of an object, however, the gains are scaled down significantly compared to the neck gains.

Proposition 1 (describing Algorithm 1)

Assume that the robotic head with independent neck and eye kinematics described in section 4.4 is used to track a moving object within a confined finite volume away from the robot. Then, the proposed optimization scheme (4.35), in conjunction with the feedback scheme (4.39)-(4.45), leads to an exponentially stable visual tracking error of the target.

Proof: From (4.37), (4.39), (4.41), (4.43) and (4.45) we obtain a visual servoing scheme given by:

$$\dot{\phi} = -\lambda \begin{bmatrix} \frac{\alpha^*}{\alpha_d |y|} & 0 \\ 0 & \frac{\beta^*}{\beta_d |y|} \\ \frac{sgn(\alpha_d) |\alpha - \alpha^*|}{\alpha_d |y|} & 0 \\ 0 & \frac{sgn(\beta_d) |\beta - \beta^*|}{\beta_d |y|} \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix} \tag{4.46}$$

The time derivative of the image servoing error with components e_x, e_y can be written as:

$$\dot{e} = L\,\dot{\phi} \tag{4.47}$$

where L is the image Jacobian of the neck-eye system. Due to the neck-eye mechanical configuration, we will assume the Jacobian has the following structure:

$$L = \begin{bmatrix} a & 0 & b & 0 \\ 0 & d & 0 & e \end{bmatrix} \tag{4.48}$$

where $a > 0$, $b > 0$, $c > 0$ and $d > 0$. According to the mechanical configuration of the hardware and the cameras, the velocity of the feature position s_x in x-pixel coordinate varies according to the velocity of α and θ. In addition, positive $\dot{\alpha}$ and $\dot{\theta}$ result in positive $\dot{s}_x$, negative $\dot{\alpha}$ and $\dot{\theta}$ result in negative $\dot{s}_x$, as guaranteed by assuming that the cameras in the eyes are place in proper orientations. Hence, we can conclude that and are always positive. Similarly, c and d are always positive. If the approximation of the image Jacobian $\hat{L}$ equals the actual image Jacobian L then $L\hat{L}^+$ equals to identity. But in the case the pseudo inverse of the $\hat{L}$ defined as in equation (4.46), the product $L\hat{L}^+$ is no longer identity, and becomes:

$$\lambda L\hat{L}^+ = \frac{\lambda}{|y|}\begin{bmatrix} p & 0 \\ 0 & q \end{bmatrix} \tag{4.49}$$

where

$$p = \frac{\alpha^*}{\alpha_d}a + \frac{sgn\,(\alpha_d)\,|\,\alpha - \alpha^*\,|}{\alpha_d}b \tag{4.50}$$

$$q = \frac{\beta^*}{\beta_d}c + \frac{sgn\,(\beta_d)\,|\,\beta - \beta^*\,|}{\beta_d}d \tag{4.51}$$

From the solution of optimization problem (4.35), Θ^*, we can calculate the optimal eyes angle vector Φ^*. Furthermore, Θ^* has the same sign as Θ_d from equations (4.21), (4.22) and (4.23) otherwise the head would rotate in the opposite direction. Since $|\alpha_d| \geq |\alpha^*|$, $|\beta_d| \geq |\beta^*|$ and $sgn(\alpha_d) = sgn(\alpha^*)$, $sgn(\beta_d) = sgn(\beta^*)$, it follows that $p > 0$, $q > 0$, and therefore:

$$\frac{\lambda}{|y|}\begin{bmatrix} p & 0 \\ 0 & q \end{bmatrix} > 0 \qquad \forall\lambda > 0 \tag{4.52}$$

Finally, we can conclude that the case where $L \neq \hat{L}$ the tracking error is also exponentially stable given that $\hat{L}$ is defined as (4.46).

4.5.2 Optimization using Actor-Critic TD Methods

Instead of solving optimization problem (4.35) directly, here we propose to employ actor-critic methods, such as the Temporal Difference learning method that has independent memory structures of policy component and value function component. The policy component is referred to as "actor" due to the fact that it is used to determine actions [102]. Similarly, the value function component is referred to as "critic" since it makes a judgment of the "actions" made by the "actor". The learning that is done by the actor-critic methods is considered to be on-policy learning. This is referred to the fact that the "critic" critiques the current policy (actions) that is currently being executed by the "actor". The critique is usually defined as a TD error. This value is the given by the critic and propagate all learning for "actor" as well as the "critic". Generally, the "critic" is a state value function. After the "actor" performs the action, the "critic" evaluates whether or not the action performed is better. This is defined as the TD error δ_t:

$$\delta_t = r_{t+1} + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{4.53}$$

where V is the current value function, t is current time step, γ is the discount factor, s_{t+1} is the current state vector, s_t is the previous state vector and r_{t+1} is a reward function defined as follows:

$$r_{t+1} = R\left(s_{t+1}, p(s_t)\right) \tag{4.54}$$

where $p(s)$ is the policy function, $R(.)$ is a function of state vector and policy. The value function is defined as:

$$V_{t+1}(s) = \begin{cases} V_t(s) + \alpha\, \delta_t & \text{if } s = s_t \\ V_t(s) & \text{otherwise} \end{cases} \tag{4.55}$$

where α is the learning rate. The TD error is used to evaluate the action performed by the actor. The critic component is basically an on-policy learning of the value function V. The actor component is updated by:

$$p_{t+1}(s) = \begin{cases} p_t(s,\, a) + \alpha\, \delta_t & \text{if } a = a_t \text{ and } s = s_t \\ p_t(s,\, a) & \text{otherwise} \end{cases} \tag{4.56}$$

where $p_t(s,\, a)$ is the preference for taking action a at time t for state s and δ_t is a TD error defined in (4.53) and a is the action taken by the agent (learner). The equation (4.56) can be rewritten to use eligibility traces as follows:

$$p_{t+1}(s,\, a) = p_t(s,\, a) + \alpha\, \delta_t\, e_t(s,\, a) \tag{4.57}$$

where $e_t(s,\, a)$ denotes the eligibility trace at current time t for state-action pair $(s,\, a)$. The eligibility traces can be generalized as follows:

$$e_t(s,\, a) = \begin{cases} \gamma\, \lambda\, e_{t-1}(s,\, a) + 1 - p_t(s,\, a) & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\, \lambda\, e_{t-1}(s,\, a) & \text{otherwise} \end{cases} \tag{4.58}$$

where λ is the decay factor for exponentially decaying memory trace of the $TD(\lambda)$ algorithm. A value function can be represented by a function approximator. As a result, the value function can be updated by weight updates. In this chapter, we use multi layer feed forward network as function approximators. The value function (4.55) can now be written in terms of neural network representation as follows:

$$V_{\vec{w}_v} = \sum_{i=0}^{M} w_i\, \Phi\left(\sum_{j=0}^{P} w_{ij}\, s_i \right) \tag{4.59}$$

where $\Phi(.)$ is an activation function, w_i are neural network weights, P is the length of state s and M is the number of activation units. Similarly, (4.57) can also be written in a neural network representation form as follows:

$$p_{\vec{w}_p}(s) = \sum_{i=0}^{M} w_i^P \, \Phi \left(\sum_{j=0}^{P} w_{ij}^P \, s_i \right) \tag{4.60}$$

The eligibility traces equation (4.61) can now be updated as follows:

$$e_t(s,\, a) = \lambda \, \gamma \, e_{t-1} + \frac{\partial \, V_{\vec{w}_v}(s)}{\partial \, \vec{w}_v} \tag{4.61}$$

The activation function used in this chapter is the Log sigmoid function

$$\phi(.) = \frac{1}{1 + e^{-x}} \tag{4.62}$$

The weight update equation for the value function is defined as follows:

$$\vec{w}_v = \vec{w}_v + \eta \, \delta_t \, e_t \tag{4.63}$$

where η is the learning rate of the NN training.

Similarly, the weight update for (4.60) is as follows:

$$\vec{w}_p = \vec{w}_p + \eta \, \delta_t \, e_t^P \tag{4.64}$$

where e_t^P is the eligibility traces of the actor structure and is defined as follows:

$$e_t^P(s,\, a) = \lambda \, \gamma \, e_{t-1}^P + \frac{\partial \, p_{\vec{w}_p}(s)}{\partial \, \vec{w}_p} \tag{4.65}$$

where λ is the decay factor and γ is the discount factor is defined in (4.53).

Theoretical proof for convergence of the $TD(\lambda)$ with function approximators is presented in [103]. Tsitsiklis and Van Roy [103], proves that the discrete policy evaluation using function approximators converges when learning is performed along the trajectories of $TD(\lambda)$. [103] also proved that the error of the value function is bounded by

$$E \leq \frac{1 - \lambda \, \gamma}{1 - \gamma} E^* \tag{4.66}$$

where E^* is the optimal quadratic error of the function approximator. According to (4.57), the more λ is close to 1 the more accurate the approximation will be. The important part of this reinforcement learning scheme is how to select the reward function properly so that all the important aspects of realistic head-eye motion distribution are taken into account. The reward function is a function of states and actions as defined in (4.54). We define the state vector s as follows:

$$s = [\Phi \;\; \Theta \;\; \dot{\Phi} \;\; \dot{\Theta} \;\; y \;\; \chi]^T \tag{4.67}$$

where $\dot{\Phi}$ is the head angular velocity and $\Phi = [\alpha \;\; \beta \;\; \gamma]^T$, Θ is the eye angular displacement and $\Theta = [\theta_l \;\; \psi_l \;\; \theta_r \;\; \psi_r]$, y is the distance of an object to be tracked from the eyes and χ is the image projection in pixel coordinate of the object to be tracked as seen through the eyes cameras.

We can see that information such as the distance of an object and image projection of an object to be tracked is important factors to determine the outcome behavior of the neck-eye motion distribution of the humanoid robot head. In the case of human, this information comes to us naturally without us making any effort. The estimation of the distance of the object to be tracked can be done due to the stereovision structure of the eyes. The image projection of the object is obtained through the processing of the information from the retinas by the brain. This provices rough estimates of how far the object is off from the center of the eyesight. For the humanoid robot, the distance information described above can also be obtained, and it is equally as important that the information mentioned above is available as it is a prerequisite for realistic head-eye motion distribution. Assuming that y and χ are available, and c_1, c_2, c_3 and c_4 are weighting factors, we can form a reward function to take into account the following behaviors:

$$r_1 = -c_1 \left(\chi_x^2 + \chi_y^2 \right) \tag{4.68}$$

to reward the action that corresponds to bringing the object closer to the center of eyesight.

$$r_2 = -c_2 \left(\dot{\Phi}^T \dot{\Phi} \right) \tag{4.69}$$

to reward the action that corresponds to moving the neck slower than the eyes, since the neck inertia is a lot larger, thus avoiding turning the head abruptly.

$$r_3 = -c_3 \left(\Theta^T \Theta \right) \tag{4.70}$$

to penalize the action that corresponds to keeping the eye off center for a period of time, as we want to gradually turn the head toward the object. Finally,

$$r_4 = -c_4 \begin{cases} \Phi^T \Phi & \text{if } y \text{ is large} \\ \Theta^T \Theta & \text{if } y \text{ is small} \end{cases} \tag{4.71}$$

to reward the action according to the distance of an object as we want to only rotate the eyes when the object is too close to the eyes (to imitate the saccadic eye movement). The total reward function:

$$R = r_1 + r_2 + r_3 + r_4 \tag{4.72}$$

The following Algorithm summarizes the reinforcement learning approach adapted for the neck-eye motion coordination:

Description of Algorithm 2:

Step 1: Learning of optimal neck-eye motion using $TD(\lambda)$

1. Define the state variables vector as in equation (4.68)
2. Do the value function iteration by updating the TD error defined in equation (4.53) where the reward function is as described in equation (4.72)
3. Update NN weights of value function (4.59), as well as policy function (4.60) using (4.61) - (4.65)

4. Repeat learning episodes until optimal value function V^* is reached
5. Stop updating the NN weights

Step 2: Control using visual feedback with optimal policy

1. The optimal policy function obtained from the latest update of step 2 and 3 can now be used for control of the robot system as in equation (4.60)

Through the proposed optimization scheme we can accomplish saccadic motion without requiring the hardware to incorporate designs taken from eyes kinesiology studies [81, 8]. Unlike the optimization approach of directly solving equation (4.35), this Actor-Critic approach does not need a separate module for tracking error correction due to unmodeled kinematics of the head mechanisms since the tracking error is already taken into account through the use of reward function (4.68). With this proposed approach, we are not restricted to any particular hardware and/or design as long as we have access to the required information as mentioned in (4.67). In summary, this proposed Actor-Critic reinforcement learning for neck-eye motion distribution of humanoid robot method is platform independent. In the next chapter, we present simulation results as well as experiment results to show the effectiveness of the proposed method.

4.6 Simulation and Experiment Results

4.6.1 EKF Pose Estimation Results

The performance of the proposed estimation algorithm is demonstrated in this section through several experiments we conducted. The first set of experiments shows the estimation performance and accuracy when measurement noise is not present (e.g. by using a software generated trajectory). The second set of experiments investigates the accuracy of the scheme when measurement noise is present due to the image-

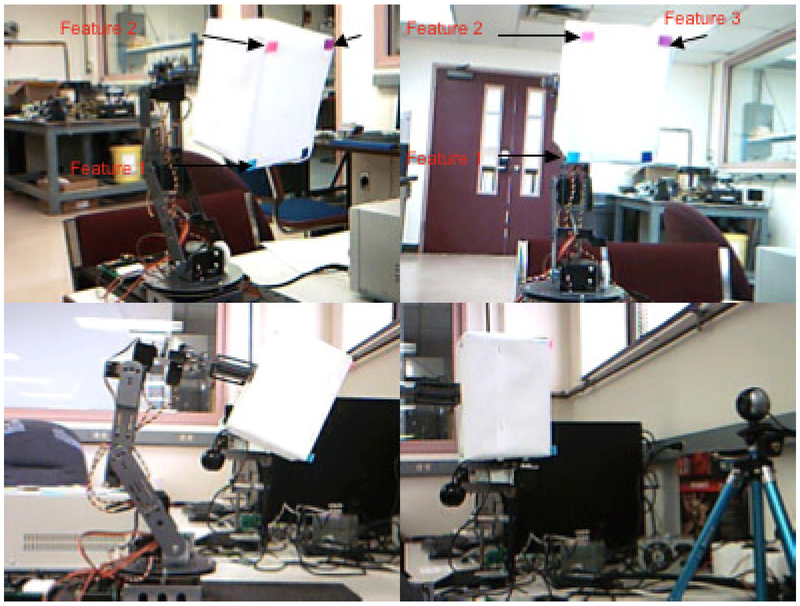


Figure 4.8. Experiment setup showing the robot arm holding an object facing a motorized pan/tilt camera. The object used in the experiment is a white box with colored markers on it. The figure shows multiple views of the setup and the features that are tracked.

processing algorithm. The third experiment involves the tracking performance of the visual servoing scheme, and the final experiments tracks the 6D pose of a human head facing the camera.

In the experiments conducted, the camera parameters values were obtained by calibration, resulting in $f_x = 493.5$, $f_y = 459.4$, $c_0 = 160$, $r_0 = 120$ and $T = 0.04$. The sampling time T is based on the measured frames per second rate, which was 26 fps. Fig. 4.8 shows the object that was used for pose estimation and tracking, while undergoing different sets of motion. Three non-collinear feature point (P_1, P_2, P_3) were used and their nominal coordinates are given (in inches) as $P_1 = [-3,\, 2.1,\, 0]^T$, $P_2 = [-3,\, -2.1,\, 0]^T$ and $P_3 = [3,\, -2.1,\, 0]^T$.

4.6.1.1 Generated Trajectory Inputs

In this experiment, a set of feature points were generated based on the actual model of the object to mimic the feature points with no measurement noise. The trajectory consists of motion in all direction and orientation. The initial pose estimates are $[x,\ y,\ z] = [0,\ 0,\ 20]^T$.

The estimation results are shown in Fig. 4.9 - 4.11. From the estimated pixel coordinates $[c\ r]^T$ for each feature point, an error function is defined as:

$$e = \left(c_m^2 + r_m^2\right)^{1/2} - \left(c^2 + r^2\right)^{1/2} \tag{4.73}$$

where $[c_m\ r_m]^T$ are obtained from the measurement set.

Fig. 4.9 plots the generated trajectory that is used as an input to the EKF pose estimation filter. The trajectory is composed of simultaneous sinusoidal rotation and translation with different frequencies in all axes. Fig. 4.10 plots the position and orientation estimates obtained using EKF pose estimation approach. Fig. experimentposeestimationekfnomeasurementnoisenormailizederror:1 plots the normalized estimation error. According to the plots, the estimation error is low since it is assumed that the generated trajectories contain neither measurement noise nor feature extraction error.

4.6.1.2 Actual Trajectory Inputs

In the second set of experiments, the EKF pose estimation algorithm is used to estimate position and orientation of the actual object, using the experiment setup shown in Fig. 4.8. Fig. 4.12 plots the feature trajectories in x pixel coordinate. Fig. 4.13 plots the feature trajectories in y pixel coordinate. The object is attached to the tip of the robot arm as shown in Fig. ekfposeexperimentsetuprobotbox:1 and in this experiment we only rotate the object along z-axis (roll), the trajectory is

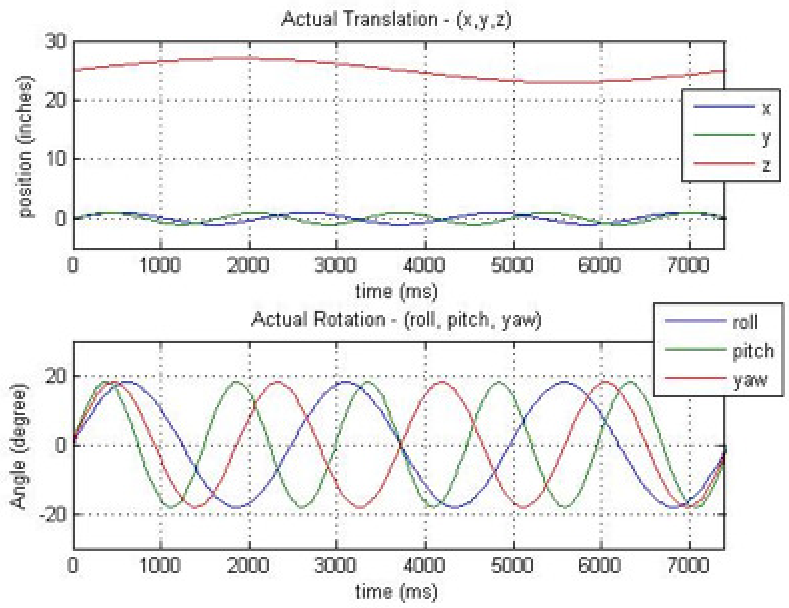


Figure 4.9. Pose Estimation using EKF where no measurement noise is presented - generated trajectory of the object.

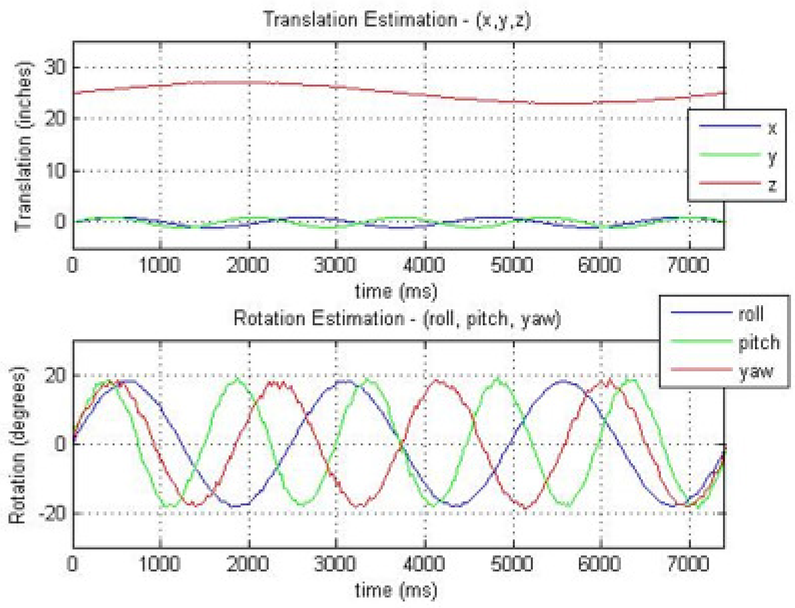


Figure 4.10. Pose Estimation using EKF where no measurement noise is presented - object position and orientation estimates.

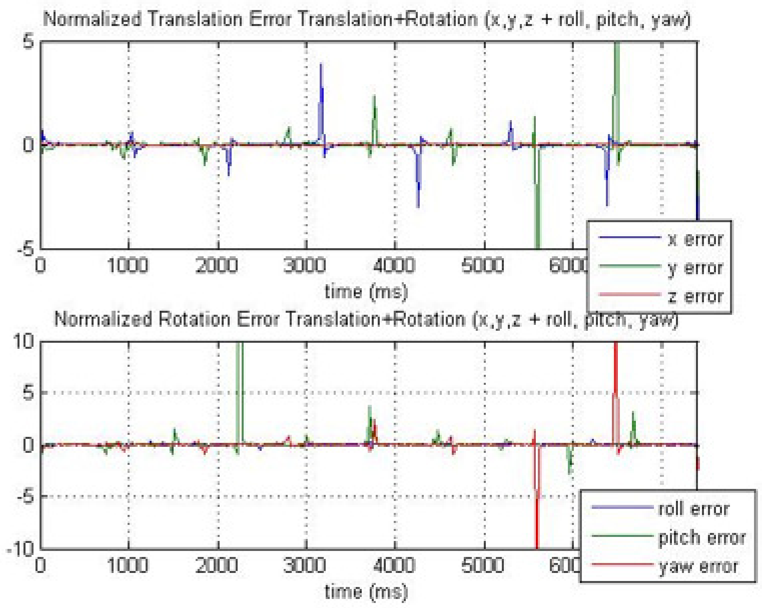


Figure 4.11. Pose Estimation using EKF where no measurement noise is presented - normalized estimation error.

also sinusoidal. We extract 3 features recursively from the object and use them as measurements. We can see that there is noise and false detection present in the extracted features. Fig. 4.14 plots the position and orientation estimates obtained using EKF pose estimation approach. Fig. 4.15 plots the error function defined in 4.73. The error is the difference in Euclidean distance between extracted features and estimated features. We can see that the errors stay within a -20 to 20 pixel bound, and that the feature extraction noise is eliminated by the Kalman Filter.

4.6.2 EKF Pose Estimation and Tracking Results

In this experiment, the camera tracks the object through a motorized pan and tilt mechanism. Tracking was performed using the position vector estimates obtained from the EKF. Since the dynamics of the two servomotors were not known, a system

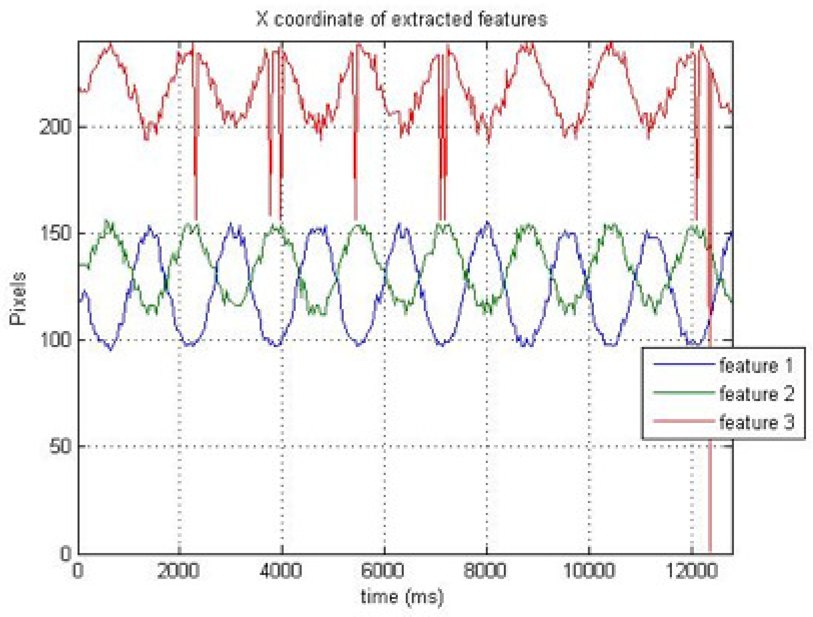


Figure 4.12. Pose Estimation using EKF where measurement noise is prominent - feature trajectories in x pixel coordinate.

