AUGMENTED REALITY TECHNIQUES IN ROBOTICS AND THEIR .NET IMPLEMENTATIONS

by

Yusuf ADIBELLİ

August 2007

AUGMENTED REALITY TECHNIQUES IN ROBOTICS AND THEIR .NET IMPLEMENTATIONS

by

Yusuf ADIBELLİ

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electronics Engineering

August 2007
Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Muhammet KÖKSAL
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Onur TOKER
Supervisor

Examining Committee Members

Prof. Dr. Muhammet KÖKSAL

Assoc. Prof. Dr. Onur TOKER

Assoc. Prof. Dr. Halil Rıdvan ÖZ

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assist. Prof. Dr. Nurullah ARSLAN
Director

Date
August 2007

AUGMENTED REALITY TECHNIQUES IN ROBOTICS AND THEIR .NET IMPLEMENTATIONS

Yusuf ADIBELLİ

M. S. Thesis - Electronics Engineering
August 2007

Supervisor: Assoc. Prof. Dr. Onur TOKER

ABSTRACT

Telerobotics is a scheme that allows humans to extend their manipulative skills over a network by combining human's cognitive skills and robot's specialized working abilities. Efficient control of the robot over LAN in the presence of time delays and data loss is a dynamic research field. The purpose of this work is to implement a reliable teleoperation of Mitsubishi RV-2AJ robot over a LAN. In order to pursue this goal, a completely distributed telerobotic framework is developed using .NET Remoting technology to provide multithreaded environment for real-time interaction between client and server side components. Computer vision and force feedback techniques are implemented and their performance analysis are evaluated in order to enhance the maneuverability of the operator telemanipulating the robot.

Keywords: Computer Vision, Telerobotics, Augmented Reality, Multithreaded, Force Feedback

ROBOTİKTE SANAL + GERÇEK DÜNYA TEKNİKLERİ VE BUNLARIN .NET UYGULAMALARI

Yusuf ADIBELLİ

Yüksek Lisan Tezi – Elektronik Mühendisliği
Ağustos 2007

Tez Yöneticisi: Doç. Dr. Onur TOKER

ÖZ

Telerobotik insanın bilgiye ve idrake dayalı becerileri ile robotun uzmanlaşmış çalışma yeteneklerini birleştirerek insanların ağ üzerinden kullanma kaabiliyetini genişleten bir sistemdir. Robotun LAN üzerinden var olan zaman gecikmeleri ve veri kayıplarına rağmen verimli kontrolu dinamik araştırma alanlarındandır. Bu çalışmada Mitsubishi RV-2AJ robotun LAN üzerinden güvenli bir şekilde kullanımı amaçlanmaktadır. Bu amacı gerçekleştirmek için de; client ve server bileşenleri arasında gerçek zamanlı bir etkileşim için multithread yapılı bir ortam oluşturulması, tamamen dağıtılmış telerobotik yapısı .NET technolojisi kullanılarak geliştirilmiştir. Görüntü işleme ve geribesleme etkisi teknikleri uygulanmış ve operatorün robotu elle kullanım manevra kaabiliyetini artırmak için bu tekniklerin performans analizleri yapılmıştır.

Anahtar Kelimeler: Görüntü İşleme, Telerobotik, Artırılmış Gerçeklik, Multithread, Geri Besleme Etkisi

DEDICATION

Dedicated To My Parents

ACKNOWLEDGEMENT

This dissertation could not have been written without Assoc. Prof. Dr. Onur TOKER who not only served as my supervisor but also encouraged and challenged me throughout my academic program. He patiently guided me through the dissertation process, never accepting less than my best efforts. I thank him all.

I also express gratitude to Research Assistants Cihan Ulaş, Melek Oktay, Mustafa Sarıöz and Taner Çevik who have previous works and creative ideas on this thesis subject.

Thanks go to the committee member Assist. Prof. Tuğrul Yanık for his valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, motivation and patience.

At last but not least, I want to thank my friends Dinçer Özer, Ergun Şakalar, and Cemil Dizman with their encouragements. And I am thankful especially to Sinan Kocaman with his extraordinary understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
Q.o.S	Quality of Service
GPS	Global Positioning System
LAN	Local Area Network
MAN	Metropolitan Area Network
MOMR	Multi-Operator-Multi-Robot
SOSR	Single-Operator-Single-Robot
OS	Operating System
GUI	Graphical User Interface
VR	Virtual Reality
VE	Visual Enhancements
TCP/IP	Transmission Control Protocol / Internet Protocol
ATM	Asynchronous Transfer Mode
HIC	Human Interactive Computer
TIC	Interactive Computer
HMD	Head Mounted Display
DOF	Degree Of Freedom
BBS	Bulletin Board System
RPC	Remote Procedure Call
OMG	Object Management Group
IDL	Interface Definition Language
COM	Component Object Model
SOAP	Simple Object Access Protocol

CHAPTER 1

INTRODUCTION

Robots are employed to make exacting routine works, alternating from the common place to difficult and from the relatively safe to the highly dangerous. The idea "human supervisory control" has the potential to bring robotics out of the laboratory and into the difficult and messy world. Traditionally, researchers have focused on automation which seeks to build all necessary capabilities into a machine to achieve a particular task, including sensors, actuators, servos and algorithms. In contrast to automatic control, supervisory control puts a human back "in the loop". In the supervisory control systems, human and machine work together.

Remote-controlled robots or teleoperation -manipulation of an object (in this case a robot) - is one way to combine the intelligence and maneuverability of human beings with the precision and durability of robots (A. Monferer, 2002). Therefore, a telerobot can be defined as a electromechanical tool or device containing sensors and actuator that effectively extends an operator's sensorimotor system, perceiving and manipulative ability, to a remote environment. Shortly, when the operator and the machine are not collocated but are connected by some communication channel, the system said to be telerobotic. Particularly, a telerobot system consists of (Telerobot System, 2004):

i. A master arm connected to a client
ii. A slave arm connected to a server station,
iii. A-stereo vision system to provide 3D views of slave-remote-scene (a visual sensor, an image processing system, and system controller).

Teleoperated and under supervisory control robots have been used in a variety of application areas in maintenance and repair including those which are (Nof, 1998):

a) Hazardous to humans such as
   i. nuclear decommissioning, inspection, and waste handling
   ii. bomb disposal and minefield clearance
   iii. unmanned underwater inspection, search and rescue
   iv. Power Line Maintenance

b) Humans adversely affect the environment such as
   i. Medical applications
   ii. Clean room operations

c) Impossible for humans to be satiated in such as
   i. Deep space
   ii. Nano robotics

Depending on the technological developments, especially, in robotic explorations, robotics is finding many applications in these areas. In addition, removing the human operators from the operation sites is an effective way to save the human lives and reduce the production costs. However, in most of these areas, we need humans in the control loop because of their high level skills (intelligence and maneuver ability). Also, machine technology advanced insufficiently to operate autonomously and intelligently in complex, unstructured and often cluttered environments.

In mechanical, electrical, computer and control systems engineering, telerobotics has become the most rapidly developing area due to the utilization of robots in many industries and, in addition, robots ensure many advantages such as being able to perform set routines with less cost, more quickly and accurately than humans. Instead of using routine programs in maneuvering the robots, telerobotics technology allows human to operate robots from a remote world and helps human to make decisions while telemanipulating them in real time. Future of the telerobotics seems extremely promising depending on the development of more powerful and efficient computers. However, flexibility problem with which these teleoperated robots can be used is a great concern to both users and telerobotic researchers even if having a positive effect of technological development.

Compensation of time delay between operator and telerobot interface is an active research area in telerobotics. It is a great obstruction for manual control of the remote manipulator if time delay between the control input given by the operator and the consequent feedback of its control actions visible on the display is in sense. When control loop’s time delay exceeds half of the time period at a particular frequency, a continuous closed-loop control becomes unstable at that particular frequency. There are many examples which have been shown in literature when the human operator in the control loop can avoid such kind of instability by using a “move and wait strategy” that the operator makes small incremental moves in an open loop fashion and then waits for a new update of the position of the telerobot. Reasons of the time delay that occurs in communication are:

i. Large distance between local and remote site,
ii. A low speed of data transmission,
iii. Computers are processing at a different stages,
iv. All of the above.

For example, continuous teleoperation in earth orbit or deep space by human operators on the earth’s surface is seriously impeded by signal transmission delays. Imposed this time delay is due to the limits on the speed of light (radio transmission) and computer processing at sending and receiving stations and satellite relay stations. For vehicles in low earth orbit, round-trip delays (the time from sending a discrete signal until any receipt of any feedback pertaining to the signal) are minimally 0.4 s; for vehicles on or near the moon these delays are typically 3 s. But in reality a round trip time delay of up to 6 seconds is common, owing to multiple reflections of signal through the satellites. For the underwater teleoperation which is used having a 1700m/s speed sonar signals in water for data transmission, the remote manipulator is not directly connected with cables to the controlling site. A round trip time-delay of 10 seconds is common for teleoperation in deep sea (Sheridan, 1992).

Further from the communication speed and the distance between the remote and local sites, appreciable time delay occurs due to the signal processing and data storage in computer buffers at various stages between the local and the remote sites.

Moreover, additional problems caused by the digital communication channels such as internet is an extra uncertainty in the actual magnitude of the time delay. Decreasing the existence of the problems like time delay and data losses that telerobotic have been facing today is possible if a dedicated channel is used for communication. Dedicated medium, commonly, is not possible and feasible because of the high economic costs. Having a dedicated medium, always, is not a perfect solution that eliminating time delay problem. For example, time delay is inevitable for satellite operations because of the large distances.

Another problem that is faced with in many practical situations in digital communication is bandwidth issue because of not having a direct cabling between remote and local world. Many digital communication devices such as modems have a lower bandwidth. Using internet over a modem as a communication channel imposes several limitations on bandwidth. For example, the present inventions of modems allow operating in the voice band from 300 to 3400 Hz, and also in the ADSL band, which extends, nearly, above 3400 Hz (Bellenger and Russell, 1999). Due to the requirement of the very high bandwidth, the most difficult signal to transmit is video signal. 30 Megabyte/second bandwidth is necessary to transfer an uncompressed standard video signal. Streaming of this type of data can be supported by using Local Area Network (LAN).

Lower bandwidth causes to decrease the video signal quality available at the display because of the drop of the rate at which:

i. Frame Rate, the rate at which video frames are displayed on a monitor per second
ii. Display Resolution, number of pixels (or maximal image resolution) that can be displayed on the screen
iii. Grayscale, the range of gray tones between black and white as displayed on a monitor or in an image

Depending on the quality of the video signal, operator will also be affected adversely by decrease in the frame rate, resolution and grayscale (Sheridan, 1992).

Using telepresence system that is receiving sensory information or feedback from the robot that creates a sense of being present at the remote site and allows a satisfactory degree of technical performance usually helps to increase operator's performance on his/her tasks. Being able to interpret the remote scene and undertake the task in the remote environment effectively and efficiently is accomplished by having sufficient visual information. Because telepresence system displays high quality visual information about the remote world. Therefore, operator while he is in local environment feels physically present in the remote world. Position of the cameras is another important factor to get reality of remote world as visual information.

There are three principals and independent determinants of the sense of presence or the state of being present in a remote environment in the telerobotic literature such as:

i. Having an ability to modify the remote environment (to be able to change objects in the remote environment or their relationship to one another)
ii. The extent of the sensory information (same level sensor information that operator would have if sensors were physically in the remote environment)
iii. The control of the sensors (being able to modify or change the position of the sensing device)

Using single camera to take pictures from a single side does not increase the user's perception on visual scene. It is not possible to know the real sizes of the objects by observing them from a single point. Three-Dimensional (3D) positions of points can be estimated only by observing in at least two images that are taken from slightly different viewing angles. This is the reason of rising of the stereo vision term that is process of combining multiple images of a scene to obtain 3D geometric information. Generating 3D images increases operator's perception by providing the illusion of depth, height, shading and perspective within the scene. The most stereo images are generated by using only two images or a pair of cameras.

According to above discussion, it is obvious that the development in telerobotic is restricted by absence of an economical high bandwidth communication medium and as well as the unavailability of a Quality of Service (Q.o.S) that capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks for this real-time communication (Cisco, 2007). Due to provide an effective, efficient and precise real-time interface in a local-remote environment depending on the stereo views of the remote world, it is necessary and undeniable to have a high-band width communication channel and guaranteed Q.o.S.

1.1 Thesis Objectives

The dissertation "Augmented reality techniques and their .NET application" presents a telerobotic system (Kazerooni et al., 1993)-(Sooyong et al., 1998) that consists of a vision-based client station (operator) and server station (slave robot arm) which are interconnected by a computer network connected through a 100 Mbps Ethernet LAN. The client-server (operator and slave arm) system implemented using a robust *Visual Studio .NET with C# (VS.NET/C#)* programming environment together with the TCP/IP (Transmission Control Protocol & Internet Protocol) socket programming to provide real-time connectivity through the LAN.

A real-time vision system based on two similar web cameras (WebCams) monitor the robot arm motion to control a tele-robot. To get a stereo illusion, stereo visualization based on the views of the left and right camera images of the robot is generated after proper synchronization. Eye shuttering glasses, to view the robot scene, are used to see remote environment in three dimensions from the client station.

The thesis work is divided into two major categories:

1. Telerobotics
2. Computer Vision

1.1.1 Telerobotics

A Reliable Telerobotic System:

Developing a fully distributed Object Oriented approach to implement Robot-Server-Client interaction (Al-Harthy, 2002).

Improvement on joystick-robot interaction performance by using extrapolation algorithms.

1.1.2 Computer Vision

Development of the client-server framework and grabbing, transferring and displaying of 3D stereo data over a Local Area Network. For the 3D effect on client side, different methods used to generate 3D image such as sync-doubling, line blanking and page flipping.

1.2 Organization of the Work

Literature survey relevant to thesis objectives will be given briefly in chapter 2. Robot fundamentals are given and Mitsubishi RV-2AJ is introduced in detail in Chapter 3. A Multi-Thread Distrubuted Telerobotic Framework is present in Chapter 4 and, also, backbone of the Telerobotic Control System is explained in detail. Chapter 5 gives you an idea about performance evaluation of the used prediction algorithm. Augmented Reality system for the developed telerobotic framework will be given in Chapter 6. We conclude in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

The literature review is subdivided into two main categories based on the two outstanding areas of this work:

1. Network Telerobotic
2. Stereo Vision and Augmented Reality

2.1 Network Telerobotic

Research on field of robots has begun to increase and gains a momentum in recent years due to a number of rising technologies such a Global Positioning System (GPS), small computers and machine vision. These robots operating under ambiguous and unstructured environment assure to allow performing real tasks with much greater safe and comfort than is presently possible. Demand of computer network based telerobotic system is increasing due to the high costs of dedicated communication links between local and remote environments. Network Telerobotics is a natural product of this demand. It primarily deals with the topics related to the utilization of a computer network such as LAN/WAN, for the development of extremely efficient telerobotic systems.

Paolucci et al. (Paolucci et al., 1996) discussed a teleoperation using packed switched computer networks as a communication resource. Experiments were carried out with different kinds of computer networks considering with different types of data traffic such that varying values of data loss, delay and jitter to evaluate the performance of teleoperation system. Two different PC were connected to a Network using an Ethernet Adapter. Two kinds of network were considered: an Ether-net based Local Area Network (LAN) and a Metropolitan Area Network (MAN). It is shown that when the packed size is increased from 64 to 1024 bytes, the network delay is also increased from a mean value of 5.6 ms to 13.4 ms with a minimum value of delay equal to 5.4 ms due to the computational overheads for LAN and 30.2 ms to 90 ms due to the routing algorithm and queuing. LAN performs well even in the presence of traffic caused by other users until the total network congestion, which, of course, causes to system to be completely unpredictable. But even with a better performance, Q.o.S guarantee cannot be provide for LAN and Internet. Random time delays occur due to the absence of Q.o.S. a real time process can go unstable when the time delay exceeds a certain limit. The performance is more degraded with added delays and jitter on MAN possibly presence of different routers and queuing algorithms.

Teleoperation performance was tested on a network simulator. An important result is that the operator performance is quiet insensitive to a fairly small data loss due to the high sampling rate with respect to data frequency contents. In addition, if the same quantity of data is supplied but spaced at regular intervals, increase in the operator performance is observed.

Introduction of time delay causes an almost linearly decrease in operator performance. Jitter produces a disturbance in velocity due to varying intervals between samples. Introduction of a buffer can decrease the jitter but increase the cost of the delay. A tradeoff can be negotiated between the two parameters. To get better performance out of the telerobotic system, a new predictive algorithm is applied to utilize the model of the actuator. The model is located at both master and slave sites. Master site model gives immediate visual feedback to operator to enhance his performance while the slave site model is used to periodically update the parameters of the actuator by comparing the predictive and actual outputs. Actuator dynamic model is obtained using least square recursive estimator with an exponential forgetting factor.

A collision-free coordinated rate control scheme is discussed by Chong et al. (Chong et al., 1999) for Multi-Operator-Multi-Robot (MOMR) teleoperation using a network with communication time delay. Multi-robot collaboration have a significant advantage over a Single-Operator-Single-Robot (SOSR). However, the effect of the time delay will cause more severe problems that seriously affect their performance in MOMR teleoperation systems than in SOSR systems due to the unpredictable nature, in local display, of the slave arm under the control of remote operator. Because the distance between two operators is so long, one of them can not get immediately command issued by the other operator. Therefore, it is obvious to pose the danger of collisions in slave arms. This type of collaboration, known as unconstrained collaboration, in which each operator has the freedom to control his/her slave arm independently from the other slave arm, is very sensitive to time delays. Operator usually performs to *a wait and move strategy* in order to avoid from collisions which causes to decrease the productivity, considerably.

Simulation experiments managed using OpenGL and network delay simulator showed that the occurrence of collisions even when a virtual thickness corresponding to the time-delay is added to the slave manipulator model in local display. Although there was no network delay, some collisions occur because of human error. By authors, a new approach is suggested that the usage of velocity rate control which scales the velocity commands issued by the operator considering the relative positions of the slave arms. If they are too near, the velocity commands will be scaled down, otherwise they will be sent as they are. However, if the distance is too small, the velocity commands will become zero neglecting the operator, completely. This approach causes not to happen the collisions completely but decrease the level of maneuverability of the joystick because of scaling effect.

Yeuk et al. in (Ho et al., 1999) discussed component-based distributed control for teleoperations using DCOM and JAVA. A model based supervisory control is proposed at the remote environment which is the substructure preparation project at the bottom of a volcano in Japan. A component based distributed control of the system is used to fulfill the certain requirements such as high level of robustness in deployment of the complete system and ease in upgrading the system. Based upon the remote environment model, a supervisory control is implemented at the remote site. JAVA / DCOM to maximize system flexibility are employed to realize component infrastructure and internet is used as a communication backbone. Complete isolation is from the network protocol is obtained using components.

JAVA and DCOM, each one has some unique characteristics, are used for component developments. JAVA is basically an operating system transparent software language but the use of Virtual Machine makes it a bit slow than Operating System

(OS) optimized complied DCOM objects. Therefore, DCOM is used in all interface components except Path Planner GUI and Database interface. Path Planner GUI and Database interface are written in JAVA because of the simplicity with no difficulty real-time constraint. For that project, the heart of the supervisory system is DCOM / ActiveX Supervisory Control Server and it maintains communication with vehicle objects, direct manual control as well as sensor integration server components. It is provided to the operator that video stream as well as a 3D graphical model of the current remote environment. Generally, the difference between the video stream and 3D graphical model should not be much, but if there is, the Supervisory Control will transfer the control to the operator to initiate necessary actions.

Yeuk et al. have further extended their study in Ho's work, in 2000. In this study, they provided the feedback by two paths, one from the Global Positioning System (GPS) data and second from the visual feedback. A camera is used to generate a visual feedback from the slave environment. Then images are snapped from the remote environment models which are identified by Visual Enhancements (VE), the position X of the vehicle is determined by minimizing the error function below based on the difference between the vehicle and position coordinates obtained from GPS and the visual feedback.

$$E=\sum_{i,j} E^2_{ij} = \sum K_{ij} \left[X_p(i,j) - P_i Tcw_i Twf_i(\underline{X})\right]^2 \qquad (2.1)$$

Here P_i, Tcw_i, $Twf_i(.)$ are coordinate transformation operators and K_{ij} is determined from the reliability of the measurement. This information is used in supervisory control and is also sent to the master site to invoke operator intervention if any critical error occurs. The system is developed to be sufficiently robust against to the addition of white noise in both robot actuator and camera planes.

John E. Lloyd et al. in (Lloyd et al, 1997) discussed estimating 3D position of the object and using it in model-based telerobotics. Model-based telerobotics, sometimes also referred to as teleprogramming, has been proposed as a means of overcoming the problems of time-delay and bandwidth limitation between a teleoperated manipulator (remote site) and an operator control station (delayed remote site). Under this framework, an operator interacts with a model of the remote site instead of a delayed remote site. This interaction is in turn used by the system to generate motion and task commands which are transmitted to the remote site. For that operation, the operator points to a known object feature in a video image of the remote site and uses two-dimensional (2D) images of these features to solve for the 3D position of the object.

In the Lloyd's study, the system consists of an *operator site* and a *remote site*, connected by communication links implemented as TCP/IP sockets. At the remote site there is a *video/vision* module which continuously collects a single camera images and processes them using a model-based vision algorithm to locate objects in the scene. Using one single camera, Lloyd in (Lloyd et al, 1997) used to pin-hole camera model with offline calibrated focal length and radial distortion for one single camera. Then a list of the objects found, along with their spatial positions, is transmitted back to the operator site. The camera image itself is also compressed and transmitted back to the operator site, where it is displayed in a separate window.

