IMPROVING PERFORMANCE OF A REMOTE ROBOTIC TELEOPERATION OVER THE INTERNET

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET SELÇUK ARSLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2005

ABSTRACT

IMPROVING PERFORMANCE OF A REMOTE ROBOTIC TELEOPERATION OVER THE INTERNET

ARSLAN, Mehmet Selçuk
M. Sc., Department of Mechanical Engineering
Supervisor: Asst. Prof. Dr. E. İlhan Konukseven

August 2005, 180 pages

In this thesis study, it is aimed to improve the performance of an Internet-based teleoperation system enabling the remote operation of a 6 DOF industrial robot. In order to improve the safety and efficiency of the teleoperation, stability and synchronization (hand-eye coordination) are considered.

The selected communication medium between the human operator and remote robot is the Internet. The variable time delays and nondeterministic characteristics of the Internet may lead to instability of the teleoperation system. Considering the disturbing effects of the Internet onto the transmission, an event-based control approach is used in order to improve the stability of the teleoperation system. Besides, a visual feedback system is developed and a force-feedback mouse is designed in order to improve synchronization between the human operator and robot during the command generation according to the feedback obtained from the control system. A client-server software application is developed to interface the human operator with remote environment.

It is observed that, using the event-based control approach in the operation makes the teleoperation stable and improves the synchronization ability. Implementation of visual feedback and force-feedback mouse to the teleoperation system improves the human operator's ability to perform remote operation.

Keywords: Teleoperation, Remote Control, Event-Based Control, Internet Time Delay, Human-Machine Interaction.

ÖZ

INTERNET ÜZERİNDEN UZAK BİR ROBOTİK TELEOPERASYONUN PERFORMANSININ GELİŞTİRİLMESİ

ARSLAN, Mehmet Selçuk
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez Yöneticisi: Y. Doç. Dr. E. İlhan Konukseven

Ağustos 2005, 180 sayfa

Bu tez çalışmasında, 6 serbestlik dereceli endüstriyel bir robotun uzaktan kumanda edilmesine imkan veren Internet tabanlı bir teleoperasyon sisteminin iyileştirilmesi hedeflenmiştir. Teleoperasyonun güvenliğini ve verimliliğini artırmak için kararlılık ve senkronizasyon (el-göz koordinasyonu) dikkate alınmıştır.

Operatör ve uzaktaki robot arasındaki iletişim ortamı olarak Internet seçilmiştir. Değişken zaman gecikmeleri ve Internet'in belirlenemeyen karakteristikleri teleoperasyon sisteminde kararsızlığa yol açabilmektedir. Internet'in iletim üzerindeki bozucu etkileri göz önüne alınarak, teleoperasyon sisteminin kararlılığını iyileştirmek için olay tabanlı kontrol yaklaşımı kullanılmıştır. Buna ek olarak, kontrol sisteminden alınan geri beslemeye göre, komut üretimi esnasında operatör ve robot arasındaki senkronizasyonu iyileştirmek için bir görsel geri besleme sistemi geliştirilmiş ve bir kuvvet geri beslemeli fare tasarlanmıştır. Operatör ile uzak ortam arasında bir ara birim oluşturmak için bir istemci-sunucu yazılım uygulaması geliştirilmiştir.

Olay tabanlı kontrol yaklaşımının kullanılmasının operasyonu kararlı hale getirdiği ve senkronizasyonu iyileştirdiği gözlemlenmiştir. Görsel geri beslemenin ve kuvvet geri beslemeli farenin teleoperasyon sistemine uygulanması, operatörün uzak operasyonu yürütebilme yeteneğini iyileştirmiştir.

Anahtar Kelimeler: Teleoperasyon, Uzaktan Kontrol, Olay Tabanlı Kontrol, Internet Zaman Gecikmesi, İnsan-Makine Etkileşimi.

To My Family

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. Prof. Dr. E. İlhan Konukseven for providing me this research opportunity, guiding me throughout the study, and his patience. I am also indebted him for his precious evaluations and feedbacks in writing this thesis.

I would like to express my gratitude to my father for causing me to pursue an M. Sc. in Mechanical Engineering, his methodological approaches and support in every stage of this study.

Special thanks to Melikşah Kandilci for his permanent support, brotherhood, and ingenuity which are always on my side as in this thesis study. I would like to thank him for his boundless helps in electronics field.

I owe thanks to Dr. Coşkun Kuş for his helps in statistics. Also I would like to thank to Emre Sezginer from METU CC Network Group for his contributions, my friend Sedat Doğru for networking helps, and Ed Gonzalez and Adam Smith for coding helps. METU BILTIR - CAD/CAM and Robotics Center, the staffs of the center and especially technician Halit Şahin are gratefully acknowledged.

I owe many thanks to my dear friends M. Mesut Tahtacı, Hakan Şahin and M. Rifat Sönmez for their support throughout this study and presence. Finally many thanks to my family, especially to my sister Ayşe Rûyem Arslan, for their invaluable and endless support, and patience.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background and Motivations

The importance of performing unmanned, large scale and challenging operations has been increasing. The human operator's expertise and intelligence are extended to distant applications through remote operations. Necessity of the presence of the human being in the application environment, restrictions, efficiency, and operation cost are the main determining factors in developing remote handling systems. The remote manipulation is not only adopted as indispensable for some applications like space teleoperations but also adopted as a performance increasing operation such as telesurgery. New branches such as teleoperation and telerobotics had been brought in robotics due to the advances in remote operation. Giving a brief discussion on the difference between teleoperation and telerobotics might be useful to describe the scope of this study.

Teleoperator is a machine that enables the human operator to extend the moving and sensing capabilities and manipulate objects physically at a distance. Teleoperators were developed in the late 1940s to operate at a distance. Advances in atomic physics created a need for the implementation of experiments remotely at the laboratory scale. Dealing with the hazardous environments which contain nuclear radiation, high temperatures, chemicals, etc. brought about the development of remote handling systems. After the development of the first master-slave mechanical manipulator, Figure 1.1, teleoperation activities in nuclear domain were started to apply to new application areas. Hereafter, teleoperation applications were spread out to medical, space, and underwater fields. Table 1.1 includes some branches of remote operation that took place in many areas since 1940s.