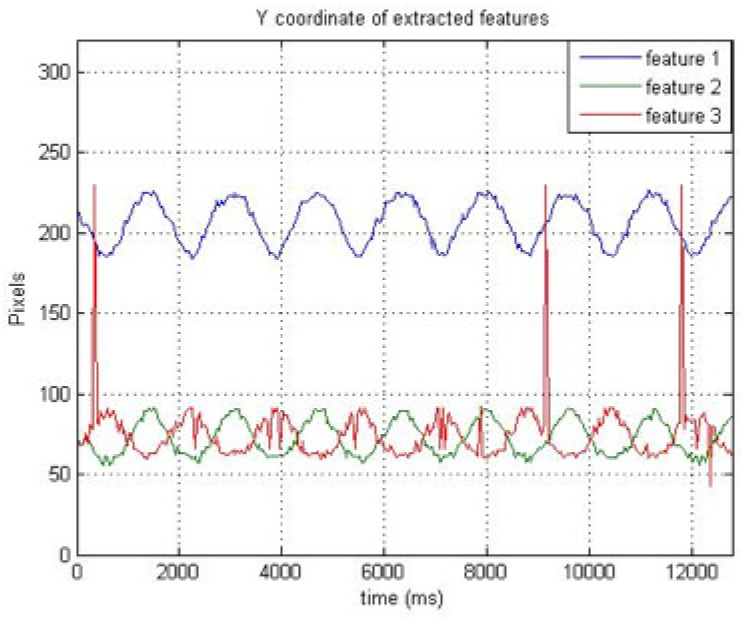


Figure 4.13. Pose Estimation using EKF where measurement noise is prominent - feature trajectories in y pixel coordinate.

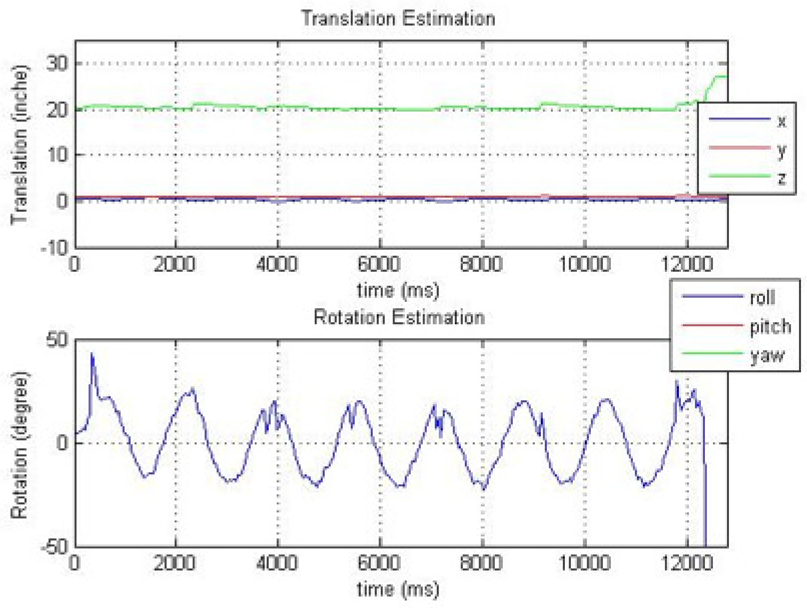


Figure 4.14. Pose Estimation using EKF where measurement noise is prominent - object position and orientation estimates.

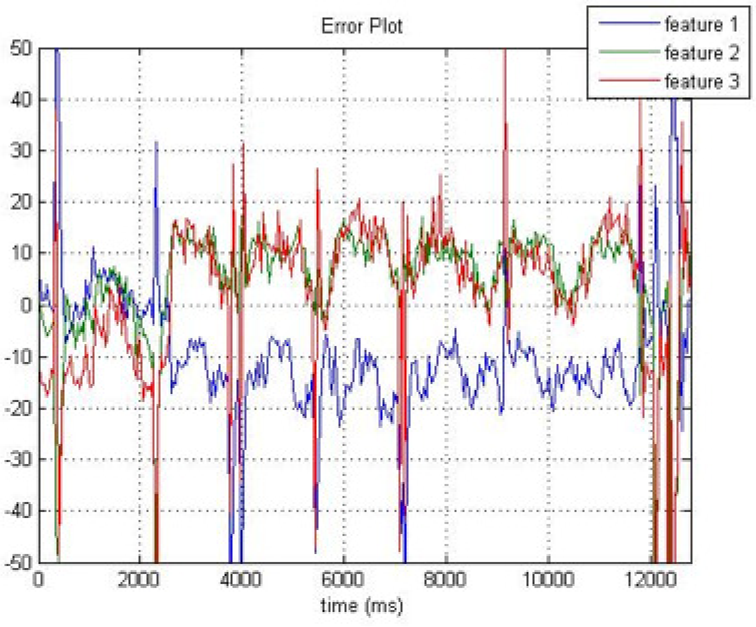


Figure 4.15. Pose Estimation using EKF where measurement noise is prominent - error function defined in (4.73).

identification algorithm using an ARX model was fitted to the data. The motors were modeled with the discrete-time transfer function given as:

$$G(z) = \frac{335.3z - 325.3}{z^2 - 0.5761z - 0.4162} \tag{4.74}$$

In order to achieve efficient performance of the motors, a tuned PID controller was designed using the Ziegler-Nichols tuner in MATLAB. The resulting controller gains were: $K_P = 7.98$, $K_I = 4.53$ and $K_D = 2.88$. Fig. 4.16 plots the convergence of the x-pixel (solid line) and y-pixel (dashed line) with untuned and tuned PID gains for the desired pixel positions set at $[120 \;\; 90]^T$. The performance of the motors with tuned PID gains is apparent, exhibiting a fast settling time of 20 time steps (at 26 fps). These controller gain values were then used for tracking of the object whose pose is estimated by the filter. Fig. 4.17 and 4.18 plot the position and orientation estimates during tracking of the object by the camera. It can be seen that by using tracking in combination with EKF pose estimation, the x, y translation estimates are close to zero. On the other hand, the depth and the orientation estimates of the object are still present, however the camera does not have enough degrees of freedom to track them. However, by using visual servoing, the object is always in the field of view.

4.6.3 Head Pose Estimation Results

In this experiment, we use an actual human face to test the pose estimation algorithm. A camera acquires the video stream of the human's face in motion. A simple routine based on Lucas-Kanade optical flow tracking [47] is implemented to keep track of 4 facial features points; right eye, left eye, nose and mouth. The facial features information in pixel coordinate obtained from Lucas-Kanade tracking is fed into the EKF pose estimation algorithm. The resulting 6D head pose estimates

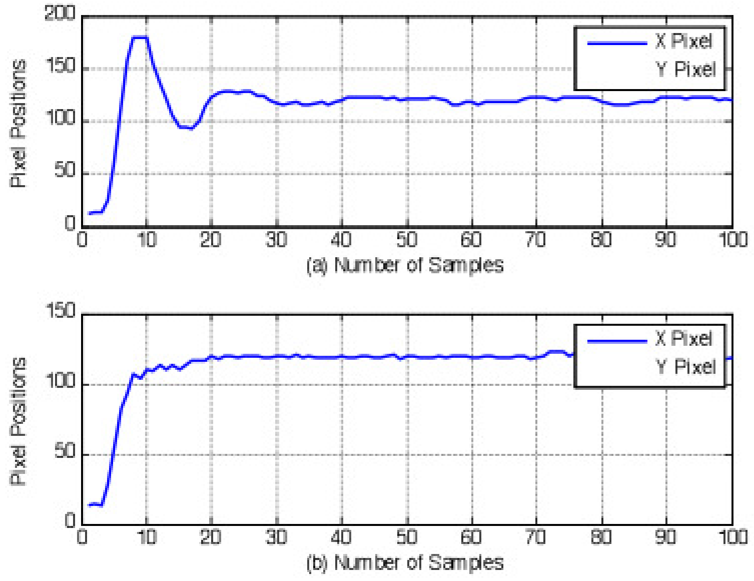


Figure 4.16. Performance of the pan/tilt tracking system, individual x and y pixel position response, (system sampling time was 0.04 seconds) (a) Untuned PID system pixel position response, (b) Tuned PID system pixel position response, starting at an initial condition with [15, 40] pixel coordinate.

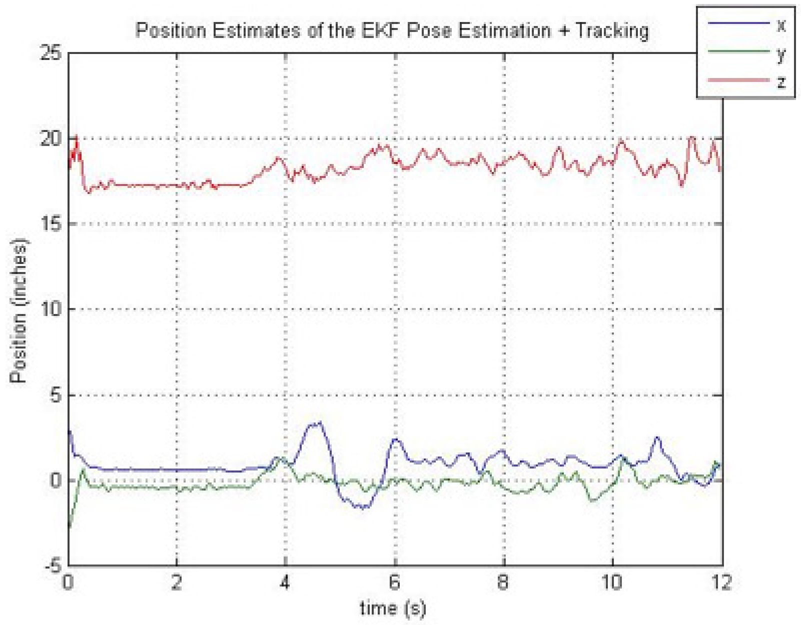


Figure 4.17. Plot of estimated position vector with tracking.

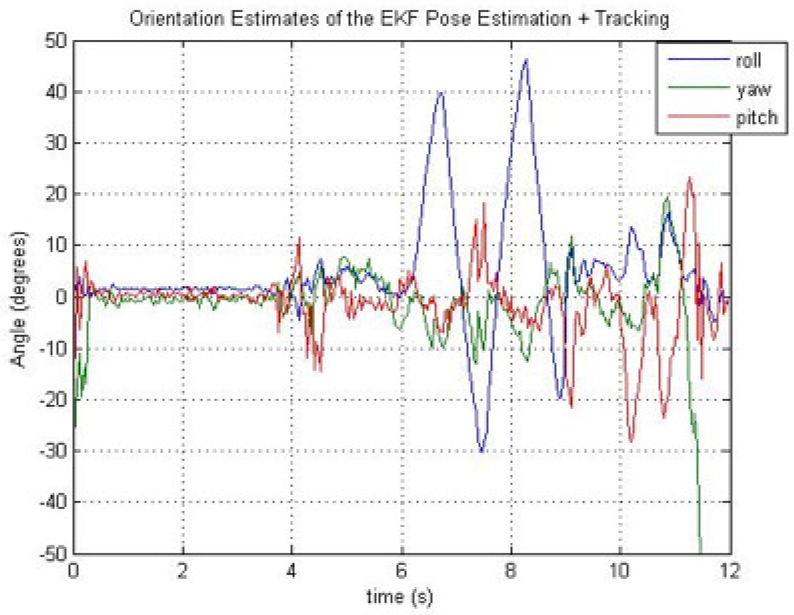


Figure 4.18. Plot of estimated orientation vector with tracking.

(position and orientation) are then passed into a 3D visualization software (Blender 3D), which updates the pose of a skull 3D model. Therefore, the 3D model “mimics” the pose of the human, at a rate of 20Hz using a USB camera, and a Visual Studio.Net application that passes the pose estimates to Blender. Fig. 4.19 shows screen captures of the head tracker application and Fig. 4.20 and 4.21 shows plots of the resulting position and orientation estimates. As we can see from Fig. 4.19 - 4.21, the estimates are close to the actual head pose.

4.6.4 Head Eye Motion Distribution Results

In this subsection, we first present details on implementation of the Algorithm 2 to learn the optimal control with a simulated robot head model. We then present experimental results of using proposed methods Algorithm 1 and Algorithm 2 with a simulated robot head model. Last, we present the results of implementing the

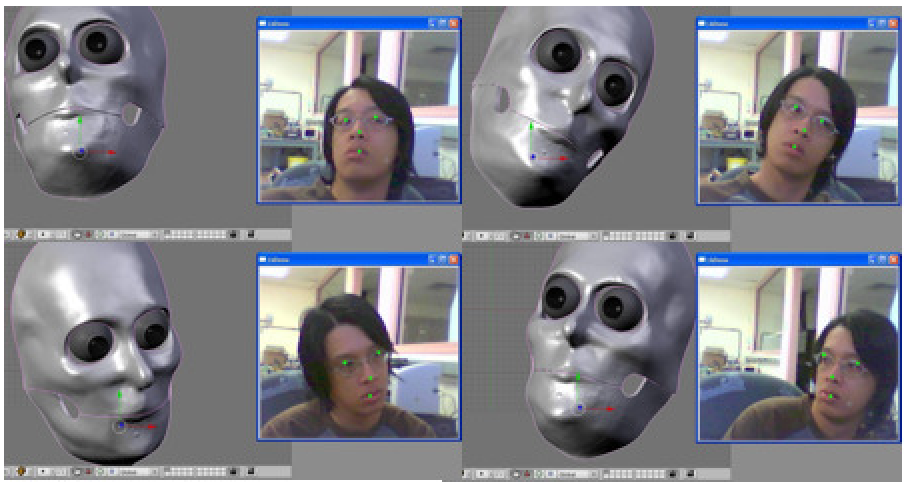


Figure 4.19. Screen captures of the input head pose video and 3D visualization of the estimated position and orientation.

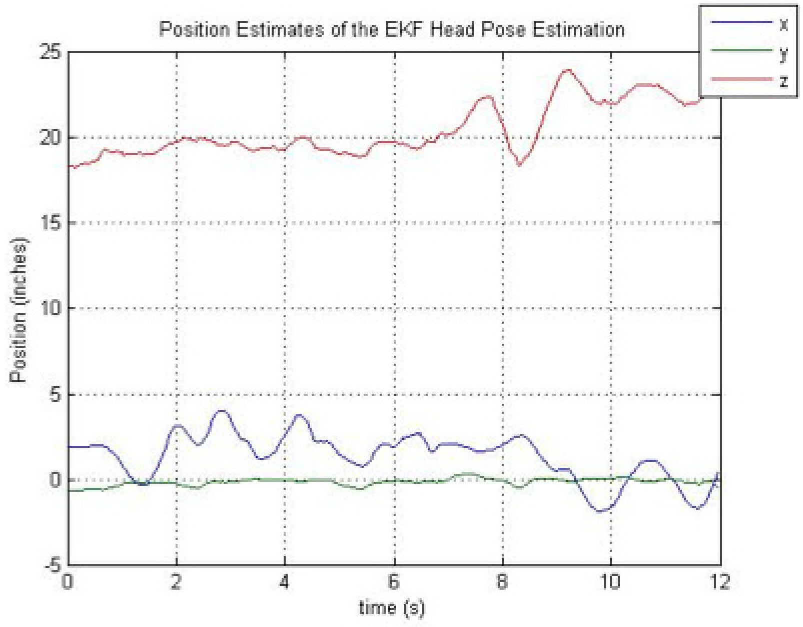


Figure 4.20. Plot of position estimates (in inches).

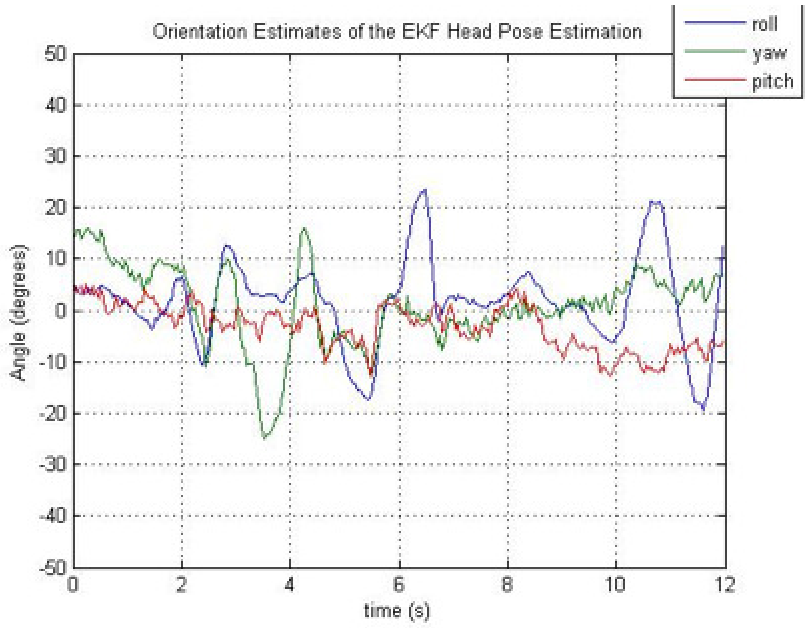


Figure 4.21. Plot of orientation estimates (in degrees) obtained from the EKF filter.

proposed Algorithm 1 on the LILLY humanoid robot head. Appendix B shows the implementation diagrams of both Algorithm 1 and Algorithm 2.

4.6.4.1 Simulation of Head-Eye Robot Structure

The kinematics of the simulated robot structure is defined as in [76]. The measurements of the simulated robot structure are taken from the actual LILLY humanoid robot head. However, the inertia, mass, friction and stiffness are estimated from our experiments. In the simulation, for simplicity we only derive a model of the robot with one eye (left) since we obtain the estimate of the object distance through other means (not through the stereo vision structure of the head). The robot head dynamics equation is defined as follows:

$$M\ddot{q} + V(q, \dot{q}) + G(q) = \tau \tag{4.75}$$

where M is the mass matrix of the robot head, $V(q,\,\dot{q})$ is the coriolis term, $G(q)$ is the gravity term and τ is the actuator torque. In this simulation, consists of angular positions of the neck around z-axis (pan) and around x-axis (tilt), angular positions of the left eye around z-axis (pan) and around x-axis (tilt). A simple dynamics equation is defined as follows:

$$M = \begin{bmatrix} I_1 & 0 & 0 & 0 \\ 0 & I_2^2 m_2 & 0 & 0 \\ 0 & 0 & I_{eyeX} & 0 \\ 0 & 0 & 0 & I_{eyeY} \end{bmatrix} \tag{4.76}$$

where $I_1 = 2$, $I_2 = 0.15$, $I_{eyeX} = 0.2$ and $I_{eyeY} = 0.2$.

$N(q,\,\dot{q}) = V(q,\,\dot{q}) + G(q)$ is defined as follows:

$$N(q,\,\dot{q}) = \begin{bmatrix} I_{neckZ}\dot{q}_1 & (m_2 + m_{eye})g\,l_2\cos(q_2) + I_{neckX}\dot{q}_2 & 0 & 0 \end{bmatrix}^T \tag{4.77}$$

where $I_{neckZ} = 0.2$, $I_{neckX} = 0.5$, $m_2 = 4$, $l_2 = 0.15$ and g is the acceleration due to gravity which equals 9.81.

The dynamical model of the robot head is used with both Algorithm 1 and Algorithm 2 in the next subsection to compare the performance with the real human head.

For the optimization approach in Algorithm 1, equation (4.35) is solved in real time to find the optimal trajectory due to the cost function in equation (4.34). For the implementation of Algorithm 2, we use $\lambda = 0.7$, $\gamma = 0.2$ and $\eta = 0.01$ for the $TD(\lambda)$ algorithm. For value function iteration (4.59), we use feed forward neural network with 2 hidden layers and 10 neurons on each layer. And we have the same structure for policy iteration (4.60). For reward function, we use $c_1 = 1.0$, $c_2 = 0.7$, $c_3 = 0.5$ and $c_4 = 0.3$. Sampling period used in implementation of both Algorithm 1 and Algorithm 2 is $T = 0.05$ sec.

After 100 episodes of 10 seconds learning (around 25,000 iterations) with random initial robot states, the value function converges. With optimal value function, we also arrive at the optimal policy for control. The optimal policy is represented by (4.60) where the NN weight vector is now optimal w_p^*.

4.6.4.2 Experiments on Comparison between Optimization Method and Actor-Critic TD Method

In this subsection, we compare the result of using Algorithm 1 and Algorithm 2 with real human for object tracking task. We set up a system where we have a simulated object moving on a computer screen. The human test subject sits in front of the camera and we capture the motion of the human head movement. Simultaneously, the moving object trajectory is used as input for simulated robot system. In other words, the simulated robot is also implemented in real time for implementation using both Algorithm 1 and Algorithm 2. This is as if the simulated robot and the human are performing a tracking of objects that have the exact same motion profile. For visualization purpose, we output the head-eye motion of the simulated robot to the computer screen through 3D Blender program so that we can compare the motion of the simulated robot head along side with the motion of the human head in real time.

We conduct 3 sets of experiment for different speeds of an object to be tracked. We use slow, medium and fast motions by having the moving object trajectory according to sinusoidal wave with different frequencies. The object to be tracked is simulated using the following:

$$\begin{aligned} x(t) &= 0.127\sin(2\pi f_1 t) \\ y(t) &= 0.7662 \\ z(t) &= 0.127\sin(2\pi f_2 t) \end{aligned} \tag{4.78}$$

where f_1 and f_2 dictate how fast the simulated object moves. For slow speed, $f_1 = 0.5$ and $f_2 = 0.3$. For medium speed, $f_1 = 1.0$ and $f_2 = 0.8$. Finally, for fast speed, $f_1 = 2.0$ and $f_2 = 2.8$. Note that the unit of simulated object position is defined in meters.

Since the information of the pupils location are available only in 2D plane due to the projection of the camera, we compare the head-eye motion by looking at the 2D projection of the left and right pupil location on to the camera plane for both real human and the simulated robot. Fig. 4.22 shows the comparison between human eye trajectory and the simulated robot eyes trajectory using optimization method for slow moving object. Fig. 4.23 shows the eyes trajectory error between the human eyes and simulated robot eyes trajectory using Algorithm 1 for slow moving object. Similarly, Fig. 4.24 and 4.25 show result using Algorithm 2. Fig. 4.26 - 4.29 show results for medium speed moving object. Fig. 4.30 - 4.33 show results for fast moving object. Fig. 4.34 shows snapshots of the experiment done with slow speed moving object. Table 4.1 lists the root mean square error of the implementations of Algorithm 1 and 2 for three different speeds.

Table 4.1. RMSE error of Algorithm 1 and 2

Speed	Algorithm 1	Algorithm 2
Slow	Left Eye: 3.25314 Right Eye: 3.35942	Left Eye: 3.99042 Right Eye: 3.97044
Medium	Left Eye: 3.32233 Right Eye: 3.20877	Left Eye: 3.32 Right Eye: 3.20743
Fast	Left Eye: 1.93574 Right Eye: 2.09657	Left Eye: 1.81412 Right Eye: 2.36035

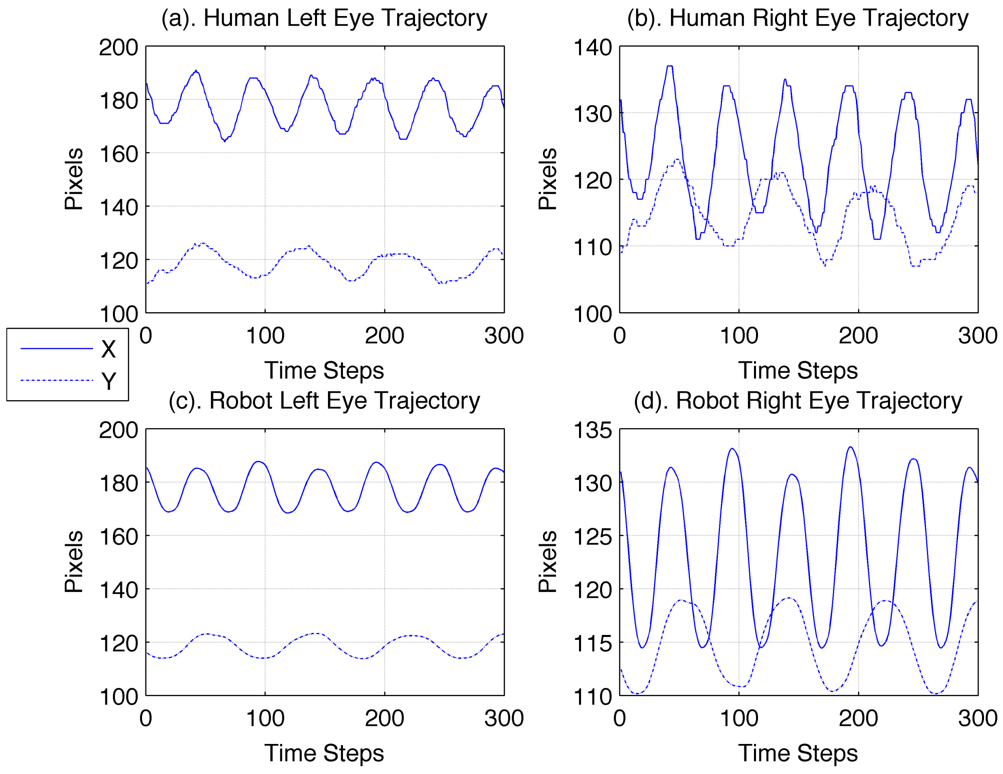


Figure 4.22. Experiment 1 using Algorithm 1 for slow moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

4.6.4.3 Experiments on LILLY Humanoid Robot Head

We now consider the effect of modelling errors, by conducting experiments with the humanoid robot head hardware. In this experiment the LILLY humanoid robot actor uses the visual feedback compensation with equations (4.39) - (4.45) in order to improve the tracking accuracy. This is done through computer programming by discretizing (4.39), (4.41), (4.43) and (4.45). Fig. 4.35 shows snapshots of the output and human face tracking result. We can see from the screenshots that the tracking error is minimized since the human face being tracked is always in at the center of the image frame.