Besides overcoming time delay, model-based teleprogramming systems permit other advantages, such as operator control of the viewpoint, the ability to test and preview actions, and the introduction of artificial graphical and kinesthetic aids for task specification. More generally, they provide the opportunity to raise the semantic level of the interaction between the operator and the manipulator system.

Mayez et al. discussed in (Mayez et al., 2005) about the performance of networked teleoperation systems which is based on timely streaming of highly-demanding dynamic media to interface human operator to the actuators and sensors of a remote robot. To minimize real-time delays a multi-threaded programming has been used to restructure sequential processing into concurrent threads that are executed in a pipelined fashion to parallelize the CPU processing with network transmission.

Telerobot is implemented using TCP/ATM in which two LANs are connected to an Asynchronous Transfer Mode (ATM) backbone. Specification of Quality-of-Service (QoS) includes application timing, reliability, clock synchronization and criticality. This is accomplished by using a constant bit rate (CBR). For suitable real-time applications, ATM connection allows a tightly constrained transmission delay. In the robot closed loop control, random delays affect stability and performance.

The client and server are run on two PCs having 2-GHz Intel P4 processors with 1GB DRAM memory. The vision server software uses MS Visual C++ with .NET framework 1.1 under Microsoft development environment 2003. The imaging device used is Microsoft DV camera and VCR. The PUMA server is implemented using MS Visual C# with the above .NET framework. Network delays are studied using three campus networks denoted by routes A, B, and C as shown on Figure 2.2.

Evaluation is carried out at the following levels:

(1)streaming force in presence of video,

(2) thread engineering and delays in live transfer of stereo video.

Mayez et al. studied on delays in live transfer of stereo video, in three sub-section, such as:

1) the delays in copying the video data from cameras to computer main memory,

2) performance of thread engineering for live video transfer in route A,

3) video performance in routes B and C.

The time delay which is due to copying a 300 video frames from cameras to memory in the case of a single stereo thread is 24 ms, and time delay in the case of single copying thread increased from 24 ms yo 60.5 ms. In the case of stereo copying thread with force thread reading and transferring data over the network, the mean stereo copying times decreased from 60.5 ms to 33.46 ms. The improvement is due to the release of CPU resources to the video copying thread.

Performance of live transfer of stereo video for route A is evaluated using a single buffer with serialized transfer. In this case the inter-arrival times of 300 stereo video frames is 86.5 ms or 11.6 fps. For good depth perception of the viewer, a frame rate greater than 10 fps is sufficient and it gives good viewing . In network B, it is noticed that the stereo and force frames preserve the same distribution pattern as in network A, but the whole distribution is slightly shifted with an increase in the average inter-arrival times to 60 ms for the stereo data. Similarly, for the route C the dominant part (90%) of the distribution remains in the range of 58-60 ms (stereo) and some scattering appears in the region 60-80 ms.

The use of .NET technology for streaming of force packets provides nearly the same inter-arrival delay for routes A, B, or C which is also comparable to the delay for one single hub using TCP sockets.

2.1.1 Supervisory Control in Telerobotics

The machine, with automatic control, controls the process or task, adapts to changing circumstances and conditions and makes decisions in pursuit of some goal. Supervisory control is defined by Sheridan (Sheridan, 1986) "in the strictest sense, supervisory control means that one or more human operators are intermittently programming and continually receiving information from a computer that itself closes an autonomous control loop through artificial efforts to the controlled process or task environment."

Buzan and Sheridan (Buzan and Sheridan, 1989) developed a dynamic user aid to help operators compensate for several second time delays in telemanipulator systems. The aid, among other approaches like state prediction, position feedback, etc., also utilized impedance control. Therefore, it provides different level of supervisory control in the remote system. Also, this study presents a dynamic extension to the kinematic predictor display concept. The predictive operator aid provides the operator with appropriate position and force predictions by using a model of the manipulator and the environment. Three predictive force reflection techniques are suggested and tested.

Using the parallel-feedback predictive controller for both position and force results in a four-part predictive operator aid which are state predictor, predictor display (position feedback), force reflector and slave impedance controller. *State Predi*ctor uses models of the manipulator and the environment, and predicts not only the states (position and velocity), but also the system configuration. The predictor display technique incorporates *dual* visual feedback, since the operator is provided with both the actual feedback and the predicted feedback using superpositioning. it is probably necessary using two monitors and separate the predicted video from the delayed real video for clarity. Force feedback gives the operator insight to the interaction, instilling confidence to move faster, or fear that one should stop. The slave impedance controller provides dynamic disturbance rejection by controlling the slave / environment interaction.

Matthew et al. in (Matthew et al., 1995) discussed a teleprogramming experiment incorporating operator supervisory control of a robot performing puncture and slice operations on the thermal blanket securing tape of satellite repair mission sub-task. The operator interface performs the dual function of providing immediate feedback to assist in specifying intentions and providmg time delayed feedback to allow assessment of the status of the remote site. The authors, in teleoperation research, claim that remote sites should be remote and by this principle, they were able to treat research issues that could not be completely simulated in a laboratory setup. They developed a layered architecture controller defining multiple layers of control. Operator direction interacts as the highest layer of the architecture and does not affect lower level behaviors of the system.

Fischer et al. (Fischer et al., 1996) have shown the necessity of hierarchical supervisory control for service task solution using a huMan Robot Interface (MRI) and the demands on information exchange between operator, MRI and service robot are outlined and categorized. The aim of this paper is to combine a semiautonomous robot system with a human operator to obtain an intelligent human-robot-system (hierarchical supervisory control). The resulting system is capable of solving different service tasks even in unknown situations. They have presented a distributed planner to control the robot system enabling both flexible robot behaviors and online operator support. The behavior of the proposed intelligent system is separated in four different abstraction levels starting from physical layer to the highest level used by the operator, for example, knowledge based robot. A semi-autonomous robot system is combined with a human operator to obtain an intelligent human robot system (hierarchical supervisory control).

Chen and Luo constructed a remote supervisory control architecture by combining computer network and an autonomous mobile robot in (Chen and Luo, 1997). Users having access to World Wide Web (WWW) can command the robot by sending primitive commands to move the robot, to grip and lift arm, to pan and tilt the camera and grab images, to speak, and get the information what robot senses through internet. This architecture offers multilevel remote control modules, namely, direct control, supervisory control and learning control modes. In supervisory mode, the robot works as a service man who provides to web users with a specific service. The server receives only a high level command, then controls the robot to perform the specific task by applying local intelligence of the mobile robot such as collision avoidance, path planning, self referencing and object recognition. One of the possible uses of this scheme is stated to be sharing of robot with multi-users via WWW.

Sheridan (Sheridan, 1997) defines a model of supervisory control. In this model, the operator as a supervisory controller, provides system commands to a human interactive computer (HIC) which consists of system status displays and data input devices. HIC improves these goals to the lower level Task Interactive Computer (TIC) which translates these higher level goals into a set of commands to the actuators that will produce the desired system performance. A sketch of this scheme is shown in Figure 2.3

Luo and Chan (Luo and Chen, 2000) have proposed a behavior programming concept to avoids disturbances of the internet latency. Primitive local intelligence of a mobile robot have been grouped into motion planner, motion executer and motion assistant. Each of a group is treated as an agent. All of these agents are integrated by centralized control architecture based on multi-agent concept. Event driven approach is applied on the robot to switch the behaviors to accommodate the unpredicted mission autonomously. For communication between the client and server, two virtual channels are used. First channel for the transmission of explored information and the second one for the commands. The high level behavior programming control of the networked robot is demonstrated to be a feasible and reliable method to reduce to interference caused by internet latency.

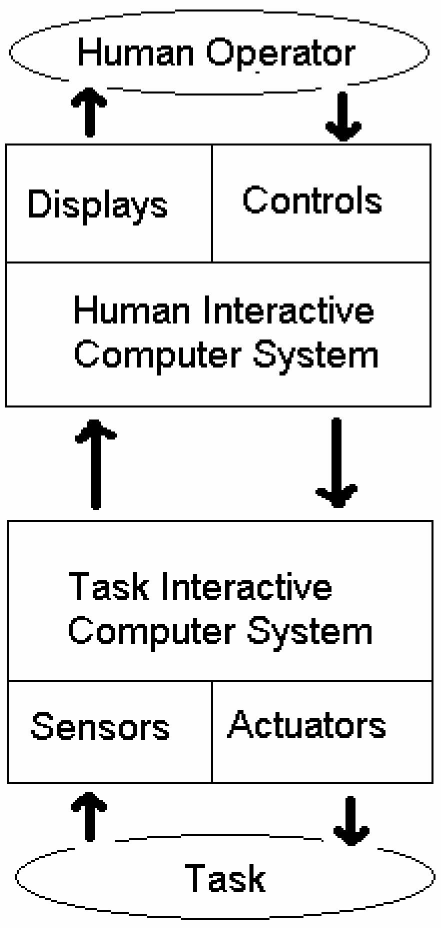

Human Operator
Displays
Controls
Human Interactive Computer System
Task Interactive Computer System
Sensors
Actuators
Task

Figure 2.1 A Model of Supervisory Control (Sheridan, 1997)

2.2 Stereo Vision and Augmented Reality

Stereo vision is a technique to grab 3D information of a scene. In addition to that meaning, it is defined as visual perception of or exhibition in three dimensions. In robotics, it is used for 3D viewing / reconstruction of the remote scene in a telerobotic environment. To supplement the perception of the person using the real data, augmented reality helps us to add additional information with the real data. In the following text, a review of the work in stereo vision and augmented reality is presented.

2.2.1 Stereo Vision

In stereo vision, we discuss different issues of 3D characteristics of a scene. Meaning of the stereo vision is the visualization of a remote scene in such a way that the viewer has a clear idea about the relative distances and depths of the objects present in the stereo image. Stereo vision has a wide range of application areas such as three dimensional map building, data visualization and robot pick and place.

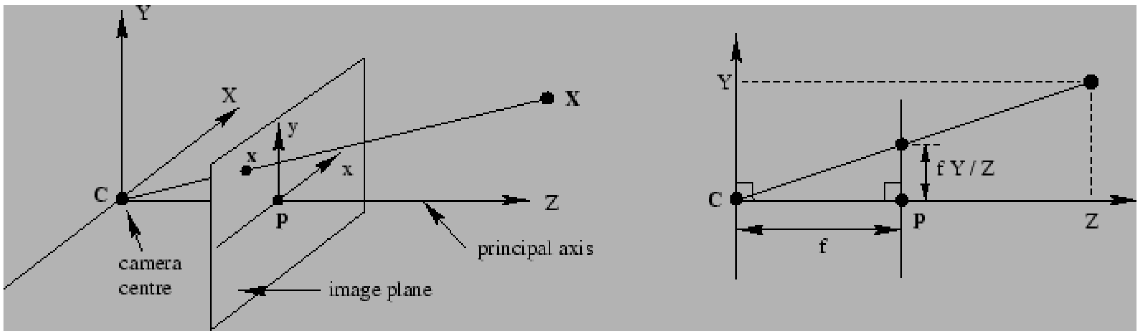


Figure 2.2 Pinhole Camera Model (Owens R., 1997)

In order to know the mapping of 3D point on 2D image, a simple camera model known as 'pinhole' camera is described. A point (X,Y,Z) in 3D space where Z is the depth of the point maps to,

$$x_{cam} = \frac{f}{Z} X \tag{2.2}$$

$$y_{cam} = \frac{f}{Z} X \tag{2.3}$$

In this equation, f corresponds to the distance from the projection plane to the projection center.

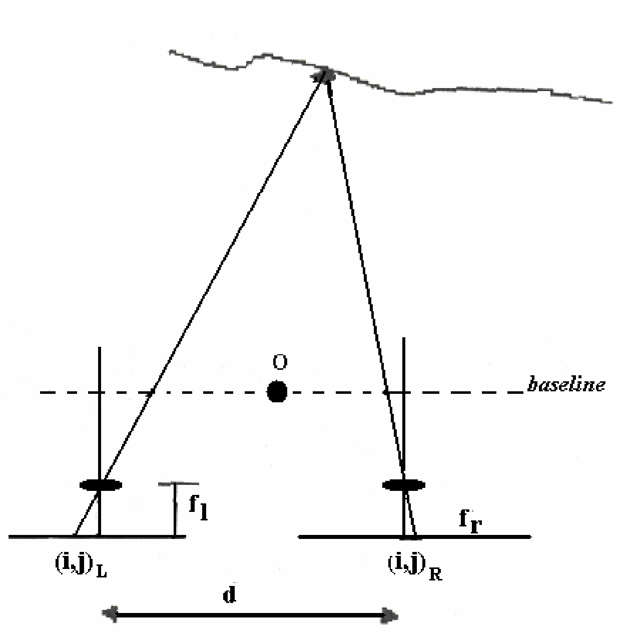


Figure 2.3 Stereo Camera Model (Iqbal, 2003)

3D camera model can be developed considering figure 2.3. We assume the following aspects in developing this camera model.

- Two cameras with their optical axes parallel and separated by a distance d.
- The line connecting the camera lens centers is called the baseline.
- Let baseline be perpendicular to the line of sight of the cameras.
- Let the x-axis of the three dimensional world coordinate system be parallel to the baseline.
- Let the origin O of this system be mid-way between the lens centers.

Using similar triangles,

$$\frac{x_1}{f_1} = \frac{x + \frac{d}{2}}{z} , \qquad \frac{x_r}{f_r} = \frac{x - \frac{d}{2}}{z} \tag{2.4}$$

where x_1, x_r are the x-coordinates of the projections of 3D point x on left and right image planes while f_1, f_r are focal lengths of left and right lenses respectively. d is the disparity or the distance by which two cameras are separated from each other.

Assuming equal focal lengths,

$$\frac{y_1}{f_1} = \frac{y_r}{f_r} = \frac{y}{z} \tag{2.5}$$

where y_1, y_r are the x-coordinates of the projections of 3D-point x on the left and right image planes. Now solving for (*x, y, z*) in the world coordinates (Iqbal, 2003),

$$x = \frac{d(x_1 + x_r)}{2(x_1 - x_r)}, \qquad y = \frac{d(y_1 + y_r)}{2(y_1 - y_r)}, \qquad z = \frac{d.f}{x_1 - x_r} \tag{2.6}$$

2.2.2 Augmented Reality

Augmented Reality (AR) is a field of computer research which deals with the combination of real world and computer generated data. At present, most AR research is concerned with the use of live video imagery which is digitally processed and "augmented" by the addition of computer generated graphics. Moreover, Augmented Reality can be defined as a variation of Virtual Reality (VR). VR is a technology that is computer generated and allows the user to interact with data that gives the appearance of a three-dimensional environment. AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therfore, AR supplements reality, rather than completely replacing the reality as is the case with Virtual Environment (VE) or VR. According to Azuma (Azuma, 1997), AR systems are required to have the following three characteristics:

1) Combines real and virtual
2) Interactive in real time
3) Registered in 3-D

In this article, at least six classes of potential applications of AR have been explored: medical visualization, maintenance and repair, annotation, robot path planning, entertainment and military aircraft navigation and targeting. In Boeing

Company, AR technology have been developing by a group to guide technician in building a wiring harness that forms part of an airplane's electrical system. See (Sims, 1994) for details.

AR can help to users by annotating objects and environments with public or private information. Rekimoto (Jun, 1995) proposed an application of annotation where a user gets information about the contents of library shelves on a hand-helps display as he walks around in the library.

Robot path planning can be facilitated using AR in situations where a large time delay is present between operator and the robot. Operator can preview the effect of the move on the local display overlaid on the remote world image. Once operator satisfied with the movement, he can send the actual command.

In combining the real and virtual worlds in an AR system, we have two choices:

1) Use Optical Technology
2) Make use of Video Technology

In an optical AR equipment, we make use of direct see-through, for example, the operator gets a direct view of the real world while the virtual objects are superimposed on optical see through mirrors in front of his eyes.

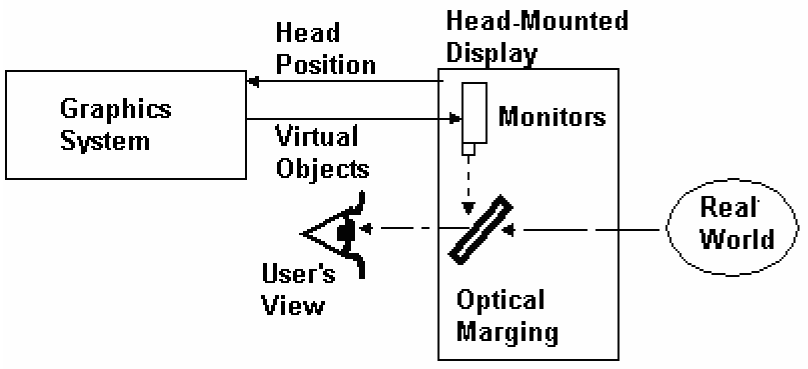


Figure 2.4 Optical See-through Augmented Reality Display

In video, the operator does not have any direct view of the real world. He uses the video input from the camera altered by a local scene generator in order to add virtual objects to the scene.

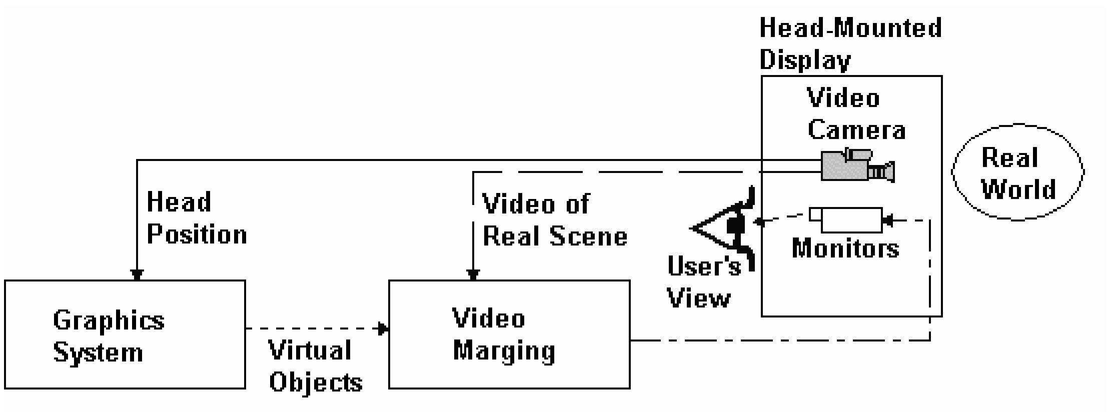


Figure 2.5 Video See-through Augmented Reality Display

There are advantages and disadvantages of both techniques. Further details is presented in (Azuma, 1997).

2.2.3 Classification of Visualization Systems Based on Used Equipments

There are many Stereo3D Image Formats such as Interleave/Interlace, Line Blanking, page flipping and sync-doubling (above-and-below).

2.2.3.1 Interleaved/Interlaced Stereoscopic

Image format in which the left and right views are combined or "woven" together, line by line. Each line alternates between the left and right view of the image.

2.2.3.2 Page Flipping /Page Flip Mode

Method of viewing stereoscopic content by using video hardware to rapidly switch the left and right eye view in temporal sync with shutter glasses.

2.2.3.3 Sync-Doubling (Above-And-Below)

The above-and-below standard is perhaps more effective and does not require additional hardware support inside the computer. In this scenario, the left view is rendered in the upper half of the screen at half of its vertical resolution, while the right view is rendered in the lower half. These images are separated by a variable number of black lines that will operate as a vertical blanking interval on final display.

2.2.3.4 Line Blanking

This method is suitable when the output on the screen is in interlaced format. The line-blanking controller hides odd lines for right eye and even for left eye. The frequency of watching is 1/2 of the monitor frequency

Each format requires different techniques and/or equipment for generation and visualization. Furthermore, they have different robustness characteristics under MPEG compression, and image/video resizing. For a detailed and comparative discussion on these modes, see the online document, Eye3D Manual (eDimensional, 2007), (Ramm, 1997)**.**

Different ways to generate 3D video content are given as:

1. Parallel Camera Configuration (Lee et al., 1996), can be used to observe with high accuracy a 3D object under magnification and depth. This is a very commonly used technique for 3D video generation. Computational aspects are simpler than a tilted case. However, it has problems especially with the near stereoscopic viewing. Most of the time some sort of video mixer may be required to convert two video streams into a single synchronized system.

2. Tilted camera Configuration (Lee et al., 1996), produce more accuracy in the horizontal direction than in the vertical direction compared to the case of parallel camera configuration. However this problem can be solved by using different horizontal and vertical scaling factors. Furthermore, this configuration provides a larger area of stereoscopic vision, such that the total area for 3D display is more, the depth resolution is enhanced, an near stereoscopic viewing is better than the parallel configuration. Computational aspects are more complicated and demanding compared to the parallel case.

3. Nu View 3D adapter consist of two LCD shutters, a prismatic beam, splitter and an adjustable mirror. Watching through the Nu View, while it is switched off, one will see two images. The mirror/prism system puts the camera lens into the center of the light rays of the left and right eye view. The shutters allow the camera lens to get only one of the views at a time. The adaptor is connected the video-out port of the

camcorder. This way the shutter can sync to the recording (50 or 60 Hz). It is a simple and practical solution to 3D video generation. See the online documentation at (Bungert, 2007) for further details.

There are basically two major classes of 3D visualization techniques. These are shuttering glasses and head mounted displays which are described as follows.