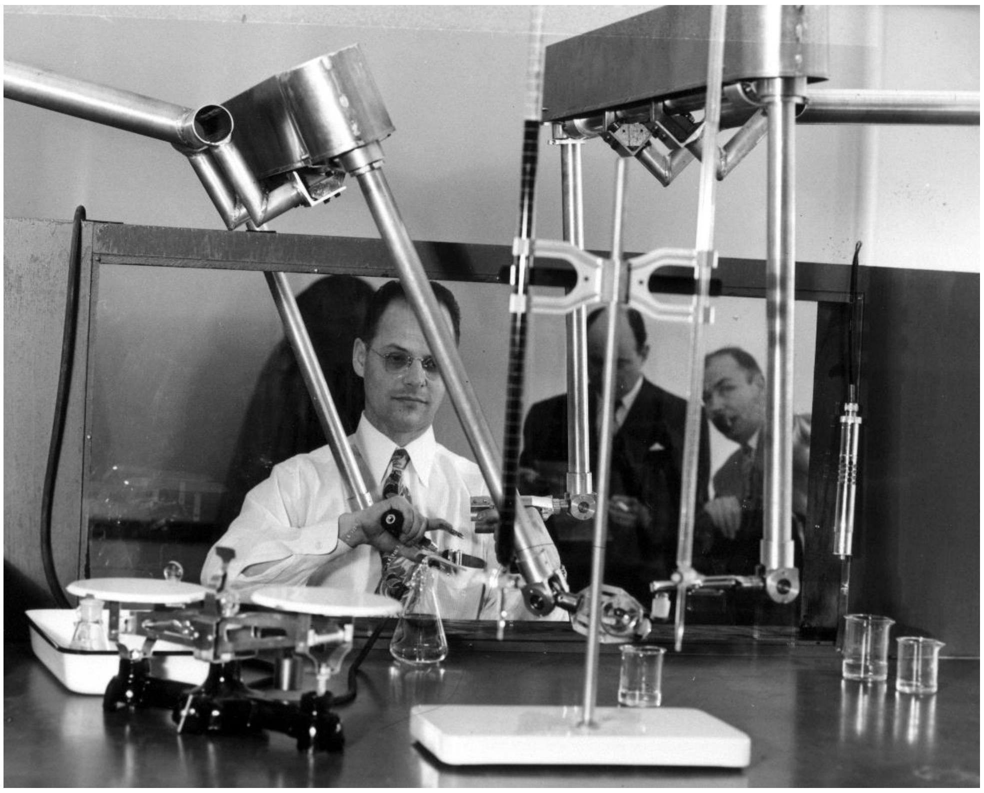


Figure 1.1 One of the First Small Scale Mechanical Link Teleoperator

Table 1.1 Areas of Application in Telerobotics and Teleoperation

Undersea	Research, rescue, ship salvage, repair and maintenance of undersea lab.
Nuclear	Hot-cell operations; inspection, rescue, cleanup and decontaminations in emergency situations
Chemical & Biological	Handling toxic materials, propellants, explosives etc.
Metal processing, handling and fabrication	In forging plants and for handling large, hot metal pieces
Electronics	In super-clean rooms and in toxic atmospheres
Public services	Fire-fighting, rescue and cleanup in hazardous environments
Entertainment	Telegarden, Teleactor, Puma Paint Project [10], controlling remote telescopes and mobile vehicles, etc.
Education & Training	Manipulator trainings, experiments in remote laboratories, museum guide autonomous robots etc.
Medical	Operations, first-aid, surgery lectures, home healthcare, rehabilitation
Aerospace	Spacecraft assembly and maintenance and for the exploration of satellites and planets

A *telerobot* is a system that beneficially combines human interaction and automation in a single robot system. In the early 1970s, as digital electronics became more cost effective, interest began to emerge in the integration of automation with teleoperations as a scheme for effectively increasing the work efficiency of remote operations. This integration of automation with teleoperation is the foundation of telerobotics. Unlike manufacturing automation, remote operations in hazardous and unstructured work task environments necessitate human-in-the-loop control, or teleoperations, as a backstop to ensure safe operations. A telerobotic system provides a continuum of remote capabilities from fully manual operations through fully autonomous task execution where the mixture of operations is chosen to improve overall system performance [1]. A recent example of an industrial telerobotic system is shown in Figure 1.2.

The prefix "Tele" is derived from the Ancient Greek and means *distant, distance*. While the distance is a few meters in laboratories, some applications cause the extension of the distance as far as millions of kilometers as in aerospace applications. The increase in distance between the operator and remote device leads to a challenging problem, *time delay*. Time delay is the elapsed time until a signal arrives at the destination and is mainly related with the distance between the operator and the remote device. Most applications have private communication links, e.g. in underwater applications, sound waves or cable lines are used to transmit the data. Communication medium is also another important factor affecting the delay. For instance, in underwater applications, sound transmission is limited around 1700 m/s in water; in aerospace applications, signal transmission is limited to the speed of the light (radio transmission). The Internet is among the most popular communication media and its packet routing mechanism plays an important role in time delay since the packets are handled and routed at each node (router) on the path.

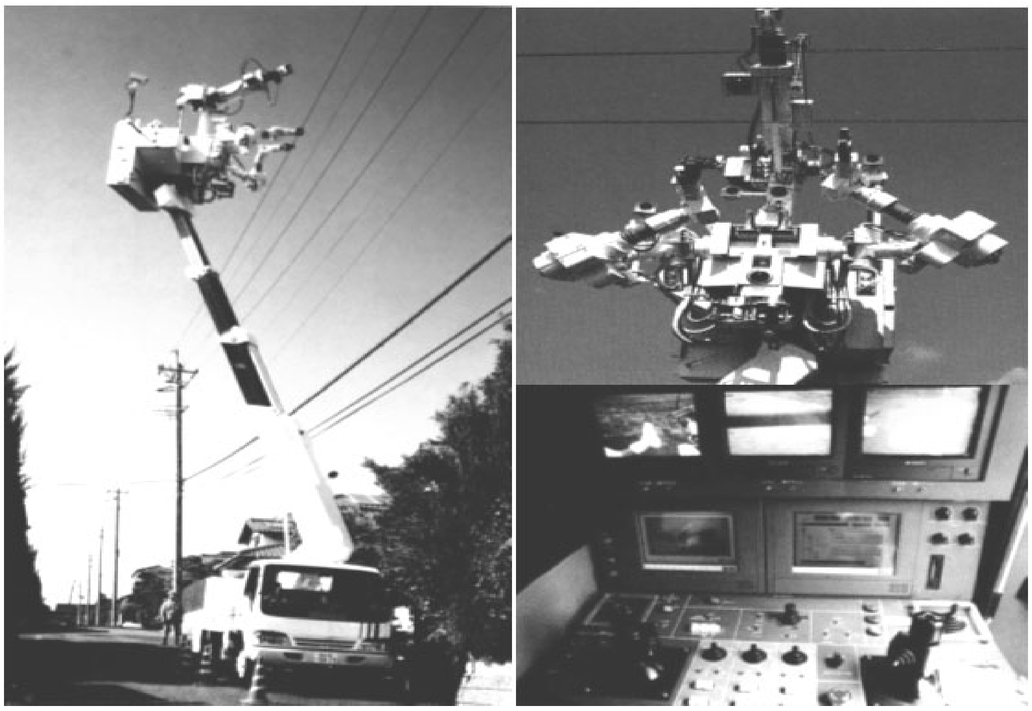


Figure 1.2 A telerobotic system for repairing high-voltage power lines.

The Internet is a network that interconnects millions of computers throughout the world and provides the exchange of information between computers at high rates. By January 2005, there are over 300 millions hosts on the Internet and this number grows exponentially according to Internet Systems Consortium (ISC) Internet Domain Survey [2]. Growing availability of the Internet has brought in a cheap and open public communication medium for remote operations. However, as Khamis et al. [3] described, the Internet brings forth many challenges in real-world applications. "These problems include restricted bandwidth, random time delay and data loss, which influence system performance. Internet performance as a communication medium has nondeterministic characteristic, which depends mainly on network load" [3]. Since the Internet was not designed and implemented for remote operations, these problems must be handled for a successful remote operation.

Stability and synchronization are important criteria affecting the performance and safety of the Internet-based remote operations. Xi and Tarn [4] state that the synchronization of the human or autonomous controller with the remote robot local controller will affect the stability as well as the dynamic performance of the system. This type of synchronization will be directly related to the properties of the Internet communications. While the control commands and the resulting feedbacks flow in the Internet, the congestion in any network brings on the buffering effect. Time delay causes the loss of synchronization between the human action and the corresponding reaction of the remote device. Therefore, desynchronization can bring forth the instability as long as the output is out of control. Feedback plays an important role in synchronization since the human operator needs feedback to make decisions for the next commands. Teleoperators are normally operated with man-in-the-loop and with supermedia. *Supermedia* is used to describe the collection of all feedback streams in teleoperations, such as haptic, video, audio, temperature and others [5]. Especially visual feedback provides the most important information since vision is the most powerful sense of the human operator. Although other types of feedback are not as effective as visual feedback, they are either utilized as supplementary feedback like force restricting operator motions or required as essential feedback like temperature changes in organs.