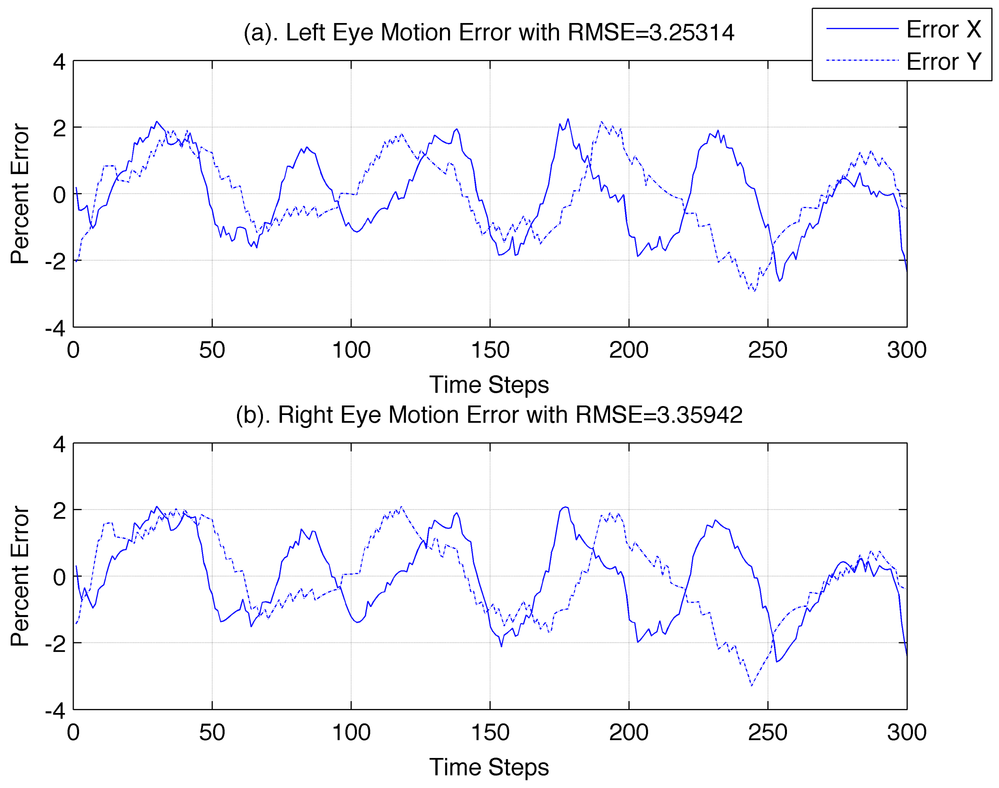


Figure 4.23. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 1 for tracking of slow moving object, the root mean square error of left eye trajectory is 3.253 and the root mean square error of right eye trajectory is 3.359.

Fig. 4.36 shows the head motion profile that is obtained for this particular experiment. Fig. 4.37 shows the distribution of motion profile between the neck and the eyes. As we can see from Fig. 4.38 that with error correction scheme outlined in section 4.5 the tracking error is kept within certain bound no matter how far the object is from the eyes of LILLY.

4.6.4.4 Experiments on Zeno Humanoid Robot Head

Zeno head is about 1/4 of a size of an actual human head. The appearance of the Zeno is based on a fictitious character developed by Hanson Robotics. Zeno looks

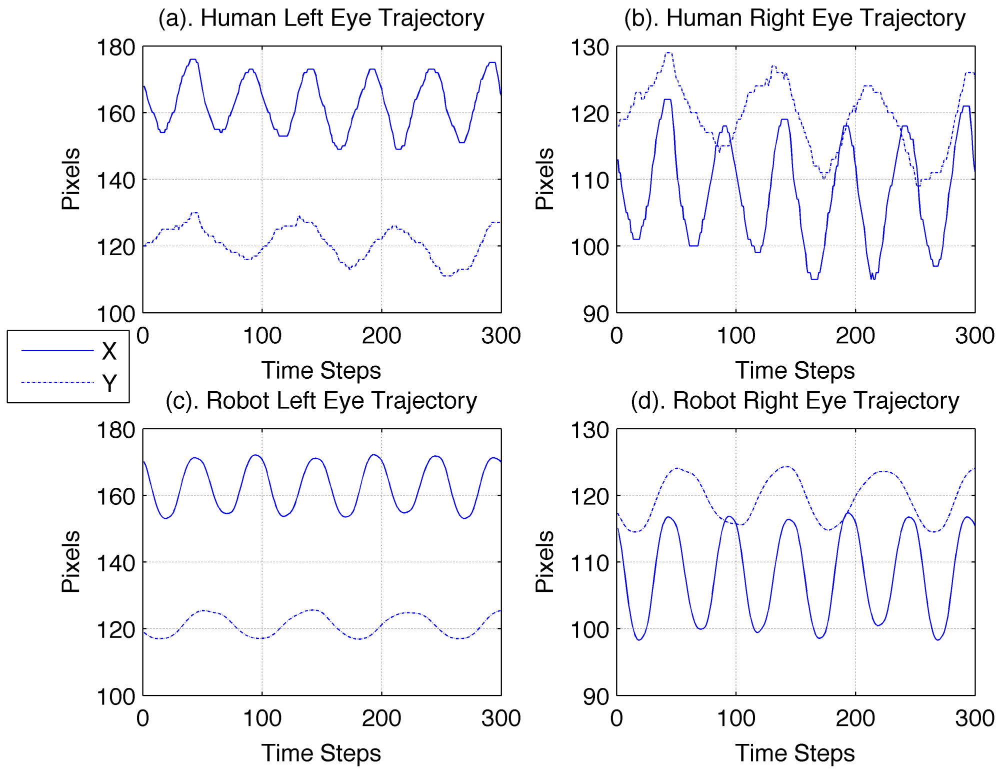


Figure 4.24. Experiment 1 using Algorithm 2 for slow moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

like a 4-7 year old child. The unique features of the Zeno include: life-like skin and the ability to generate various facial expressions. The skin is made of Frubber material, which is the intellectual property of Hanson Robotics. The Zeno head appearance is very realistic thanks to the Frubber material. The Zeno head has 3 degrees of freedom at the neck joint; it is capable of panning, tilting the head back and forth as well as left and right. It also has 2 degrees of freedom in each eye (pan and tilt). It is powered by 9 servomotors. 4 of the servos are used for generating facial expressions. The rest of the servomotors are used for control of the neck/eye motion. The Zeno head has one color video camera in the right eye, which give us the video stream

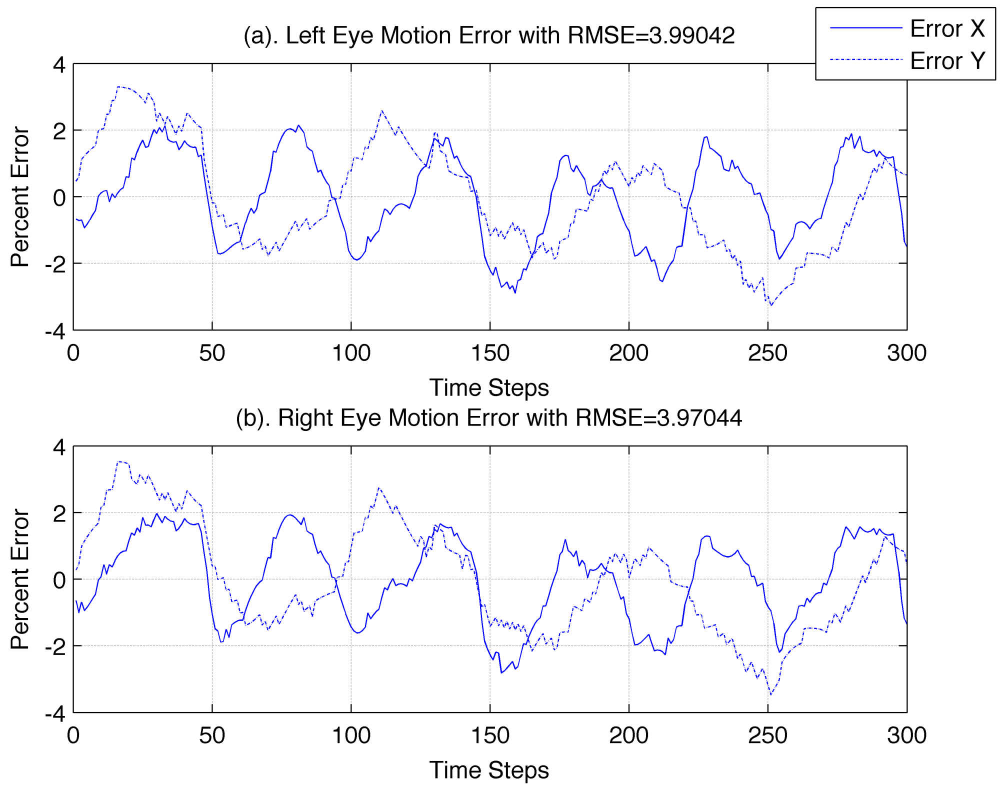


Figure 4.25. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 2 for tracking of slow moving object, the root mean square error of left eye trajectory is 3.99 and the root mean square error of right eye trajectory is 3.97.

of the view that is seen by the Zeno. Zeno head is a good platform that allows us to have an interactive / realistic humanoid robot in terms of appearance as well as motion. Fig. 4.2 and 4.39 show the Zeno head doing a human tracking with neutral facial expression.

Currently, we have successfully implemented realistic human tracking on the Zeno head (both Algorithm1 and Algorithm2). As a basis for conversational robot application, the robot head should be able to keep an eye contact with the human subject (tracking) in real-time in a realistic manner. Fig. 4.40 shows the snapshots of the Zeno head doing tracking of a person through implementation of Algorithm

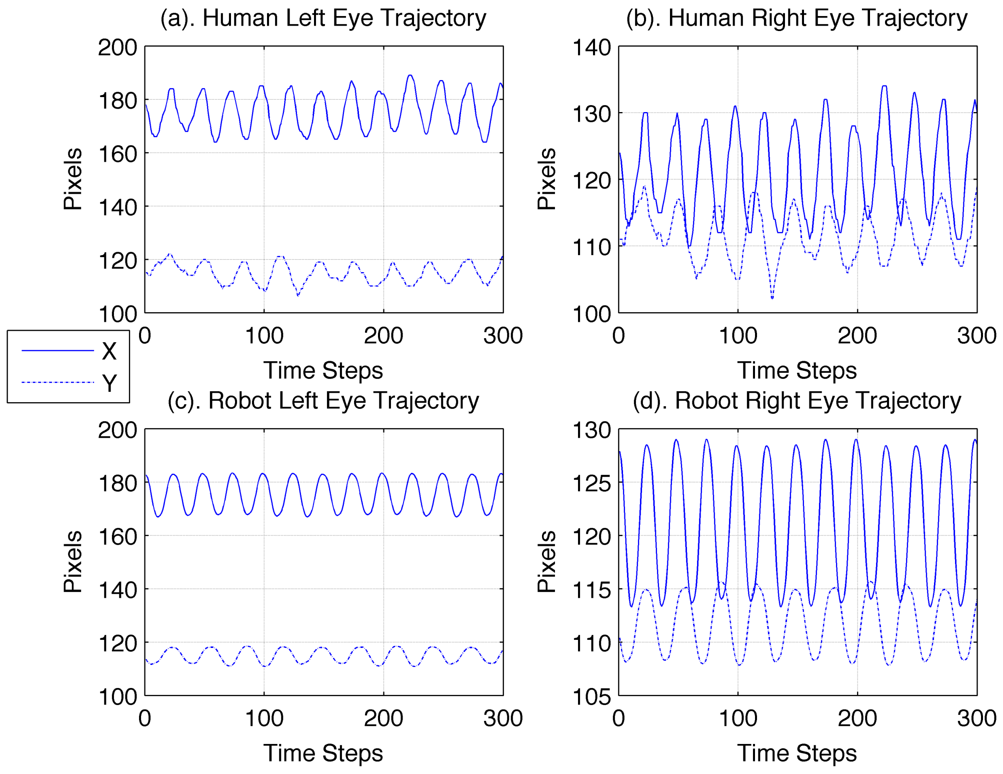


Figure 4.26. Experiment 2 using Algorithm 1 for medium speed moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

1. Fig. 4.41 shows the snapshots of the Zeno head doing tracking of a person using Algorithm 2. From the experiments, results show that both Algorithm 1 and 2 yield realistic head-eye motion of humanoid robotic actors for human tracking application. In addition, the proposed Algorithm 1 and 2 are independent of platform structure.

4.7 Summary and Future Work

4.7.1 Summary

An efficient object tracker using pose estimation and visual servoing was proposed and implemented using a EKF and then the estimates are tracked using a

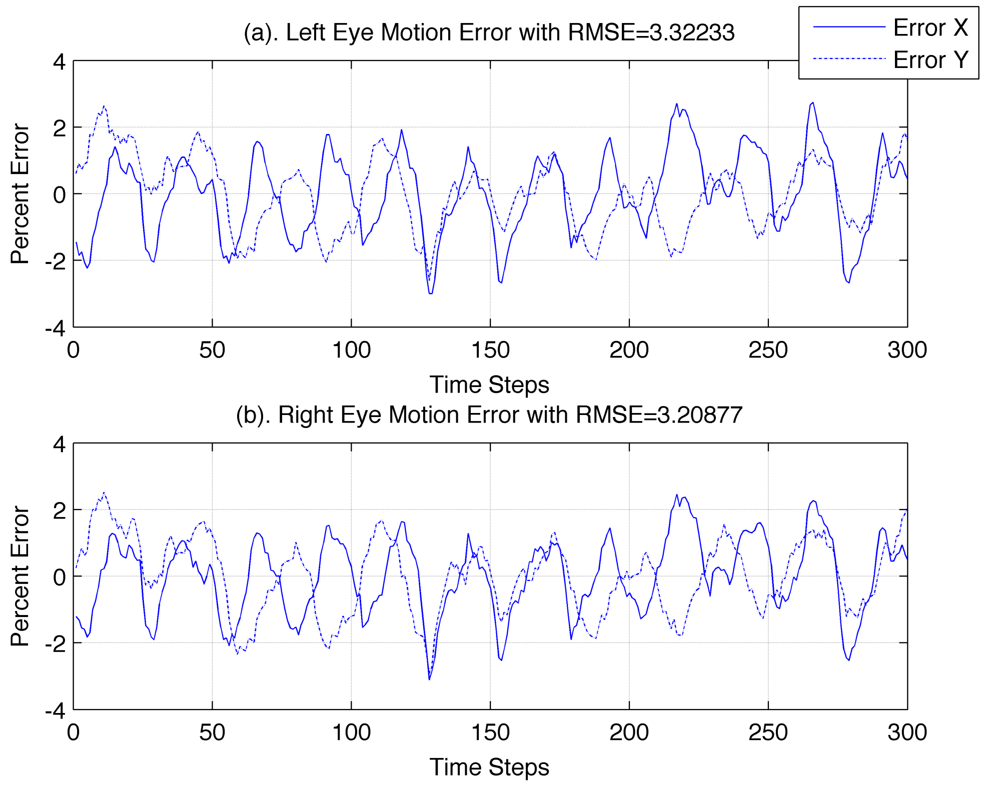


Figure 4.27. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 1 for tracking of medium speed moving object, the root mean square error of left eye trajectory is 3.322 and the root mean square error of right eye trajectory is 3.208.

tuned PID visual servoing controller. Experimental results clearly show high tracking accuracy, robustness of the EKF pose estimation filter approach. To confirm our findings, we presented experimental results of 6D pose tracking with a motorized pan and tilt camera. An efficient and robust human tracker for a humanoid robot was implemented and experimentally evaluated. We employ an optimization approach for achieving nominal head-eye pose angles, and further compensate the kinematic errors using visual feedback. Experimental results show that the visual feedback scheme can significantly improve the tracking accuracy while still maintaining the realism of

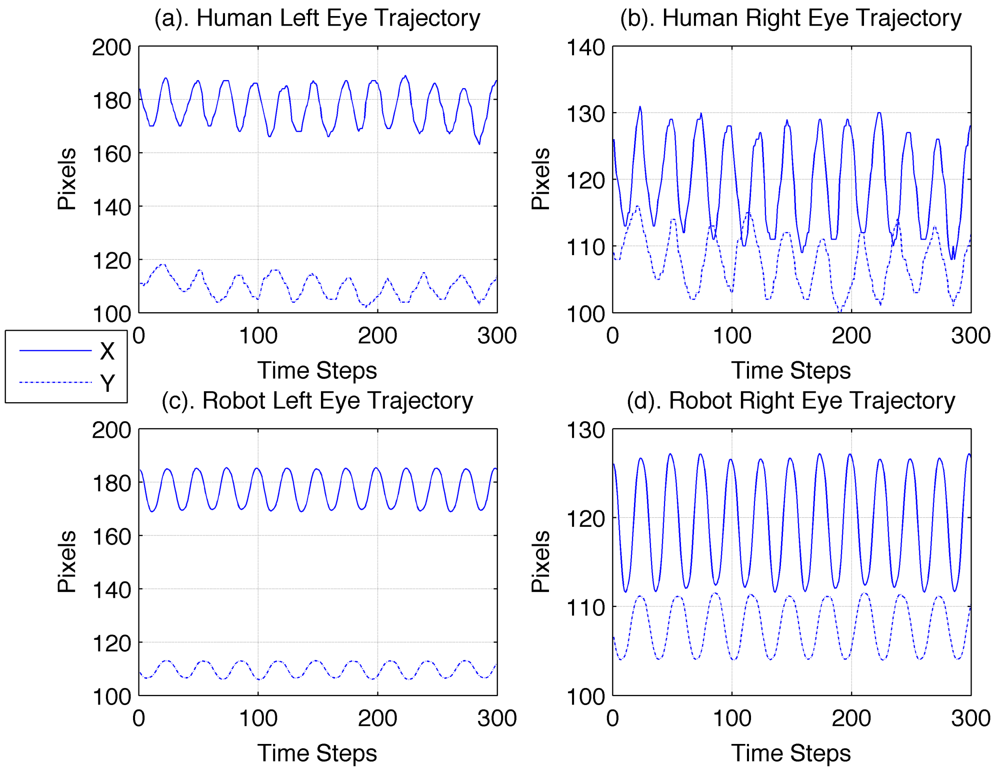


Figure 4.28. Experiment 2 using Algorithm 2 for medium speed moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

head-eye motion coordination that is generated from optimizing the objective function.

In addition, an efficient and robust human tracker for a humanoid robot was implemented and experimentally evaluated. We employ an optimization/learning approach for achieving nominal head-eye pose angles, and further compensate the kinematic errors using visual feedback. Experimental results show that the visual feedback scheme can significantly improve the tracking accuracy while still maintaining the realism of head-eye motion coordination that is generated from optimizing a suitable objective or reward function. Experimental results show that with the

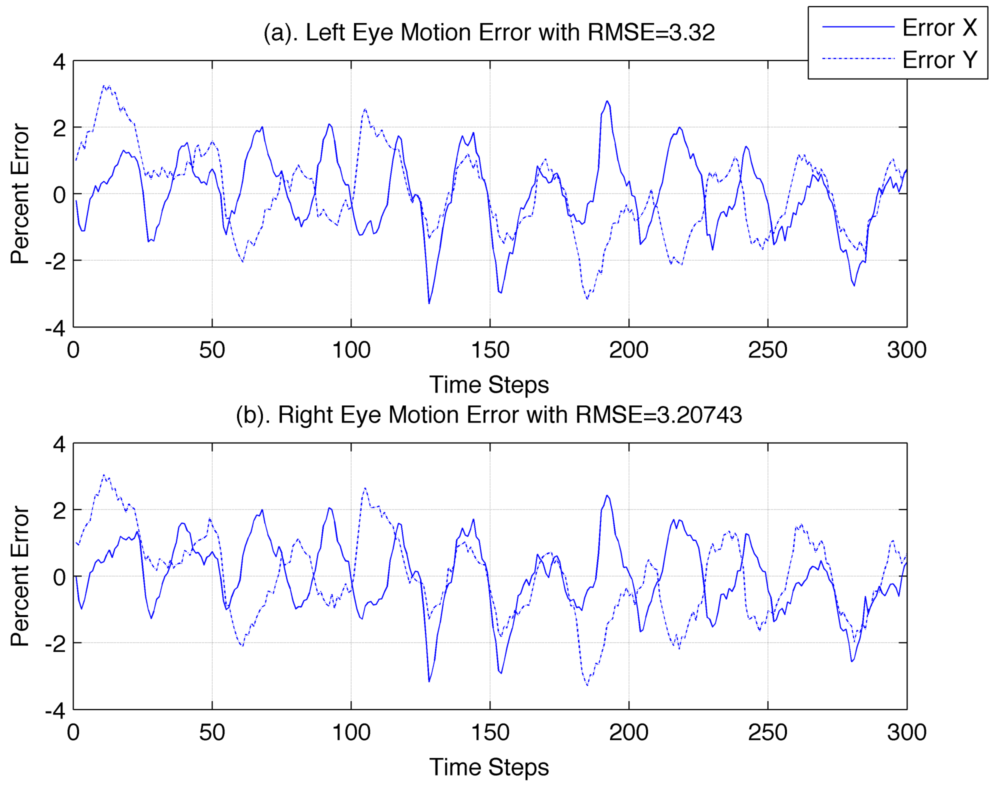


Figure 4.29. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 2 for tracking of medium speed moving object, the root mean square error of left eye trajectory is 3.32 and the root mean square error of right eye trajectory is 3.207.

proposed cost/reward functions used with the optimization methods, the head-eye motion distribution of the humanoid robot actor doing object tracking is realistic compare to the actual head-eye motion distribution of the real human. Both proposed Algorithms 1 and 2 do not require exact knowledge of hardware parameters, and generate similar solutions. Furthermore, Algorithm 2, based on reinforcement learning, can be implemented in real-time using a neural net, but requires a start learning period. The learning period can be carried out with an approximate simulation model. On the other hand, Algorithm 1, based on numerical optimization

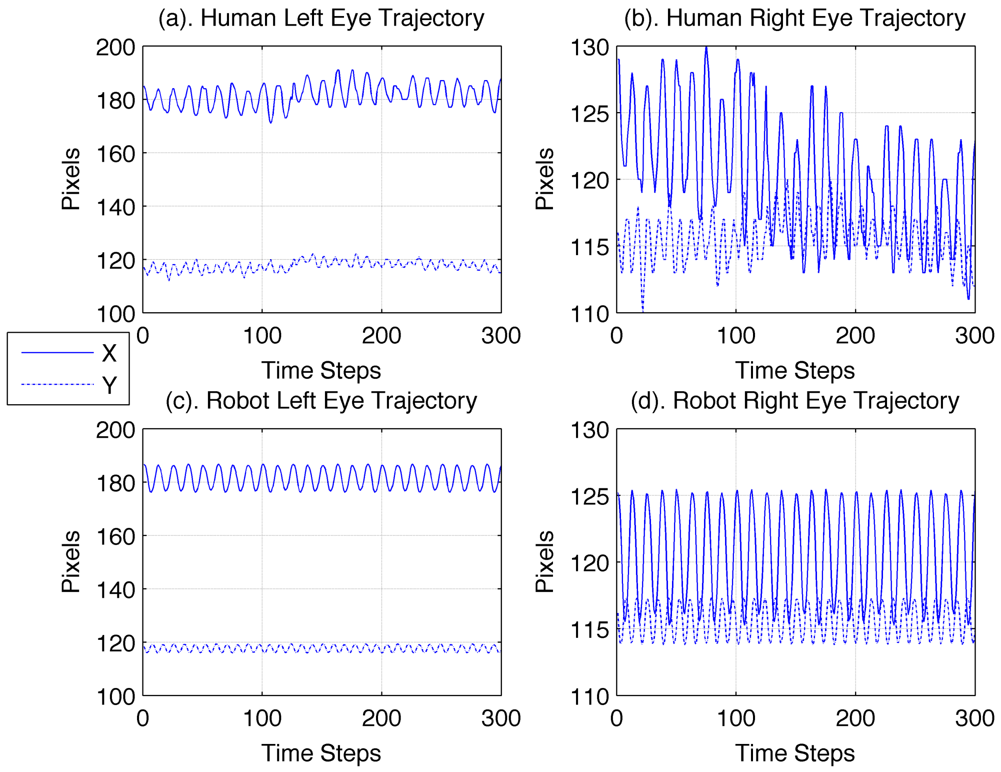


Figure 4.30. Experiment 3 using Algorithm 1 for fast moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

at each sampling step, does not require a start-up learning period, but it is more computationally intensive when implemented on-line.