1. The shutter glasses achieve stereo by using frame sequential techniques. Shutter glasses enable the subjects to see stereoscopic images. The glasses alternately "shutter," i.e. block, the viewer's left, then right, eyes from seeing an image. The stereoscopic image is alternatively shown in sequence left-image, right-image synchronously with the shuttering of the glasses (Lo and Chalmers, 2004). At low refresh frequencies, the user can experience the annoying phenomena of flickering which can affect the ability to control the robot arm. However, most of the available monitors and display adapters can support refresh frequencies equal or above 120 Hz at resolutions of 1024x768 or above. Therefore, 3D visualization with very high details is possible with most shuttering glasses. There are indeed such papers, which demonstrates the effectiveness of shuttering glasses in 3D visualization. See (Strunk and Iwamoto, 1990) for more details.

Just for the illustrative purposes, the “Eye3D Premium” shuttering glasses can support resolutions (in pixels) up to 2048x1538 at 120 Hz, and 1856x1392 at 140 Hz. These specs are available only in high-end monitors. For reasonably high resolution and high refresh rate, the existing shuttering glasses technology is more than enough.

2. Head mounted display (HMD) (Bungert, 2007) and (Lee et al. 1985) provide a much larger virtual monitor size for the user, usually in the range of 2 meters large. However, their main disadvantage is that their resolutions are either VGA or SVGA (at least the ones which are commercially available during this period of time). They are more comfortable to work with, forces to use to see the 3D object and nothing else, and there is no problem of flickering. Most of them support the INTERLEACED 3D video format, but not the so called ABOVE/BELOW format which is robust under video compression and resizing. Most HMDs also support page flipping, but this requires special drivers for each display adapter.

CHAPTER 3

ROBOT SYSTEM

Robots are very powerful elements of today's industry. They are capable of performing many different tasks and operations precisely and do not require common safety and comfort elements humans need. However, it takes much effort and many resources to make a robot function properly. Most companies that made robots in the mid-1980s no longer exist, and only companies that made industrial robots remain in the market (such as Adept Robotics, Staubli Robotics, Mitsubishi Electric, Fanuc Robotics, North America, Inc.).

3.1 ROBOT FUNDAMENTALS

3.1.1 Robot Components

1. ***Manipulator or Rover*****:** Manipulator is the main body of the robot and consists of the links, the joints, and other structural elements of the robot.

2. ***End Effector*****:** This is the part that is connected to the last joint (hand) of a manipulator, which generally handles objects.

3. ***Actuators*****:** Actuators are the "muscles" of the manipulators. Common types of actuators are servomotors, stepper motors, pneumatic cylinders and hydraulic cylinders.

4. ***Sensors:*** Sensors are used to collect information about the internal state of the robot or to communicate with the outside environment.

5. ***Controller:*** The controller rather similar to human cerebellum, and although it does not have the power of human brain, it still controls human motion. The controller receives its data from the computer, controls the motions of the actuators and coordinates the motions with the sensory feedback information.

6. ***Processor:*** the processor is the brain of the robot. It calculates the motions of the robot's joints, determines how much and how fast each joint must move to achieve the desired location and speeds, and oversees the coordinated actions of the controller and the sensors.

7. ***Software:*** There are perhaps three groups of software that are used in a robot. One is the operating system which operates the computer. The second is the robotic software, which calculates the necessary motions of each joint based on the kinematic equations of the robot. This information is sent to the controller. This may be at many different levels, from machine language to sophisticated languages used by modern robots. The third group is the collections of the routines and application programs that are developed in order to use the peripheral devices of the robots, such as vision routines, or to perform specific tasks.

It is important to note that in many systems, the controller and the processor are placed in the same unit.

3.1.2 Robot Reference Frames

Robots may be moved relative to different coordinate frames. In each type of coordinate frame, the motions will be different. Usually, robot motions are accomplished in the following three coordinate frames.

World Reference Frame, which is a universal coordinate frame, as defined by x, y, z-axes. In this case, the joints of the robot move simultaneously so as to create motion along the three major axes.

Joint Reference Frame, which is used to specify movements of each individual joint of the robot.

Tool Reference Frame, which specifies movements of the robot's hand relative to a frame attached to the hand. The x'-, y'-, z'-axes attached to the hand define the motions of the hand relative to this local frame (Niku, 2001).

3.2 MITSUBISHI RV-2AJ

The "Mitsubishi RV-2AJ" is compact industrial robot developed with Mitsubishi's advanced Technology. This robot responds to user's needs for compact and flexible facilities generated due to the recent diffusion of compact and highly accuracy products such as personal computer related devices.

The Mitsubishi RV-2AJ robot, shown in figure 3.1, is 5 axis robot arms featuring 64-bit RISC/DSP controller technology, and load capacity of 1.5 kg. The maximum speed of the robot arm is 2.200 mm/s. RV-2AJ's joint space is limited and joint limits are shown in Table 3.1 .

The robot family of Mitsubishi includes small robots, like the RV-2AJ, which has a height of 410 mm which is the maximum range of the robot arm when the arm is aligned vertically. RV-2AJ used in this project is one of the smaller robots in the Mitsubishi robot family.

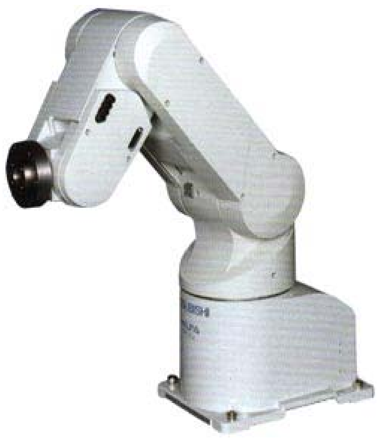


Figure 3.1 Mitsubishi RV-2AJ (Mitsubishi Electric, 2006)

To accurately position and orientate an object in 3D (three dimensional space) a robot must have 6 DOF's (degree of freedom). This means that it must be able to move the tool to an X,Y,Z position and then rotate the tool about X,Y,Z to provide the correct orientation. The Mitsubishi RV2AJ robot has only 5 DOF and therefore has reduced functionality, i.e. there will be orientations within its workspace that it will be unable to achieve. The missing DOF, J4, is the wrist pitch movement. In practice this limitation will usually cause, at worst, only minor inconvenience (HarewoodGill, 2006).

The robot 'waist' position (J1).
The robot 'shoulder' position (J2).
The robot 'elbow' position (J3).
The robot wrist 'yaw' position (J5).
The robot wrist 'roll' position (J6).

JOINT	LIMIT
J1	**-150^o to +150^o**
J2	**-60^o to +120^o**
J3	**-110^o to +120^o**
J5	**-90^o to +90^o**
J6	**-200^o to +200^o**

Table 3.1 Joint Limits

The manipulator has 5 degrees of freedom. The size of the manipulator is given by Figure 3.2.

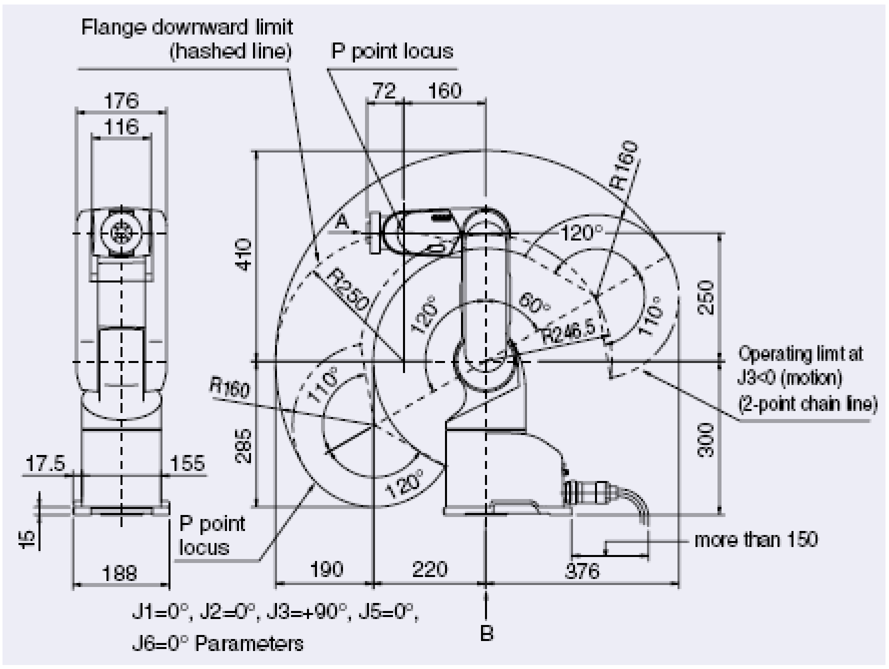

Flange downward limit (hashed line)
P point locus
176
116
72
160
410
R160
A
120°
R250
120°
60°
R246.5
110°
250
Operating limt at J3<0 (motion) (2-point chain line)
R160
110°
17.5
155
285
120°
300
15
P point locus
188
190
220
376
more than 150
J1=0°, J2=0°, J3=+90°, J5=0°, J6=0° Parameters
B

Figure 3.2 Dimensions of Mitsubishi RV-2AJ (Mitsubishi Electric, 2006)

The Mitsubishi RV-2AJ has a lot of different application areas. Some of them are shown in Figure 3.3.

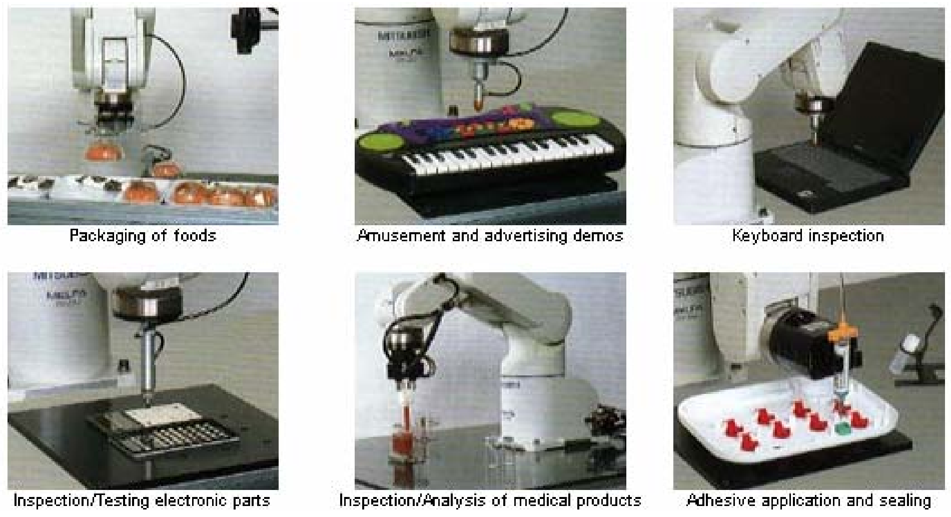

Packaging of foods
Amusement and advertising demos
Keyboard inspection
Inspection/Testing electronic parts
Inspection/Analysis of medical products
Adhesive application and sealing

Figure 3.3 Application areas of the Mitsubishi RV-2AJ
(Haklidir M. and Tasdelen, 2006)

The Mitsubishi RV-2AJ has two important parts to manipulate the Robot Arm.

1. ***The Robot Controller Unit***, which is the main part of robot arm that has a processors to calculate the motions of the robot's joints by means of determining how much and how fast each joint must move to achieve the desired location and speeds, and it controls the motions of the actuators.

It has three modes to control an operation such as Automatic Operation Mode, Automatic Operation externally Mode and Teach Mode. Automatic Controller Mode is used to run programs that are stored in the robot controller memory. The program will cycle, i.e. run continuously once started. In Automatic Operation Mode-controlled externally (via PC)-, robot has an external connection ports to connect to a computer and it can be controlled by sending data from that computer. Teach Mode is used to maneuver the robot by using Teach Pendant (TeachBox).

2. ***A Teach Pendant (or TeachBox) Unit,*** which is used to create, edit and control the program, teach the operation position and for jog feed, etc. It can not be used without enabling the Teach Mode of controller.

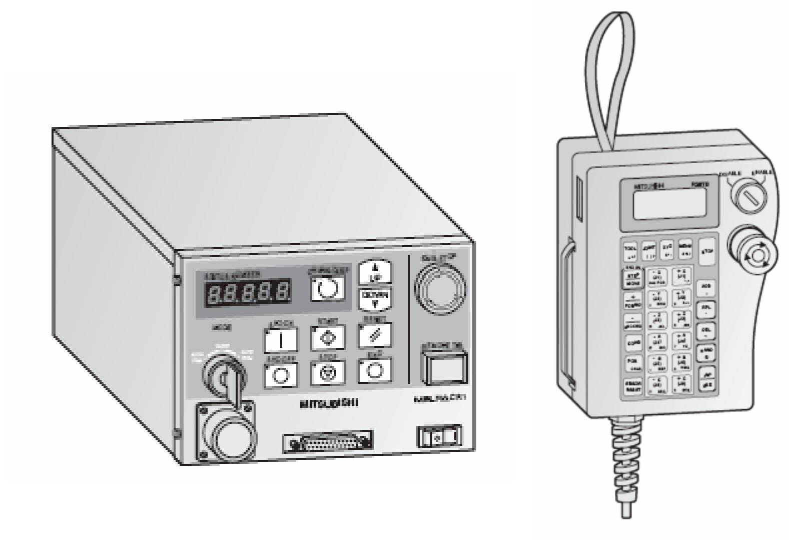
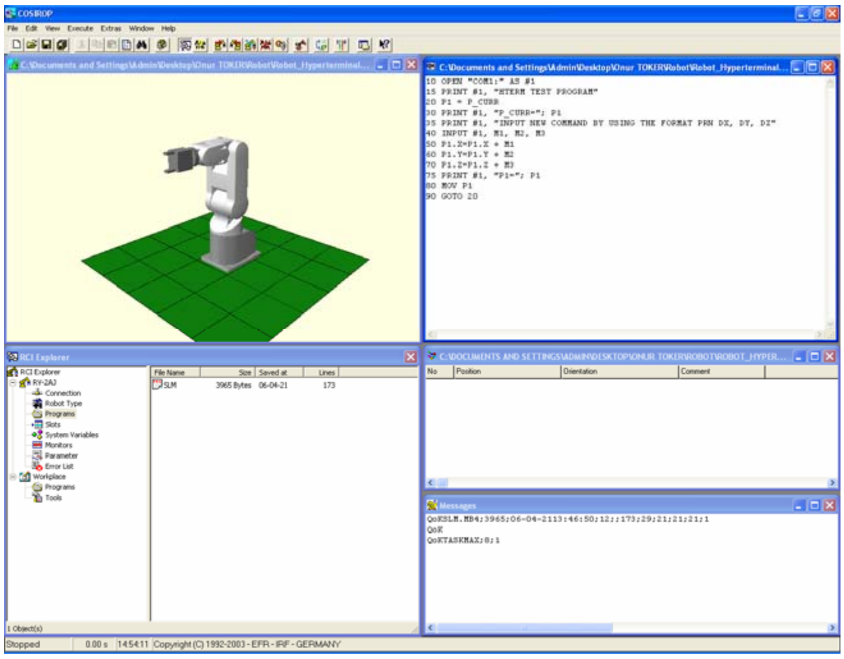


Figure 3.4 RV-2AJ Controller Box and Teach Pendant (Mitsubishi Electric, 2006)

3.3 COSIROP 2.0 PROGRAMMING SOFTWARE

COSIROP 2.0 is the tool for programming, online control, parameterization, and diagnosis of Mitsubishi MELFA Robots. COSIROP is used to develop robot programs in MELFA BASIC or Movemaster Command. Its jog operation tool is used to teach the robot and, also, it is helpful to insert positions directly into a position list. Therefore, users can exchange their programs between the PC and the robot controller via a serial interface or via an Ethernet connection. The RCI Explorer (Robot Controller Interface) is the new information-processing center of COSIROP 2.0. With the RCI Explorer users can up- and download programs, simply by drag-and-drop.

A powerful robot programming language needs an equally powerful programming environment. COSIROP allows users to create robot programs in minutes using the MELFA BASIC IV or MOVEMASTER COMMAND robot programming languages. After testing and optimizing the program they can then transfer the program to the actual robot with a couple of mouse clicks, via an efficient direct network or serial link between the PC and the robot.


COSIROP
File Edit View Execute Extras Window Help
C:\Documents and Settings\Admin\Desktop\Onur TOKER\Robot\Robot_Hyperterminal...
RCI Explorer
RCI Explorer
RV-2AJ
Connection
Robot Type
Programs
Slots
System Variables
Monitors
Parameter
Error List
Workplace
Programs
Tools
1 Object(s)
C:\DOCUMENTS AND SETTINGS\ADMIN\DESKTOP\ONUR TOKER\ROBOT\ROBOT_HYPER...
Messages

```
10 OPEN "COM1:" AS #1
15 PRINT #1, "HTERM TEST PROGRAM"
20 P1 = P_CURR
30 PRINT #1, "P_CURR="; P1
35 PRINT #1, "INPUT NEW COMMAND BY USING THE FORMAT PRN DX, DY, DZ"
40 INPUT #1, M1, M2, M3
50 P1.X=P1.X + M1
60 P1.Y=P1.Y + M2
70 P1.Z=P1.Z + M3
75 PRINT #1, "P1="; P1
80 MOV P1
90 GOTO 20
```

File Name	Size	Saved at	Lines
SLM	3965 Bytes	06-04-21	173

No	Position	Orientation	Comment

```
QoKSLM.MB4;3965;06-04-2113:46:50;12;;173;29;21;21;21;1
QoK
QoKTASKMAX;8;1
```

```
Stopped    0.00 s    14:54:11    Copyright (C) 1992-2003 - EFR - IRF - GERMANY
```

Figure 3.5 COSIROP software can be used to download, start and stop a program to the robot controller box.

While the programs are being executed they can monitor and visualize the robot with the help of COSIROP’s comprehensive control and diagnostics functions. The real-time axis speeds and motor currents are displayed clearly, together with the statuses of all the robot’s inputs and outputs. Live monitoring facilities for all the programs executed by the controller enable to track down program errors quickly and reliably.

COSIROP also provides tools for program archival and for backing up the robot's parameters and settings (Mitsubishi Electric, 2006), (Cosirop, 2006).

Other useful functions include:

- Online ”teach-in” function for robot positions
- Position display on a 3-D representation of the robot
- Syntax checking
- I/O monitor
- Variable monitor
- Online command execution
- Error diagnostics
- Position editor
- Project management

3.4 TESTS ON RV-2AJ ROBOT

Three kinds of tests are considered, respectively, to understand the aspects of robot control, robot-PC communication and parameterizations topics.

3.4.1 Control of RV-2AJ by Microsoft HyperTerminal

COSIROP does not provide the operational flexibility which can be achieved by using low-level communication and programming. In this project, we would like to teleoperate the robot from a remote location with visual feedback, and we will need more than the operational flexibility provided by COSIROP. At first, robot was controlled by Microsoft HyperTerminal which is a program that someone can use to connect to other computers, Telnet sites, and bulletin board systems (BBSs), online services, and host computers, using either modem or a null modem cable.

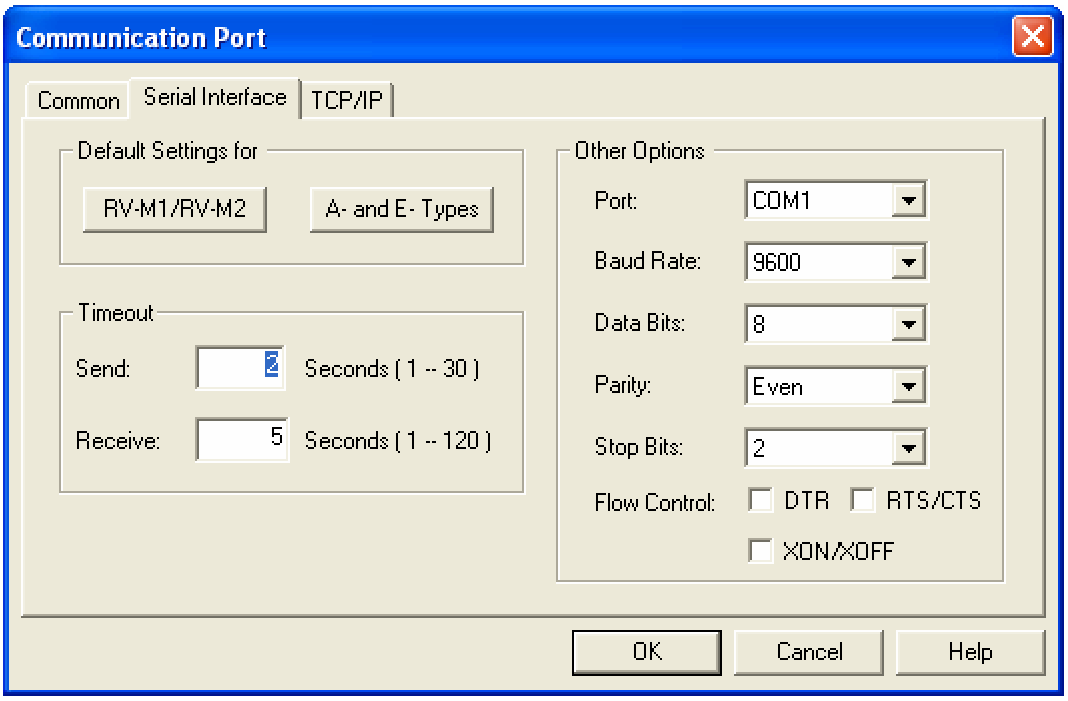
Our first goal was to be able to communicate with the robot controller box over a serial link. For this purpose, we have developed two Melfa-Basic programs, HTERM.MB4, and JHTERM.MB4, which are given below. These programs basically run on the robot controller box, listen to the serial port of the controller box, and wait for commands. Once a command is received, a move will be initiated, and then the system will wait for the next command.