1.2 Survey of Literature

The telerobot at the University of Western Australia is one of the first web-based telerobots. In this telerobotic system; the remote robot operates in a largely autonomous mode and video is fed back to the operator. Thus, the instability problem is eliminated to some extent, implementing the supervisory control. This telerobotic system has been improved and is still online [7, 8]. Puma Paint Project [9] is another exciting project, which is available online on the Internet. In this project, people can make paintings through a web-based virtual canvas. The commands are sent to the PUMA 760 robot equipped with painting equipments.

Likewise many projects [10] have been implemented on Internet-based control and research projects have been conducted. After initial implementations, researches have mainly focused on coping with the disturbing effects of communication time delay. To deal with this challenge, new control strategies are introduced and many interfaces are developed taking time delay into account. So far, many studies seeking to overcome the Internet time delay problem have been performed. Some of these studies are presented below;

The JBIT project [11] is a force-reflecting closed loop remote control scheme over the Internet. Implementing an online bandwidth estimator and data flow manager end-to-end characteristics of the Internet is monitored. Both video and control data streams are adapted in order to guarantee a safe and stable teleoperation.

Park and Park [12] introduced Variable Holding Time technique to improve the teleoperation performance. In this study event-based control method is used for Internet-based teleoperation and holding time technique is used with this control method to improve its efficiency. The events arriving at the controller are held for holding time to take fast and smooth response. Holding time is determined by evaluating the three data, which are operator's intention, similarity of commands, and environmental information, by means of fuzzy logic.

Liu et al. [13] presented a novel transport protocol, Trinomial Protocol, which is teleoperation-oriented. This protocol was roughly combination of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Some mechanisms of these protocols are improved and fused in this new protocol.

Khamis et al. [3] used supervisory control strategy to interact a remote mobile robot through the Internet. The local autonomy of the robot provides a communication at an abstract level using high level commands. Limiting the remote interaction to high level commands helps to decrease the bandwidth needs. Increasing robot's autonomy can also help to decrease the teleoperated task sensitivity to delay.

Elhajj et al. [5] introduces the synchronization of all the feedback streams in teleoperation. This synchronization is based on event-based control perspective and defined as event-synchronization. "An event-synchronized system is one in which all the signals in the system are always referencing events which are within a certain tolerance range of each other" [5]. For instance, the video frames are event-synchronized with the control signal.

Boyaci [14] developed the first test-bed for Internet based teleoperation in Turkey by utilizing the ABB IRB2000 industrial robot and implemented a client-server software application for control of the robot through the Internet. The task for test was to make drawings on a paper placed in the robot's workspace through the Internet. A fuzzy logic control model was integrated to minimize the effects of random communication delays.

Mirfakhrai and Payandeh [15] developed an autoregressive model for the Internet time delays which are measured for a week. Two different connections were probed and time delay was partially predicted. Predicted time delay value was used in a telerobotic control system which is developed using the wave variables. Wave variables are based on the passivity theory. Passivity theory is a method to generalize the concept of power and energy in a dynamic system.

A different architecture for Internet-based teleoperation was presented [16]. In that study, the time delay model, which takes into account the randomness, was developed. Based on this delay model a state space model for a telerobotic system was presented. This state space model is responsible for control, observation, and prediction.

Kosuge et al. [17] introduced the Virtual Time Delay method to deal with the variable time delay. The Virtual Time Delay equals to the maximum delay obtained from observation of the network. This method provides that time delay is constant and equals to the Virtual Time Delay. The instability problem was handled using scattering transformation method which requires the delay to be constant.

You et al. [18] developed an experimental environment, suitable for remote robotic teleoperations. This scheme allows the users to perform research experiments involving supervisory robot control, multi-sensor data fusion, communication time delay, predictive systems modeling, and local autonomous control. This work describes one of the well-equipped teleoperation systems ever presented. The system comprises the following sub-systems. 15 DOF Hand/Arm integrated sub-system which is responsible for performing tasks. Predictive simulation and overlay sub-system which is responsible for forecasting the forthcoming movement through the information obtained from simulation model. A graphical simulator which is at the operator’s site allows the simulation of the commanded tasks. The simulation of the remote device is presented to the operator directly without waiting the acknowledgement of the last task. This sub-system also allows overlapping the image coming from the predictive simulation and real image coming from remote site. The other sub-systems are: Autonomous multi-sensor integration sub-system, Virtual Reality interface, and visual/audio interaction sub-system.

Video streams occasionally saturate the bandwidth of networks and increase time delay due to its larger sizes. To overcome this problem, instead of video feedback, graphical models of the remote devices are used in Internet-based teleoperation. Park et al. [19] presented the display estimation technique. The command sent to the remote site is also used in simulation model. This simulation is displayed to the operator after a delay which is estimated based on statistical data. These displays are not predictive but they are estimative since they are displayed after a delay. It is stated that feeding to the operator estimative displays provides concurrency between remote device and operator.

Elhajj et al. [20] discussed the effect of random time delay on Internet-based teleoperation systems that include supermedia. Supermedia is the collection of multimedia, haptic and any other sensory information. The event-based control is implemented to decrease the effect of time delay on the stability, and synchronization of a teleoperation system including supermedia.

Han et al. [21] proposed a direct Internet control architecture for an Internet based mobile robot. Operator controls the robot using a local simulator without knowing the current status of the robot. Environment and the robot are simulated in this simulator. The Internet control architecture includes a user interface enabling the user to control a remote service robot, a virtual environment which has the information of the real environment, a command filter to recover the information loss of control commands, a path generator to restore the moving path of the virtual robot, and a path-following controller to guarantee that real robot follows the generated path. This mobile robot has obstacle avoidance function and obstacle information is sent to the simulator to create the obstacles in the simulator environment already known thanks to virtual environment.

Xi and Tarn [4] introduces a new method for action synchronization and control of telerobotic systems. Traditionally, a robotic system's tasks and actions are synchronized and controlled according to the action reference, time. "The random communication delay has resulted in the loss of synchronization of time based action references among different entities of the robotic system. This can easily cause the instability" [4]. This problem is solved introducing non-time based action reference parameter. This parameter is relevant to the real-time sensor measurement and the task.

1.3 Scope of the Thesis

A teleoperation system, Figure 1.3, can be represented by five subsystems: the human operator, the human interfaces (e.g. master devices, visual display etc.), the communication media, the slave (e.g. actuators, sensors, etc.), and the environment [6]. The basic components constitute the attained teleoperation system is described in the following paragraphs in accordance with this general scheme.

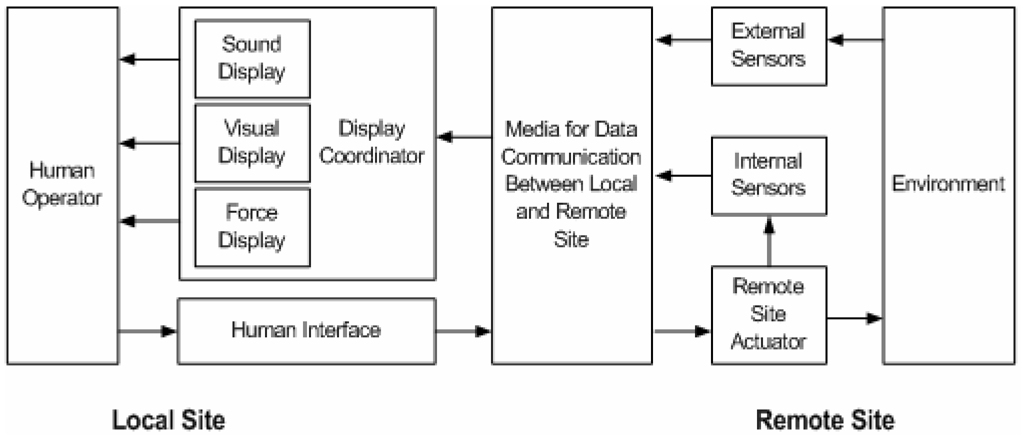


Figure 1.3 General Teleoperation System

In this study, the remote device is a 6 DOF industrial robot (ABB IRB2000) equipped with a pen holder apparatus and a pen. The task is to draw figures on a paper or to track the lines, which are drawn previously, using the robot which is controlled over the Internet. The robot controller unit can be serially linked to a computer in order to communicate. The robot is controlled sending the Cartesian coordinates, orientation and velocity information from the server computer to the controller unit. The control commands are sent to the server side from the client side. The human operator is at the client side and is responsible for generating the commands. The commands are generated via a mouse with the assistance of graphical joystick or drawing board interfaces.