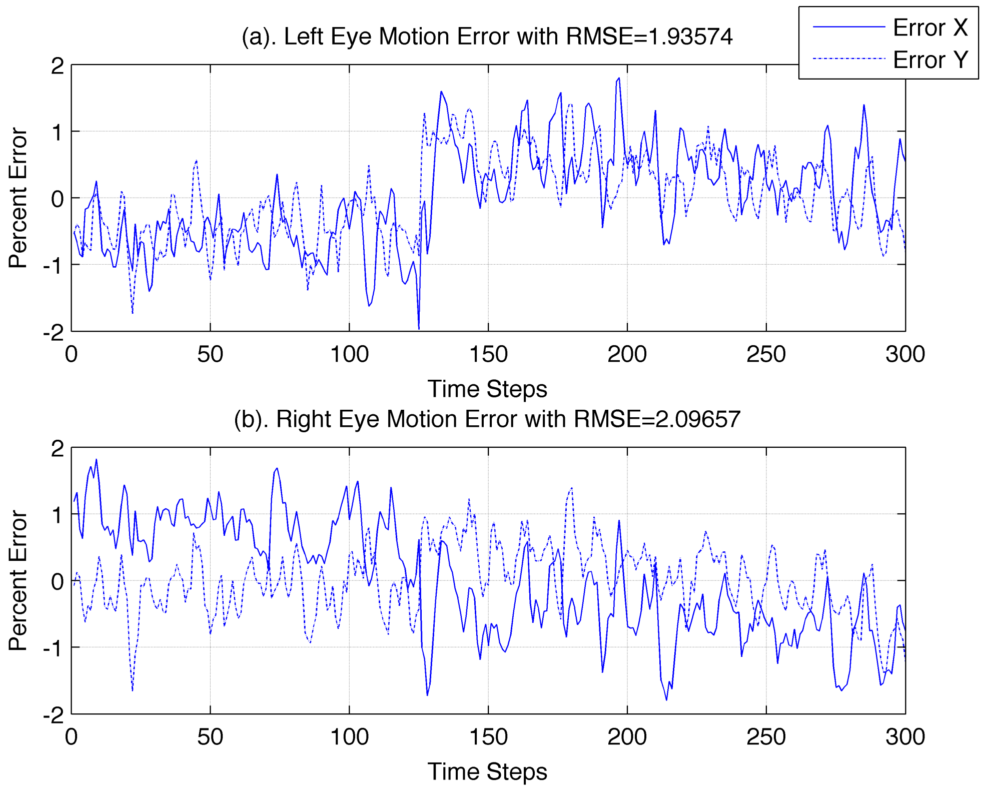


Figure 4.31. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 1 for tracking of fast moving object, the root mean square error of left eye trajectory is 1.93 and the root mean square error of right eye trajectory is 2.09.

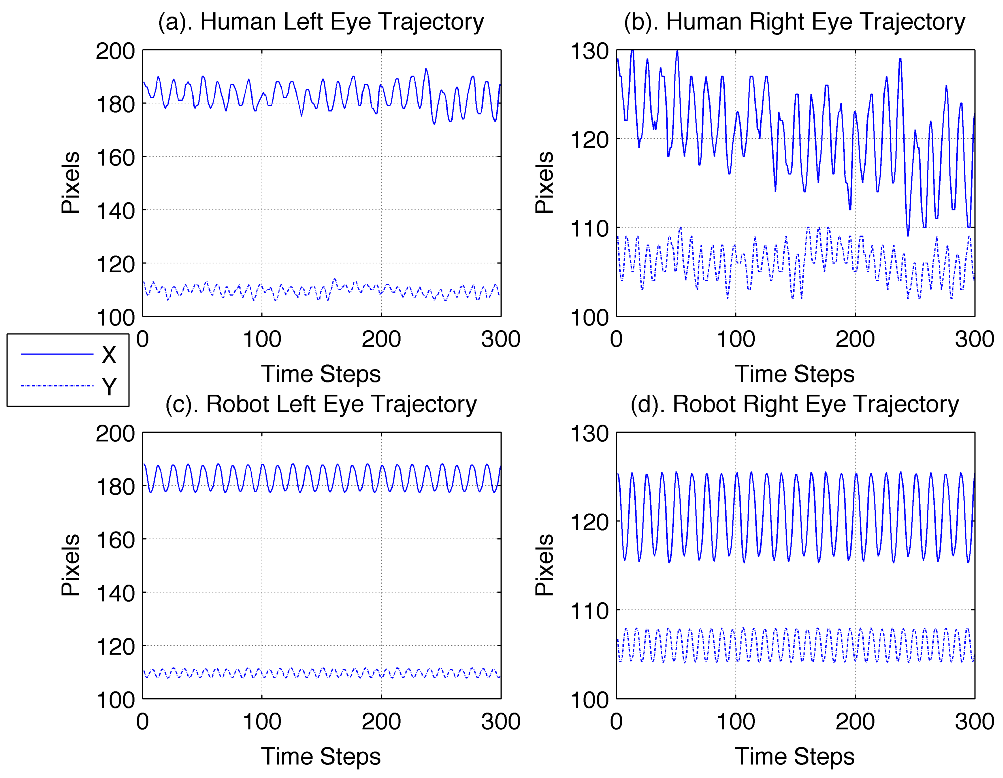


Figure 4.32. Experiment 3 using Algorithm 2 for fast moving object tracking (a) Human's left pupil trajectory in pixel coordinate obtained from video stream of the experiment (b) Human's right pupil trajectory (c) Robot's left pupil trajectory in pixel coordinate (d) Robot's right pupil trajectory.

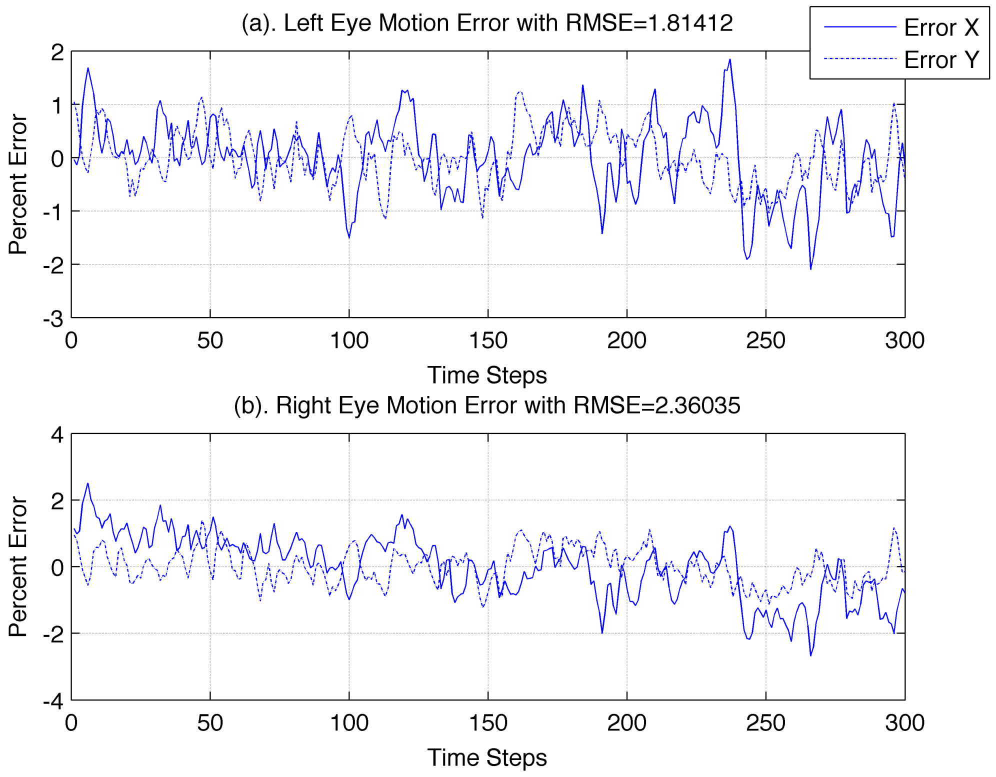


Figure 4.33. Percent error between the real human pupils' trajectory and the simulated robot pupils' trajectory using Algorithm 2 for tracking of fast moving object, the root mean square error of left eye trajectory is 1.81 and the root mean square error of right eye trajectory is 2.36.

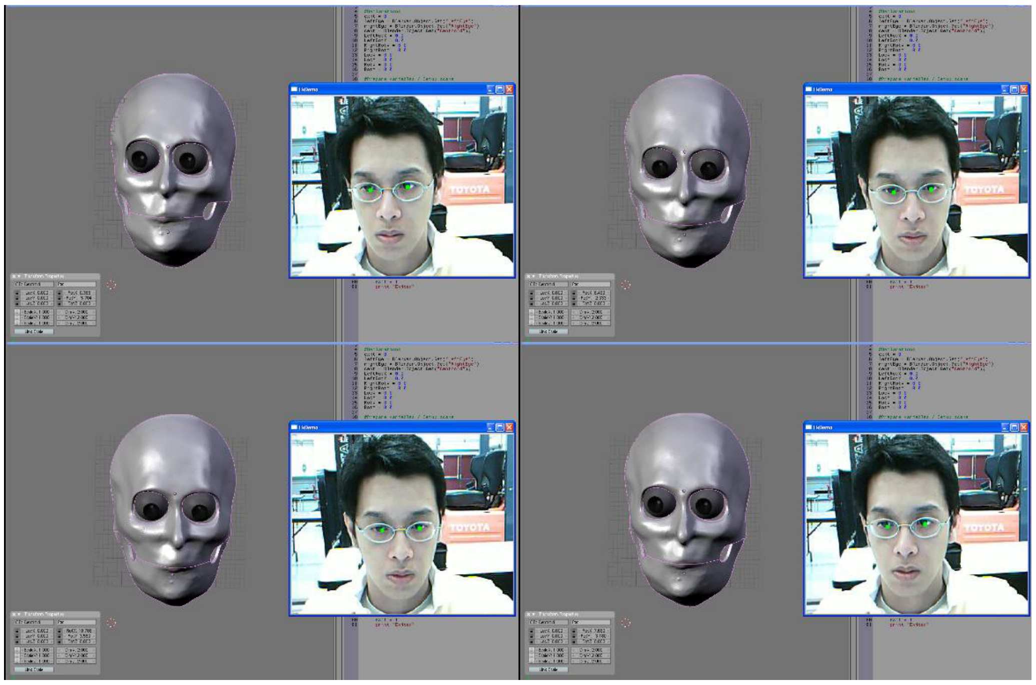
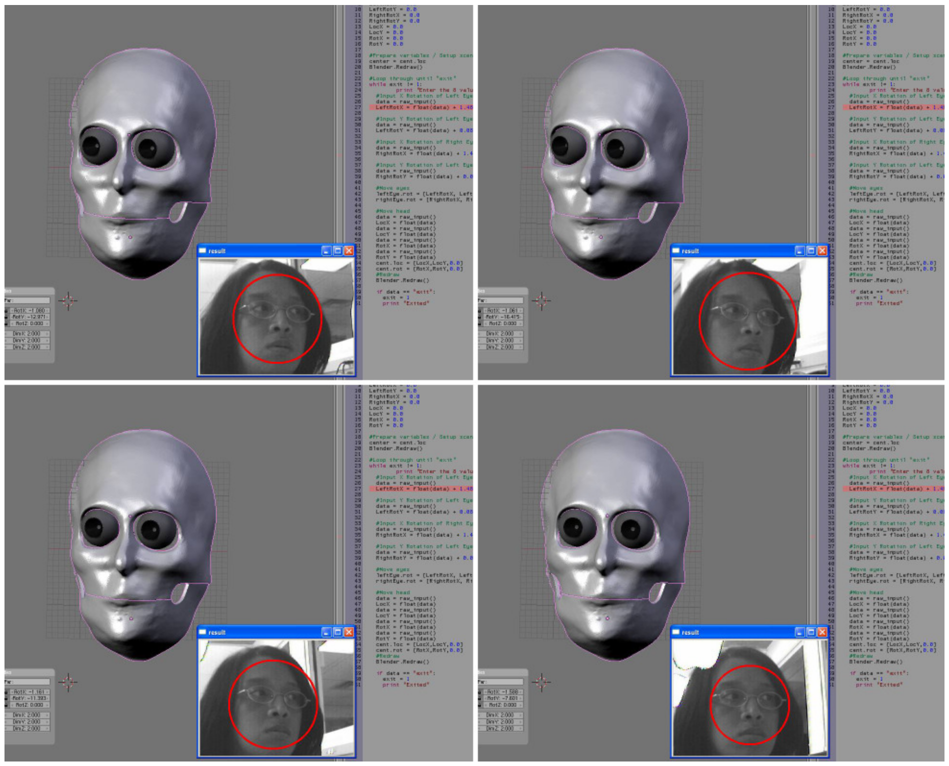


Figure 4.34. Snapshots of comparison between a human and a humanoid robot actor doing a tracking of a person using optimization method.



```
LeftRotY = 0.0
RightRotX = 0.0
RightRotY = 0.0
LocX = 0.0
LocY = 0.0
RotX = 0.0
RotY = 0.0

#Prepare variables / Setup scen
center = cent.loc
Blender.Redraw()

#Loop through until "exit"
while exit != 1:
        print "Enter the 8 valu
  #Input X Rotation of Left Eye
  data = raw_input()
  LeftRotX = float(data) + 1.48

  #Input Y Rotation of Left Eye
  data = raw_input()
  LeftRotY = float(data) + 0.08

  #Input X Rotation of Right Ey
  data = raw_input()
  RightRotX = float(data) + 1.4

  #Input Y Rotation of Left Eye
  data = raw_input()
  RightRotY = float(data) + 0.0

  #Move eyes
  leftEye.rot = [LeftRotX, Left
  rightEye.rot = [RightRotX, Ri

  #Move head
  data = raw_input()
  LocX = float(data)
  data = raw_input()
  LocY = float(data)
  data = raw_input()
  RotX = float(data)
  data = raw_input()
  RotY = float(data)
  cent.loc = [LocX,LocY,0.0]
  cent.rot = [RotX,RotY,0.0]
  #Redraw
  Blender.Redraw()

   if data == "exit":
     exit = 1
     print "Exited"
```

Figure 4.35. Snapshots of the humanoid robot actor doing a tracking of a person with error compensation scheme being implemented, the small window on the right is an image that is seen by the robot head through a camera in the one of the eyes.

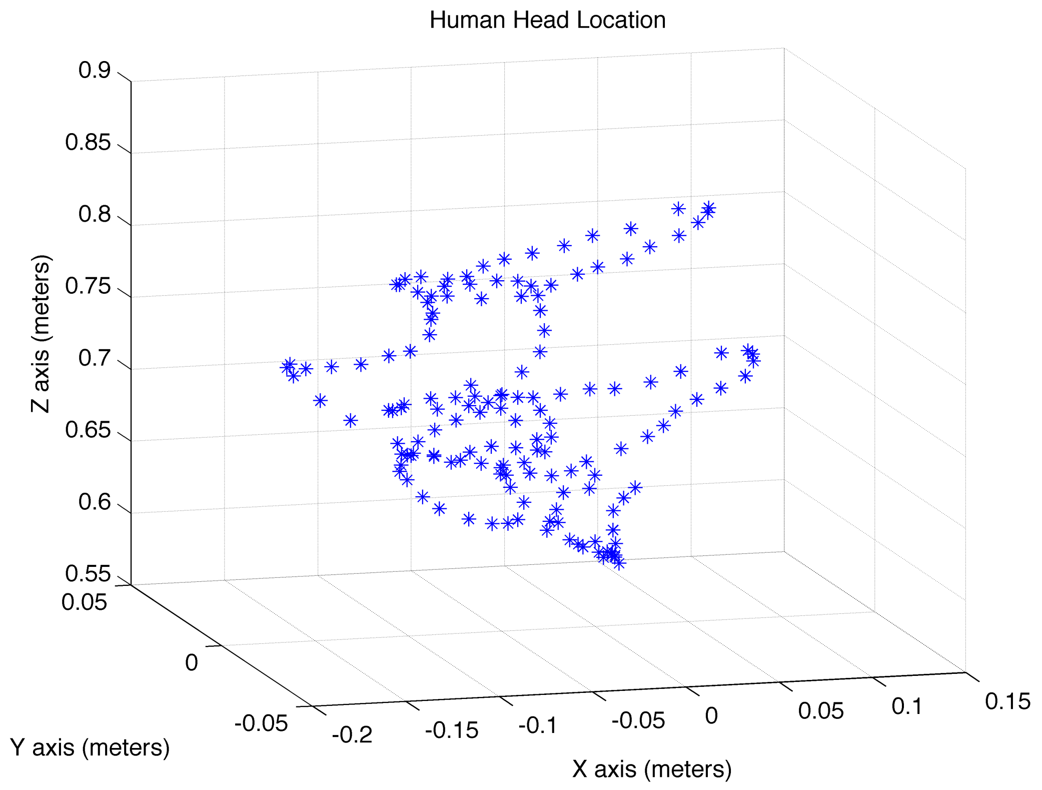


Figure 4.36. Actual human head location profile to keep track of by the humanoid robot head.

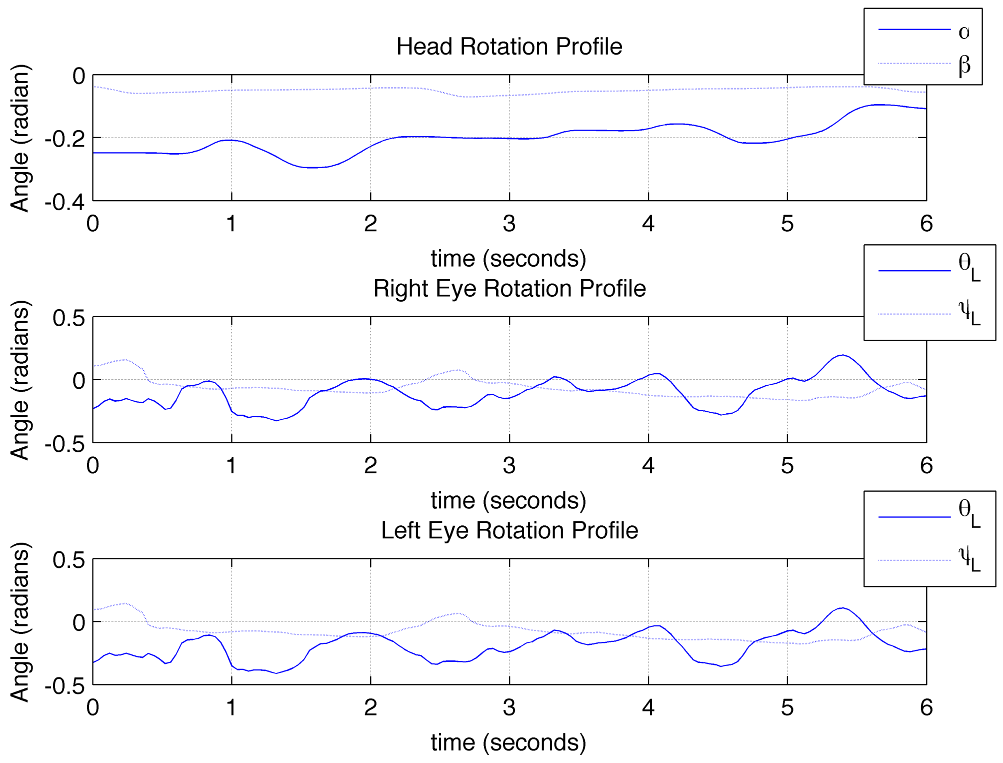


Figure 4.37. Optimal angular motion profile distribution between the neck and the eyes.

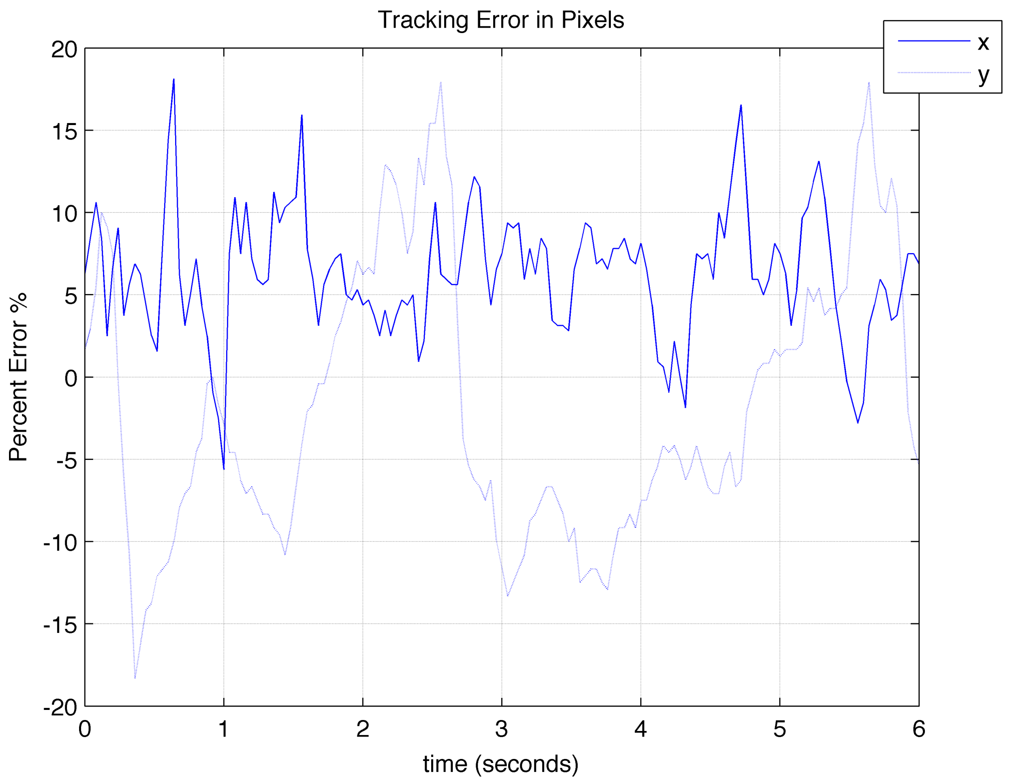


Figure 4.38. Tracking error expressed in camera frame (pixels) for tracking with visual feedback correction scheme.

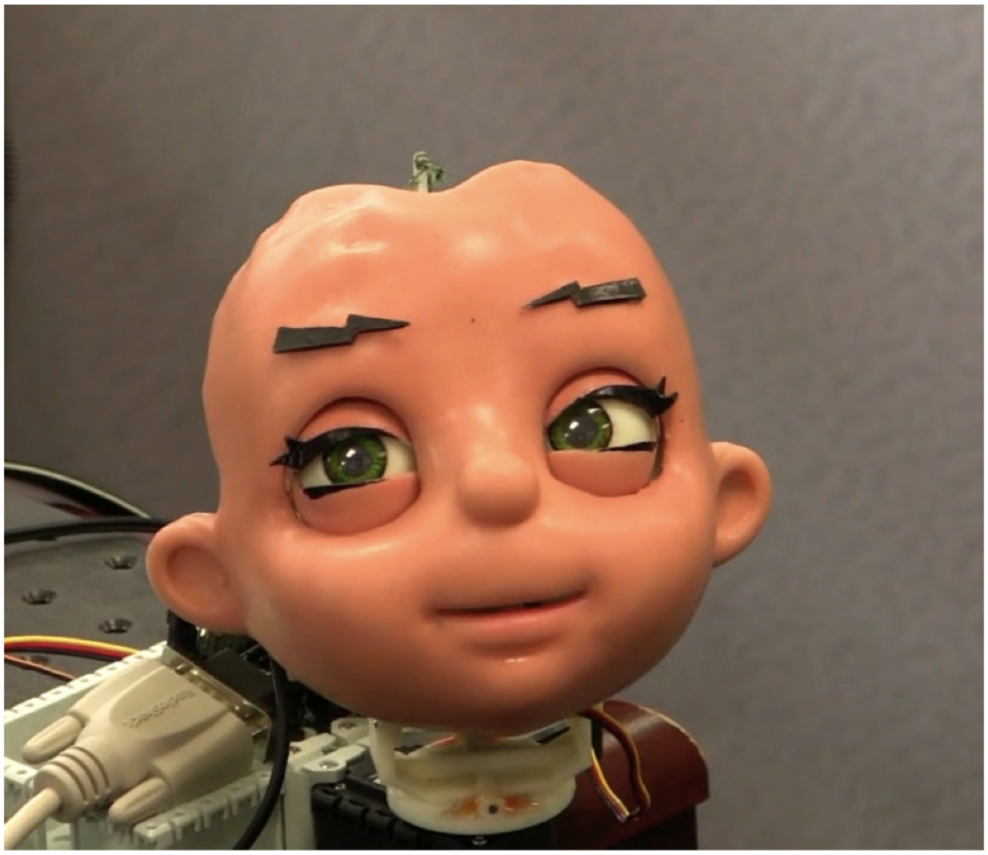


Figure 4.39. Frontal view of the Zeno head doing human tracking.