The programs HTERM.MB4, and JHTERM.MB4 can be downloaded to the controller box by using the COSIROP software. Although program load, start, and stop type operations are also achieved by some kind of serial communication protocol between the robot controller box, and the PC running the COSIROP software, we have no detailed information about this. Because of this lack of information, we have to use COSIROP for downloading a new program. However, once a program is downloaded to the robot controller box, COSIROP application can be closed, and the robot can be operated by using our own application.

The full information about the communication protocol between the robot controller box, and the PC running the COSIROP software is really valuable, because once we have this level of detailed information, complete program download, start, and stop type operations can also be done from our own application, completely eliminating the COSIROP software. Although it will be very nice to have this level of control from our own application, from an operational point of it is not that critical to be able to download, start, and stop a new program from our own application. Because, in our project the program running on the robot controller box will be fixed, so it does not matter whether the program is loaded to the controller box by COSIROP or by our own application.

The main complexity will be in the program running on the PC (Namely our own application), but not on the program running on the robot controller box. In the teleoperation version, we will have two programs on two different PCs, and again the main complexity will be on the programs running on the PCs.

In Figure 3.5, the COSIROP software main window is shown, and in Figure 3.6, serial communication parameters for connection between the robot controller box and the PC running the COSIROP software is shown.


Communication Port

Common | Serial Interface | TCP/IP

Default Settings for: RV-M1/RV-M2 | A- and E- Types

Timeout
Send: 2 Seconds (1 -- 30)
Receive: 5 Seconds (1 -- 120)

Other Options
Port: COM1
Baud Rate: 9600
Data Bits: 8
Parity: Even
Stop Bits: 2
Flow Control: ☐ DTR ☐ RTS/CTS ☐ XON/XOFF

OK | Cancel | Help

Figure 3.6 Serial communication parameters for connection between the robot controller box and the PC running the COSIROP software

3.4.1.1 *Robot Controller Box Melfa Basic IV Program for HyperTerminal Control*

In robot side program, some parameters are defined to use them as a joint input values. After opining COM port of robot, defined position and joint parameters are initialized by current coordinates in x, y, z form and joint type data at the current position of robot. Then program checks the data on input terminal, in this case serial port, for manipulation. Therefore, robot is ready to be controlled from external inputs which are sent by HyperTerminal. But user has to send J1 (joint 1) parameter with the parameter PRN instead of any X, Y, Z coordinate values.

For example, in the HyperTerminal side, user should write to turn robot's J1-axis +90 degrees, "PRN 90". After user command, program calculates user's parameter corresponds to joint angle in radial format. Then, robot joint 1 is moved with respect to the calculated value. A detailed controller box program is given below. For detailed information about Melfa Basic IV commands, see the Appendix A.

```
01 DEF POS P1
02 DEF JNT J1
04 DEF DOUBLE DJ1, DJ2, DJ3, DJ4, DJ5
10 OPEN "COM1:" AS #1
15 PRINT #1, "JOYSTICK TEST PROGRAM V1.0"
20 P1 = P_CURR
25 J1 = J_CURR
30 PRINT #1, "P_CURR="; P1
32 PRINT #1, "J_CURR="; J1
35 PRINT #1, "INPUT NEW J1 BY USING THE FORMAT PRN J1"
40 INPUT #1, DJ1
50 J1.J1 = 3.14 * DJ1 / 180
60 MOV J1
90 GOTO 20
```

3.4.2 Control of RV-2AJ by Microsoft Visual Studio .NET/C# GUI

To control RV-2AJ robot by using .NET/C# Graphical User Interface (GUI), same controller box program written in Melfa Basic IV, shown above, is used. Interface of the .NET/C# controller unit was designed on Microsoft Visual Studio 2005. In the program, we have two buttons, one of them to move robot J1 at positive direction and the other one to move at negative direction. At every click on positive J1 button, program sends a "***PRN 5"*** -5^o movement- command to the robot using serial communication port and **"PRN -5"** for negative button click event. The user interface of the system is shown in figure 3.7.

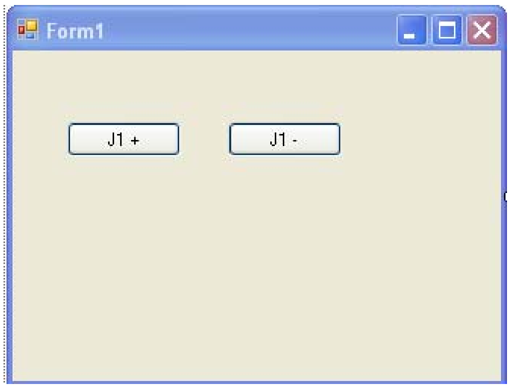
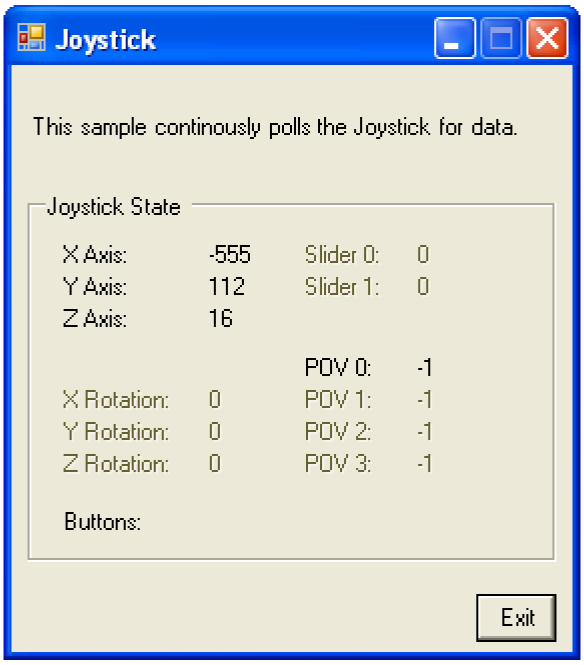


Figure 3.7 GUI of RV-2AJ Controller (written in C#)

3.4.3 Control of RV-2AJ by Joystick

Finally, we have also tested an analog joystick in the Visual C# 2005 environment. By using the DirectInput API of DirectX framework, we were able to read X, Y, and Z, namely three different analog coordinates, plus a couple of buttons, which may be used as digital inputs / mode change inputs.



Joystick

This sample continously polls the Joystick for data.

Joystick State

```
X Axis:       -555     Slider 0:   0
Y Axis:       112      Slider 1:   0
Z Axis:       16

                       POV 0:      -1
X Rotation:   0        POV 1:      -1
Y Rotation:   0        POV 2:      -1
Z Rotation:   0        POV 3:      -1

Buttons:
```

Figure 3.8 Joystick experiment with DirectInput API of DirectX.

CHAPTER 4

A MULTI-THREAD DISTRUBUTED TELEROBOTIC FRAMEWORK

A telerobotic system consists of master and slave stations which are connected by a computer network. In order to establish a reliable working relationship between master and slave arms, different plans are used to transfer master arm commands over to the slave arm. Distributed application programming is one of the schemes to establish a reliable connection between master and slave arms. Basically different items are realized as software components and then these components communicate with each other using distributed application programming paradigm. This is strictly an object oriented approach and promises all the benefits of object oriented programming like software reusability, easy extensibility, less time in debugging, data encapsulation, etc.

There are three most dominating distributed object technologies which are given as

1) CORBA,
2) .NET
3) JAVA / RMI

CORBA is an abbreviation for Common Object Request Broker Architecture and RMI stands for Remote Method Invocation. These are extensions of traditional object-oriented systems that allow the objects to be distributed across a heterogeneous network. The objects may reside in their own address space outside the boundary of an application or on a different computer than the application and they can still be referenced as being part of the application.

All of these, three distributed object technologies, are based on a client/server approach implemented as a network calls being transported on network protocols like HTTP, TCP/IP, etc. RPC (Remote Procedure Call) is the basic idea behind the CORBA and RMI technologies. In this approach, the local (client) and remote (server) ends are replaced by stubs thus making possible for both the client and server to use local calling conventions for remote methods. In order to avoid the hard and error prone implementations of network calls directly to the client and server objects, the distributed technology standards address the complex networking interactions through abstraction layers and hide the networking issues in order to let the programmer concentrate on developing the core logic of the application.

4.1 AN OVERVIEW OF THE DISTRIBUTED OBJECT TECHNOLOGIES

Here it is presented a brief overview of the three above mentioned technologies offering support for distributed programming.

4.1.1 CORBA

CORBA is an open distributed object computing infrastructure standardized by OMG (Object Management Group) (OMG, 2002), (Gupta, 2003). CORBA is the most widely used middle ware standard in the non-Windows market. ORB (Object Reference Broker) is the core of CORBA architecture. All the CORBA objects interact with each other transparently using ORB regardless of whether these objects are local or remote. IIOP (Internet Inter-ORB Protocol) was developed in the CORBA 2.0 as a means for the communication between ORBs from different vendors. IIOP runs on top of TCP/IP. Every CORBA object must be declared in IDL (Interface Definition Language), a language to declare the interfaces and methods of a CORBA server object.

4.1.2 .NET

The .NET architecture by Microsoft has replaced the Distributed Component Object Model (DCOM), previously used for distributed computing on, mainly, Windows based machines. In .NET, the COM (Component Object Model) is replaced by CLR (Common Language Runtime) that supports and integrates components developed in any programming language conforming to CLR specifications. .NET is a loosely coupled architecture for distributed applications.

The remote access is based on XML and SOAP (Simple Object Access Protocol) technologies. It also supports JAVA like object references and garbage collection but it has no JVM (JAVA Virtual Machine) like interpreter. IL (Intermediate Language) code is compiled by JIT (Just-In- Time) compiler to native machine code prior to execution. Compiled IL code executes on top of a portable API (Application Programming Interface) that enables future platform independence.

.NET provides two main strategies to use distributed objects, 1) Web services and, 2) .NET Remoting. Web services involve allowing applications to exchange messages in a way that is platform, object model, and programming language independent. Web services use XML and SOAP to form the link between different objects. Remoting, on the other side, relies on the existence of the common language runtime assemblies that contain information about data types. For the closed environments where faster connections are required, .NET remoting is an ideal solution cutting the overhead caused by object and data serialization through XML (DevHood, 2001), (Gupta, 2003).

4.1.3 JAVA/RMI

It is a standard developed by JavaSoft. JAVA has grown from a programming language to three basic and completely compatible platforms; J2SE (JAVA 2 Standard Edition), J2EE(JAVA 2 Enterprise Edition) and J2ME(JAVA 2 Micro Edition). RMI supports remote objects by running on a protocol called the JRMP(JAVA Remote Method Protocol). Object serialization is heavily used to marshal and unmarshal objects as streams. Both client and server have to be written in JAVA to be able to use object serialization. The JAVA server object defines interfaces that can be used to access the objects outside the current (JVM) JAVA Virtual Machine from another JVM that could reside on a different computer. A RMI registry on the server machine holds information of the available server objects and provides a naming service for RMI. A client acquires a server object reference through the RMI registry on the server and invokes methods on the server object. The server objects are named using URLs and the client acquires the server object reference by specifying the URL.

4.2 MOTIVATION FOR USING .NET FRAMEWORK

The system development support for .NET based components in most common languages like Visual Basic, Visual C++ and C# is excellent when using Microsoft Visual Studio as an integrated development environment. Components developed in any of the above languages as well as other languages conforming to CLR specifications, can be used easily in different applications and can interact with components developed in different languages. JAVA and CORBA support multiple inheritances while .NET does not. However, multiple inheritance at the interface level is provided in the .NET framework which compensates for the unavailability of the former. In comparison to DCOM, .NET provides object and data serialization through a firewall making it more dependable on even the internet. In addition, there is no need for component registration on the server side. The application just requires an access to server assembly which contains the implementation of server objects as well as the meta-data for these objects.

.NET components are self-describing: type signatures and other information is embedded in the components. This allows a lot of reflection on types, and it makes it possible for services such as the Visual Studio debugger to work across different languages. This level of debugging for components developed using different languages and in one environment is still missing in CORBA and JAVA. Microsoft technologies are a very good choice for organizations that mainly use Windows OS to run mission-critical applications (Gupta, 2003).

In conclusion, although JAVA has been used in many telerobotic systems but we choose .NET framework because of the following reasons:

1. We target to use the proposed framework on a commodity LAN where Microsoft Technologies give optimized performance. JAVA is recommended for Internet-Based cross platform environments.

2. .NET components can be easily deployed to work across firewalls.

3. CLR (Common Language Runtime) used by .NET Framework is similar to Java Virtual Machine because it also compiles the source code into platform-independent byte code (Mayez et al., 2003).

In our case, we need a real-time distributed system that will run on two lab PCs with Windows XP and commodity 100 Mbps LAN. .NET based distributed components prove to be an excellent choice for the proposed framework.

By using the distributed programming, network protocol issues can be avoided in the sense that the distributed framework itself takes care of all the network resources and data transfer over the network. In other words, distributed components based approach gives us complete isolation from network protocols. The framework can decide either to use TCP or HTTP protocols. All of the components are created using Visual C# as programming language.

In order to describe the complete system, we need to explain individual components and their interactions with each other when they co-exist in a distributed application. For simplicity we can divide the components in two groups, i.e., server side components and client side components.

4.3 SERVER SIDE COMPONENTS

On the server side, we have following components;

1. Mitsubishi RV-2AJ CR1 Controller Program
2. .NET Server Component & Server User Interface

4.3.1 Mitsubishi RV-2AJ CR1 Controller Program

RV-2AJ component is the heart of the server side framework because it deals with the commands sent by PC used as a server and the responses of the robot arm’s coordinates sent to PC. In other words, commands are issued from the client to the server side component will be read as they are issued to the robot and whenever the robot changes its states the component updates itself automatically to reflect these changes.

In the RV-2AJ CR1 Controller, the kernel program is written in COSIROP program in Melfa Basic IV programming language. After downloading program to Controller CR1, it is started by the user. Then, the communication between the Controller Unit and COSIROP Software is closed to establish a communication with the Server Program.

Controller Kernel Program (CKP) has eight input parameters which are sent by the server PC, in fact by the client PC, connected to RV-2AJ by RS-232 Communication Control Protocol. Moreover, CKP consists of five different subroutines which are controlled with respect to the input parameters sent by the server. The program flowchart is shown in Figure 4.1.

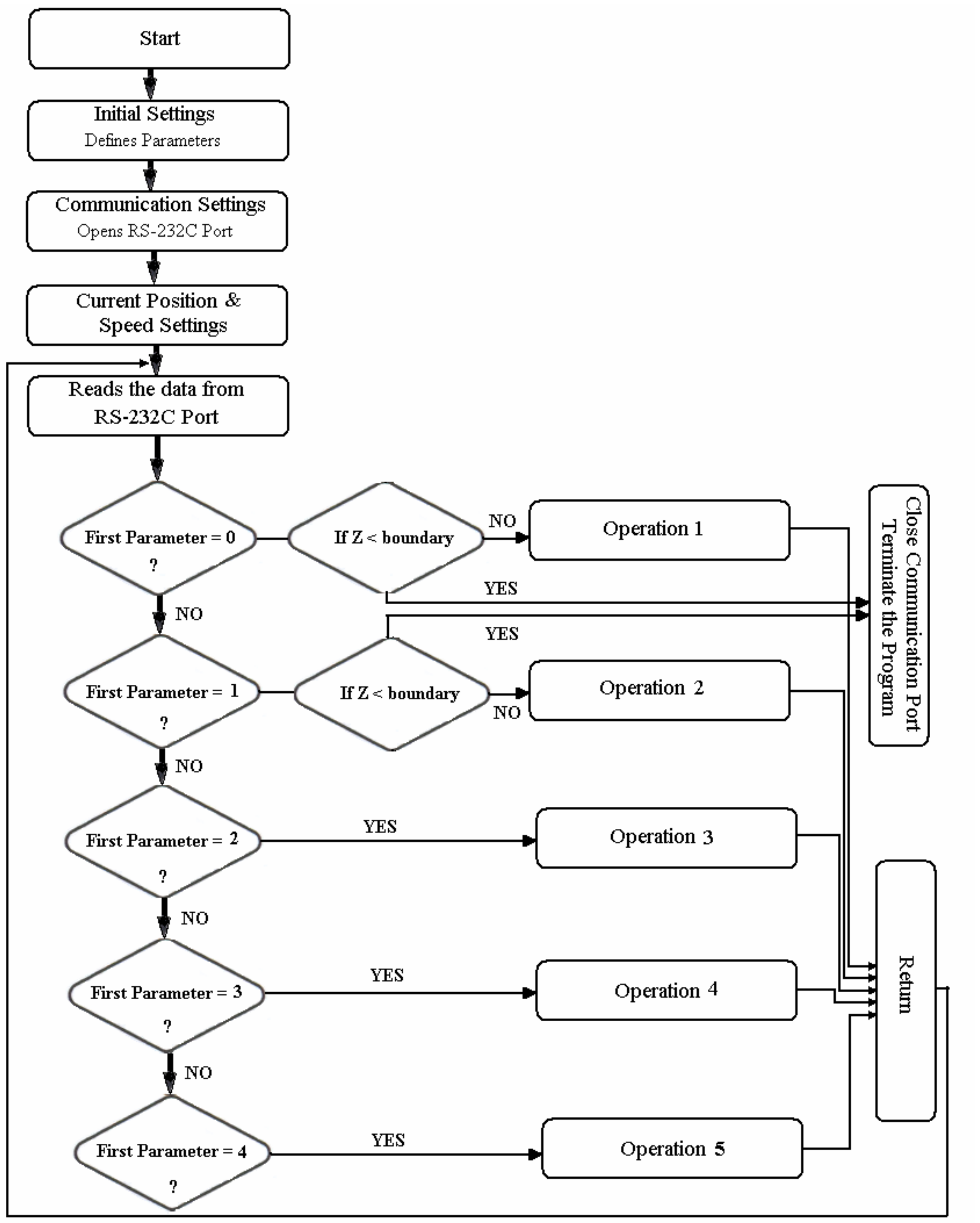


Figure 4.1 Mitsubishi RV-2AJ CR1 Controller Program Flowchart

In the program, we have eight parameters defined as integer value. The first parameter M0 is checked when any input data is read from COM port. Controller Program defines which subroutine will be called with respect to M0 value.

```
IF M0=0 THEN GOTO Operation 1
IF M0=1 THEN GOTO Operation 2
IF M0=2 THEN GOTO Operation 3
IF M0=3 THEN GOTO Operation 4
IF M0=4 THEN GOTO Operation 5
```

4.3.1.1 Operation 1

Operation 1 subroutine is written to move the robot arm in joint reference frame. In the subroutine, joint type data at the current position is stored to joint variable J1, input parameters M1, M2,…M6 are scaled with the scaling factor 0.001 to establish a small reliable incremental movement. These scaled parameters are added to current joint values initially stored. Then, this joint variable J1 is converted to cartesian coordinates X, Y, Z to prevent the robot arm from the collisions by checking the Z position with the previously defined boundary value. If the converted Z coordinate value less than boundary value, the program is terminated. Otherwise, robot arm is moved in joint reference frame with respect to input data. After movement, hand is opened or closed with respect to M7 input data. If M7 is equal to 1, hand will be opened and if it is 0, hand will be closed. After saving the robot current position coordinates to P1 coordinate variable program sends the joint and cartesian coordinates of the robot arm back to the server. And it returns to read the input COM port.

4.3.1.2 Operation 2

Operation 2 subroutine is written to move the robot arm in world reference frame or in X, Y, Z Cartesian coordinates. Firstly, cartesian coordinate position data at the current position is stored to position variable P1, input parameters M1, M2, M3 are scaled with the scaling factor 0.01 to establish a small reliable incremental movement. These scaled parameters M1, M2, M3 are added to current position values P1.X, P1.Y and P1.Z respectively. Then, this position variable P1’s Z coordinate is checked with the previously defined boundary value to prevent the robot arm from the collisions. If the modified Z coordinate value less than boundary value, the program is terminated. Otherwise, robot arm is moved in world reference frame with respect to input data. After movement, M7 input data is checked and hand is closed or opend depends on the M7 parameter. If M7 is equal to 1, hand will be opened and if it is 0, hand will be closed. After saving the robot current joint coordinates to J1 joint variable program sends the joint and cartesian coordinates of the robot arm back to the server as in the form of :

```
"P=J2.J1, J2.J2, J2.J3, J2.J4, J2.J5, J2.J6, P1.X, P1.Y, P1.Z”.
```

And it returns to read the input data from COM port.

4.3.1.3 Operation 3

If M0 is equal to 2, program goes to subroutine 3 to set the robot arm position to a specific coordinates. But coordinate initialization is done in Joint Reference Frame to provide a robust and dynamic movement. Sometimes coordinate initialization in cartesian reference frame caused to undesired result that robot arm moves to different positions in a repeated operation 3 calls. Then, program checks the M7 to control the hand motion. After program sends the joint and cartesian coordinates of the robot arm back to the server, it is saved the robot current position coordinates to P1 and joint coordinates to J1 coordinate variable. And it returns to read the input data from COM port.

4.3.1.4 Operation 4

Subroutine 4 is directly related to client interface program. In client GUI, we have last position button to store and move to last position. Therefore, last position data is sent in the format of M0 is equal to 3. then, program moves to cartesian coordinates P1.X to M1, P1.Y to M2 and P1.Z to M3, respectively with the maximum speed of the robot arm. Then, it checks the robot hand. At the end of that operation, it store current joint coordinates to J1 and sends robot current positions back to the server.