The human operator is at the core of the control system. Regarding the teleoperation, functions of a human operator can be identified as task-setter, error computer, planner, and decision maker. Thanks to sensory information, the human operator monitors the performance of the teleoperation while generating commands. Sensory information which describes the most current situation of the teleoperator is fed to the human operator. *Sight* and *touch* are the two most important and mostly used human senses which are highly convenient for teleoperation. The teleoperation system discussed in this thesis study includes both visual and force feedback in order to provide reliable and efficient operation.

Video system is used in order to provide a visual feedback from the robot environment. In addition, virtual reality simulation feedback is provided to the operator as a supplementary feedback. The 3D graphical model of the robot simulates the executed commands. On the other hand, the force feedback is provided the operator via a designed force-feedback mouse. The human operator is stimulated kinesthetically by means of applied force while generating commands via force-feedback mouse.

The communication medium between the robot site and the human operator site is the Internet. Although the Internet is a cheap and readily accessible communication medium, its non-deterministic characteristics make the teleoperation prone to instability. The random time delay, network buffering effects and disconnections experienced over the Internet are the main difficulties encountered in the Internet based remote control.

In order to implement the teleoperation, human-machine interface software is developed utilizing a high-level programming language. This interface allows the human operator to communicate with the remote robot. The information is exchanged between the remote site and the human operator. This user interface consists of two components: server side (robot side) and client side (human operator side). Server part of the interface communicates with robot, cameras and client part. Commands sent from client are processed, transmitted to the robot or cameras, and acknowledged commands are sent to the client. Server part also includes a path generator module to recover the missing commands in the command sequence. The client part includes the modules which are: communication with server, camera displays, graphical joystick interface, drawing board interface, simulation display of virtual reality model of the robot, communication with force feedback mouse, and round-trip time (RTT) measurement. The operator directly interacts with the designed graphical user interfaces in this software. These graphical user interfaces enable the human operator to generate commands or to view the video and graphical displays.

The human-machine software is so designed that it handles the information traffic between the client and server sides. The video feedback and the control signals, which are commands and acknowledgment messages of these commands, constitute the information traffic. The Internet time delays on the communication link affect this traffic. This may result with instability. To overcome this problem, the control strategies regarding the Internet time delay are developed. The stability concept considered in this thesis study is not same as the stability concept used in automatic control systems. In this thesis study, the stability describes whether the teleoperation system will be able to follow the input command. If the output is out of the control then it is accepted that the teleoperation system is unstable. The control sub-system of the teleoperation system plays an important role in the performance and stability of the system. To deal with the random time delay, non-time based control strategy is implemented. The action reference is not directly related with time, but the task is the event based action reference parameter. So, the communication delay has little effect on the operation, especially the stability. Each command sent by the operator is determined as an event. When the command is executed, the positive acknowledgement message is sent to the operator. But this is not sufficient alone for continuing to generate and send commands. The video feedback is checked to determine whether it is captured on the execution time or not. The command generation is allowed if the video and acknowledgement message belong to the same event. This implies the synchronization of all feedbacks.

1.4 Objective of the Study

The remote interaction is a special type of human robot interaction. The operator and the robot are far away from each other but linked through a communication medium. In the Internet-based teleoperation, the distance between operator and the remote environment necessitates the extension of human intelligence to the teleoperator environment and transfer of the sensory information to the operator. The established communication system is responsible for transferring the information and plays an important role in the remote control since its reliability and performance have a significant effect on the performance of teleoperation. Besides that, the feedback system increases the efficiency and quality of the remote interaction. According to the application type, to feed the sufficient sensory or other information back to the operator provides the operator takes place in the operation effectively.

Providing a control system coping with the disturbing effects of the Internet time delay also increases the performance of the operation. The efficient cooperation of the feedback system, control system and the human operator is significant and desired. Adapting the human operator to the remote interaction requires the well-designed human-machine interfaces. The match between the operator's capabilities and the motion capabilities of the teleoperator indicates a good human robot interaction. While the operator controls the teleoperator with his or her hands, he or she most probably views the visual feedback and the provided displays. Although the Internet time delay has disturbing effects over the remote control, to some extent, synchronizing the motion of teleoperator with the human operator's action improves the hand-eye coordination of the human operator.

The purpose of this thesis study is to implement a reliable and high-performance teleoperation by providing the human operator with visual and force feedback. In this context, it is tried to cope with the disturbing effects of the Internet time delay onto stability and synchronization.

1.5 Outline of the Thesis

In this thesis, the main components of the developed teleoperation system are introduced and how they are constituted is described. The chapters are organized in order that each chapter includes preliminary information for the next chapters. The outline of this thesis is as follows:

Chapter 2 describes the teleoperation hardware architecture. The elements constituting the teleoperation system are discussed.

Chapter 3 is dealt with the communication link of the teleoperation system. This link is the Internet and its characteristics are described. How and why the time delay occurs is explained in this chapter. Networking issue which is related to the developed communication software is also described.

Chapter 4 describes the feedback mechanism of the teleoperation system. Hardware and software parts of the visual and force feedback mechanism are discussed in detail.

Chapter 5 explains the control architecture of the teleoperation system which is developed for improving the stability and synchronization.

Chapter 6 describes the developed software covering all parts of the software except for those explained in the previous chapters.

Chapter 7 is devoted to conclusions, discussions and future works.

CHAPTER 2

TELEOPERATION HARDWARE ARCHITECTURE

2.1 Overview

This chapter describes the main architecture of the designed teleoperation system. The designed teleoperation system architecture consists of the following hardware and software components: the test bed, the communication medium, the feedback system, the control system and the human-machine interface. Firstly the overall system is introduced and after that its hardware components are described in this chapter.

2.2 Main Architecture of the Teleoperation System

In general, a teleoperation system consists of two components, the master (local system) and the slave (remote system). The human operator specifies the commands by means of a master. The master system is simply an interface which might be only a mechanical device or an integration of a mechanical device and a software program. The tasks are forwarded to the slave and the information relevant to the execution of the task is provided to the operator.

The slave system is also an interface through which directly interacts with the environment. The slave device can be more complex than master device since the interaction with the environment may require higher degrees of freedom, more sensory capabilities, dexterity, higher power rates, etc. The operation may also necessitate the autonomy.

In this study, the master simply consists of the force-feedback mouse and the graphical user interface software. Position coordinates are generated with the cooperation of a force-feedback mouse and a graphical joystick interface or a drawing board interface module. The commands generated by the operator are assigned to the slave with the cooperation of the client and server side interfaces. Whether the commands are completed successfully or not the master forwards the positive or negative acknowledgement messages and the related video feedback to the operator.

The slave consists of an industrial robot, video cameras, a video server, and user interface software. Commands taken from the master are forwarded to the robot and acknowledgement messages received from the robot related to these commands are processed by the slave system and sent to the master for evaluation. The slave side interface also includes a path generation algorithm to generate commands in case the command packets arrive disorderly in a timely manner.