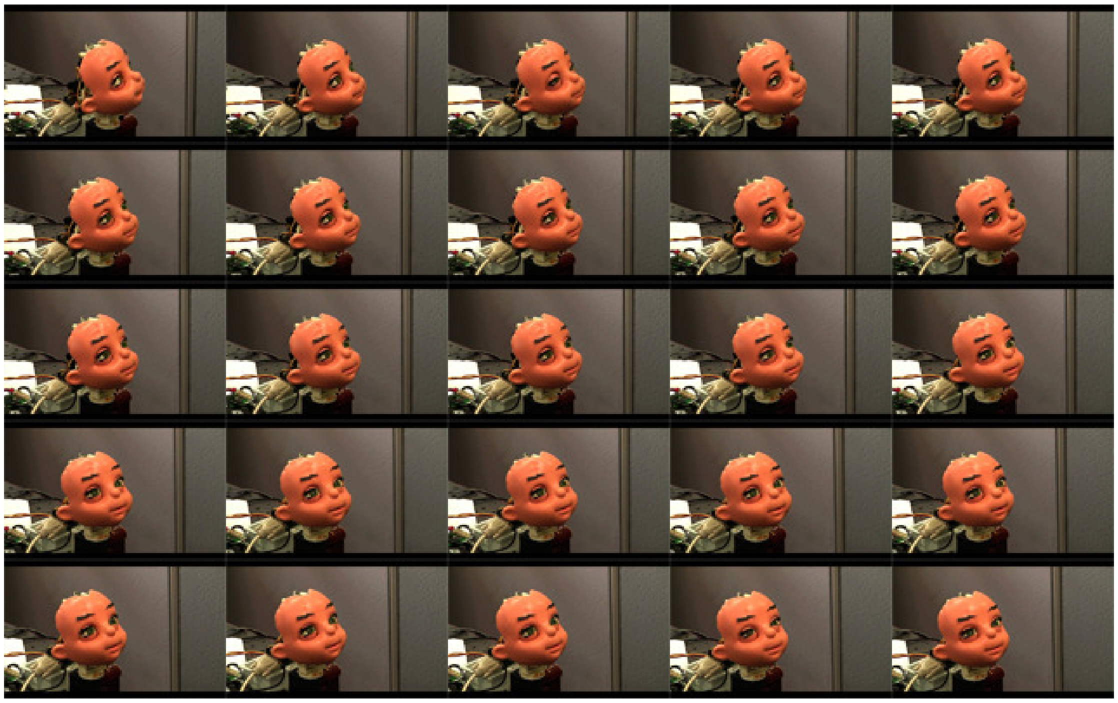


Figure 4.40. Snapshots of the Zeno humanoid robot doing a tracking of a person with error compensation scheme being implemented (Algorithm 1).

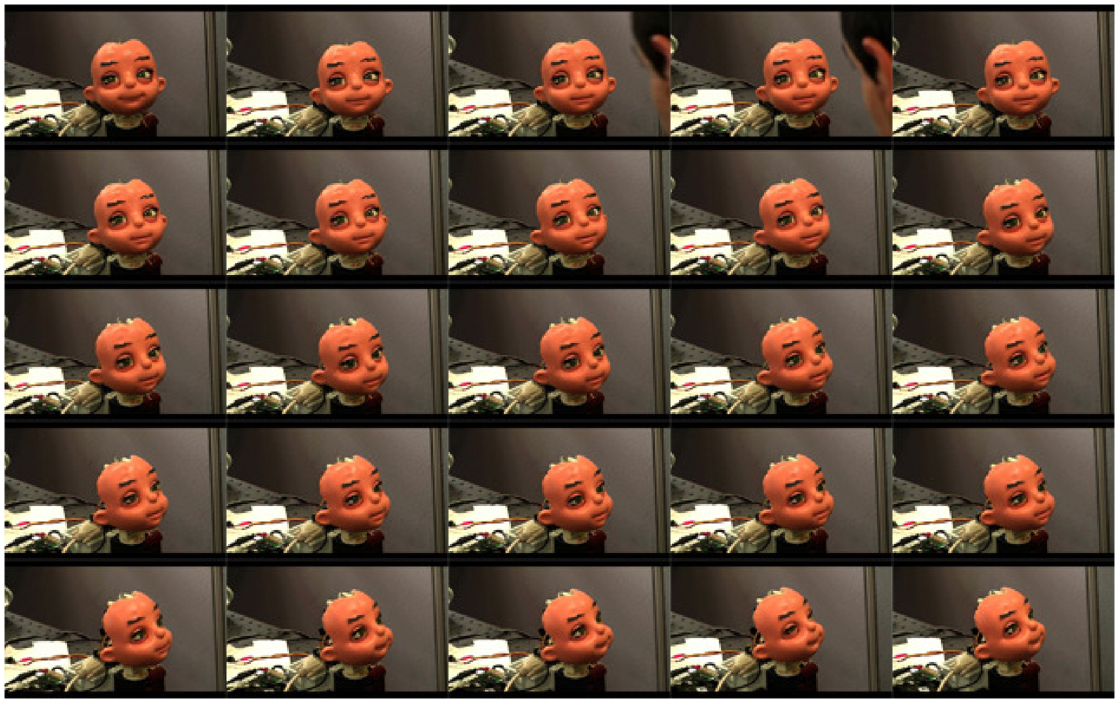


Figure 4.41. Snapshots of the Zeno humanoid robot doing a tracking of a person with actor-critic reinforcement learning of optimal control scheme being implemented (Algorithm 2).

CHAPTER 5

INTERACTION WITH MULTIPLE DEGREES OF FREEDOM ROBOT

The supervisory control of multiple robots is a very demanding application. There are challenging tasks that can be achieved using multiple robots and/or robots with multiple degrees of freedom. It is challenging to efficiently control multiple robots / robots with degrees of freedom with a simple/intuitive interface by a single operator. In this chapter, we propose the use of Reinforcement Learning for intuitive interface mapping. Based on interaction with the environments, we can determine the optimal interface mapping through the process of Reinforcement Learning. The novelty of this method is the use of changing reward functions based on qualitative performance evaluation for the Reinforcement Learning algorithm. We show that the use of proposed reward functions can result in optimal/intuitive interface mapping for multiple robots / robots with degrees of freedom control applications.

5.1 Reinforcement Learning

Reinforcement learning is a methodology of learning through interaction with the environment to achieve a certain goal. In reinforcement learning, the idea is to make use of agents and environments. The agent is the entity that learns the environments and makes decisions. The objective is for the agent to learn the optimal policy (control) through interacting with the environment by maximizing the cost/reward function over a certain period of time. The state-space is defined as a set of all possible states that the system can be in. Similarly, action-space is defined as all the possible actions that the agent can take. The applications for reinforcement learning range from discrete event decision-making processes such as games, i.e., chess, checker and backgammon to complicated control problems. One of the strengths of reinforcement learning is that it is capable of dealing with problems that the models are unknown or highly complex.

The agent interacts with an environment, at each discrete time step $t = 0, 1, 2, 3, \ldots$ a numerical reward value $r = R(s,\, a)$ is given based on current environment state $s_t \in S$ and the current action $a_t \in A(s_t)$. An agent then arrives at the next state s_{t+1}. The relationship between actions and states is denoted by agent policy $\pi(s)$. The cumulative future rewards, value function is defined as $V^{\pi}(s)$ for the current policy π. The goal of reinforcement learning is to find the optimal policy π^* that corresponds to the optimal value function $V^*(s)$. To find the optimal policy, the method of policy iteration is used. It also involves the process of finding the current value function $V^{\pi}(s)$, which is called policy evaluation. Given the current value function $V^{\pi}(s)$, the policy π can be improved. The process of improving policy is called policy improvement. After repeating the policy iteration process over and over we can arrive at the optimal policy π^*. There are different standard methods that are used for policy iterations, namely, Dynamic Programming, Monte Carlo method and Temporal Difference learning. In this dissertation, we will be focusing on Temporal Difference learning.

Temporal difference (TD) learning is an approach to learning how to predict a quantity that depends on future values of a given signal:

$$Y_t = y_{t+1} + \gamma\, y_{t+2} + \gamma^2\, y_{t+3} + \ldots = \sum_{i=1}^{\infty} \gamma^{i-1}\, y_{t+1} \tag{5.1}$$

$$\begin{aligned} Y_t &= y_{t+1} + \gamma\, \left[y_{t+2} + \gamma\, y_{t+3} + \gamma^2\, y_{t+4} + \ldots\right] \\ &= y_{t+1} + \gamma\, Y_{t+1} \end{aligned} \tag{5.2}$$

ALGORITHM: TD(λ) WITH FUNCTION APPROXIMATION

```
$\vec{w}_v \leftarrow$ an arbitrary initial value
$\vec{w}_p \leftarrow$ an arbitrary initial value
$\vec{e} \leftarrow \underline{0}$   (dimension of $\vec{e}$ = dimension of $\vec{w}_v$)
$\vec{e}^{\,p} \leftarrow \underline{0}$   (dimension of $\vec{e}^{\,p}$ = dimension of $\vec{w}_p$)

for each episode do
    $s \leftarrow$ random initial state
    while not end of episode do
        $s_{t+1} \leftarrow p_{\vec{w}_p}(s_t)$
        $\delta_t \leftarrow r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$
                $e_{t+1} \leftarrow \lambda \gamma e_t + \frac{\partial V_{\vec{w}_v}(s_t)}{\partial \vec{w}_v}$
        $e^p_{t+1} \leftarrow \lambda \gamma e^p_t + \frac{\partial p_{\vec{w}_p}(s_t)}{\partial \vec{w}_p}$
        $\vec{w}_v \leftarrow \vec{w}_v + \eta \delta_t e_{t+1}$
        $\vec{w}_p \leftarrow \vec{w}_p + \eta \delta_t e^p_{t+1}$
        $s_t \leftarrow s_{t+1}$
    end while
end for
```

Figure 5.1. Algorithm: actor-critic $TD(\lambda)$ with function approximator.

The Temporal difference error and the value function are already defined in the previous chapter as in (4.53) and (4.55), respectively. We will be using the actor-critic reinforcement learning approach for our implementations in later sections. The details on actor-critic reinforcement are already given in the previous chapter, (4.53) - (4.65). The implementation algorithm of the actor-critic $TD(\lambda)$ with function approximations is shown in the Fig. 5.1.

5.2 Interface Mapping

In this chapter, we are interested in building a generalized method to learn the most intuitive interface mapping to be used with single user control of either a robot with multiple degrees of freedoms or multiple robots. It is obvious that the word "intuitive" it is quite vague as there is no quantitative measure to make a judgment. With the current advancement in the interface device technology, it is challenging to utilize the device to its full potential, or there may not be an immediate intuitive way to make use of the device. An example of an interface device that might be a good candidate for us to figure out the "intuitive" input/ output mapping is shown in Fig. 1.6 and 1.8. Also, intuitiveness is something subjective and may vary from user to user. In this chapter, we present details of interface mapping.

5.2.1 Definition of Interface Mapping

Suppose we have a device that give continuous sensory outputs as a result of manipulating the device by a human operator. Let's assume that the reading that we get from a device has n elements:

$$u = [u_1 \;\; u_2 \;\; \dots u_n]^T \tag{5.3}$$

where u is the raw input from a device and $u \in \Re^n$.

We can define a general form of a dynamical interface mapping using a non-linear state-space model.

$$\dot{x} = f(x,\, u) \tag{5.4}$$

$$y = C\, x \tag{5.5}$$

where x is the state of the mapping, y is the output of the mapping and $y \in \Re^m$ and $C \in \Re^{m \times n}$ is the output matrix of the mapping.

With the above equations (5.4) and (5.5) the mapping is allowed to have dynamic behavior. This might be useful in some cases where the output of the mapping is used to control a fast system (or even system with no dynamics at all) so that the relationship between interface inputs and outputs can exhibit some degree of change. Examples of such systems are; pointing devices.

Since we want to learn the optimal mapping for a particular task / user, the mapping should be adjustable. We represent the mapping using multi layer feed forward network (just like what we have in the previous chapter). (5.4) now becomes:

$$\dot{x} = f_{\vec{w}}(x,\, u) \tag{5.6}$$

where $f_{\vec{w}}(.)$ denote a multi layer feed forward network similar to (4.59) and (4.60).

However, for interfaces that are used with tasks such as robot control, it might be a better idea to work with static mapping since there is already the dynamics effect of the actual robot system. An example of a simple interface device to be used with robot control task is shown in Fig. 1.6. So, for static interface mapping the relationship between interface inputs and outputs becomes:

$$y(u) = \sum_{i=0}^{p} w_i\, \Phi_i \left(\sum_{j=0}^{q} w_{ij}\, u_i \right) \tag{5.7}$$

or it can be rewritten in a compact form as:

$$y(u) = f_{\vec{w}}(\vec{w},\, u) \tag{5.8}$$

We can see that now the interface mapping is adjustable through $\vec{w}$, and we denote $\vec{w}^*$ as the optimal weight for the interface mapping. This can also be considered as the most intuitive mapping. The next chapter presents a novel approach to update the interface mapping so that the weight of the multi-layer feedforward network $\vec{w}$ approaches $\vec{w}^*$ using actor-critic reinforcement learning approach presented in the previous chapter.

5.3 Learning the Interface Mapping

From the previous section, we now have the definition of the interface mapping. We focus on finding the optimal weight $\vec{w}^*$ for a particular system and task. In this chapter, we present details on reinforcement learning of the optimal weight tuning, and also, we discuss the use of reward function that is related to the qualitative performance metrics defined for a particular task and are set by the user.

5.3.1 $TD(\lambda)$ for mapping weight update

To fit the weight update process under the actor-critic reinforcement learning scheme, we need to clearly define the actor structure as well as the critic structure. First of all, we need to define state-space of the system. The current environment state, in this case, is defined as the current weight of the interface mapping as defined in (5.8):

$$s_t \in S_w \tag{5.9}$$

where S_w define the state-space of mapping weights.

The state of the environment (weight $\vec{w}$) changes due to the current action. The new state is a function of current state and the current action, as follows:

$$s_{t+1} = F(s_t, p_t) \tag{5.10}$$

where $F(.)$ is an arbitrary function. Let assume that the Taylor series expansion of $F(.)$ takes the following form:

$$F(s) = F(0) + F'(0)s + \frac{F''(0)}{2!}s^2 + \frac{F'''(0)}{3!}s^3 + \dots \tag{5.11}$$

We take the first order approximation of (5.11) to be our state propagation, and substituting the $F'(0)s$ term with the action term $p_t(s_t)$ and then turn it into

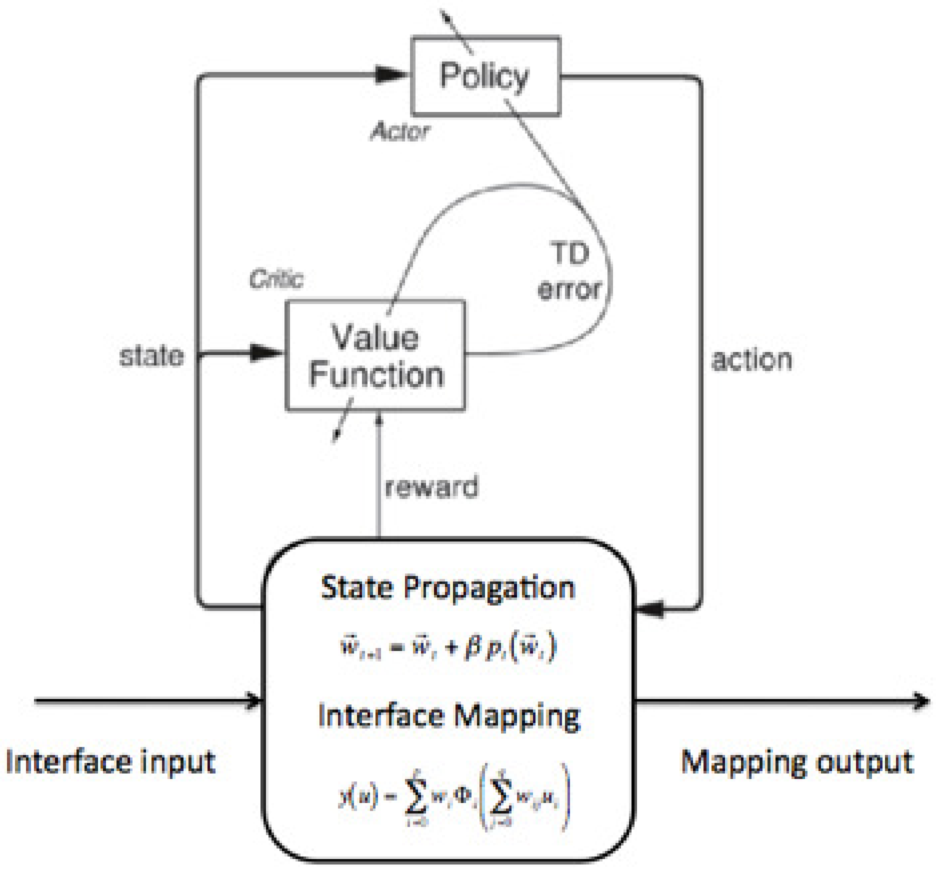


Figure 5.2. The complete $TD(\lambda)$ learning of the interface mapping system.

a discrete update of environment state $\vec{w}$. The state propagation of the mapping is defined as follows:

$$s_{t+1} = s_t + \beta\, p_t(s_t) \tag{5.12}$$

where β represents a step size.

The complete system of actor-critic $TD(\lambda)$ learning of an interface mapping is shown in Fig. 5.2. Note that in the diagram a state vector of the environment is represented as $\vec{w}_t$ instead of s_t. The learning of the mapping can be done according to the algorithm laid out in Fig. 5.1.

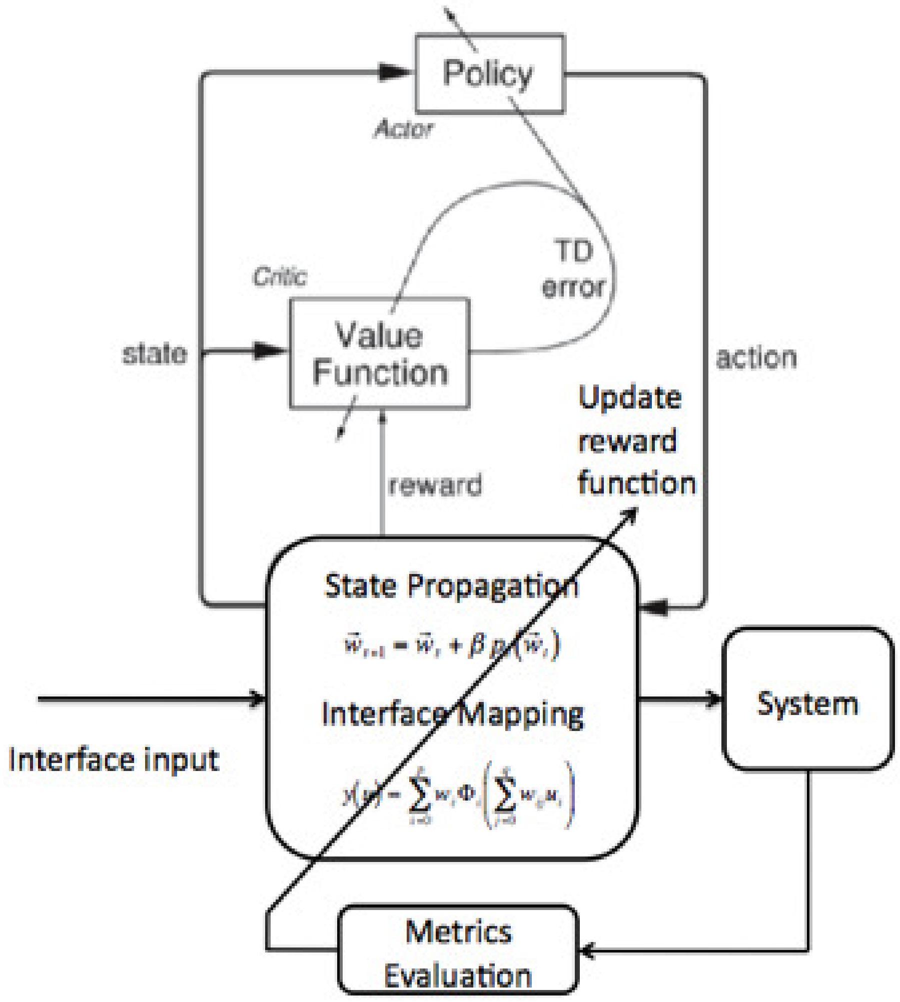


Figure 5.3. The complete $TD(\lambda)$ learning of the interface mapping system with reward function update scheme implemented.

5.3.2 Reward Functions and Performance Metrics

A reward function has to be set so that the mapping of the interface converges to the optimal one after certain amount of learning episodes have been completed assuming there exist an optimal mapping for a task of interest for every user. The optimal mapping can vary from user to user as different users might not find the optimal mapping for one particular user to be the most preferable one. In this subsection, we introduce a systematic methodology to update a reward function according to performance metrics. The reward function obtained will be used with the $TD(\lambda)$ learning of the interface mapping system as shown in Fig. 5.3. Suppose that at a particular episode $u_k(t)$ a certain metric PM_m is used to evaluate mapping performance, a real number value p_k is assigned to $u_k(t)$ to indicate how good the mapping is. k is the number of episodes that have been evaluated according to the chosen performance metric. We have:

$$\{u_k(t),\, p_k\} \qquad k > 0,\;\; t_s < t \leq t_f \tag{5.13}$$

where t_s and t_f is start time and end time of the episode.

The reward function is defined as:

$$R = \frac{1}{k}\sum_{n=1}^{k} p_k\, P\{u_k(t)\,|\,u(t)\} \tag{5.14}$$

where $u(t)$ is the current episode input profile.

Since $u_k(t)$ is a continuous function, to evaluate the probability of $P\{u_k(t)\,|\,u(t)\}$ we can use Hidden Markov Model. Appendix A provides details on Hidden Markov Model implementation for input profile recognition.

The performance metric that we can use for updating the reward function (5.14) can be arbitrary. The metric can be both quantitative and qualitative. An example of quantitative metrics are, for example, time it take to complete a task, accuracy, tracking error and etc. From implementation point of view, using qualitative metrics is more complicated since we need a way to get feedback from the user. Appendix C provides a list of widely used performance metrics for robot control applications.