4.3.1.5 Operation 5

If M0 is equal to 4, program goes to subroutine 5 to set the robot arm speed. We declared an integer variable SORA (Speed of Robot Arm) and M1 input parameter is equalized to SORA. Then, robot arm speed is set to M1 speed using SPD instruction.

For more detail, look at the Mitsubishi RV-2AJ Controller Program for .NET Application at Appendix B.

4.3.2 .NET Server Component & Server User Interface

A server is a computer that handles requests for data transfers, and other network services from other computers (ie, clients). In our project, server (remote environment) is the computer which is connected to Mitsubishi RV-2AJ robot arm and handles the client computer's (local environment) requests.

An interface is a set carrying definitions of public methods and properties. It serves as a contract for any component that implements this interface. In other words, any component that inherits or implements the definitions contained in an interface must provide the implementation of all the methods or properties enumerated in the interface. This scheme is needed in .NET based distributed applications because any client that accesses or executes the methods of a component on the server needs an access to the server assembly or component. By giving a reference to an interface that the server component implements, we can hide the actual component or assembly from the client. This provides security from potential unsafe clients as well as gives the developers freedom to the easily amend the logic of the server methods while the interface remains unchanged for all the clients because the interface is only a definition, the implementation being only inside the component. Server user interface is shown in figure 4.2.

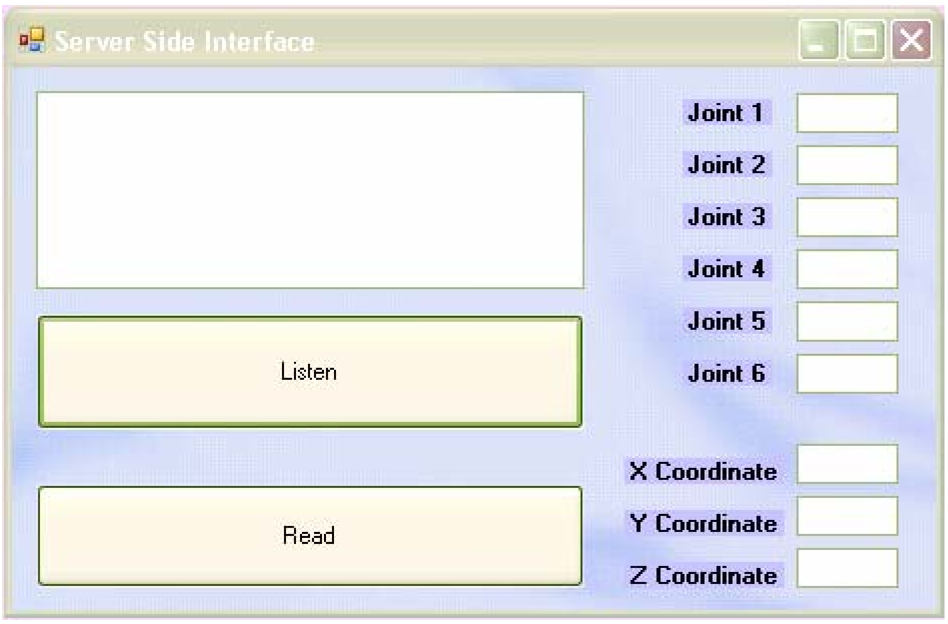


Figure 4.2 Server Side Graphic User Interface

A block diagram explaining the role of user interface and Server Component in the hierarchy of the system on server side is shown in figure 4.3. It is clear from the figure that server side logic is implemented in four layers. The last layer in the hierarchy is the physical layer consisting of robot RV-2AJ. On the highest level of the hierarchy is the human operator that might interact with the system using a GUI (Graphical User Interface).

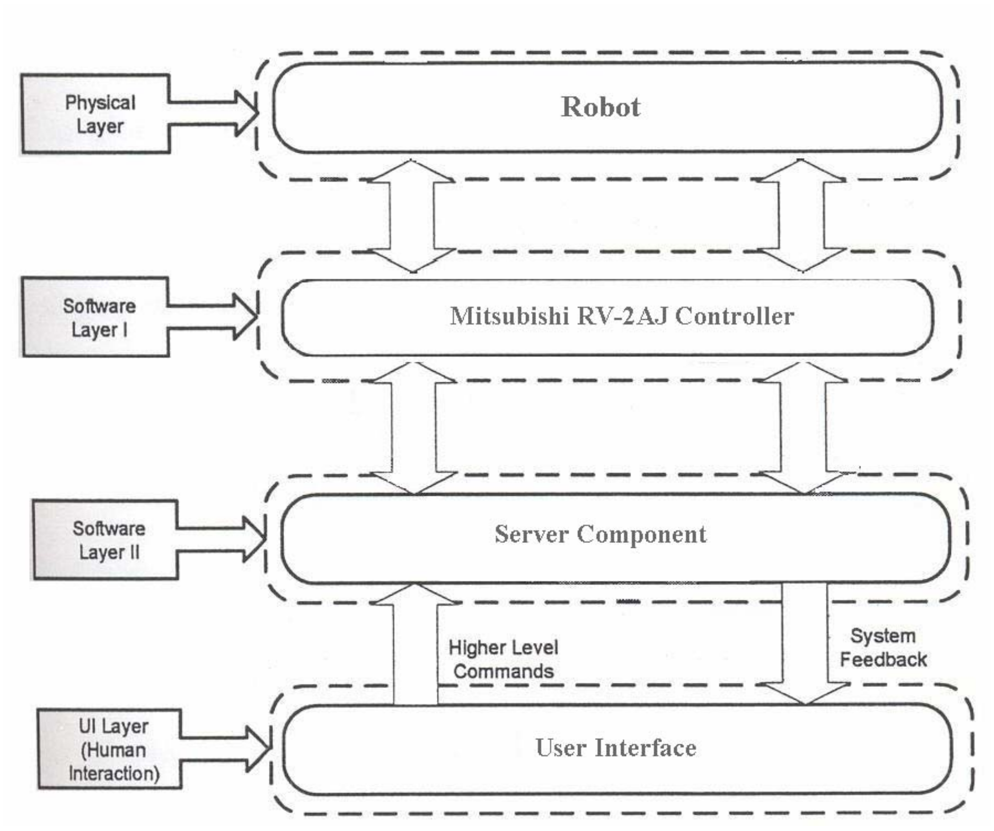


Figure 4.3 Component Hierarchy on the Server Side

4.3.2.1 TCP/IP Socket Programming

A *socket* is one endpoint of a two-way communication link between two programs running on the network. When a computer program needs to connect to a local or wide area network such as the Internet, it uses a software component called a socket. The socket opens the network connection for the program, allowing data to be read and written over the network. It is important to note that these sockets are software, not hardware, like a wall socket

A socket consists of the pair <IP Address,Port>. A port can be defined as an integer number between 1024 and 65535. This is because all port numbers smaller than 1024 are considered *well-known* - for example, telnet uses port 23, http uses 80, ftp uses 21, and so on.

A TCP connection consists of a pair of sockets. Sockets are distinguished by client and server sockets. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system. If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing to or reading from their sockets (Damian, 2006), (Sun Microsystems, 2007).

The power of network programming in .NET platform cannot be denied. Socket programming is the core of network programming in Windows and Linux, and today the .NET platform implements it in a powerful way.

Socket programming in C#: The *'System.Net.Sockets'* namespace contains the classes that provide the actual .NET interface to the low-level Winsock APIs. In network programming, apart from which programming language to use there are some common concepts like the IP address and port. IP address is a unique identifier of a computer on a network and port is like a gate through which applications communicate with each other. In brief, when we want to communicate with a remote computer or a device over the network, we should know its IP address. Then, we must open a gate (Port) to that IP and then send and receive the required data.

One of the biggest advantages noticed in the .NET network library is the way IP address/port pairs are handled. It is a fairly straightforward process that presents a welcome improvement over the old, confusing UNIX way. .NET defines two classes in the System.Net namespace to handle various types of IP address information:

- IPAddress
- IPEndPoint

In the .NET Framework, one can create connection-oriented communications with remote hosts across a network. To create a connection-oriented socket, separate sequences of functions must be used for server programs and client programs:

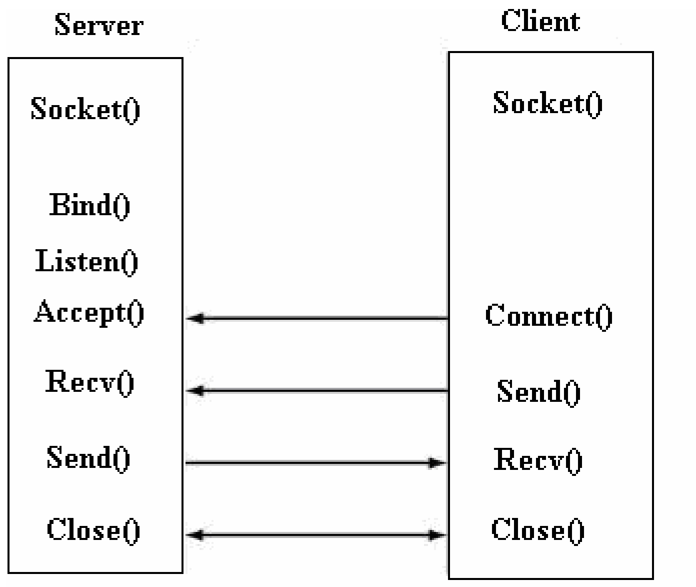


Figure 4.4 Server-Client Communication Orientation

On the Server, we have four tasks to perform before a server can transfer data with a client connection:

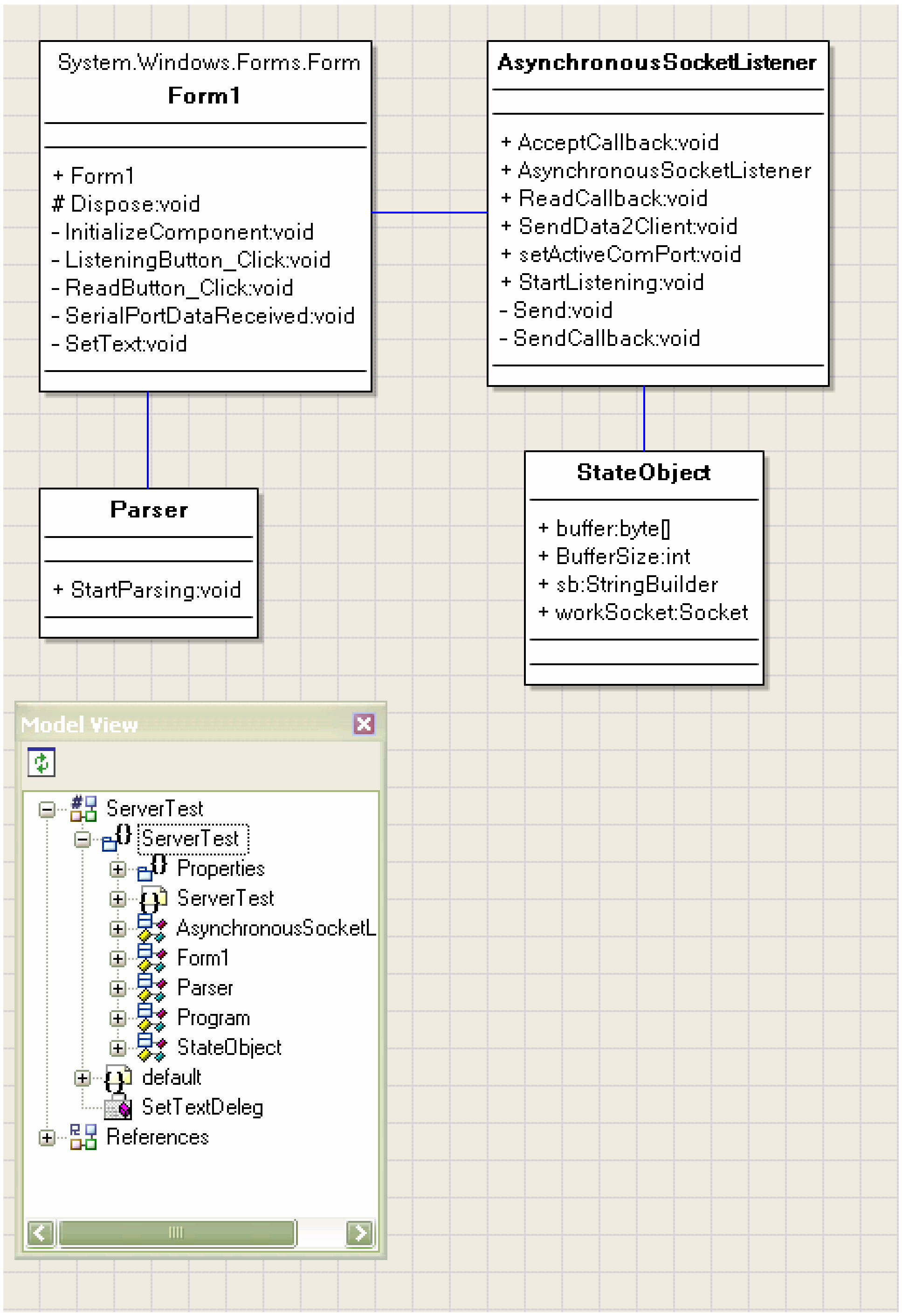
1. Create a socket.
2. Bind the socket to a local IPEndPoint.
3. Place the socket in listen mode.
4. Accept an incoming connection on the socket.

In addition to Server, on the Client part, we have a working TCP server, and we can create a simple TCP client program to interact with it. There are only two steps required to connect a client program to a TCP server:

1. Create a socket.
2. Connect the socket to the remote server address (Blum, 2006).

4.3.2.2 ServerTest Program

ServerTest Program has four classes written in Visual Studio .NET/C# programming language. They are *Form1*, *AsynchronousSocketListener*, *Parser* and *StateObject* shown in figure 4.5 as a UML (Unified Modeling Language) diagram. *Form1* class deals with user interface components such as initialization of components, button click events and serial port communication. Construction and initialization of the socket data buffer is established in *StateObject* class**.** *AsynchronousSocketListener* class is the essential part of the *ServerTest* program which establishes a TCP/IP socket and listens to the socket for a client to make a connection request. *Parser* is an object which parses the serial port data sent by the RV-2AJ.


System.Windows.Forms.Form
Form1

+ Form1
Dispose:void
- InitializeComponent:void
- ListeningButton_Click:void
- ReadButton_Click:void
- SerialPortDataReceived:void
- SetText:void

AsynchronousSocketListener

+ AcceptCallback:void
+ AsynchronousSocketListener
+ ReadCallback:void
+ SendData2Client:void
+ setActiveComPort:void
+ StartListening:void
- Send:void
- SendCallback:void

Parser

+ StartParsing:void

StateObject

+ buffer:byte[]
+ BufferSize:int
+ sb:StringBuilder
+ workSocket:Socket

Model View

```
ServerTest
  ServerTest
    Properties
    ServerTest
    AsynchronousSocketL
    Form1
    Parser
    Program
    StateObject
  default
  SetTextDeleg
References
```

Figure 4.5 Server Component Model View and UML Diagram of Server

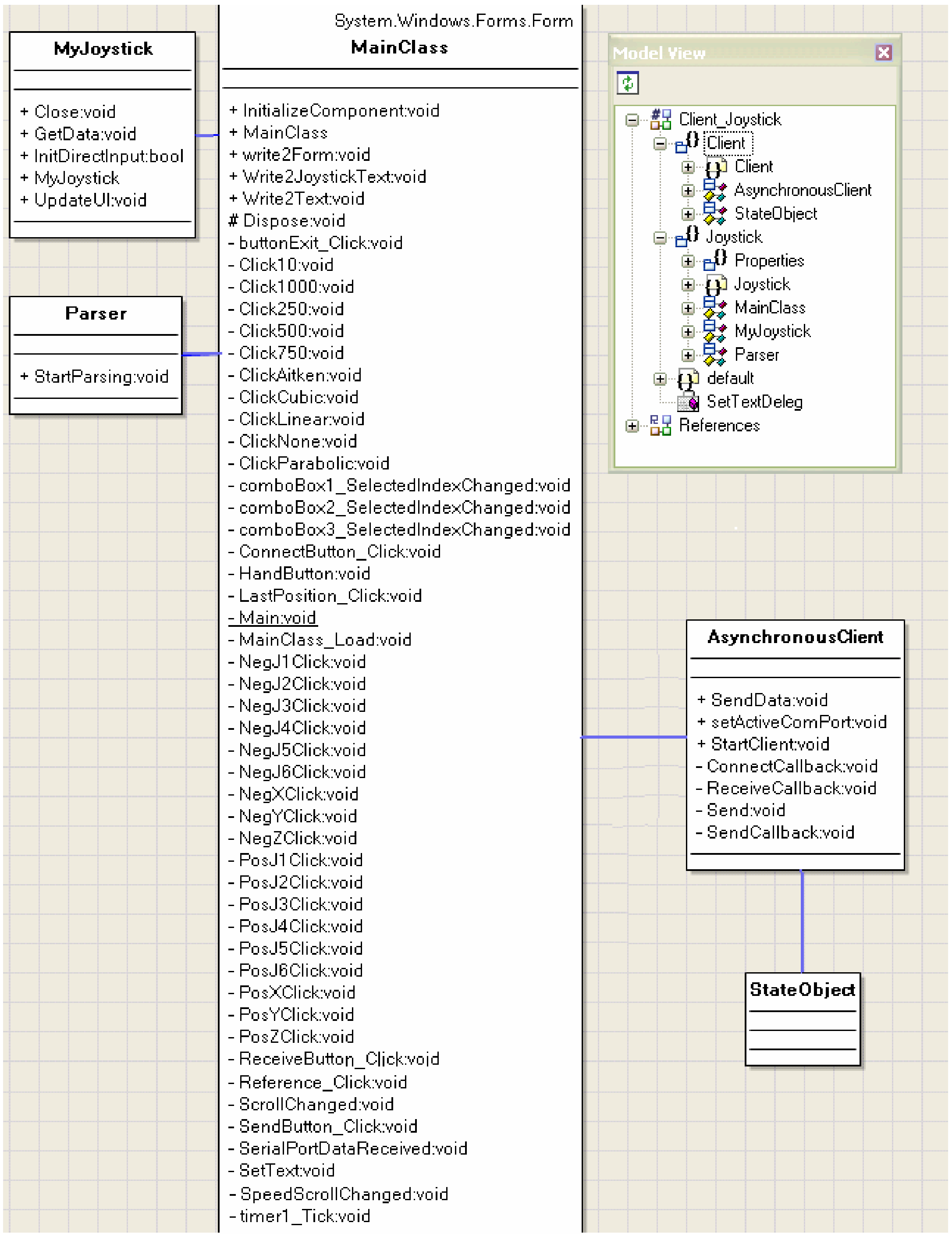


Figure 4.6 Client Component Model View and UML Diagram of Client

4.4 CLIENT SIDE COMPONENTS

4.4.1 .NET Client Component & Client User Interface

In computing, a client is a system, or a computer program or terminal that accesses a remote service, requests information or services from another computer (a server) on the network.

The client side, *Client_Joystick* interface, in this distributed environment contains the *Client* component, to reference the server side component through .NET Remoting, as well as *Joystick* component, to implement slave arm's all functionality to the master arm located in local environment.

UML diagram of Client Side Components is shown in figure 4.6.

4.4.1.1 Client Component

Client component contains all the definitions to execute methods on Mitsubishi RV-2AJ. With the help of *Client_Joystick* interface we can also get or set the public properties of the above mentioned two components located on the server side. In the beginning, after the client side program is executed and initialized, user can communicate with the server side program by clicking connection button. Once the network connection with the server established, the client sends the reference position coordinates to initialize the slave robot arm.

Client component, as in *ServerTest* Program, has *AsynchronousClient* class to create a socket to establish a server-client communication and read-write operation to this socket.

4.4.1.2 Joystick Component

This component implements all the functionality required to interact slave robot arm with a master arm, in this case a joystick.