The main architecture is given schematically in Figure 2.1. This architecture is based on two main parts, client side and server side. The communication link between them is accomplished through the Internet. The communication protocol group used in this link is Transmission Control Protocol / Internet Protocol (TCP/IP).

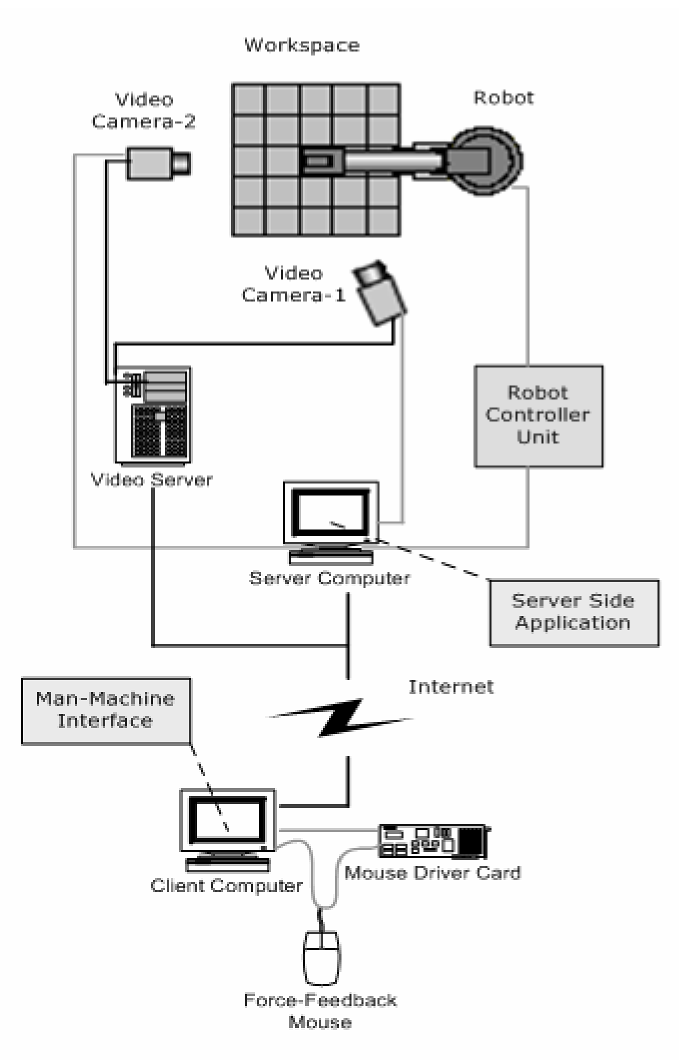


Figure 2.1 Main Architecture of the Developed Teleoperation System

The end system, which is the computing device connected to the Internet, on the server side of this link is a desktop computer which runs the server side interface application. The interface application developed for the server side communicates with the client side over the Internet, provides computing facilities related to the operation, communicates with the robot’s controller unit, and cameras via serial port. There are two video cameras in the workspace. One of them is placed near the robot’s gripper in order to visualize the detailed movements; the other provides a panoramic view. These cameras are connected to a video server at the server side. The video server is not directly connected to the server computer but it is connected to the Internet. It transmits the video signals through the Internet by using TCP/IP.

The computer which is at the client side may be any computer using Windows 98 or higher operating system and has a connection to the Internet. The developed application for the client side provides communication with the server side through the Internet. It allows the operator to generate 2D position coordinates, measures the round-trip time delay, displays the video and/or graphical feedback, and controls the force-feedback mouse. Force-feedback mouse is connected to the client computer through serial port. It generates force according to information sent from client application. This information is produced by the control algorithm to restrict or to allow the command generation.

In the following sections, the components constituting the hardware architecture of the teleoperation system are described.

2.3 Test Bed

The statement of test bed is used to designate the setup enabling remote interaction with the environment. The test bed is implemented to test the proposed teleoperation system over the Internet, and it is at the METU BILTIR-CAD/CAM and Robotics Center at the METU campus. The task performed by using this test bed is to follow the paths, which are drawn on a paper previously, using the industrial robot which is controlled over the Internet. The auxiliary components are used in the test bed in order to perform this task. The remote workspace mainly consists of ABB IRB2000 industrial robot, its controller unit, two cameras, a video server and a server computer. The auxiliary components are a pneumatic pinch type gripper, a pen holder, camera attachments, a compressor, and a drawing table. This test bed is shown in Figure 2.2.

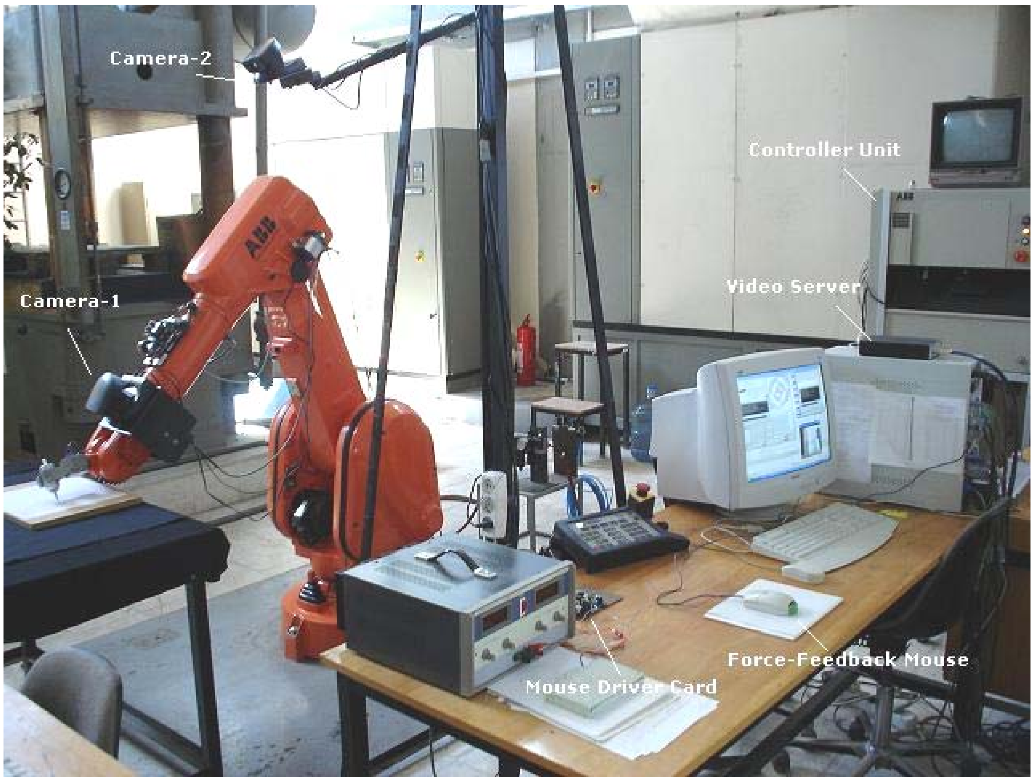


Figure 2.2 View from the Test Bed Environment

2.3.1 Robot

The controlled remote device is a robot and it performs the operator's commanded actions. The robot is ABB IRB2000 industrial robot and primarily designed for arc welding, sealing and gluing operations. Also it can be used for applications such as assembly, water jet cutting, laser cutting, and material handling. The following sections describe the mechanical unit, the controller unit, and the communication protocols and how the robot is maneuvered.

2.3.1.1 Mechanical Unit

The robot has six degree of freedom (DOF) and all of the joints are revolute. The movement structure and the main parts of the robot can be seen in Figure 2.3. All of the actuators are servo-controlled, brushless AC motors, specially adapted for each axis. Although the robot may handle a load of up to 10 kg within a wide range accurately. The permitted load depends on distance from wrist. Its incremental movement is approximately 0.13mm and repeatability value is less than 0.1mm in both directions. The measurement system consists of a resolver on each motor shaft and a measurement board mounted on the robot. The resolver is used for gathering speed and position data.