5.4 Experiment Results

A set of experiments on interface mapping using reinforcement approach is conducted using setup show in Fig. 5.4. We use 3D stylus as an input device. With

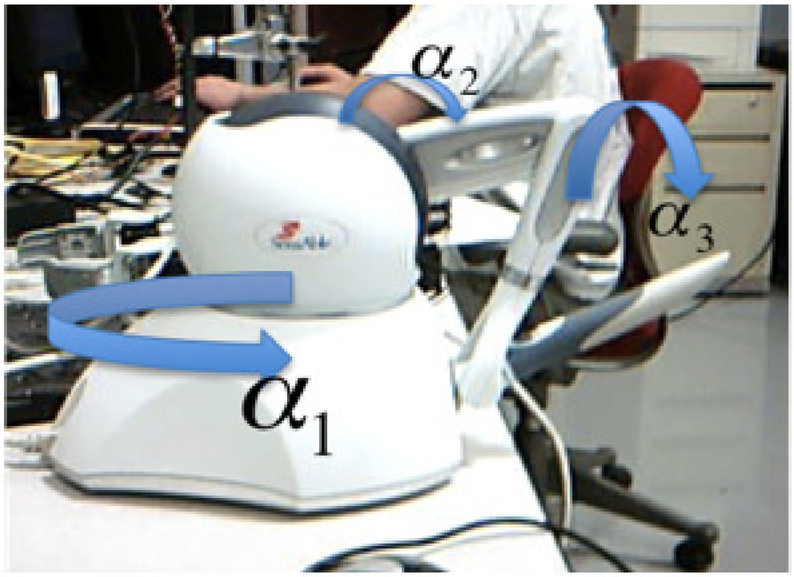


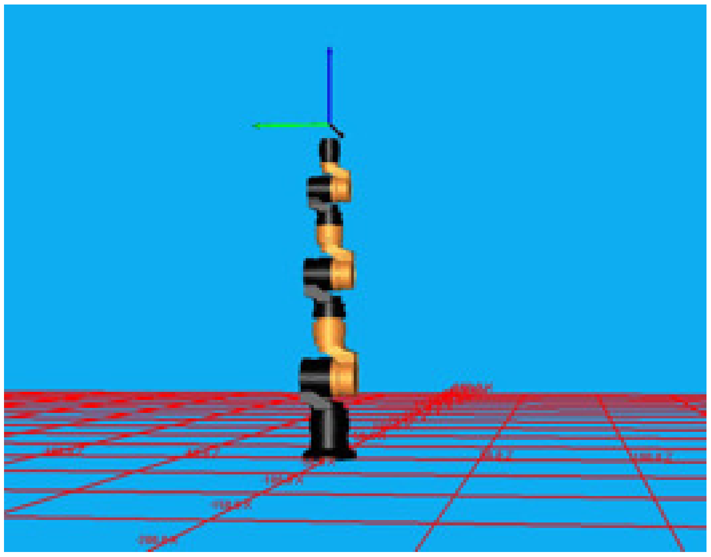
Figure 5.4. 3D stylus used as an input device.

this device, we can access joint angular position data as well as 3D position of the tip of the stylus.

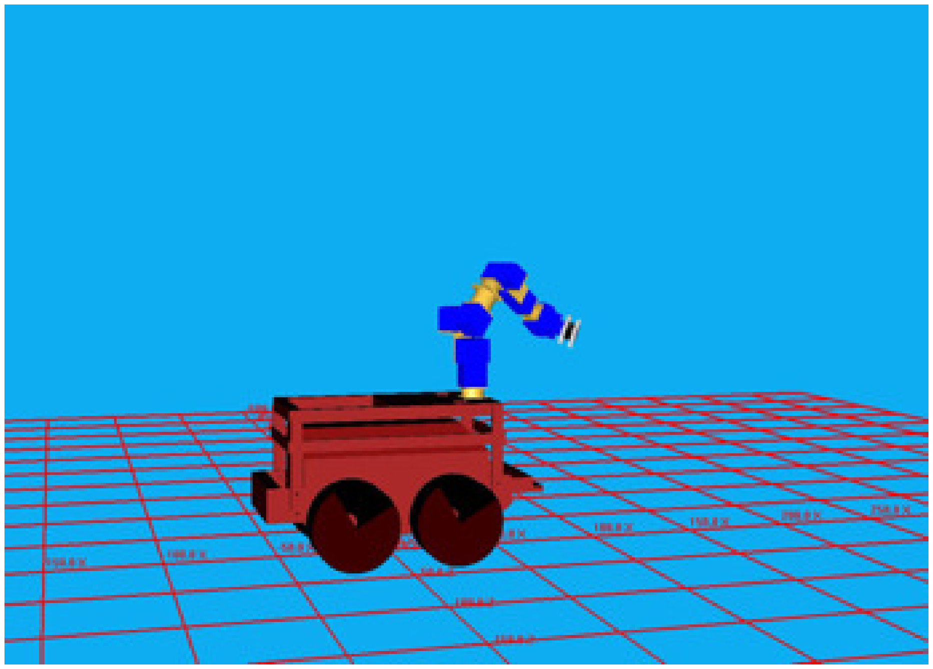
Two different simulated robot systems are used in conjunction with the 3D stylus to conduct experiments. Fig. 5.5 shows the screen capture of 7 degrees of freedom articulated robot arm and 6 degrees of freedom robot arm mounted on a mobile platform.

5.4.1 Experiment 1

Manipulating a 7 DOFs robot arm using a 3D stylus where the interface mapping is (5.8). The task is to position the end effector of the robot arm along a prescribed trajectory. The performance metric used in this experiment is the tracking error. The lower the error the higher the reward will be. Each episode lasts 15 seconds; the prescribed trajectory is the same for all episodes. For the HMM, we use 15 hidden states and calculate the reward using (5.14) and $k = 10$. We use $\lambda = 0.7$, $\gamma = 0.2$ and $\eta = 0.01$ for the $TD(\lambda)$ algorithm. For value function iteration (4.55),



(a)



(b)

Figure 5.5. (a) 7 degrees of freedom articulated robot arm (b) 6 degrees of freedom robot arm mounted on a mobile platform.

we use feed forward neural network with 2 hidden layers and 10 neurons on each layer. And we have the same structure for policy iteration (4.60). Fig. 5.6 shows that

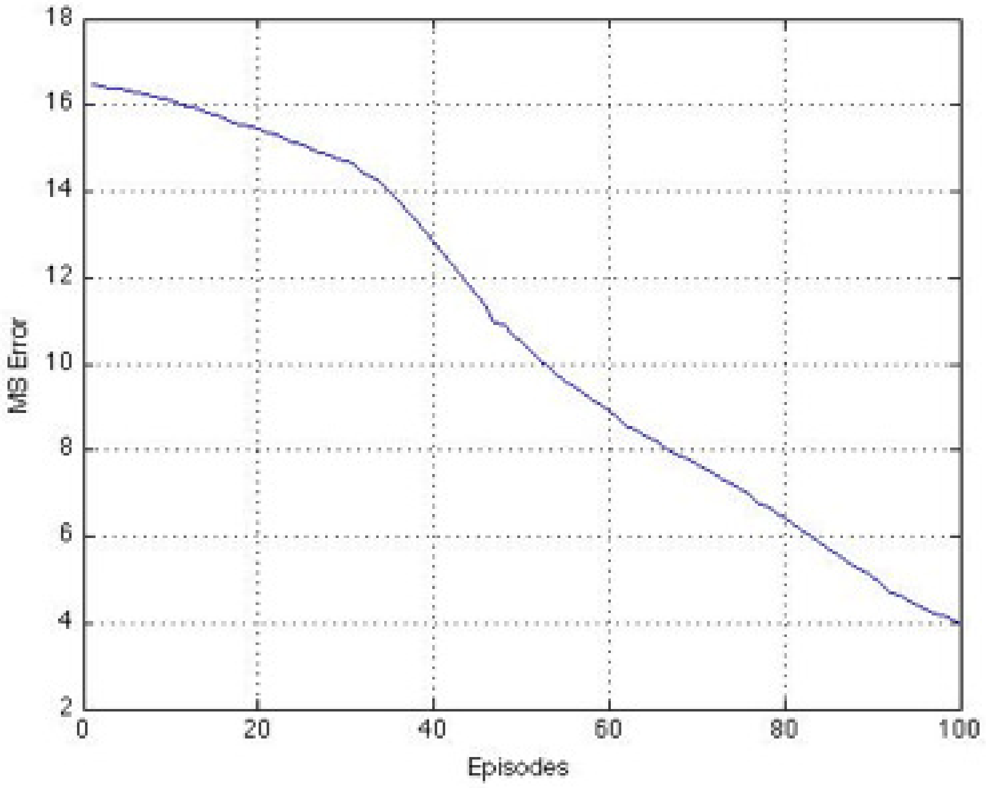


Figure 5.6. Tracking error decreases as the user continue using the system.

the tracking error of the robot end-effector and the prescribed trajectory decreases as the user the number of learning episodes increases. It shows that robot system has becoming easier to control using the interface device with the proposed mapping update method.

5.4.2 Experiment 2

Manipulating a 6 DOFs robot arm mounted on a mobile platform using a 3D stylus where the interface mapping is (5.8). The task is to pick an object from start point and place it at the prescribed location. The performance metric used in this experiment is the time it takes to complete the task. Each episode lasts 20 seconds;

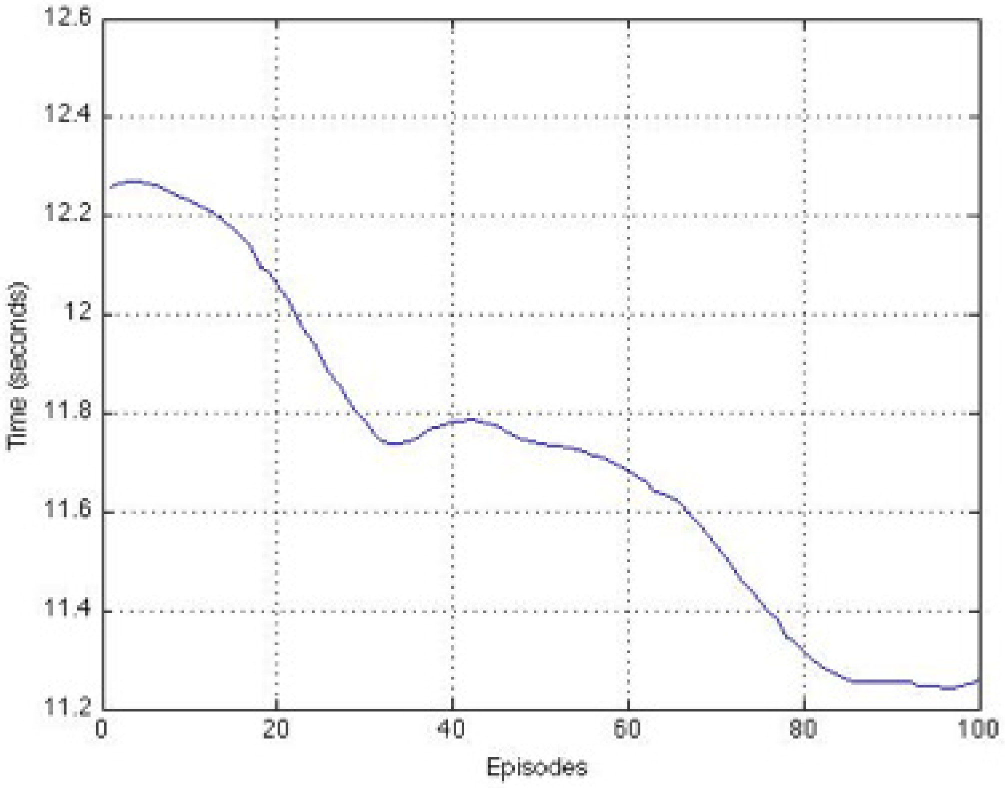


Figure 5.7. Time take to complete the task decreases as the user continue using the system.

the prescribed location is the same for all episodes. For the HMM, we use 15 hidden states and calculate the reward using (5.14) and $k = 10$. We use $\lambda = 0.7$, $\gamma = 0.2$ and $\eta = 0.01$ for the $TD(\lambda)$ algorithm. For value function iteration (4.55), we use feed forward neural network with 2 hidden layers and 10 neurons on each layer. And we have the same structure for policy iteration (4.60). Fig. 5.7 shows that the time spent completing the pick and place task decreases as the number of episode increases. Note that after it reaches 80 episodes the performance cannot be improved any further.

Figure 5.8. The brain activity sensor is worn by the user.

5.4.3 Experiment 3

In this experiment, we would like to control the articulated robot arm shown in Fig. 5.9 using a brain activity sensor shown in Fig. 1.8. The task that we are interested in is position the end-effector of the robot arm that has a camera attached to a certain position and orientation. The input from the brain activity sensor will be used with the control the robot joints angular position. The robot has 5 degrees of freedom and we implement the proposed learning of the interface mapping, shown in Fig. 5.3, to find the optimal mapping for this particular this.

The user wearing the brain activity sensor is shown in Fig. 5.8. The user is presented with a user interface shown in Fig. 5.10. From the user's point of view, the user is doing visual servoing as the user tries to position the camera at the end-effector of the robot arm so that the object of interest is centered on the screen.

The emotiv head band (brain activity sensor) is capable of sensing level of emotions. We use this capability of the device as the one of the performance indications for this experiment. We use $\lambda = 0.7$, $\gamma = 0.2$ and $\eta = 0.01$ for the $TD(\lambda)$ algorithm. For value function iteration (4.55), we use feed forward neural network with 2 hidden layers and 10 neurons on each layer. And we have the same structure for policy iteration (4.60). The reward function used in this experiment is the time it takes to complete the task - moving the end-effector of the robot so that the object move from the initial location as shown in Fig. 5.11 to the desired final location (center) as shown in Fig. 5.12.

There are 3 different emotions that can be sensed using the brain activity sensor device. So, we define the indication of mental work load as "emotion energy":

$$EE = \sum_{k=0}^{3} \int_{0}^{t_f} g_k(t)dt \tag{5.15}$$

where t_f denotes the final time, $g_k(t)$ represents a function of time for each emotions. In [104], electrodermal activity (EDA) was used to distinguish stress from cognitive load.

We start with initial randomized NN weights. After 10 successful task completions, we measure the emotion levels as shown in Fig. 5.15. The time that take to complete the task at 11^{th} episode was $t = 42.6$ seconds and the resulting emotion energy was 1644.07. We continued the learning for 10 more episodes (task completions) and at the 21^{th} episode the task completion time was 37.2 seconds and the emotion energy was 1467.91. The emotions level of the final episode is shown in Fig. 5.16. The robot arm initial configuration is shown in Fig. 5.13 and the final robot arm configuration of the final episode is shown in Fig. 5.14.

In this experiment, we can see that the intuitive mapping from the brain activity sensor input to the robot manipulator joint space is not immediately available. In addition, the robot manipulator was used to position the camera that is attached to the end-effector to a desired location, in other words, the user performed image-based visual servoing of a redundant mechanism. The user utilized the visual feedback information to perform the manipulation of the manipulator. With a randomized initial mapping, the user was able to complete the specified visual servoing task. However, it is obvious from the time that took the user to complete the task that the user had some difficulty completing the task. We implemented the proposed $TD(\lambda)$ learning of the interface mapping (as depicted in Fig. 5.3) to update the mapping between brain activity sensor and the manipulator joint space position control. As the user continued performing the same task the mapping was updated by proposed algorithm. After a certain numbers of task completions (episodes) we can see from the time that the user took to complete the visual servoing task and the mental work load indication (5.15) that the overall performance of the user performing this particular task has improved.

5.5 Summary

Reinforcement learning allows more flexibility for the interface mapping application by not just limiting to training using input/output data set. Different kind of information can be incorporated into reward function. Universal function approximation property of the neural networks makes it possible to extend to higher input/output dimension systems. From the experiment results, we can that it is possible to use the proposed algorithm with arbitrary systems with arbitrary interface devices. The reward function and the performance metrics must be appropriately selected to suit the task to be perform. The experiment also show that the proposed algorithm can be generalized to case of multiple inputs / multiple outputs system as we have successfully shown that the interface mapping was improved with a robot

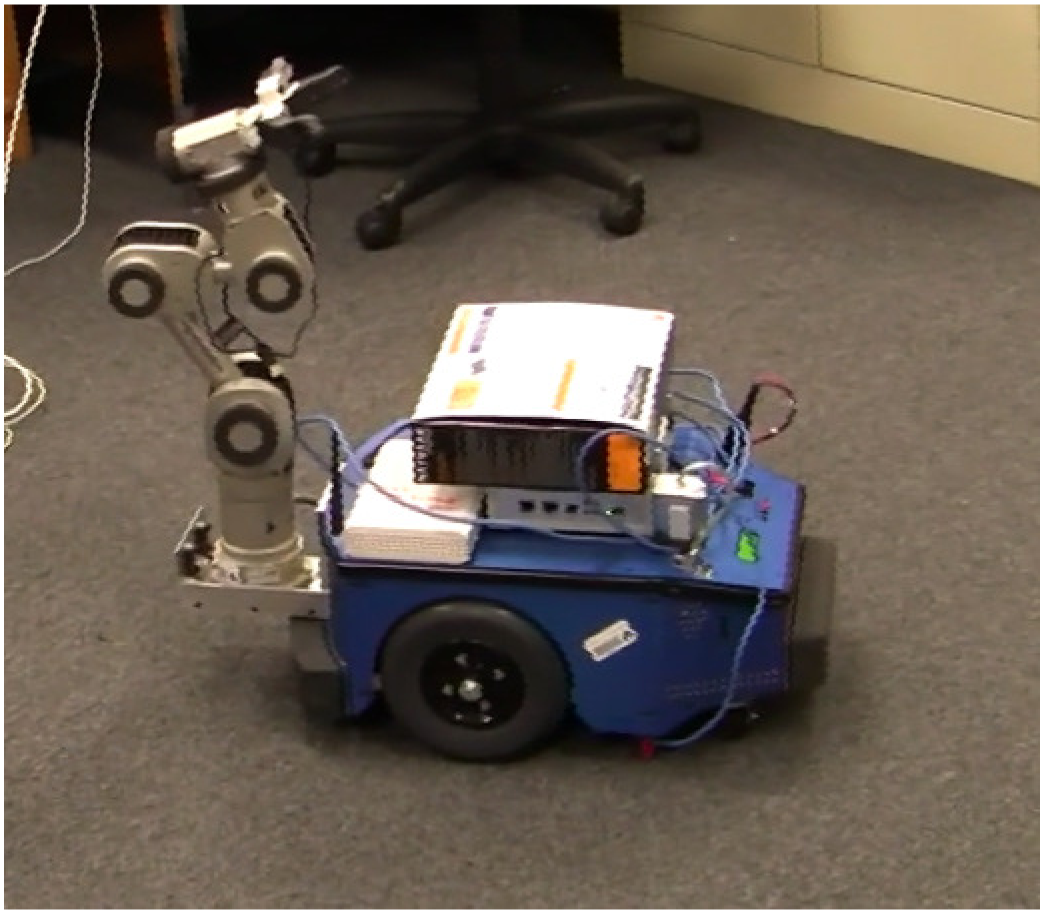
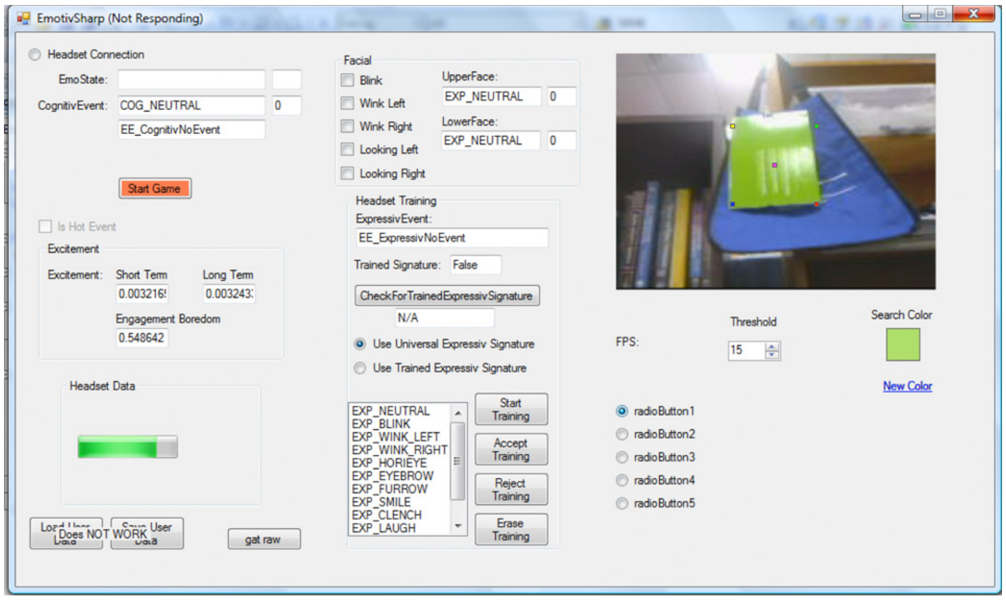


Figure 5.9. The robot arm to be controlled through the use of the brain activity sensor.

system that has multiple degrees of freedom. The mapping itself is represented by the feedforward neural networks. Even though the convergence of the neural network training is not guaranteed, we were able to show that the mapping was converging to some optimal values. From one of the experiments, we see that the training could not be improved any further after a certain number of iterations. We can assume that the mapping has reached its optimal value but we cannot say for sure that the obtained mapping was the optimal one. It is difficult to determine whether or not the acquired mapping is optimal as there are several factors that contribute to the optimality of the interface mapping, for instance, an optimal mapping for one user might not be


EmotivSharp (Not Responding)

Headset Connection

EmoState:

CognitivEvent: COG_NEUTRAL 0

EE_CognitivNoEvent

Start Game

Is Hot Event

Excitement

Excitement: Short Term 0.003216! Long Term 0.003243:

Engagement Boredom 0.548642

Headset Data

Load User Data | Does NOT WORK | Save User Data | gat raw

Facial

- Blink
- Wink Left
- Wink Right
- Looking Left
- Looking Right

UpperFace: EXP_NEUTRAL 0

LowerFace: EXP_NEUTRAL 0

Headset Training

ExpressivEvent: EE_ExpressivNoEvent

Trained Signature: False

CheckForTrainedExpressivSignature

N/A

[✓ Use Universal Expressiv Signature]

Use Trained Expressiv Signature

```
EXP_NEUTRAL
EXP_BLINK
EXP_WINK_LEFT
EXP_WINK_RIGHT
EXP_HORIEYE
EXP_EYEBROW
EXP_FURROW
EXP_SMILE
EXP_CLENCH
EXP_LAUGH
```

Start Training | Accept Training | Reject Training | Erase Training

FPS:

Threshold 15

Search Color 🟩

New Color

[✓ radioButton1]

radioButton2

radioButton3

radioButton4

radioButton5

Figure 5.10. A user interface for experiment 3 that is seen by the user.

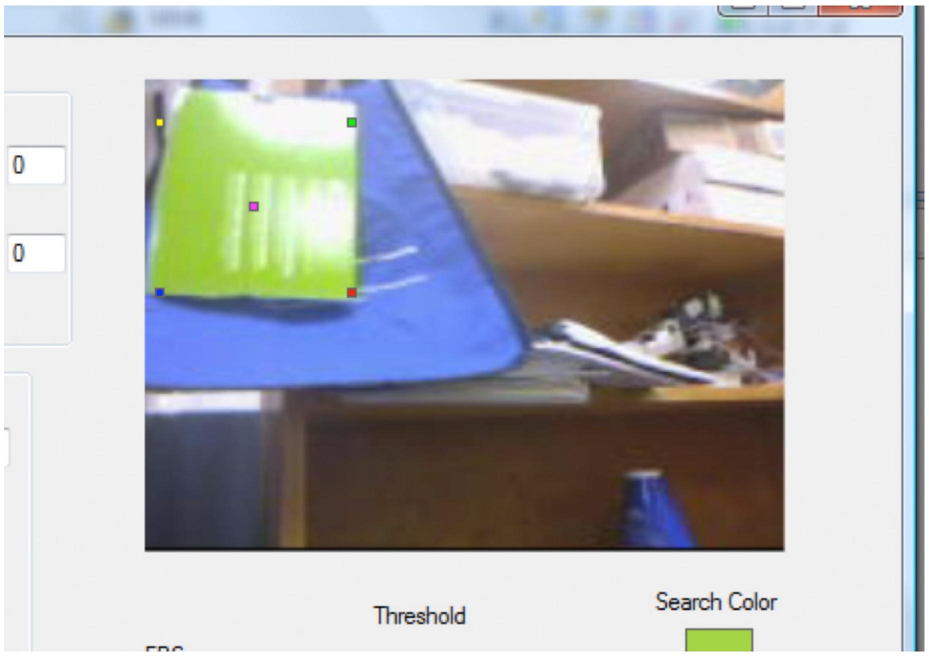


Figure 5.11. Initial location of the object to be tracked with respect to the camera frame that is used for all the episodes of experiment 3.