To handle the Kontorland Joystick Device by .NET, Microsoft DirectX API is used. Microsoft DirectX is a collection of Application Programming Interfaces (API) for handling tasks related to multimedia, especially game programming and video, on

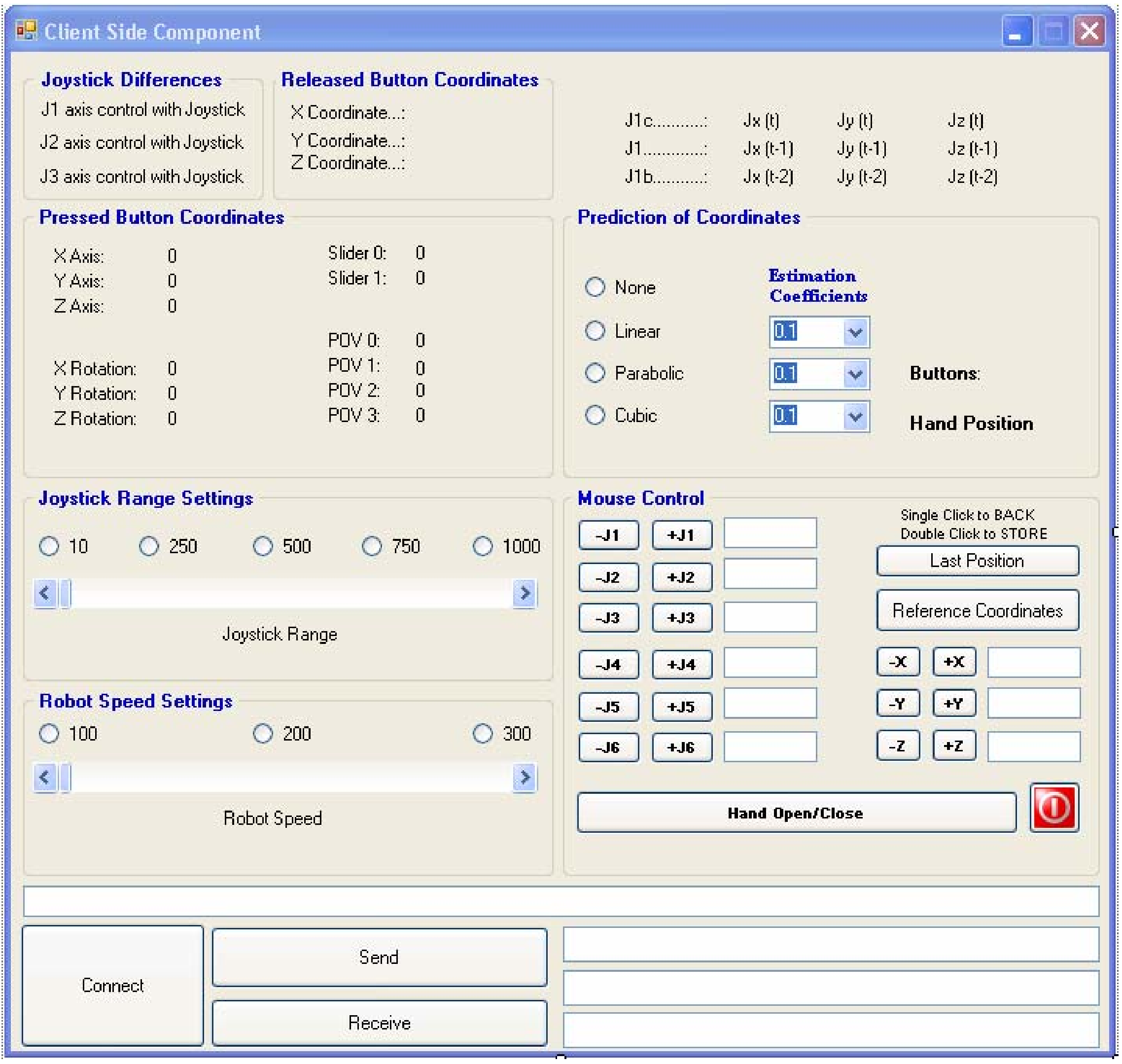
Microsoft platforms. Direct3D is widely used in the development of computer games for Microsoft Windows. DirectX is also used among other software production industries, most notably among the engineering sector because of its ability to quickly render high-quality 3D graphics using DirectX-compatible graphics hardware.

In the *MyJoystick* class, we have *Close (), GetData (), InitialDirectInput (), MyJoystick (), UpdateUI ()* methods. *InitialDirectInput ()* method is used to initialize all functions of the Joystick Device. After *GetData ()* method checks device's validation, it reads device state values. *UpdateUI ()* method updates old state values with new ones depends on the pushed buttons. *MyJoystick ()* method is a constructor to create a new Joystick object to be used by other threads.

Construction and initialization of the socket data buffer is established in *StateObject* class**.** *AsynchronousClient* class is the communication part of the *Client_Joystick* program to make a connection request with *ServerTest* which establishes a TCP/IP socket. *Parser* is an object which parses the serial port data sent by the RV-2AJ.

MainClass class is the essential part of the client side components and it supplies a direct interaction interface to the operator to manipulate the robot arm. *Client_Joystick* Graphic User Interface shown in figure 4.7 provides user:

- To control RV-2AJ Hand
- To set Joystick Sensitivity
- To set RV-2AJ Speed Parameter
- To move RV-2AJ in Joint Reference Frame,
- To move RV-2AJ in World Reference Frame,
- To maneuver RV-2AJ to reference coordinate
- To establish a TCP/IP socket communication with Server
- To view transferred and non-transferred Joystick Movement values
- To predict more realistic coordinates by Coordinate Estimation panel



X Axis: 0
Y Axis: 0
Z Axis: 0

X Rotation: 0
Y Rotation: 0
Z Rotation: 0

Slider 0: 0
Slider 1: 0

POV 0: 0
POV 1: 0
POV 2: 0
POV 3: 0

Linear: 0.1
Parabolic: 0.1
Cubic: 0.1

Figure 4.7 Client Side Gaphic User Interface

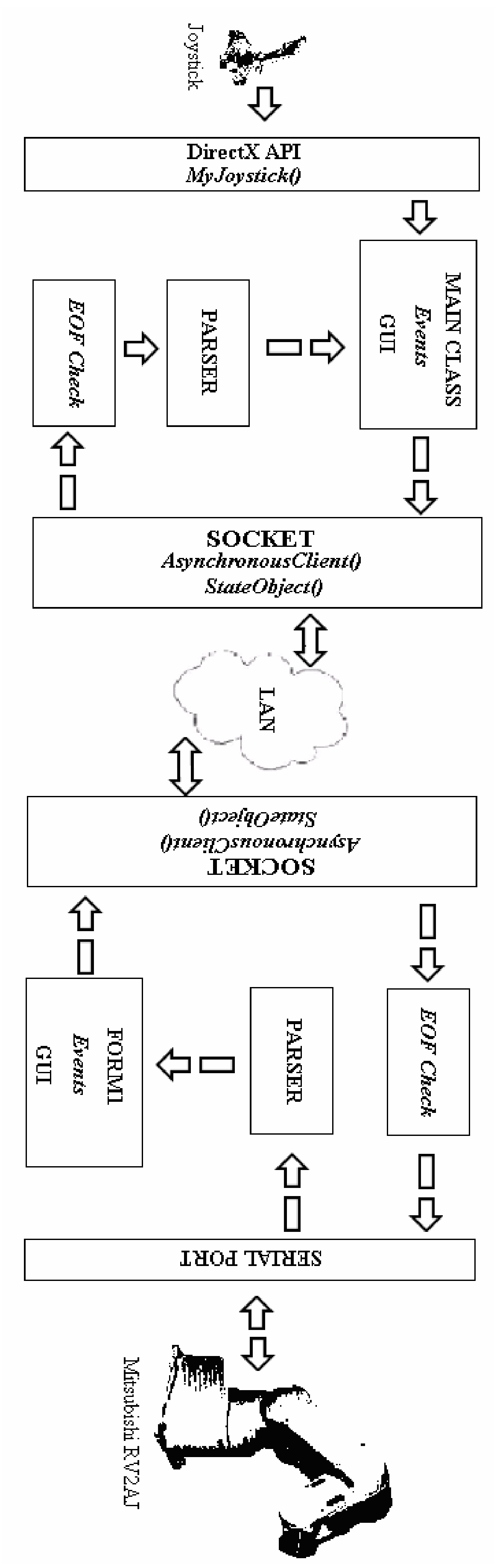


Figure 4.8 Server and Client Side Integrated Scheme

CHAPTER 5

PERFORMANCE EVALUATION

There exist many day-to-day activities that require continuous human control for their successful and safe completion. Driving an automobile, riding a bicycle, flying an aircraft are three examples. Each of these tasks involves the human being acting as a feedback element in a control system. The importance of such human feedback activity in the operation of many engineering systems has led to the development of a separate discipline called manual feedback control, or more simply, manual control. As a distinct discipline, manual control is approaching its sixtieth year of existence.

The past history of systems affecting an area of interest is fundamental *to* the success of forecasting. Atmospheric systems usually change slowly, but, continuously with time. That is, there is continuity in the weather patterns on a sequence of weather charts. When a particular pressure system or height center exhibits a tendency to continue without much change, it is said to be persistent. These concepts of persistence and continuity are fundamental *forecast* aids. Forecasting is done by using many mathematical methods. Extrapolation is the one of them to get accurate forecasting results.

In mathematics, ***extrapolation*** is the process of constructing new data points outside a discrete set of known data points. It is similar to the process of interpolation, which constructs new points between known points, but its results are often less meaningful, and are subject to greater uncertainty. The extrapolation procedures used in forecasting may vary from simple extrapolation to the use of more complex mathematical equations and analog methods based on theory. The forecaster should extrapolate past and present conditions to obtain future conditions (Sheridan, 1997), (Brezinski, 1991).

5.1 EXTRAPOLATION METHODS

5.1.1 Linear Extrapolation

This means creating a tangent line at the end of the known data and extending it beyond that limit. Linear extrapolation will only provide good results when used to extend the graph of an approximately linear function or not too far beyond the known data. A linear extrapolation can be done easily with a ruler on a written graph or with a computer.

$$\hat{x}_n = x_{n-1} + w_1 \cdot (x_{n-1} - x_{n-2}) \tag{5.1}$$

5.1.2 Polynomial Extrapolation

A polynomial curve can be created through the entire known data or just near the end. The resulting curve can then be extended beyond the end of the known data. Polynomial extrapolation is typically done by means of Lagrange interpolation or using Newton's method of finite differences to create a Newton series that fits the data. The resulting polynomial may be used to extrapolate the data.

$$\hat{x}_n = x_{n-1} + w_1 \cdot (x_{n-1} - x_{n-2}) + w_2 \cdot (x_{n-2} - x_{n-3}) \tag{5.2}$$

5.1.3 Cubic Extrapolation

A conic section can be created using three points near the end of the known data. It means, Cubic extrapolation is using a polynomial in which the highest power is the third power. $f(x) = Ax^3 + Bx^2 + Cx + D$ where $A \neq 0$. If the cubic section created is an ellipse or circle, it will curve back on itself. A parabolic or hyperbolic curve will not, but may curve back relative to the X-axis. This type of extrapolation could be done with a conic sections template on a written graph or with a computer.

$$\hat{x}_n = x_{n-1} + w_1 \cdot (x_{n-1} - x_{n-2}) + w_2 \cdot (x_{n-2} - x_{n-3}) + w_3 \cdot (x_{n-3} - x_{n-4}) \tag{5.3}$$

5.2 PERFORMANCE EVALUATION TESTS

Server side components are: (1) RV-2AJ CR1 Controller Program, (2) .NET Server Component & Server User Interface.

.NET Server Component acts as a software proxy of the robot for which commands are issued. The CR1 Controller Program reads robot current robot joint $\boldsymbol{\theta_p(t)}$ as a 6 **x** 1 vector and current robot cartesian $\mathbf{X_p(t)}$ as a 3 x 1 vector. In addition to CR1 Controller, .NET Server Component reads the incremental cartesian motion $\mathbf{\Delta X}$ produced by joystick and sent by the client as a 3 x 1 vector. A command for an incremental cartesian motion $\mathbf{\Delta X}$ is sent directly to robot. A command for an incremental cartesian motion is specified in hand frame translation $\mathbf{\Delta X}$ (3 x 1) and orientation matrix $\mathbf{\Delta M}$ (3 x 3). RV-2AJ computes the new robot hand position
$\mathbf{X_{new}} = \mathbf{X_p(t)} + \mathbf{\Delta X}$. RV-2AJ computes the inverse kinematics for X_{new} and finds the corresponding joint vector $\boldsymbol{\Delta\theta}$ which is sent to robot.

The force sensing component *MyJoystick(),* implemented in a separate thread, reads the master robot arm, a joystick in our project, and creates a stream of incremental cartesian force directed to the slave arm, remote side robot arm. In the client side .NET program, we have a timer which provides a mechanism for executing our joystick reading method at specified intervals. At every *TimerTick()* event, *MyJoystick()* class checks whether any change occurred at joystick position, or not. If there is a change at master arm position, the difference between the previous checked motion and current one is sent to server in a robot incremental cartesian motion format.

Performance evaluation experiments under different conditions were carried out on the server side distributed .NET framework. The connection between slave robot arm and PC is established by RS-232 port. We used a PC which has 3.00 GHz P-IV microprocessor and 512 MB RAM. Each force data packet contains 8 double values, 6 one for cartesian coordinates and the other two ones to set robot configuration – motion type, hand position - which equal 8 x 8 = 64 bytes.

The experiments are done based on the estimation techniques which are explained in section 5.1. Only the force vector is sent over a RS-232 communication port. The joystick timer's interval was set to 200 milliseconds. It means, joystick motion was checked at every 0.2 seconds and incremental change was transferred to robot if robot movement completed after previous incremental data sent.

In the experiments, performance of the estimation techniques is defined as the matching of desired and actual positions. Therefore, performance evaluation of the system is carried out by measuring the differences between the desired positions which are produced by joystick and actual positions which are the current coordinate values of robot arm. The error analysis of the different estimation techniques was calculated by using these differences. In the error calculation, we have used Least Square Method which is aimed at minimizing the sum of squared deviations of the observed values for the actual variable from those desired by the model.

5.2.1 Least Squares Method (LSM)

The method of least squares assumes that the best-fit curve of a given type is the curve that has the minimal sum of the deviations squared (*least square error*) from a given set of data.

Suppose that the data points are $(x_1,y_1), (x_2,y_2),....., (x_n,y_n)$ where, x is the independent variable and y is the dependent variable. The fitting curve $f(x)$ has the deviation (error) d from each data point, i.e., $d_1 = y_1 - f(x_1), d_2 = y_2 - f(x_2),..., d_n = y_n - f(x_n)$. According to the method of least squares, the best fitting curve has the property that (eFunda, 2007):

$$\Pi = d_1^2 + d_2^2 + \ldots + d_n^2 = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \text{a minimum} \tag{5.4}$$

In the experiments, we have used 3 different approaches to increase the performance of the robot control with joystick. By using joystick, a rectangle shape with X and Y dimension without any depth, ignoring Z dimension, is drawn 3 times. Experiments are carried out by using the Linear, Parabolic and Cubic Extrapolation algorithms. For every extrapolation methods, different estimation coefficients are used to measure the sensitivity of the performance. All of the experiments are repeated three times by comparing more than 140 samples of actual and desired positions to get more reliable and accurate results and MATLAB program, shown in Appendix C, was used to calculate, plot and compare these results. Experiment setups and results are shown in Table 5.1.

None			
Est. Coef.	**No**		
X error	11,18%		
Y error	11,60%		
Z error	0%		
LINEAR EXTRAPOLATION			
Est. Coef.	**0,1**	**0,5**	**0,9**
X error	9,61%	11,79%	29,22%
Y error	12,24%	12,47%	30,68%
Z error	0%	0%	0%
PARABOLIC EXTRAPOLATION			
Est. Coef.	**0,1**	**0,5**	**0,9**
X error	6,03%	12,37%	44,29%
Y error	5,46%	13,56%	49,09%
Z error	0%	0%	0%
CUBIC EXTRAPOLATION			
Est. Coef.	**0,1**	**0,5**	**0,9**
X error	3,08%	7,39%	x
Y error	3,35%	13,69%	x
Z error	0%	0%	x

Table 5.1 Experiment Results

5.2.2 Experimental Results

5.2.2.1 Experiment Result without Using Estimation Techniques

In this experiment, we sent only the joystick previous and current motion differences that are read with 200 millisecond time intervals to the robot arm. *TransferredData*(x_{n-1})=*JoystickCurrentPosition*(p_{n-1}) – *JoystickPreviousPosition*(p_{n-2})

$$x_{n-1} = p_{n-1} - p_{n-2} \qquad \hat{x}_n = x_{n-1} \tag{5.5}$$

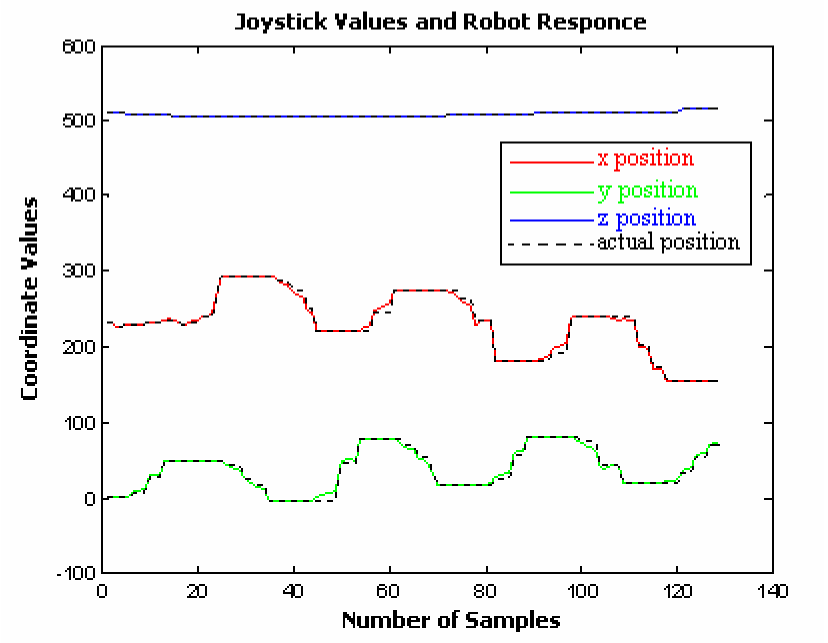


Figure 5.1 Joystick Incremental Motion (Desired) and Robot Response (Actual) (None of Estimation Techniques)

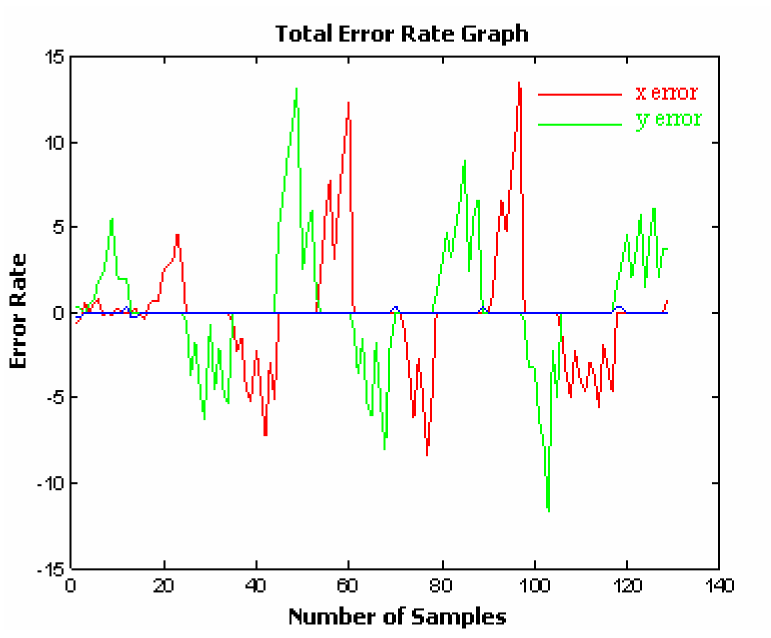


Figure 5.2 Desired and Actual Position Error (None of Estimation Techniques)

5.2.2.2 Experiment Result Using Linear Extrapolation

$$x_{n-1} = p_{n-1} - p_{n-2} \qquad \hat{x}_n = x_{n-1} + w_1 \cdot (p_{n-1} - 2p_{n-2} + p_{n-3}) \qquad (5.6)$$

In this experiment, estimation coefficient is $w_1 = 0.1$.