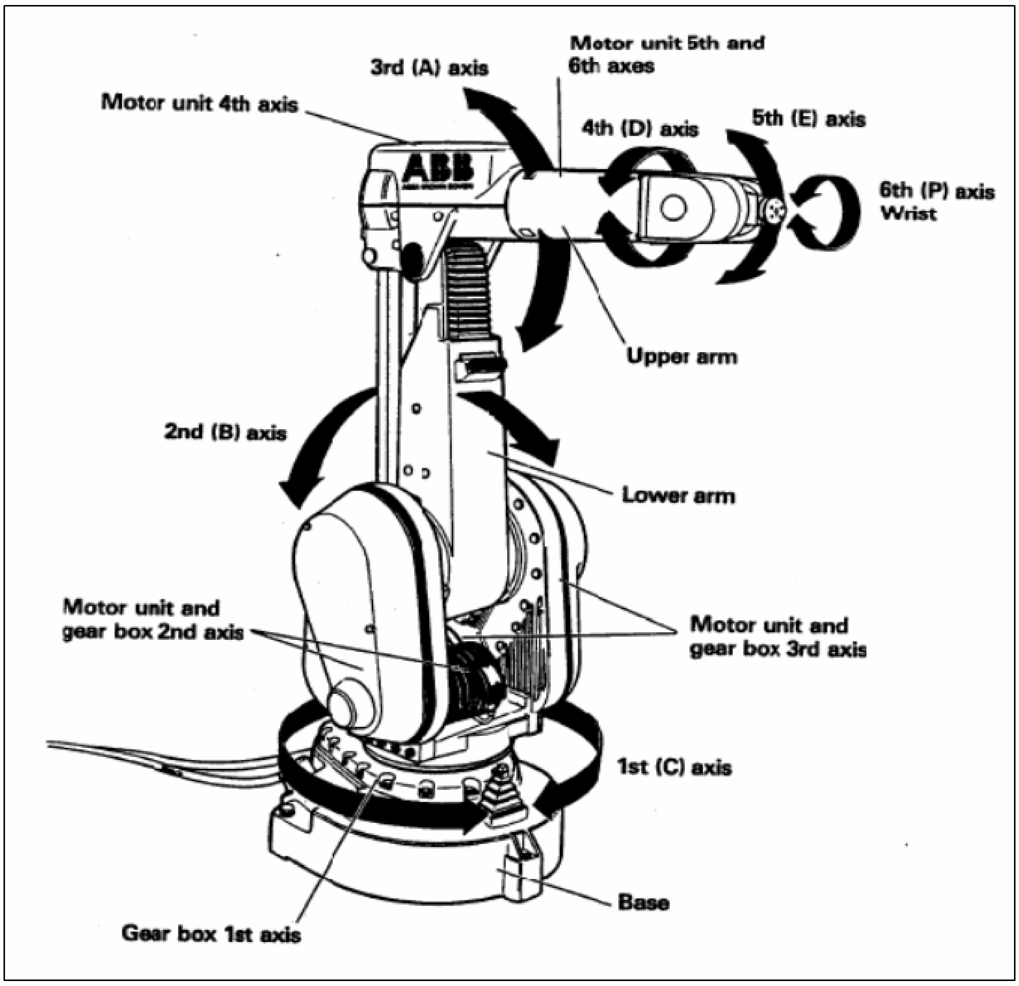


Figure 2.3 Movement Structure and the Main Robot Parts [22]

Movement pattern of each joint can be described as follows:

Axis 1 (C): Turning of the complete mechanical unit.
Axis 2 (B): Forward and reverse movement of the lower arm.
Axis 3 (A): Up and down movement of the upper arm.
Axis 4 (D): Turning of the complete wrist unit.
Axis 5 (E): Bending of the wrist around the wrist center.
Axis 6 (P): Turning of the mounting flange.

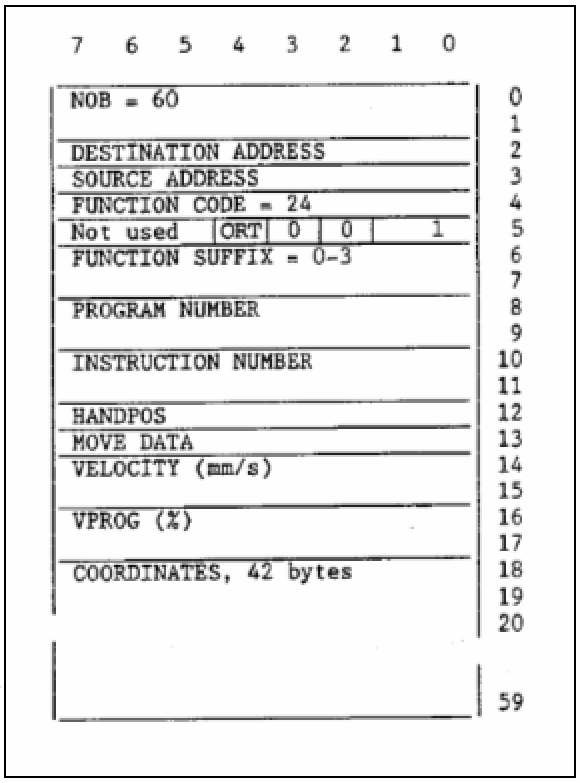
2.3.1.2 Controller Unit and Communication Protocols

ABB IRB2000 has S3 type controller unit [22], and this controller has the ability to perform a communication by using digital I/O, analogue I/O or computer link to a superior controller. Also it has capability to control up to 7 external axes. The computer link permits the connection of the robot to an external computer. Through this link it is possible to change operation modes, start and stop robot executing, read and write data to/from robot, and receive information about events and errors that occur in the robot system. The computer interface must be in agreement with V24, V28/RS232. The communication speed is 9600 baud (bits/second). The character size is 8 bits + even parity and one stop bit.

The computer link software consists of two protocols, ADLP 10 and ARAP. ADLP 10 (ABB Data Link Protocol) is the communication protocol. It is a line procedure for asynchronous communication between two stations in a hierarchic system. Transmission control characters are used in this protocol to define the nature of the information in the succeeding character string or to transmit instructions to another station with regard to the communication process. For instance, ACK is used to define the positive acknowledgement and it is sent by the robot. STX represents the start of text and is used to indicate the beginning of the text in a telegram (the series of characters transmitted between robot and computer). The hexadecimal value equivalents of control characters are used in transmission.

ARAP (ABB Robot Application Protocol) describes the exchange of messages and the message format. These messages are responsible for controlling the robot. A message can consists of one or more telegrams. A telegram is a series of characters transmitted between the robot and a computer, using the line procedure ADLP 10. A character is an ASCII code consisting of 8 bits + parity bit. A telegram consists of a header and a data field. All data within a telegram should be binary coded. The telegram's header contains information on the number of bytes the telegram has, the source address, the destination address, the function code specifying the function of the message, the telegram type and the function suffix. A telegram's data field consists of the transmitted data and the error code.

Figure 2.4 depicts an example telegram used to maneuver the robot from computer. The length of this telegram is 60 bytes. The first 8 bytes constitutes the header of the telegram and the rests are for data field [23]. All data constituting a telegram is combined and sent to the controller unit as a sequence of bytes. For detailed information on communication between robot and computer, it can be referred to [14, 24].