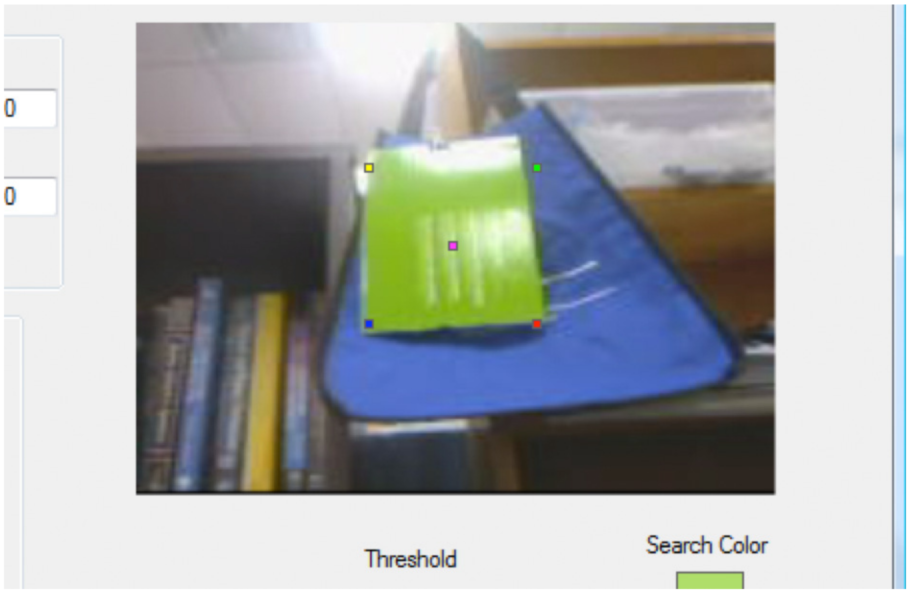


Figure 5.12. Final location of the object - the task will consider complete after reaching this desired location.

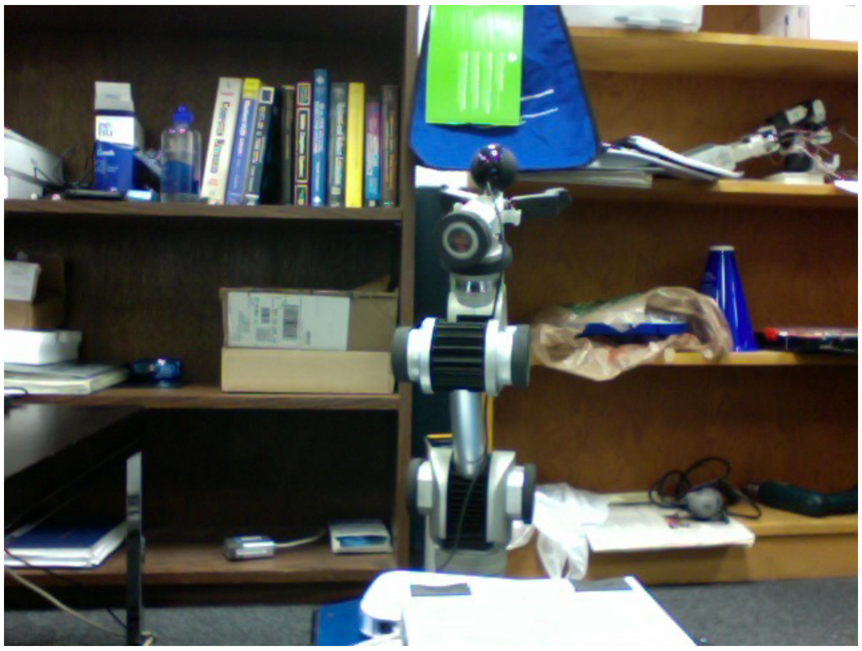


Figure 5.13. Initial configuration of the robot arm for experiment 3.

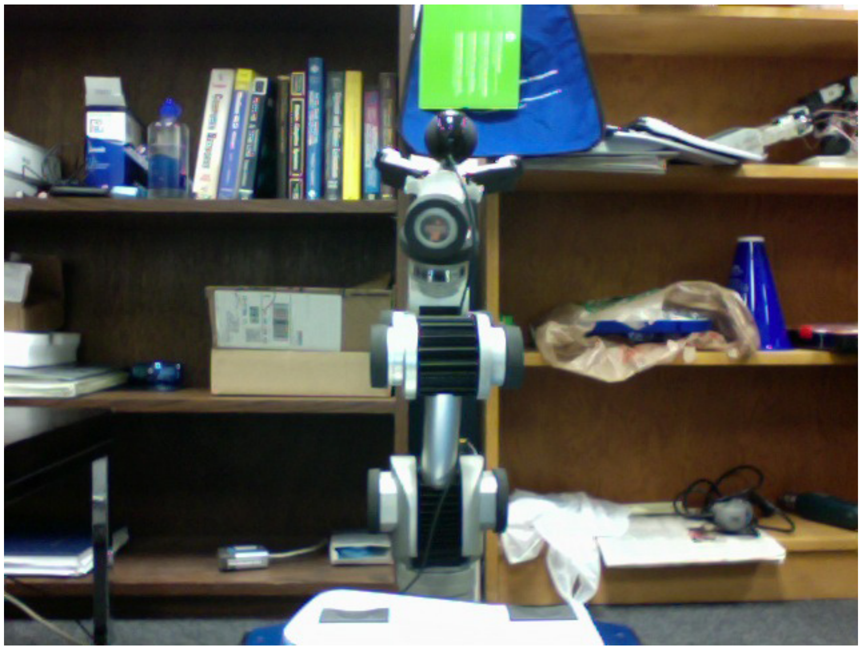


Figure 5.14. Final robot arm configuration after completing the task of the final episode.

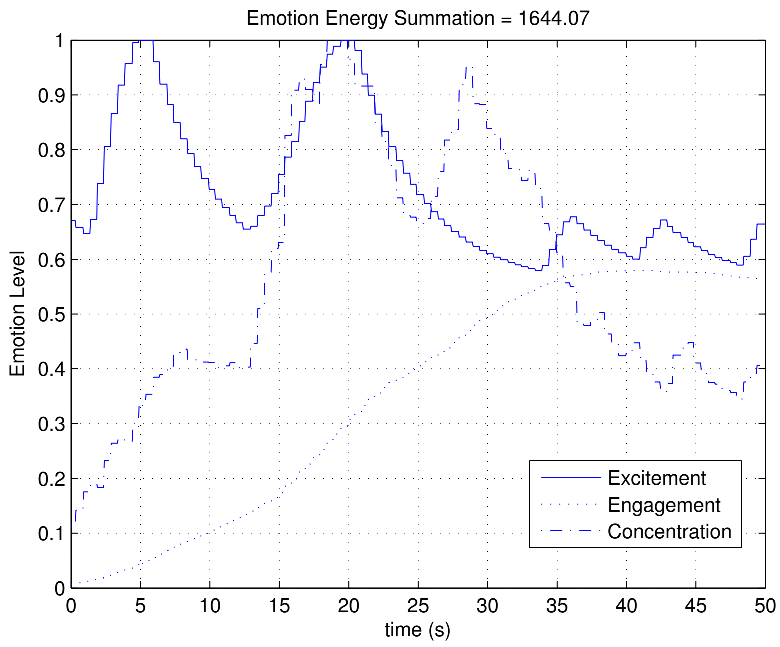


Figure 5.15. Experiment 3 - emotion level plots of the 11th episode.

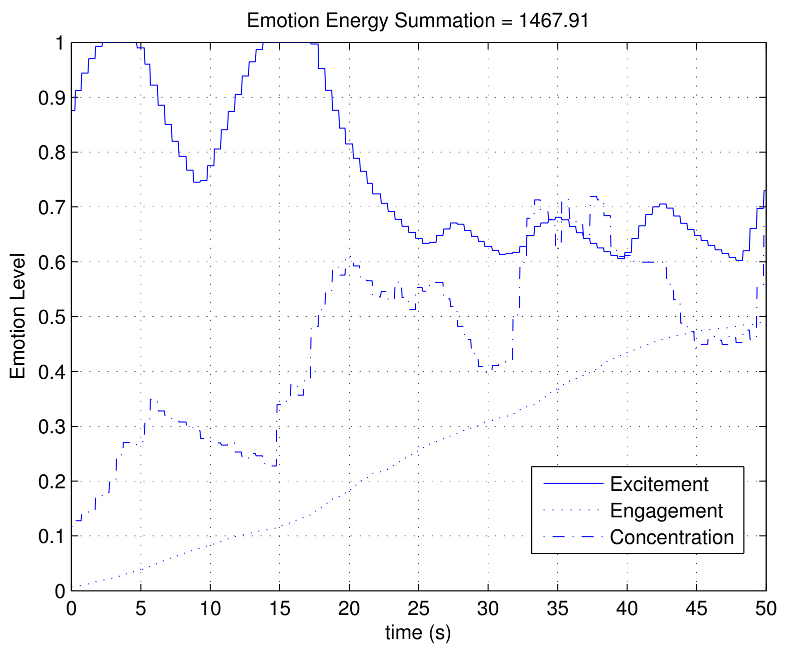


Figure 5.16. Experiment 3 - emotion level plots of the 21th episode.

an optimal mapping for another users. We also noticed that there is a learning on the user's part. By performing the same task repeatedly, the user would definitely go through some learning process regardless of updating of the mapping. This definitely has some effect on the convergence of the interface mapping. Further investigation on the convergence of the mapping and the decoupling of the user's learning and the learning of the interface mapping will be addressed in future work.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

6.1.1 Physical Interaction with Robots

In this part of the dissertation, we were interested in realism of physical interactions with robot manipulators. Motivating applications are safe robots, assistive robots and devices etc. The important element that the motivating applications have in common is the need to accurately sense the physical interaction force in all dimensions. With the introduction of the Extended Kalman Filter force estimation scheme, we are able to accurately estimate unmeasured force components. The estimated force information is used in conjunction with force control method, i.e., impedance control to perform assistive task by specifying the cartesian impedance of the robot manipulator.

The proposed Extended Kalman Filter force estimation eliminates the need for 3D force sensor at the interaction point. Along with the virtual end-effector concept, the interaction can occur at any point along the robot manipulator chain. The proposed HRI method would allow the user to physically interact with the robot in an intuitive manner. Assistive robots and physical guidance of the robot manipulator would now be possible without the need of force/ torque sensor, plus the interaction point is not only limited to just the end-effector of the robot manipulator as long as 1D force sensor is available.

6.1.2 Interaction with Robots through Vision

In this part of the dissertation, the focus was on the realism of interaction with robots through vision. This part mainly applies to humanoid robots. We were able to extract head pose information from a single camera by the extended Kalman filter estimation. We are also interested in generating realistic head-eye motion distribution for humanoid robot head doing human tracking. We proposed the use of two different methods to improve the realism of humanoid robots. The first method is the head-eye motion distribution using an online optimization approach. The other method is the reinforcement learning of the motor control of the humanoid robot head-eye mechanism. Both methods yield realistic motion of the humanoid robot tracking a person. The advantage of the first method is that it does not require learning but the down side is that it is more computationally expensive than the other method, as it needs to solve an optimization problem every time step. The second method does require some learning time at startup. With both of the methods for distributing humanoid robot head-eye motion, the schemes can be implemented on any humanoid robot, as both of the methods are independent of the robot head kinematics.

6.1.3 Interaction with Multiple Degrees of Freedom Robots

In this part of the work, we show that the interface mapping of an arbitrary system can improved toward the optimal mapping. We started with the assumption that the given interfaces were not the most intuitive interface devices to be used with the given tasks. Through the use of the proposed reinforcement learning algorithm, we were able to improve the performance of the task completion by updating the interface mapping. Experiment results show the effectiveness of the proposed algorithm through different indications. We can now use the proposed mapping update algorithm with a system that does not have good interface or we can map a simple interface to be used to control multiple robots.

6.2 Future Work

6.2.1 Physical Interaction with Robots

The effectiveness of the proposed method has been confirmed through simulations and experiments. In the experiment the virtual end-effector was fixed to just one location, we need to generalize the virtual end-effector point of interactions and make it so that multiple interactions can occur at multiple location simultaneously. Also, we need to further investigate the experimental use of an artificial skin sensor array with the proposed method on robot manipulators, when the skin sensor becomes available.

6.2.2 Interaction with Robots through Vision

Future work includes algorithm implementation on different robot platforms, and in actual conversational interaction without robot. Furthermore, facial expression synthesis should be implemented on the humanoid through the use of compliant skin to further enhance the realism of HRI.

6.2.3 Interaction with Multiple Degrees of Freedom Robots

From the experiments the interface mapping converges to certain values, but we have not really investigated the convergence of the proposed learning of the interface mapping algorithm. The reason was because the function that we are trying to optimize is not known and cannot be easily modeled that is why we use reinforcement learning approach to begin with. For future work, we need to investigate the convergence of the learning of the interface mapping. We also need to study the effect of human learning on the learning process of the interface mapping. Also, we need to implement the algorithm with different robot systems, and collect data from different users. Finally, we will work on improving computation efficiency of the algorithm. Incorporate more performance justification criteria into the algorithm, and incorporate the reward function in a more efficient manner.

APPENDIX A

PHANTOM OMNI HAPTICS DEVICE KINEMATICS AND DYNAMICS

In this appendix, we present kinematics and dynamics of the Phantom Omni haptics device.

A.1 Kinematics

We find the kinematics of this device by defining the twists, the first four twists are given by:

$$\begin{aligned}\omega_1 &= \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \omega_2 = \begin{bmatrix}-1 & 0 & 0\end{bmatrix}^T \omega_3 = \begin{bmatrix}-1 & 0 & 0\end{bmatrix}^T \omega_4 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \\ \nu_1 &= \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \nu_2 = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \nu_3 = \begin{bmatrix}0 & 0 & l_2\end{bmatrix}^T \nu_4 = \begin{bmatrix}0 & -l_3 & l_2\end{bmatrix}^T\end{aligned} \tag{A.1}$$

where d_i is the link length of the i^{th} joint. The transformation between base and virtual end-effector frames at $q = 0$ is given by:

$$g_{st}(0,p) = \begin{bmatrix} I_{3\times 3} & \begin{pmatrix} -a_4 \\ -(l_3+l_4) \\ l_2 \end{pmatrix} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{A.2}$$

where a_i is the link offset of the i^{th} joint. The homogeneous transformation of the Phantom Omni haptics device becomes:

$$g_{st}(q,p) = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

where

$$h_{11} = C_1C_4 + S_4S_1S_2S_3 - S_4S_1C_2C_3,$$

$$h_{12} = -S_1(C_2S_3 + S_2C_3),$$

$$h_{13} = C_1S_4 - C_4S_1S_2S_3 + C_4S_1C_2C_3,$$

$h_{14} = -a_4C_1C_4 - a_4S_4S_1S_2S_3 + a_4S_4S_1C_2C_3 + S_1S_2C_3l_3 + S_1S_2C_3l_4 + S_1C_2S_3l_3 + S_1C_2S_3l_4 + S_1C_2l_2$,

$h_{21} = -(C_2S_3 + S_2C_3)S_4$,

$h_{22} = C_2C_3 - S_2S_3$,

$h_{23} = (C_2S_3 + S_2C_3)C_4$,

$h_{24} = a_4S_4C_2S_3 + a_4S_4S_2C_3 - C_2C_3l_3 - C_2C_3l_4 + S_2S_3l_3 + S_2S_3l_4 + S_2l_2$,

$h_{31} = -S_1C_4 + S_4C_1S_2S_3 - S_4C_1C_2C_3$,

$h_{32} = -C_1(C_2S_3 + S_2C_3)$,

$h_{33} = -S_1S_4 - C_4C_1S_2S_3 + C_4C_1C_2C_3$,

$h_{34} = a_4S_1C_4 - a_4S_4C_1S_2S_3 + a_4S_4C_1C_2C_3 + C_1S_2C_3l_3 + S_1S_2C_3l_4 + S_1C_2S_3l_3 + C_1C_2S_3l_4 + C_1C_2l_2$,

and $S_i = sin(q_i)$, $C_i = cos(q_i)$

A.2 Dynamics

Using conventional Lagranges equation, we obtained dynamical model of the Phantom Omni haptics device as follow:

$$M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \tag{A.4}$$

where

$M_{11} = -m_4l_3^2C_3^2 + I_1 + m_2C_2^2l_2^2 + m_3C_2^2l_2^2 + m_3l_3^2 + m_4C_2^2l_2^2 + m_4l_3^2 - m_3l_3^2C_3^2 + 2m_4C_2l_2l_3S_3 + 2m_3C_2l_2l_3S_3$,

$M_{12} = M_{21} = 0$,

$M_{13} = M_{31} = 0$,

$M_{14} = M_{41} = m_4a_4(S_4C_1C_2l_2 + S_4l_3C_1S_3 - C_4S_1C_2l_2 - C_4l_3S_1S_3)$,

$M_{22} = l_2^2(m_2 + m_3 + m_4)$,

$M_{23} = M32 = l_2(-m_3S_2l_3C_3 + m_3C_2l_3S_3 - m_4S_2l_3C_3 + m_4C_2l_3S_3 + m_4C_2l_4S_4)$,

$M_{24} = M42 = -m_4l_2S_2a_4(S_1S_4 + C_1C_4)$,

$M_{33} = m_3l_3^2 + m_4l_3^2 + 2m_4l_3l_4 - 2m_4l_3l_4C_3^2 + m_4l_4^2 - m_4l_4^2C_3^2$,

$M_{34} = M_{43} = m_4l_3C_3a_4(S_1S_4 + C_1C_4)$,

$M_{44} = m_4a_4^2$,

$I_1 = 1.5$,

$l_2 = 0.135$,

$l_3 = 0.14$,

$l_4 = 0.1$,

$a_4 = 0.05$,

$m_2 = 0.25$,

$m_3 = 0.2$ and $m_4 = 0.1$.

The Coriolis term is defined as follow:

$$C(q,\dot{q}) = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \tag{A.5}$$

where

$C_1 = m_4a_4C_4\dot{q}_4^2C_1C_2l_2 + m_4a_4C_4\dot{q}_4^2l_3C_1S_3 + m_4a_4\dot{q}_4^2S_1C_2l_2 + m_4a_4S_4\dot{q}_4^2l_3S_1S_3 - 2m_4l_2^2C_2\dot{q}_1S_2\dot{q}_2 + 2m_4l_3^2C_3\dot{q}_3\dot{q}_1S_3 + 2m_4l_3C_3\dot{q}_3l_2C_2\dot{q}_1 - 2m_2l_2^2C_2\dot{q}_1S_2\dot{q}_2 - 2m_3l_2S_2\dot{q}_2l_3\dot{q}_1S_3 - 2m_3l_2^2C_2\dot{q}_1S_2\dot{q}_2 + 2m_3l_3C_3\dot{q}_3l_2C_2\dot{q}_1 + 2m_3l_3^2C_3\dot{q}_3\dot{q}_1S_3 - 2m_4l_2S_2\dot{q}_2l_3\dot{q}_1S_3$,

$C_2 = l_2(m_4C_2l_3\dot{q}_3^2 - m_4S_2S_1a_4C_4\dot{q}_4^2 + m_4C_2l_4C_3\dot{q}_3^2 + m_4S_2C_1a_4\dot{q}_4^2 + m_3C_2l_3\dot{q}_3^2 + m_3l_2S_2C_2\dot{q}_1^2 + m_3S_2l_3\dot{q}_1^2S_3 + m_3S_2l_3S_3\dot{q}_3^2 + m_2l_2S_2C_2\dot{q}_1^2 + m_4l_2S_2C_2\dot{q}_1^2 + m_4S_2l_3\dot{q}_1^2S_3 + m_4S_2l_3S_3\dot{q}_3^2)$,

$C_3 = -m_4 l_2 S_2 \dot{q}_2^2 l_4 S_3 + 2 m_4 l_3 C_3 \dot{q}_3^2 l_4 S_3 - m_4 l_3 C_1 C_3 a_4 S_4 \dot{q}_4^2 - m_3 l_3 S_3 l_2 S_2 \dot{q}_2^2 - m_3 l_3 C_3 l_2 C_2 \dot{q}_1^2 - m_3 l_3^2 C_3 \dot{q}_1^2 S_3 - m_4 l_3 C_3 l_2 C_2 \dot{q}_1^2 - m_4 l_3^2 C_3 \dot{q}_1^2 S_3 - m_1 l_3 C_3 l_2 C_2 \dot{q}_2^2 - m_4 l_3 C_3 l_2 C_2 \dot{q}_2^2 - m_4 l_3 S_3 l_2 \dot{q}_2^2 + m_4 l_4^2 C_3 \dot{q}_3^2 S_3 + m_4 l_3 S_1 C_3 a_4 C_4 \dot{q}_4^2$,

$C_4 = -m_4 a_4 (S_4 l_2 C_2 \dot{q}_2^2 S_1 + 2 S_4 l_2 S_2 \dot{q}_2 C_1 \dot{q}_1 + S_4 l_2 C_2 S_1 \dot{q}_1^2 + S_4 l_3 S_1 \dot{q}_1^2 S_3 - 2 S_4 l_3 C_1 \dot{q}_1 C_3 \dot{q}_3 + S_4 l_3 S_1 S_3 \dot{q}_3^2 + C_4 l_2 C_2 \dot{q}_2^2 C_1 - 2 C_4 l_2 S_2 \dot{q}_2 S_1 \dot{q}_1 + C_4 l_2 C_2 C_1 \dot{q}_1^2 + C_4 l_3 C_1 \dot{q}_1^2 S_3 + 2 C_4 l_3 S_1 \dot{q}_1 C_3 \dot{q}_3 + C_4 l_3 C_1 S_3 \dot{q}_3^2)$,

The gravity matrix is defined as follow:

$$G = \begin{bmatrix} 0 \\ g l_2 C_2 (m_2 + m_3 + m_4) \\ g S_3 (m_3 l_3 + m_4 l_3 + m_4 l_4) \\ 0 \end{bmatrix} \tag{A.6}$$

The joint viscous friction coefficient matrix is defined as follow:

$$D = \alpha I_4 \tag{A.7}$$

where $\alpha = 0.05$.

APPENDIX B

IMPLEMENTATION DIAGRAMS OF ALGORITHM 1 AND 2

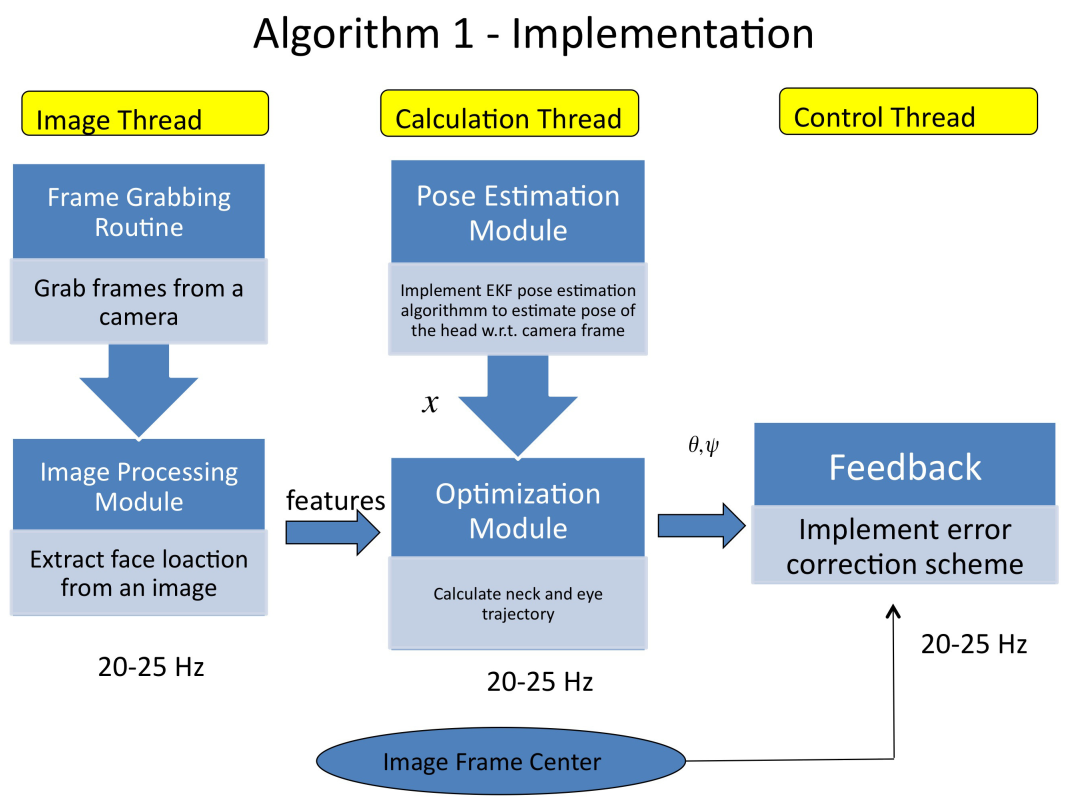

Algorithm 1 - Implementation
Image Thread
Calculation Thread
Control Thread
Frame Grabbing Routine
Grab frames from a camera
Pose Estimation Module
Implement EKF pose estimation algorithhmm to estimate pose of the head w.r.t. camera frame
x
θ, ψ
Image Processing Module
Extract face loaction from an image
features
Optimization Module
Calculate neck and eye trajectory
Feedback
Implement error correction scheme
20-25 Hz
20-25 Hz
20-25 Hz
Image Frame Center

Figure B.1. Implementation Diagram of the Algorithm 1.