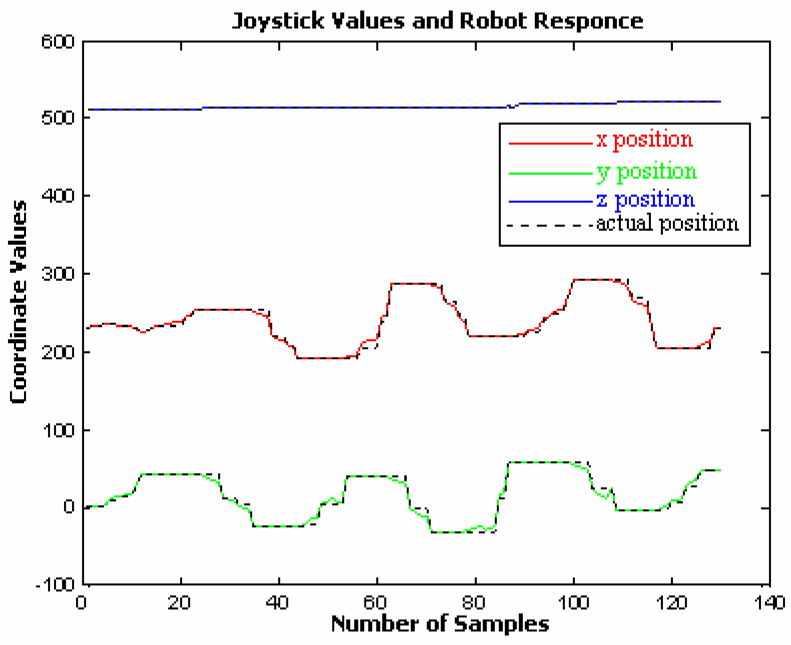


Figure 5.3 Linear Extrapolated Desired Motion and Actual Motion

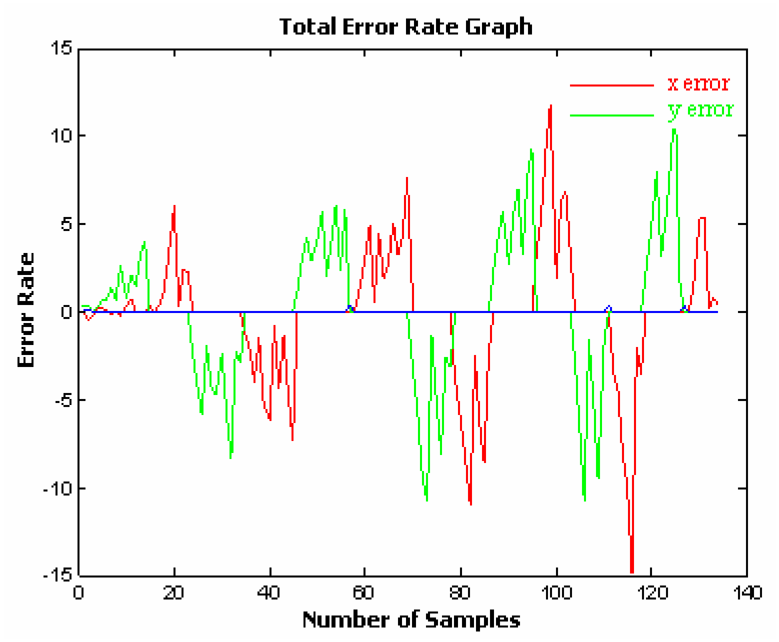


Figure 5.4 Desired and Actual Position Error

5.2.2.3 Experiment Result Using Parabolic Extrapolation

$$x_{n-1} = p_{n-1} - p_{n-2} \tag{5.7}$$

$$\hat{x}_n = x_{n-1} + w_1 \cdot (p_{n-1} - 2p_{n-2} + p_{n-3}) + w_2 \cdot (p_{n-2} - 2p_{n-3} + p_{n-4})$$

(5.8)

where, $w_1 = w_2 = 0.1$

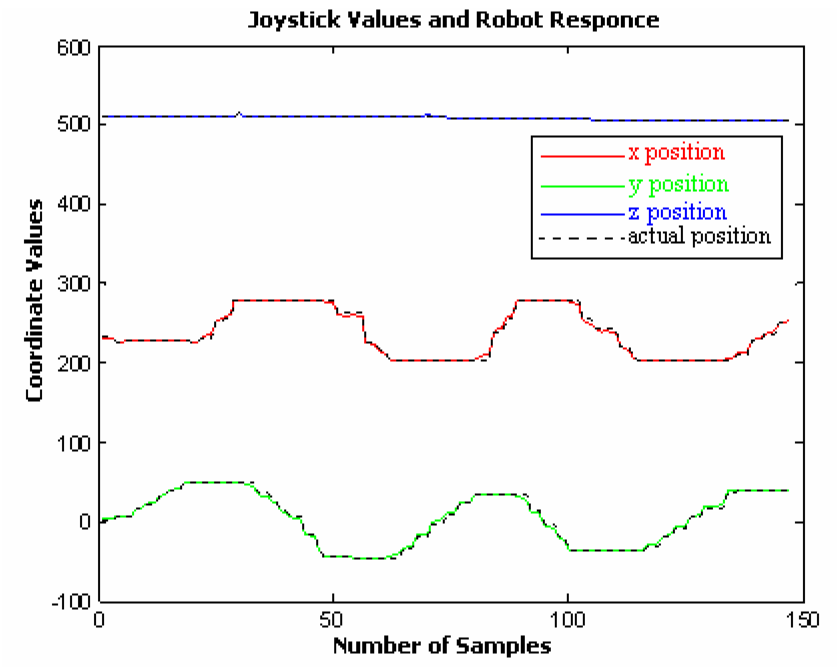


Figure 5.5 Parabolic Extrapolated Desired Motion and Actual Motion

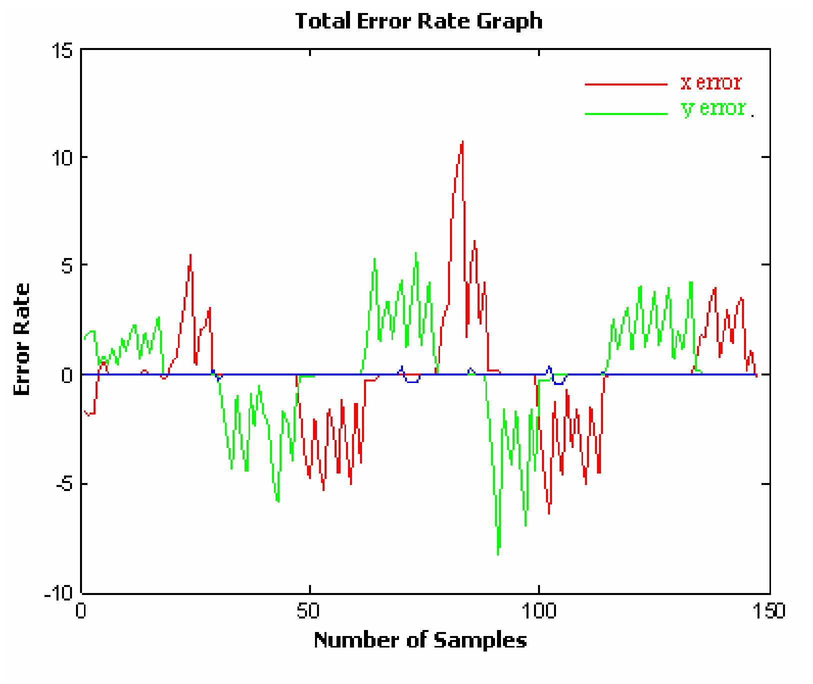


Figure 5.6 Desired and Actual Position Error

5.2.2.4 Experiment Result Using Cubic Extrapolation

$$x_{n-1} = p_{n-1} - p_{n-2} \tag{5.9}$$

$$\hat{x}_n = x_{n-1} + w_1 \cdot (p_{n-1} - 2p_{n-2} + p_{n-3}) + w_2 \cdot (p_{n-2} - 2p_{n-3} + p_{n-4}) + w_3 \cdot (p_{n-3} - 2p_{n-4} + p_{n-5}) \tag{5.10}$$

where, $w_1 = w_2 = w_3 = 0.1$

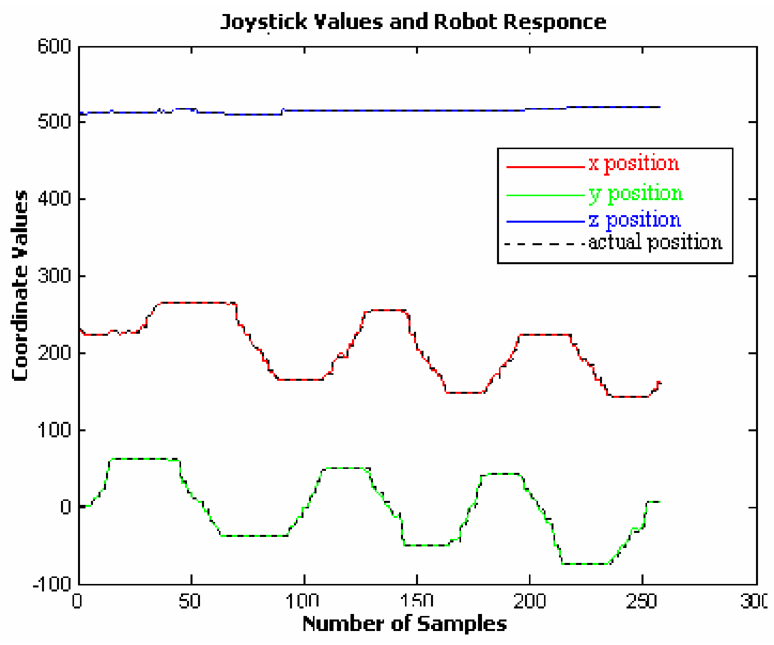


Figure 5.7 Cubic Extrapolated Desired Motion and Actual Motion

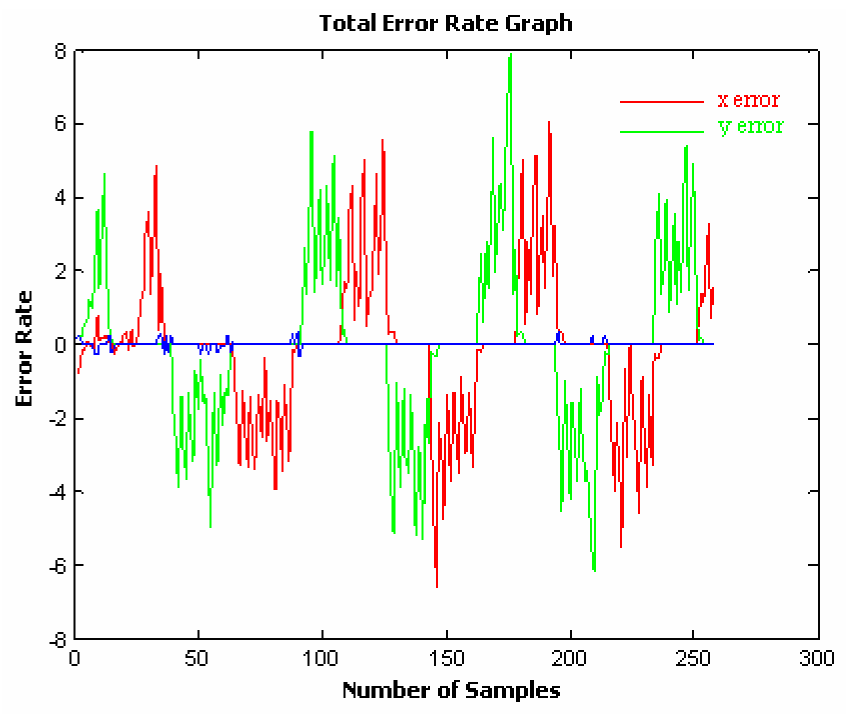


Figure 5.8 Desired and Actual Position Error

5.3 COMPARISION

When we compare the results with respect to the estimation techniques tested with different estimation coefficients, the most reliable and accurate algorithm can be defined as Cubic Extrapolation in the case of its estimation coefficients are equal to 0.1. For Linear and Parabolic Extrapolations, when estimation coefficient increases from 0.1 to 0.5 or 0.9, error rate also increases with respect to increase rate. For the Cubic Extrapolation, similar results are obtained. Even performance of Cubic Extrapolation when its estimation coefficient w is equal to 0.9 could not be measured because predicted coordinates exceeded the robot workspace.

We try to measure the affect of the time interval which can be defined as the sampling time of reading the joystick or transferred data intervals. But, increase on the time interval, 200 ms to 800 ms, affected the error with an observable rate. It means, error increased more than 2 times when interval increased four times.

CHAPTER 6

AN AUGMENTED REALITY SYSTEM FOR TELEROBOTICS

Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it.

According to Azuma (Azuma, 1997)**,** AR systems are required to have the following three characteristics:

1) Combines real and virtual
2) Interactive in real time
3) Registered in 3-D

AR can help to users by annotating objects and environments with public or private information. Robot path planning can be facilitated using AR in situations where a large time delay is present between operator and the robot. Operator can preview the effect of the move on the local display overlaid on the remote world image. Once operator satisfied with the movement, he can send the actual command.

6.1 VISUALIZATION SYSTEMS BASED ON USED EQUIPMENTS

Visualization systems are classified based on the used equipments. There are many Stereo3D Image Formats:

- Interleave/Interlace, image format in which the left and right views are combined or "woven" together, line by line. Each line alternates between the left and right view of the image.

- Line Blanking, this method is suitable when the output on the screen is in interlaced format. The line-blanking controller hides odd lines for right eye and even for left eye.

- Page flipping, method of viewing stereoscopic content by using video hardware to rapidly switch the left and right eye view in temporal sync with shutter glasses.

- Sync-doubling (above-and-below), the above-and-below standard is perhaps more effective and does not require additional hardware support inside the computer. In this scenario, the left view is rendered in the upper half of the screen at half of its vertical resolution, while the right view is rendered in the lower half. These images are separated by a variable number of black lines that will operate as a vertical blanking interval on final display.

Each format requires different techniques and/or equipment for generation and visualization.

For 3D visualization, we used one of the basically two major classes of 3D visualization techniques which is shuttering glasses. The shutter glasses achieve stereo by using frame sequential techniques. Shutter glasses enable the subjects to see stereoscopic images. The glasses alternately "shutter," i.e. block, the viewer's left, then right, eyes from seeing an image. The stereoscopic image is alternatively shown in sequence left-image, right-image synchronously with the shuttering of the glasses.

Most of the available monitors and display adapters can support refresh frequencies equal or above 120 Hz at resolutions of 1024x768 or above. Therefore, Eye3D Premium Shuttering Glasses support 3D visualization with very high details as shown in figure 6.1. For more details, visit the web page (eDimensional Company, 2007).

Figure 6.1 Eye3D Premium Shuttering Glasses for 3D Visualization (eDimensional, 2007)

Just for the illustrative purposes, the “Eye3D Premium” shuttering glasses can support resolutions (in pixels) up to 2048x1538 at 120 Hz, and 1856x1392 at 140 Hz. These specs are available only in high-end monitors.

In the Eye3D Premium product, we have LCD Glasses, LCD Glasses Controller Box and IR Transmitter. LCD Glasses Controller Box gets the VGA output signal of the PC, and generates a modified VGA signal for the monitor, and control/synchronization signals for the IR transmitter, which in turn controls the LCD Glasses.

6.2 VISUALIZATION SYSTEMS BASED ON USED IMAGE AND VIDEO GENERATION TECHNIQUES

At the beginning of project, we tried to generate artificial 3D images to understand the basics of the 3D visualization. For this purpose, we generated two different images artificially, one for the 1st eye and the other one for the 2nd eye, as shown in figure 6.2.

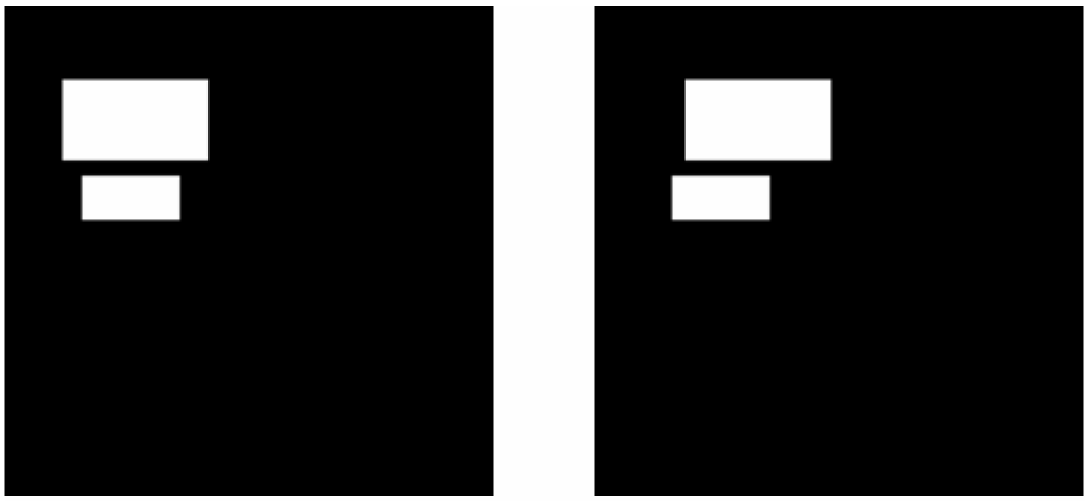


Figure 6.2 Artificially Generated Images for the 1st and the 2nd eye

Two different methods, Interleaved and Sync-Doubling Mode, are used to get an artificial 3D scene. To generate an artificial 3D image in Interleaved Mode, a MATLAB Program is used as given in Appendix D. Using this program; we create an image in which the left and right views of artificial images shown above are combined line by line. Combined image is shown in figure 6.3.

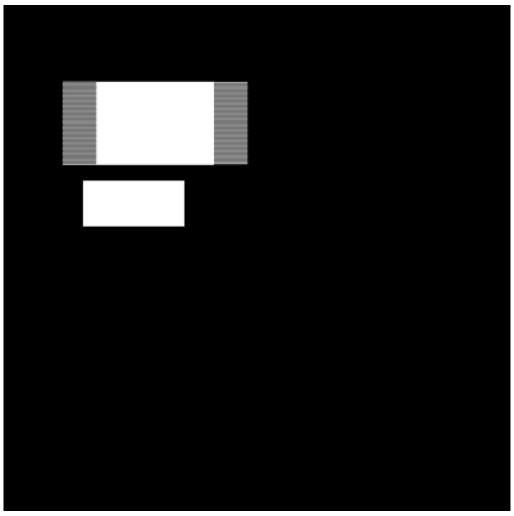


Figure 6.3 Artificially Generated 3D Image in Interleaved Mode

This image, when viewed on a CRT monitor with LCD glasses, and controller operating in Interleaved Mode, will result an artificial 3D scene. This has been tested. It has been verified that both eyes see the lower rectangle in the same position, and upper one in different positions.

To generate an artificial 3D image in Sync-Doubling Mode, images shown above are used; in this case, images are combined as in a single image, one above the other, as shown in figure 6.4.

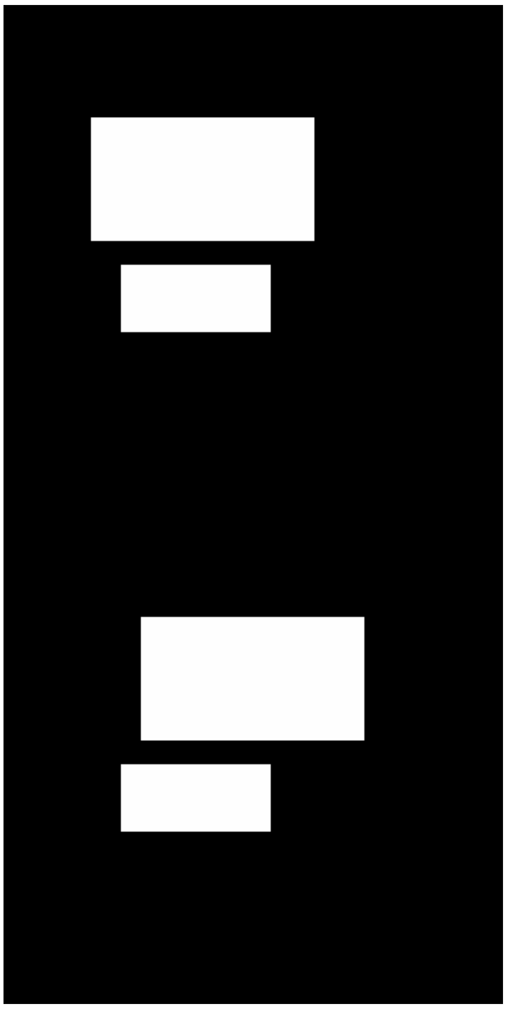


Figure 6.4 Combined 1st and the 2nd eye Artificially Generated Images

This image, when viewed on a CRT monitor with LCD glasses, and controller operating in Sync-Doubling Mode, will result an artificial 3D scene as shown below.

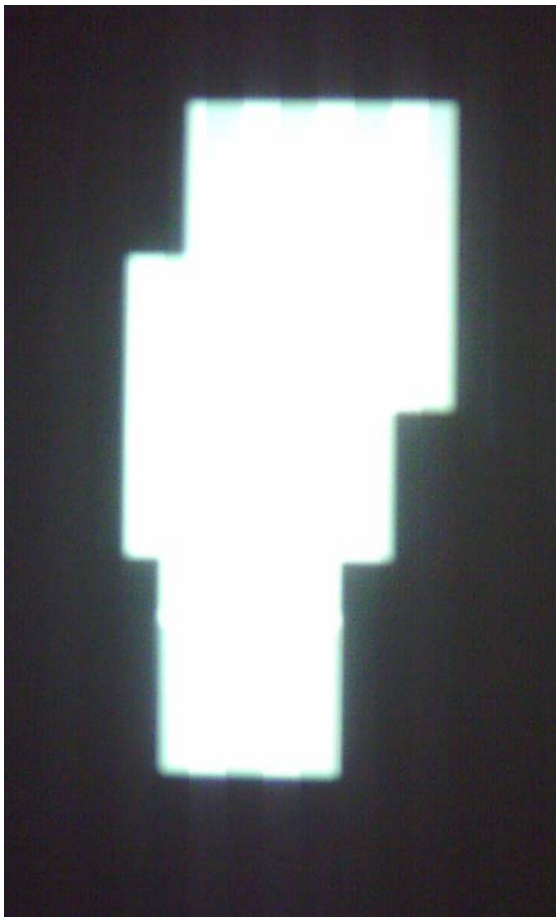


Figure 6.5 Artificially Generated 3D Image in Sync-Doubling Mode

6.3 3D LIVE VIDEO GENERATION AND 3D VISUALIZATION IN SYNC-DOUBLING MODE

To generate 3D video content, we used Parallel Camera Configuration, shown in figure 6.6, which can be used to observe with high accuracy a 3D object under magnification and depth. This is a very commonly used technique for 3D video generation. One is used for the left eye image, and the other is used for the right eye images. Approximate camera separation is 6 cm.