7 6 5 4 3 2 1 0	
NOB = 60	0 1
DESTINATION ADDRESS	2
SOURCE ADDRESS	3
FUNCTION CODE = 24	4
Not used | ORT | 0 | 0 | 1	5
FUNCTION SUFFIX = 0-3	6 7
PROGRAM NUMBER	8 9
INSTRUCTION NUMBER	10 11
HANDPOS	12
MOVE DATA	13
VELOCITY (mm/s)	14 15
VPROG (%)	16 17
COORDINATES, 42 bytes	18 19 20 59

Figure 2.4 Telegram for Maneuvering of the Robot from Computer [23]

2.3.1.3 Maneuvering the Robot

The robot is maneuvered by sending the velocity, position, and orientation values of the robot's tip point from connected computer to the controller unit. The velocity of the tip point is expressed in millimeters per second. Position and orientation values are specified with respect to the base coordinate frame. Coordinate frames of the ABB IRB2000 robot are shown in Figure 2.5. All frames are defined in Cartesian coordinate system. The position vector of the hand, *P*, points from the origin of the base coordinate frame, O_0, to the origin of the hand coordinate frame, O_5. The position vector is defined by X, Y and Z coordinates, and it describes the position of the robot's tip point.

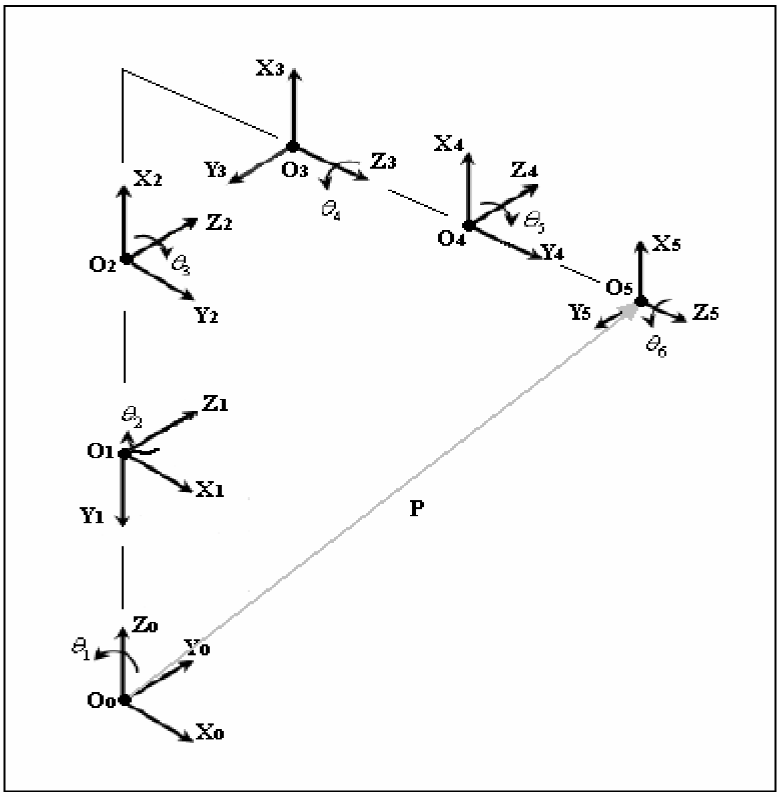


Figure 2.5 Coordinate Frames of the ABB IRB2000 Robot

In order to orient the robot's hand using roll, pitch and yaw angles, the orientation of the hand coordinate frame must be described with respect to the base coordinate frame. The rotation matrix, Euler angles or quaternions can be used to rotate the hand coordinate frame. Quaternions do not yield mathematical singularities, offer computationally easy operations and yield well-conditioned numerical solutions to orientation problems. Since quaternions offer a much better method of representing coordinates, they are used in robotics, computer graphics, chemistry, etc. The ABB IRB2000 robot also uses quaternions to specify the orientation of the wrist. The wrist orientation is expressed in quaternion coordinates. A quaternion is a four-element vector,

$$q = (q_1, q_2, q_3, q_4) \tag{2.1}$$

A quaternion vector represents a rotation by an angle (*θ*) about a unit vector (*nx, ny, nz*). A unit quaternion can be represented in the following format [25]:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)nx \\ \sin(\theta/2)ny \\ \sin(\theta/2)nz \end{bmatrix} \tag{2.2}$$

Elements of the quaternion vector are obtained from the rotation matrix with Eulerian (roll, pitch and yaw) angle representation.

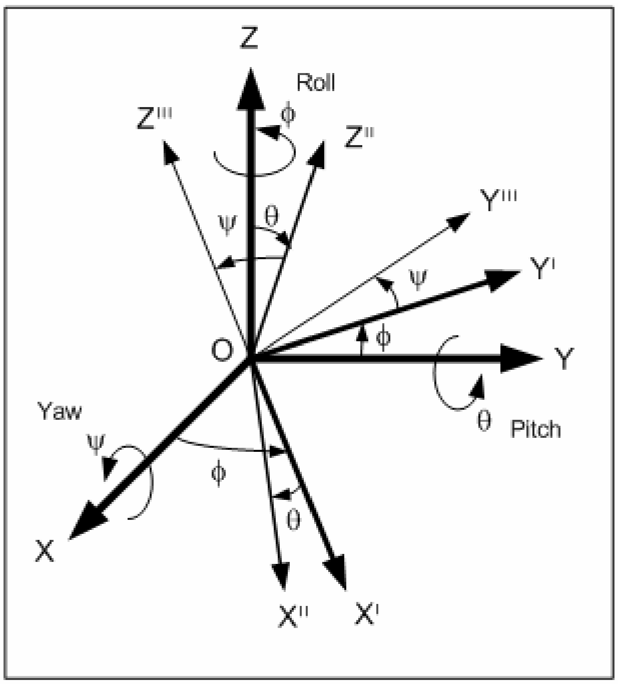


Figure 2.6 Euler ZYX (Roll, Pitch, Yaw) Sequence

The orientation of the robot's wrist is described by the Euler ZYX sequence. This sequence simply describes the rotations in terms of the roll, pitch and yaw angles. Euler ZYX sequence is shown in Figure 2.6. Firstly, the reference coordinate system is rotated by angle ϕ about the *OZ* axis, then it is rotated by θ angle about the OY^{I} (rotated *OY*) axis, and finally it is rotated by ψ angle about the OX'' (rotated OX') axis. The rotation matrix about the OX axis with ψ angle is

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \qquad (2.3)$$

Similarly, the rotation matrix about the OY axis with θ angle is

$$R_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \qquad (2.4)$$

Finally, the rotation matrix about the OZ axis with ϕ angle is

$$R_z = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.5)$$

Multiplication of the above matrices (2.3, 2.4 and 2.5) gives the resultant Eulerian rotation matrix, (2.6).

$$R_zR_yR_x = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$

$$= \begin{bmatrix} c\theta c\phi & s\psi s\theta c\phi - c\psi s\phi & c\psi s\theta c\phi + s\psi s\phi \\ c\theta s\phi & s\psi s\theta s\phi + c\psi c\phi & c\psi s\theta s\phi - s\psi c\phi \\ -s\theta & s\psi c\theta & c\psi c\theta \end{bmatrix} \qquad (2.6)$$

where,

c = cos;   s = sin

The rotation matrix can also be obtained from the elements of quaternion vector [26]:

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \quad (2.7)$$

Each element of the quaternion vector is obtained by using the equality of two rotation matrices, (2.6) and (2.7). The first element, q_0, is considered as a scalar and the other three as a vector in three-dimensional space.

In the developed software, Euler angles are used to obtain (2.6). Applying some calculations the equality of (2.6) and (2.7) gives the elements of quaternion vector. Thus, the orientation parameters (ϕ, θ, ψ) are converted to quaternion vector and the parameters (X, Y, Z, q_0, q_1, q_2, q_3) are passed to the robot by adding them to the maneuvering telegram.