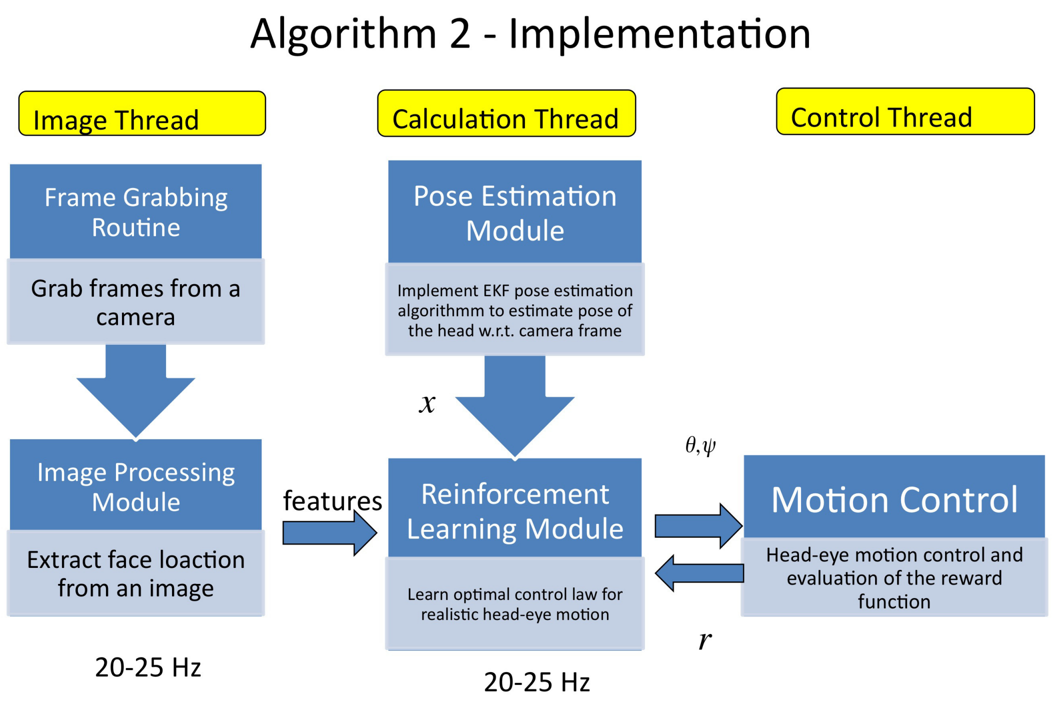

Algorithm 2 - Implementation
Image Thread
Calculation Thread
Control Thread
Frame Grabbing Routine
Grab frames from a camera
Pose Estimation Module
Implement EKF pose estimation algorithmm to estimate pose of the head w.r.t. camera frame
x
θ,ψ
Image Processing Module
Extract face loaction from an image
features
Reinforcement Learning Module
Learn optimal control law for realistic head-eye motion
Motion Control
Head-eye motion control and evaluation of the reward function
r
20-25 Hz
20-25 Hz

Figure B.2. Implementation Diagram of the Algorithm 2.

APPENDIX C

LIST OF WIDELY USED PERFORMANCE METRICS FOR ROBOT CONTROL APPLICATIONS

List of general human-robot performance metrics [86]

Navigation

- Global navigation
- Local navigation
- Obstacle encounter

Effectiveness measurements

- Percentage of navigation tasks successfully completed
- Coverage of area
- Deviation from planned route
- Obstacles that were successfully avoided
- Obstacles that were not avoided, but could be overcome

Efficiency measurements

- Time to complete the task
- Operator time for the task
- Average time for obstacle extraction

Workload measurements

- Number of operator interventions per unit time
- Ratio of operator time to robot time

Perception

- Passive Perception
  - Detection measures
  - Recognition measures
  - Absolute judgments of distance, size, or length
  - Relative judgments of distance, size, or length
  - Platform relative judgments
  - Absolute estimates of robot velocity

  - Estimates involving relative motion
- Active Perception
  - Efficiency
  - Effort
  - Detection accuracy for targets within sensor range
  - Efficiency as time to search or non-overlapping coverage
  - Coverage as percentage of potential sensor coverage
  - Operator confidence in sensor coverage
  - Efficiency
  - Identification errors
  - Degree of operator fusion

Management

- Fan out
- Intervention response time
- Level of autonomy discrepancies

Manipulation

- Degree of mental computation
- Contact errors

Social

- Interaction characteristics
- Persuasiveness
- Trust
- Engagement
- Compliance

System Performance

- Quantitative performance

  - Effectiveness
  - Efficiency
- Subjective ratings
- Appropriate utilization of mixed-initiative
  - Percentage of requests for assistance made by robot
  - Percentage of requests for assistance made by operator
  - Number of interruptions of operator rated as non-critical

Operator Performance

- Situation awareness
- Workload
- Accuracy of mental models of device operation

Robot Performance

- Self-awareness
- Human awareness
- Autonomy

List of supervisory control of multiple robots performance metrics [88]

- Interaction Efficiency
- Neglect Efficiency
- Attention Allocation Efficiency

List of human supervisory control metric classes and subclasses [90]

- Mission Effectiveness
- Automation Behavior Efficiency
- Human Behavior Efficiency
  - Attention allocation efficiency

  - Information processing efficiency
- Human Behavior Precursors
  - Cognitive precursors
  - Physiological precursors
- Collaborative Metrics
  - Human/automation collaboration
  - Human/human collaboration
  - Automation/automation collaboration

List of performance metric evaluation criteria [90]

- Experimental Constraints
- Comprehensive Understanding
- Construct Validity
- Statistical Efficiency
- Measurement Technique Efficiency

List of generalizable metric classes [91]

- Mission Effectiveness
- Human Behavior Efficiency
- Robot Behavior Efficiency
- Human Behavior Cognitive Precursors
- Human Behavior Physiological Precursors

Collaborative Metrics [91]

- Team Behavioral Action Efficiency
- Team Cognition Efficiency

- Robot Collaboration Efficiency

REFERENCES

[1] J. Ghan and H. Kazerooni, "System identification for the berkeley lower extremity exoskeleton (bleex)," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 3477 –3484.

[2] S. Sosnowski, K. Kuhnlenz, and M. Buss, "Eddie - an emotion-display with dynamic intuitive expressions," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, Sept. 2006, pp. 569 –574.

[3] C. Breazeal and B. Scassellati, "Challenges in building robots that imitate people," pp. 363–390, 2002.

[4] J.-H. Oh, D. Hanson, W.-S. Kim, Y. Han, J.-Y. Kim, and I.-W. Park, "Design of android type humanoid robot albert hubo," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 1428 –1433.

[5] T. Takubo, K. Inoue, T. Arai, and K. Nishii, "Wholebody teleoperation for humanoid robot by marionette system," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 4459 –4465.

[6] J. Rajruangrabin and D. Popa, "Enhancement of manipulator interactivity through compliant skin and extended kalman filtering," in *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, Sept. 2007, pp. 1111 –1116.

[7] J. Rajruangrabin, P. Dang, D. Popa, F. Lewis, and H. Stephanou, "Simultaneous visual tracking and pose estimation with applications to robotic actors," in

World Congress in Computer Science 2008. Proceedings, IPCV 08, July 2008, pp. 497–503.

[8] G. Cannata and M. Maggiali, "Implementation of listing's law for a tendon driven robot eye," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 3940 –3945.

[9] J. Rajruangrabin and D. O. Popa, "Realistic and robust head-eye coordination of conversational robot actors in human tracking applications," in *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '09. New York, NY, USA: ACM, 2009, pp. 1:1–1:7. [Online]. Available: http://doi.acm.org/10.1145/1579114.1579115

[10] J. Rajruangrabin and D. Popa, "Robot head motion control with an emphasis on realism of neckeye coordination during object tracking," *Journal of Intelligent & Robotic Systems*, pp. 1–28, 2010, 10.1007/s10846-010-9468-x. [Online]. Available: http://dx.doi.org/10.1007/s10846-010-9468-x

[11] J. Rajruangrabin and D. O. Popa, "Reinforcement learning of interface mapping for interactivity enhancement of robot control in assistive environments," in *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '10. New York, NY, USA: ACM, 2010, pp. 70:1–70:3. [Online]. Available: http://doi.acm.org/10.1145/1839294.1839377

[12] J. Gudi no-lau and M. A. Arteaga, "Dynamic model and simulation of cooperative robots: a case study," *Robotica*, vol. 23, no. 5, pp. 615–624, 2005.

[13] (2008) Robotics industry statistics. [Online]. Available: http://www.roboticsonline.com

[14] K. Ikuta and M. Nokata, "General evaluation method of safety for human-care robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3, 1999, pp. 2065 –2072 vol.3.

[15] E. L. Faulring, J. E. Colgate, and M. A. Peshkin, "The cobotic hand controller: Design, control and performance of a novel haptic display," *Int. J. Rob. Res.*, vol. 25, no. 11, pp. 1099–1119, 2006.

[16] Fre, J. My, F. Michaud, and M. Lauria, "Pushing a robot along - a natural interface for human-robot interaction," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3440 –3445.

[17] C. H. Park, J. W. Yoo, and A. Howard, "Transfer of skills between human operators through haptic training with robot coordination," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 229 –235.

[18] W. Park, S. Kwon, and J. Kim, "Real-time estimation of thumb-tip forces using surface electromyogram for a novel human-machine interface," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 205 –210.

[19] V. Lumelsky, M. Shur, and S. Wagner, "Sensitive skin," *Sensors Journal, IEEE*, vol. 1, no. 1, pp. 41 –51, June 2001.

[20] T. Someya, T. Sekitani, S. Iba, Y. Kato, H. Kawaguchi, and T. Sakurai, " A large-area, flexible pressure sensor matrix with organic field-effect transistors for artificial skin applications," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 27, pp. 9966–9970, 2004. [Online]. Available: http://www.pnas.org/content/101/27/9966.abstract

[21] H. Shinoda and H. Oasa, "Passive wireless sensing element for sensitive skin," in *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 2, 2000, pp. 1516 –1521 vol.2.

[22] Z. Jiang, K. Funai, M. Tanaka, and S. Chonan, "Development of Soft Tribo-Sensor Using PVDF Film for Skin Surface Contour Measurement," *Journal of Intelligent Material Systems and Structures*, vol. 10, no. 6, pp. 481–488, 1999. [Online]. Available: http://jim.sagepub.com/content/10/6/481.abstract

[23] V. Maheshwari and R. F. Saraf, "High-Resolution Thin-Film Device to Sense Texture by Touch," *Science*, vol. 312, no. 5779, pp. 1501–1504, 2006. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/312/5779/1501

[24] W. Stiehl and C. Breaeal, "A sensitive skin for robotic companions featuring temperature, force, and electric field sensors," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 1952 –1959.

[25] Y. Tadesse, S. Priya, H. Stephanou, D. Popa, and D. Hanson, "Piezoelectric actuation and sensing for facial robotics," *Ferroelectrics*, vol. 345, pp. 13–25, June 2006.

[26] H. Kazerooni, T. Sheridan, and P. Houpt, "Robust compliant motion for manipulators, part i: The fundamental concepts of compliant motion," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 2, pp. 83 – 92, June 1986.

[27] H. Kazerooni, "Direct-drive active compliant end effector (active rcc)," *Robotics and Automation, IEEE Journal of*, vol. 4, no. 3, pp. 324 –333, June 1988.

[28] H. Kazerooni and T. I. Tsay, "Control and stability analysis of cooperating robots," in *American Control Conference, 1988*, June 1988, pp. 1382 –1386.

[29] H. Kazerooni and T. Tsay, "Compliance control and unstructured modeling of cooperating robots," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, Apr. 1988, pp. 510 –515 vol.1.

[30] H. Kazerooni, “Stability and performance of robotic systems worn by humans,” in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, May 1990, pp. 558 –563 vol.1.

[31] ——, “Robust, non-linear impedance control for robot manipulators,” in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, Mar. 1987, pp. 741 – 750.

[32] J. Wen, D. Popa, G. Montemayor, and P. Liu, “Human assisted impedance control of overhead cranes,” in *Control Applications, 2001. (CCA ’01). Proceedings of the 2001 IEEE International Conference on*, 2001, pp. 383 –387.

[33] H. Krebs and N. Hogan, “Therapeutic robotics: A technology push,” *Proceedings of the IEEE*, vol. 94, no. 9, pp. 1727 –1738, Sept. 2006.

[34] S. Singh and D. Popa, “An analysis of some fundamental problems in adaptive control of force and impedance behavior: theory and experiments,” *Robotics and Automation, IEEE Transactions on*, vol. 11, no. 6, pp. 912 –921, Dec. 1995.

[35] S. Kobayashi, A. Muis, and K. Ohnishi, “Sensorless cooperation between human and mobile manipulator,” in *Industrial Technology, 2005. ICIT 2005. IEEE International Conference on*, Dec. 2005, pp. 811 –816.

[36] H. Kazerooni, D. Fairbanks, A. Chen, and G. Shin, “The magic glove,” in *Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004 IEEE International Conference on*, vol. 1, Apr. 2004, pp. 757 – 763 Vol.1.

[37] H. Wang, K. Low, and M. Wang, “Combined impedance/direct control of robot manipulators,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 3605 –3610.

[38] K. Reed, M. Peshkin, M. Hartmann, J. Patton, P. Vishton, and M. Grabowecky, “Haptic cooperation between people, and between people and machines,” in

Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, Oct. 2006, pp. 2109 –2114.

[39] S. Muller-Schneiders, T. Jager, H. Loos, and W. Niem, "Performance evaluation of a real time video surveillance system," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, Oct. 2005, pp. 137 – 143.

[40] P. Oh and R. Stanciu, "Visual servoing to help camera operators track better," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 7 –7.

[41] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82 –90, Dec. 2006.

[42] ——, "Visual servo control. ii. advanced approaches [tutorial]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 109 –118, Mar. 2007.

[43] E. Malis, F. Chaumette, and S. Boudet, "2d 1/2 visual servoing stability analysis with respect to camera calibration errors," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 2, Oct. 1998, pp. 691 –697 vol.2.

[44] K. H. An and M. J. Chung, "Robust unified stereo-based 3d head tracking and its application to face recognition," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4056 –4061.

[45] P. Mittrapiyanuruk, G. DeSouza, and A. Kak, "Calculating the 3d-pose of rigid-objects using active appearance models," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, May 2004, pp. 5147 – 5152 Vol.5.

[46] P. Wunsch, S. Winkler, and G. Hirzinger, "Real-time pose estimation of 3d objects from camera images using neural networks," in *Robotics and Automation,*

1997. Proceedings., 1997 IEEE International Conference on, vol. 4, Apr. 1997, pp. 3232 –3237 vol.4.

[47] M. Abidi and T. Chandra, "Pose estimation for camera calibration and landmark tracking," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, May 1990, pp. 420 –426 vol.1.

[48] T. Shakunaga, "An object pose estimation system using a single camera," in *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, vol. 2, July 1992, pp. 1053 –1060.

[49] J. Wang and W. Wilson, "3d relative position and orientation estimation using kalman filter for robot control," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 2638 –2645 vol.3.

[50] Y. K. Yu, K. H. Wong, and M. Chang, "Recursive three-dimensional model reconstruction based on kalman filtering," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 3, pp. 587 –592, June 2005.

[51] V. Lippiello, B. Siciliano, and L. Villani, "Position and orientation estimation based on kalman filtering of stereo images," in *Control Applications, 2001. (CCA '01). Proceedings of the 2001 IEEE International Conference on*, 2001, pp. 702 –707.

[52] M. Ficocelli and F. Janabi-Sharifi, "Adaptive filtering for pose estimation in visual servoing," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 2001, pp. 19 –24 vol.1.

[53] Y. Kay and S. Lee, "A robust 3-d motion estimation with stereo cameras on a robot manipulator," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, Apr. 1991, pp. 1102 –1107 vol.2.

[54] V. Lippiello, B. Siciliano, and L. Villani, "3d pose estimation for robotic applications based on a multi-camera hybrid visual system," in *Robotics and Au-*

tomation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, May 2006, pp. 2732 –2737.

[55] F. Lewis, *Optimal Estimation.* New York: John Wiley & Sons, 1986.

[56] H. Yoon, D. Kim, S. Chi, and Y. Cho, "A robust human head detection method for human tracking," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 4558 –4563.

[57] H. Ahn, D. Kim, J. Lee, S. Chi, K. Kim, J. Kim, M. Hahn, and H. Kim, "A robot photographer with user interactivity," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 5637 –5643.

[58] W. Wilson, C. Williams Hulls, and G. Bell, "Relative end-effector control using cartesian position based visual servoing," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 684 –696, Oct. 1996.

[59] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 932 –946, July 2002.

[60] P. Martinet and J. Gallice, "Position based visual servoing using a non-linear approach," in *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 1, 1999, pp. 531 –536 vol.1.

[61] J. Armstrong Piepmeier, G. McMurray, and H. Lipkin, "A dynamic quasi-newton method for uncalibrated visual servoing," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1595 –1600 vol.2.

[62] J. Pages, C. Collewet, F. Chaumette, and J. Salvi, "Plane-to-plane positioning from image-based visual servoing and structured light," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, Sept. 2004, pp. 1004 – 1009 vol.1.

[63] K. F. MacDorman and T. Minato. (2005) Cogsci-2005 workshop: Toward social mechanisms of android science. [Online]. Available: http://www.androidscience.com/theuncannyvalley/proceedings2005/uncannyvalley.html

[64] M. Mori, "Bukimi no tani (the uncanny valley) (in japanese see [macdorman 05] for translation)," *Energy*, vol. 7, pp. 33 –35, 1970.

[65] C. Bartneck, T. Kanda, H. Ishiguro, and N. Hagita, "Is the uncanny valley an uncanny cliff?" in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, Aug. 2007, pp. 368 –373.

[66] M. Shimada, T. Minato, S. Itakura, and H. lshiguro, "Uncanny valley of androids and its lateral inhibition hypothesis," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, Aug. 2007, pp. 374 –379.

[67] S. Woods, K. Dautenhahn, and J. Schulz, "The design space of robots: investigating children's views," in *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*, Sept. 2004, pp. 47 – 52.

[68] J. Goetz, S. Kiesler, and A. Powers, "Matching robot appearance and behavior to tasks to improve human-robot cooperation," in *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, Nov. 2003, pp. 55 – 60.

[69] B. Scassellati, "Investigating models of social development using a humanoid robot," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 4, July 2003, pp. 2704 – 2709 vol.4.

[70] D. Omrc anden and A. Ude, "Redundant control of a humanoid robot head with foveated vision for object tracking," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4151 –4156.

[71] N. Mitsunaga, T. Miyashita, H. Ishiguro, K. Kogure, and N. Hagita, "Robovie-iv: A communication robot interacting with people daily in an office," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 5066 –5072.

[72] D. H. Kim, S. U. Jung, K. H. An, H. S. Lee, and M. J. Chung, "Development of a facial expression imitation system," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 3107 –3112.

[73] C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati, "Active vision for sociable robots," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 31, no. 5, pp. 443 –453, Sept. 2001.

[74] S. Gurbuz, T. Shimizu, and G. Cheng, "Real-time stereo facial feature tracking: mimicking human mouth movement on a humanoid robot head," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, Dec. 2005, pp. 363 –368.

[75] K. Harada, K. Hauser, T. Bretl, and J.-C. Latombe, "Natural motion generation for humanoid robots," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 833 –839.

[76] S. Singh, S. Pieper, D. Popa, and J. Guinness, "Control and coordination of head, eyes and facial expressions of virtual actors in virtual environments," in *Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on*, Nov. 1993, pp. 335 –339.

[77] P. Sharkey, D. Murray, and J. Heuring, "On the kinematics of robot heads," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 3, pp. 437 –442, June 1997.

[78] O. Deniz, M. Castrillon, J. Lorenzo, C. Guerra, D. Hernandez, and M. Hernandez, "Casimiro: a robot head for human-computer interaction," in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, 2002, pp. 319 – 324.

[79] K. Berns and T. Braum, "Design concept of a human-like robot head," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, Dec. 2005, pp. 32 –37.

[80] L. Gu and J. Su, "Gaze control on humanoid robot head," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 2, 2006, pp. 9144 –9148.

[81] G. Cannata, M. D'Andrea, and M. Maggiali, "Design of a humanoid robot eye: Models and experiments," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, Dec. 2006, pp. 151 –156.

[82] K. Itoh, H. Miwa, Y. Nukariya, M. Zecca, H. Takanobu, S. Roccella, M. Carrozza, P. Dario, and A. Takanishi, "Development of a bioinstrumentation system in the interaction between a human and a robot," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 2620 –2625.

[83] I. Iturrate, L. Montesano, and J. Minguez, "Robot reinforcement learning using eeg-based reward signals," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4822 –4829.

[84] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2397 –2403.

[85] T. Hester, M. Quinlan, and P. Stone, "Generalized model learning for reinforcement learning on a humanoid robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2369 –2374.

[86] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common metrics for human-robot interaction," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, ser. HRI '06. New York, NY, USA: ACM, 2006, pp. 33–40. [Online]. Available: http://doi.acm.org/10.1145/1121241.1121249

[87] J. Drury, J. Scholtz, and H. Yanco, "Awareness in human-robot interactions," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 1, Oct. 2003, pp. 912 – 918 vol.1.

[88] J. W. Crandall and M. L. Cummings, "Developing performance metrics for the supervisory control of multiple robots," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, ser. HRI '07. New York, NY, USA: ACM, 2007, pp. 33–40. [Online]. Available: http://doi.acm.org/10.1145/1228716.1228722

[89] J. Crandall and M. Cummings, "Identifying predictive metrics for supervisory control of multiple robots," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 942 –951, Oct. 2007.

[90] B. Donmez, P. E. Pina, and M. L. Cummings, "Evaluation criteria for human-automation performance metrics," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS

’08. New York, NY, USA: ACM, 2008, pp. 77–82. [Online]. Available: http://doi.acm.org/10.1145/1774674.1774687

[91] P. Pina, M. L. Cummings, J. W. Crandall, and M. D. Penna, “Identifying generalizable metric classes to evaluate human-robot teams,” in *HRI Workshop on Performance Metrics for Human-Robot Interaction*, 2008.

[92] A. Bechar, Y. Edan, and J. Meyer, “An objective function for performance measurement of human-robot target recognition systems in unstructured environments,” in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 1, Oct. 2004, pp. 118 – 123 vol.1.

[93] J. C. Maida, C. K. Bowen, and J. Pace, “Improving robotic operator performance using augmented reality,” *Human Factors and Ergonomics Society Annual Meeting Proceedings*, vol. 51, pp. 1635–1639(5), 1 September 2007.

[94] M. Goodrich and J. Olsen, D.R., “Seven principles of efficient human robot interaction,” in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 4, Oct. 2003, pp. 3942 – 3948 vol.4.

[95] J. Saleh and F. Karray, “Towards generalized performance metrics for human-robot interaction,” in *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, June 2010, pp. 1 –6.

[96] K. F. Kraiss, *Advanved Man-Machine Interaction: Fundamentals and Implementation*. Springer, 2006.

[97] Y.-S. Ryu and S.-Y. Oh, “Automatic extraction of eye and mouth fields from a face image using eigenfeatures and ensemble networks,” *Applied Intelligence*, vol. 17, no. 2, pp. 171–185, 2002.

[98] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” pp. 726–740, 1987.

[99] P. Dang, H. Stephanou, F. Ham, and F. Lewis, “Facial expression recognition using a two stage neural network,” in *Control Automation, 2007. MED ’07. Mediterranean Conference on*, June 2007, pp. 1 –7.

[100] Y. Wu and Z. Hu, “Pnp problem revisited,” *J. Math. Imaging Vis.*, vol. 24, no. 1, pp. 131–141, 2006.

[101] C. Li, B. Liang, and W. Xu, “Autonomous trajectory planning of free-floating robot for capturing space target,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 1008 –1013.

[102] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: Bradford Books, MIT Press, 2002.

[103] J. Tsitsiklis and B. Van Roy, “An analysis of temporal-difference learning with function approximation,” *Automatic Control, IEEE Transactions on*, vol. 42, no. 5, pp. 674 –690, May 1997.

[104] C. Setz, B. Arnrich, J. Schumm, R. La Marca, G. Troster, and U. Ehlert, “Discriminating stress from cognitive load using a wearable eda device,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 2, pp. 410 –417, Mar. 2010.

BIOGRAPHICAL STATEMENT

Jartuwat Rajraungrabin was born in Bangkok, Thailand, in 1981. He earned his B.S. degree in Control Engineering from King Mongkuts Institute of Technology, Bangkok, in 2002, and M.S. in Electrical Engineering from the University of Southern California, Los Angeles, in 2005. He received his Ph.D. degree in Electrical Engineering from The University of Texas at Arlington in 2010.