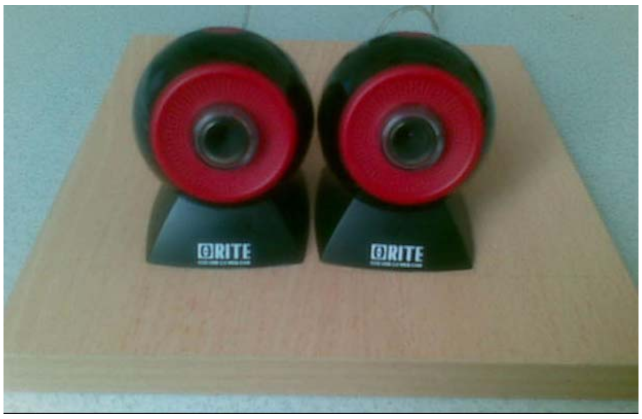
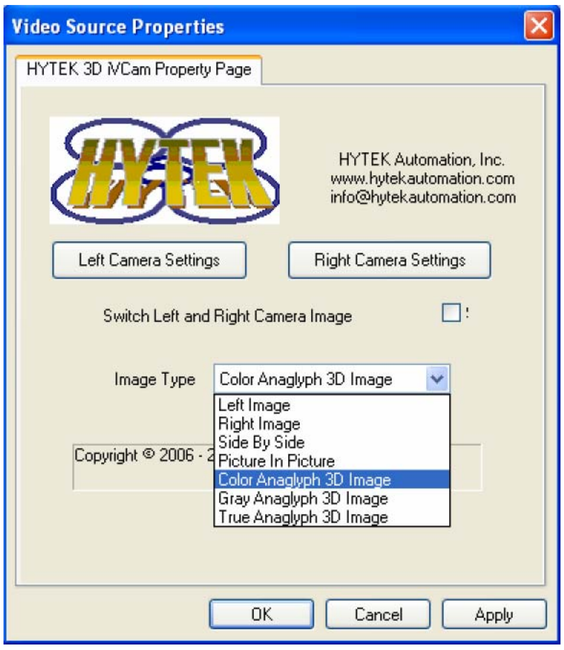


Figure 6.6 Used Parallel Camera Configuration

One AMCAP windows is opened. AMCap is a small yet fully functional video capture and preview application compatible with Microsoft DirectShow. It is based on the sample *AMCap* source code from the Microsoft DirectX 9 SDK. Two webcams are connected to PC via USB port. To combine left camera and right camera images in the only one AMCAP window, we used HYTEK General Stereo 3D Camera Driver. HYTEK software helped us to turn a pair of webcams into stereo 3D video generator. It has many choices to create a 3D images. We used *Up and Below* image format which combines a pair of camera images in a one window placed with one above the other.



HYTEK Automation, Inc.
www.hytekautomation.com
info@hytekautomation.com

Switch Left and Right Camera Image ☐

Image Type: Color Anaglyph 3D Image

Left Image
Right Image
Side By Side
Picture In Picture
Color Anaglyph 3D Image
Gray Anaglyph 3D Image
True Anaglyph 3D Image

Copyright © 2006 - 2

Figure 6.7 HYTEK General Stereo 3D Camera Driver Interface

Transferring the video images is established by the Active WebCam video program. It captures images up to 30 frames per second from any video device including USB, analog cameras, TV-boards, camcorders, and from network IP cameras. The program performs simultaneous recording and broadcasting from unlimited number of cameras. Also, the captured video can be viewed using any Internet browser

CHAPTER 7

CONCLUSION

Real-time control of telerobots in the presence of time delays and data loss is an significant research area. Different techniques have been applied to realize a reliable and efficient telerobotic framework. Previously DCOM(Distributed Component Object Modeling) has been used in the implementation of a component based telerobotic framework by Yuek et al. (Ho et al., 2000). DCOM, however, has some limitations related to deployment on remote machines and requires the registration of distributed components before interfacing with them. Microsoft .NET based components are an ideal update to the DCOM and use highly optimized network socket connections for inter-object communications, (Microsoft, 2007).

This work uses the above mentioned .NET based distributed components for the design and development of a reliable telerobotic scheme. Because telerobotics encourages the transfer of all possible real-time data from the remote to client side, force feedback and video of the remote scene have now become essential elements of a good telerobotic system. .NET technologies can offer excellent platform to build such multi-streaming application.

Primarily we have considered,

1) the development of an efficient system to transfer stereo video data from the server to client,

2) to output this video data to the user in order to provide a 3D view of the remote scene,

3) development of a real-time telerobotic framework

4) to explore augmented reality as a way to compensate for network delays in telerobotics. All of these areas are addressed adequately in this text while providing a

valuable insight into the use of latest software trends in solving multi-disciplinary problems.

7.1 CONTRIBUTIONS

Brief account of the contributions made through this thesis work is given below:

1. Different output techniques for stereo video are implemented with eye-shuttering glasses like Interleaved, Line Blanking, Over-Under and their performance is evaluated.

2. A component based framework for telerobotics is designed, implemented and its performance is evaluated to study the effects of multi-threading on real-time telerobotics that facilitates:

(a) Controlling a robot over LAN in real-time, and

(b) At the same time, providing 3D views of the remote scene

(c) Evaluating different algorithms to increase the performance of Joystick-Robot Arm compensation.

This scheme has significantly reduced the network delays in a given telerobotic scenario while providing a very reliable connection between client and server sides.

3. Different geometric working frame is provided for the operator to enhance his maneuverability in the remote environment.

REFERENCES

Al-Harthy A.. “*Design of a telerobotic system over a Local Area Network”*. M.Sc. Thesis, King Fahd University of Petroleum and Minerals, January 2002.

Azuma Ronald T., “*A Survey of Augmented Reality”*, In Presence: Teleoperators and Virtual Environments 6, 355-385, August 1997.

Bellenger D. M., Steven P. Russell, “*Dual band modem for high bandwidth, communications”*, 1999
http://www.freepatentsonline.com/5982768.html

Brezinski C. and M. Redivo Zaglia, “*Extrapolation Methods. Theory and Practice”*, North-Holland, 1991

Bungert Christoph, 2007
http://www.stereo3d.com/nuview.htm

Buzan F. T. and Thomas B. Sheridan, “*A Model-Based Predictive Operator Aid For Telemanipulators With Time Delay”*, Proc.IEEE Int.Conf. on System Man. And Cybernetics, 1:138-143,1989

Chen Tse Min, Ren C. Luo, “*Remote Supervisory Control of An Autonomous Mobile Robot Via World Wide Web”*, Proc. of ISIE ’97,1:SS60-SS64,1997

Cisco, “*Managed Voice-Smoothing the Transition to Communications”*, 2007
www.cisco.com/en/US/netsol/ns458/net_value_proposition0900aecd80151369.html

Chong N., K. Ohba, T. Kotoku, K. Komoriya, N.Matsuhira, K. Tanie, “*Coordinate Rate Control of Multiple Telerobot Systems with Time Delay”*, Proc.IEEE Int.Conf. on System Man. And Cybernetics, Pages V1123-V1128, October 1999.

COSIROP 2.0 “*Programming Software for Mitsubishi Industrial Robots”*, 2006
http://www.mitsubishi-automation.com/products/software_COSIROP_content.htm

Damian Mirela,” *TCP Sockets”*, August 23, 2006
http://www.csc.villanova.edu/~mdamian/Sockets/TcpSockets.htm

Eck David J., “Introduction *to Programming Using Java 4th Edition”*,2004
http://oopweb.com/Java/Documents/IntroToProgrammingUsingJava/VolumeFrames.html

eDimensional, 2007
https://edimensional.com/support.php

eDimensional Company, 2007, https://edimensional.com/product_info.php?cPath=21&products_id=29&osCsid=5c8a2ba365a41bea78b52c90ec6665f7

eFunda, "*The Method of Least Squares*", 2007 http://www.efunda.com/math/leastsquares/leastsquares.cfm

Fischer C., M.Buss, G. Schmidt, "*Hierarchical Supervisory Control of Service Robot Using HuMan-Robot-Interface*", Proc. of IROS, 1:1408-1416, 1996

Green P. S., J.W. Hill, J. F. Jensen, and A. Shah, "*Telepresence surgery*", IEEE Eng. Med. Biol. Mag., vol. 14, no. 3, pp. 324–329, May/Jun. 1995.

Gupta Radhika, "*Focused Framework Comparison: Remote Object Access using CORBA, J2EE and .NET*", CPSC 689-608: Industrial Frameworks for Distributed Systems Project 2, November 3, 2003

Haklidir M., Taşdelen I., *Modelıng And Sımulatıon Of An Anthropomorphıc Robot Arm By Usıng Dymola,* Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, May 29-31, 2006: 537-546

HarewoodGill Douglas (MSc Robotics), "*Workspace 5 Student Manual*", 2006

Ho Y. E.; H. Masuda; H. Oda; L. W. Stark; "*Distributed Control for Teleoperations*". Proc. Of the 1999 IEEE/ASME International Conf. on Adv. Intelligent Mechatronics, pages 323-325, September 1999.

Ho Y. E.; H. Masuda; H. Oda; L. W. Stark; "*Distributed Control for Teleoperations*". IEEE/ASME Transactions on Mechatronics, 5(2):100-109, June 2000

Introduction to the .NET Framework tutorial, 2001. http://www.devhood.com/training_modules/dist-a/Intro.NET/intro.net.htm

Iqbal A., Multistream, "*Real-time Control of a Distributed Telerobotic System*", June 2003

Jun R., "*The Magnifying Glass Approach to Augmented Reality Systems*", Proc. of ICAT'95, 1995

Kazerooni T., H. Tsay, and K. Hollerback, "*A controller design framework for telerobotic systems*", IEEE Trans. Contr. Syst. Technol., vol. 1, no. 1, pp. 50–62, Mar. 1993.

Lee, S. Bekey, G. Bejczy, A, "*Computer control of space-borne teleoperators with sensory feedback*", Proc. IEEE International Conference on Robotics and Automation. Volume: 2, page: 205- 214, Mar 1985

Lee S., S. Ro, J. Park, C. Lee, "*Optimal3D Viewing with Adaptive Stereo Displays: A Case of Tilted Camera Configuration*", ICAR '97, 1991

Lee S., S. Lakshmanan, S. Ro, J. Park, C. Lee, "*Optimal 3D Viewing with Adaptive Stereo Displays for Advanced Telemanipulation"*, International Conference on Intelliigent Robot and Systems. Pages 1007-1014, 1996

Lloyd John E., Jeffrey S. Beis, Dinesh K. Pai, David G. Lowe, "*Model-based Telerobotics with Vision"*, Proc. IEEE Int. Conf. Robotics and Automation, pp.1297-1304, 1997

Lo C. H., and A. Chalmers, "*Stereo Vision for Computer Graphics: The Effect that Stereo Vision has on Human Judgments of Visual Realism"*, ACM Transactions on Design Automation of Electronic Systems (TODAES), Pages: 238 - 271 ,2004

Luo R. C., T.M. Chen. "*Development of the multibehavior-based mobile robot for remote supervisory control through the internet"*. IEEE/ASME Transactions on Mechatronics, 5(4): 376-385, December 2000.

Matthew R. Stein, Richard P. Paul$, Paul S. Schenker and Eric D. Paljug, "*A Cross-Country Teleprogramming Experiment"*, Proc. IEEE/RSJ Int. Conf. On Intelligent Robots and Systems, 1:21-26,1995

Mayez A. Al-Mouhamed, Onur Toker, Asif Iqbal, "*Design of a Multi-Threaded Telerobotic Framework"*, ICECS 2003. pages: 1280 – 1283, Dec. 2003

Mayez A. Al-Mouhamed1, Onur Toker, Asif Iqbal, and Syed M.S. Islam "*Evaluation of Real-Time Delays for Networked Telerobotics"*, 3rd IEEE Intemational Conference on Industrial Informatics (INDIN), pages 351-356, 2005

Microsoft. "*MSDN Library"*.
http://msdn.microsoft.com/default.asp

Mitsubishi Electric, "*Instruction Manual, Specification Manual"*, 2006
http://www.mitsubishi-automation.com/products.html

Monferer A., "*Cooperative Robot Teleoperation through Virtual Reality Interfaces"*, Sixth International Conference on Information Visualization (IV'02) p. 243, 2002

Niku Saeed B., "*Introduction to Robotics, Analysis, Systems, Applications*". Prentice Hall, Inc. 2001.

Nof S., "*Handbook of Industrial Robotics"*, 2nd Edition, 1998

OMG. "*The Common Object Request Broker: Architecture and Specification, revision" 2.4.2.* 2002, http://www.omg.org

Owens R., *Computer Vision*, 1997
homepages.inf.ed.ac.uk/.../LECT1/node2.html

Paolucci F., M. Andrenucci, "*Teleoperatiorl Using Computer Networks: Prototypr Realization and Performance Analysis", Electrotechnical Conference MELECON'96 (8.th Mediterranean, 2-1156-1159,1996.

Ramm Andy, 1997
http://www.von-oppen.com/doc/ddj/articles/1997/9709/9709i/9709i.htm

Richard B.., "*C# network programming*", Jan 2006
http://www.codeproject.com/cs/internet/TCPIPChat.asp

Sheridan T. B., "*Human Supervisory Control of Robot Systems*". Proc.IEEE International Conference of Robotics Automation, page 1, 1986

Sheridan T. B., "*Telerobotics, Automation, and Human Supervisory Control*",
M.I.T Press, 1992

Sheridan T. B. *"Space Teleoperation through Time Delay: Review and Prognosis"*, IEEE Transactions on Robotics and Automation, vol. 9, no. 5. October 1993

Sheridan T. B.. Supervisory Control. G. Salvendy (2.Ed.) "*Handbook of Human factors and ergonomics*", pages 1295-1327.1997.

Sims, D., "*New realities in aircraft design and manufacture*", Computer Graphics and Applications, IEEE Volume 14, Issue 2, Page(s):91, March 1994

Sooyong L., D.-S. Choi, M. Kim, C.-W. Lee, and J.-B. Song, "*A unified approach to teleoperation: Human and robot interaction*", in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, vol. 1 , pp. 261–266, 1998

Strunk, L.M. Iwamoto, T. " *A linearly-mapping stereoscopic visual interface for teleoperation*", Proceedings. IROS '90, 429-436 vol.1, 1990

Sun Microsystems, 2007
http://java.sun.com/docs/books/tutorial/networking/sockets/index.html

Telerobot System, *"LCA Telerobot"*, 2004
systemhttp://vismi.kaist.ac.kr/2004/research/TelerobotSystem.htm

APPENDIX A

USED MELFA BASIC IV COMMANDS DEFINITIONS

CLOSE (Close)

Closes the designated file. If a file has been opened for input/output the CLOSE statement will sweep out the data in the buffer. Consequently, the output processing for the file can be completed properly.

END (End)

Ends the program execution.

DEF INTE/FLOAT/DOUBLE (Define Integer/Float/Double)

The variable declared with INT will be an integer type. (-32768 ~ +32767)

The variable declared with FLOAT will be a single-precision type. (+/- 1.70141E+38)

The variable declared with DOUBLE will be a double-precision type. (+/- 1.701411834604692E+308)

DEF JNT (Define Joint)

Declares a joint variable.

DEF POS (Define Position)

Declares a position variable.

GOTO (Go To)

Unconditionally branches to a designated line No. or label

HOPEN/HCLOSE (Hand Open/Close)

Commands the hand to open or close.

IF THEN ELSE (If Then Else)

A process is selected and executed according to the results of an expression.

INPUT # (Input)

Inputs data from a file (input device). All data uses the ASCII format.

MOV (Move)

Using joint interpolation operation, moves from the current position to the destination position.

MVS (Move S)

Carries out linear interpolation movement from the current position to the movement target position.

OPEN (Open)

Open a file or communication line

OPEN "COM1:" AS #1 'Open standard RS-232-C line as file No. 1.

PRINT (Print)

Outputs data into a file (including communication lines). All data uses the ASCII format.

PRINT#1,"PRINT TEST**"** 'Outputs the character string "**PRINT TEST**".

SPD (Speed)

Designates the speed for the robot's linear and circular movements.

MELFA-BASIC-IV Robot Status Variables

J_CURR	Returns the joint type data at the current position.
P_CURR	Returns the current position (X, Y, Z, A, B, C).
M_SPD	Returns the currently set speed during XYZ and JOINT interpolation.
JTOP	Converts the joint data into position data.
PTOJ	Converts the given position data into a joint data.

APPENDIX B

MITSUBISHI RV-2AJ CONTROLLER PROGRAM FOR .NET APPLICATION

```
2 DEF INTE M0
3 DEF INTE M1
4 DEF INTE M2
5 DEF INTE M3
6 DEF INTE M4
7 DEF INTE M5
8 DEF INTE M6
9 DEF INTE M7
10 DEF INTE MH
11 DEF INTE SORA
12 OPEN "COM1:" AS #1
15 P1=P_CURR
20 P2=P_CURR
25 J1=J_CURR
26 J2=J_CURR
27 SPD 100
28 PRINT #1, "P=", J1.J1, J1.J2, J1.J3, J1.J4, J1.J5, J1.J6,P1.X,P1.Y,P1.Z

40 INPUT #1,M0,M1,M2,M3,M4,M5,M6,M7
45 IF M0=0 THEN GOTO 50
46 IF M0=1 THEN GOTO 140
47 IF M0=2 THEN GOTO 220
48 IF M0=3 THEN GOTO 300
49 IF M0=4 THEN GOTO 400

50 J1=J_CURR
51 J1.J1=J1.J1+0.001*M1
52 J1.J2=J1.J2+0.001*M2
53 J1.J3=J1.J3+0.001*M3
54 J1.J4=J1.J4+0.001*M4
55 J1.J5=J1.J5+0.001*M5
56 J1.J6=J1.J6+0.001*M6
74 P2=JTOP(J1)
75 IF P2.Z<238 THEN GOTO 500 ELSE GOTO 78
78 MVS J1
80 IF M7=1 THEN HOPEN 1
81 IF M7=0 THEN HCLOSE 1
82 P2=P_CURR
85 PRINT #1, "P=", J1.J1,J1.J2,J1.J3,J1.J4,J1.J5,J1.J6,P2.X,P2.Y,P2.Z
```

```
110 GOTO 40

140 P1=P_CURR
141 P1.X=P1.X+0.01*M1
160 P1.Y=P1.Y+0.01*M2
170 P1.Z=P1.Z+0.01*M3
175 IF P1.Z<238 THEN GOTO 500 ELSE GOTO 180
180 MVS P1
190 IF M7=1 THEN HOPEN 1
195 IF M7=0 THEN HCLOSE 1
196 J2=J_CURR
199 PRINT #1, "P=",J2.J1,J2.J2,J2.J3,J2.J4,J2.J5,J2.J6,P1.X,P1.Y,P1.Z
200 GOTO 40

220 J1.J1=-0.0073628
221 J1.J2=0.0934216
222 J1.J3=1.385601
223 J1.J4=0
224 J1.J5=0.8990962
225 J1.J6=-3.16062
245 MVS J1
250 IF M7=1 THEN HOPEN 1
260 IF M7=0 THEN HCLOSE 1
261 P2=P_CURR
262 PRINT #1, "P=", J1.J1,J1.J2,J1.J3,J1.J4,J1.J5,J1.J6,P2.X,P2.Y,P2.Z
265 J1=J_CURR
275 P1=P_CURR
280 GOTO 40

300 P1.X=M1
310 P1.Y=M2
320 P1.Z=M3
330 MVS P1
331 SPD M_NSPD
340 IF M7=1 THEN HOPEN 1
350 IF M7=0 THEN HCLOSE 1
360 J2=J_CURR
370 PRINT #1, "P=",J2.J1,J2.J2,J2.J3,J2.J4,J2.J5,J2.J6,P1.X,P1.Y,P1.Z
380 GOTO 40

400 SORA=M1
410 SPD SORA
430 GOTO 40

500 CLOSE #1
510 END
```

APPENDIX C

PERFORMANCE EVALUATION PROGRAM

```
load RobotCoordinates.dat
load JoystickCoordinates.dat
[M,N]=size(JoystickCoordinates);
JoystickCoordinates(:,10)=JoystickCoordinates(:,7).*JoystickCoordinates(:,7);
JoystickCoordinates(:,11)=JoystickCoordinates(:,8).*JoystickCoordinates(:,8);
JoystickCoordinates(:,12)=JoystickCoordinates(:,9).*JoystickCoordinates(:,9);

[M,N]=size(JoystickCoordinates);
xerror(1,1)=0;
yerror(1,1)=0;
zerror(1,1)=0;
for i=1:M
  xerror(1:1)=xerror(1:1) + JoystickCoordinates(i:i,10);
  yerror(1:1)=yerror(1:1) + JoystickCoordinates(i:i,11);
  zerror(1:1)=zerror(1:1) + JoystickCoordinates(i:i,12);
end
xerror=xerror/M
yerror=yerror/M
zerror=zerror/M

figure(1)
plot(JoystickCoordinates(:,1),'r');
hold on
plot(JoystickCoordinates(:,2),'g');
plot(JoystickCoordinates(:,3),'b');
plot(JoystickCoordinates(:,4),'k-.');
plot(JoystickCoordinates(:,5),'k-.');
plot(JoystickCoordinates(:,6),'k-.');

figure(2)
plot(JoystickCoordinates(:,7),'r');
hold on
plot(JoystickCoordinates(:,8),'g');
plot(JoystickCoordinates(:,9),'b');

figure(3)
plot(JoystickCoordinates(:,10),'r');
hold on
plot(JoystickCoordinates(:,11),'g');
plot(JoystickCoordinates(:,12),'b');
```

APPENDIX D

3D STEREO IMAGE GENERATION PROGRAM

```
Q11=imread('C:\Documents and Settings\yadibelli\Desktop\matlab3D\11.bmp');
Q12=imread('C:\Documents and Settings\yadibelli\Desktop\matlab3D\12.bmp');

Q11=double(Q11(:,:,3));
Q12=double(Q12(:,:,3));
Q1=zeros(M,N);
[M,N]=size(Q11)
for i=1:2:M
   for j=1:N
      Q1(i,j)=Q11(i,j);
      Q1(i+1,j)=Q12(i+1,j);
   end
end
Q1=uint8(Q1);
imshow(Q1)

imwrite(Q1,'C:\Documents and Settings\yadibelli\Desktop\matlab3D\3d1.bmp');
```