2.3.2 Video Server

Video server is a network-attached server for video that is connected to a computer network like a LAN. A video server comprises one or more analog video inputs, image digitizer, image compressor and Web server with network or phone modem interface. Video servers digitize analog video sources and distribute images over a network. This feature enables to turn an analog camera into a network camera. A video server can deliver live video, automatically or on request, to a browser. For this thesis, Axis 2400+ video server is used. It is shown in Figure 2.7. This video server allows connecting up to four video cameras. Video outputs of the cameras are connected to the video server and the video server is connected to the Ethernet network directly. This video server has its own IP address, making it accessible from any authorized PC on the network. Real-time videos can also be viewed through a web page provided through this IP address. The detailed information about this component is given in Chapter 4.

Figure 2.7 Axis 2400 Video Server

2.3.3 Video Camera

Two video cameras are used in the proposed teleoperation system to provide the user visual feedback from robot environment. These cameras are Sony EVI-D30 pan/tilt CCD video cameras (Figure 2.8) and they are identical. EVI-D30 can be controlled from a computer through RS232. Pan, tilt, zoom and focus operations are performed using VISCA protocol. The detailed information about this component is given in Chapter 4.

Figure 2.8 Sony EVI-D30 CCD Video Camera

In teleoperation, it is important to provide the user-friendly interfaces in order for the remote interaction is comfortable. For this reason, viewpoint of the video camera and orientation of the hand are fixed in the way that the operation is performed effectively. It is tried to provide the viewpoint of a human who draws on a paper. Therefore, one of the video cameras is attached to the robot using the designed and built attachments, Appendix B. The video camera is set for pan, tilt and zoom values so that the camera points to the tip of the pen. The hand is oriented by passing the Euler angles in Euler XYZ sequence to the robot. The roll angle is 30 degrees and the other angles are 0 degree. Thus, orientation of the hand allows the operator to be able to see the tip point of the pen. On the other hand, in order to provide a panoramic view to the robot environment, Camera-2 is included to the test bed environment. The attained camera arrangements are shown in Figure 2.9 and Figure 2.10.

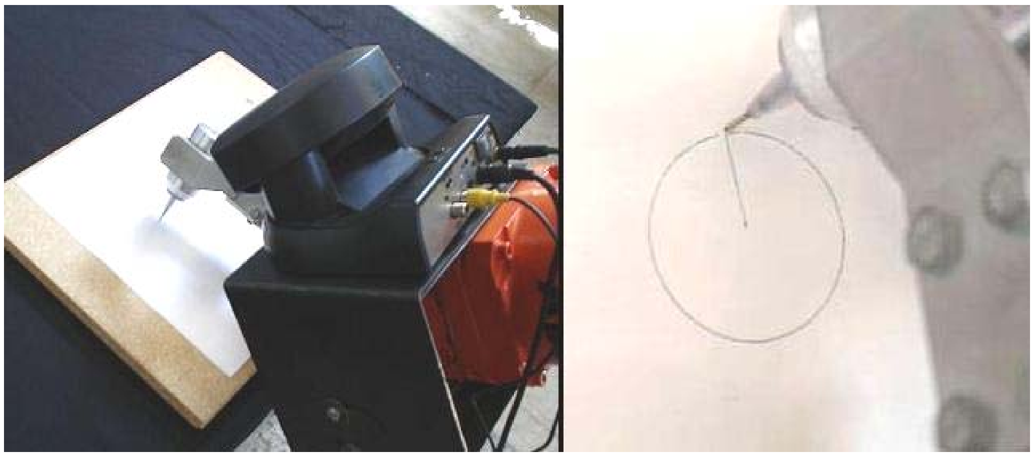


Figure 2.9 Viewpoint of Camera-1

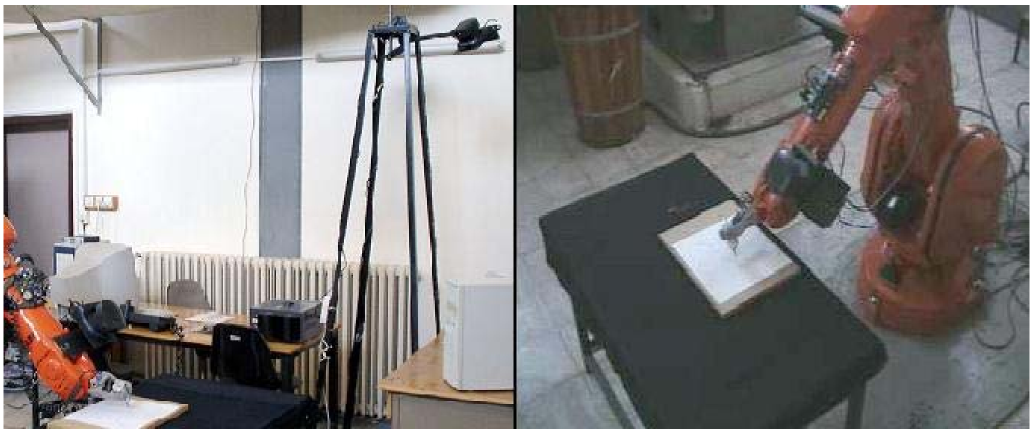


Figure 2.10 Viewpoint of Camera-2

2.3.4 Gripper and Pen Holder

In this study a pinch type pneumatic gripper and a pen holder apparatus are used in order to test the teleoperation system performance. Figure 2.11 depicts a drawing operation where pinch type pneumatic gripper is used to grip the pen holder apparatus. The pen holder apparatus is used to hold pen or pencils. A drawing paper is also placed on the table with the dimensions, H: 1.0 m, L: 1.2 m and W: 0.8 m in the workspace of the robot.

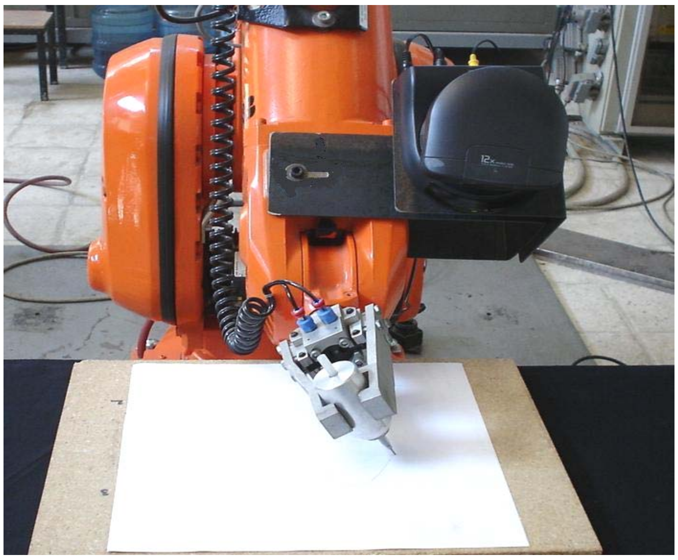


Figure 2.11 Pen Holder and Gripper Mounted to the Robot

2.4 Force-Feedback Mouse

In order to increase the safety and efficiency of the teleoperation a force-feedback mouse is used. The force-feedback mouse designed in this thesis provides physical constraints to the operator producing resistance to his movements. It is basically a powered mouse, giving the user the ability to feel the resistance in order to provide information. This mouse is used at the client side by the operator to generate maneuvering commands. If it is needed to restrict the generation of commands, the mouse resists to the movements of the operator translating digital information to the physical sensations.

In its most straight forward realization the force-feedback mouse comprises a conventional mouse, electromagnet and controller circuit Figure 2.12. Controller circuit generates electric signals to energize the electromagnet according to the received signal from client software application. When the electromagnet is energized, it produces a reluctance force to produce a resistance to the motion of the mouse. The detailed information about this component is given in Chapter 4.

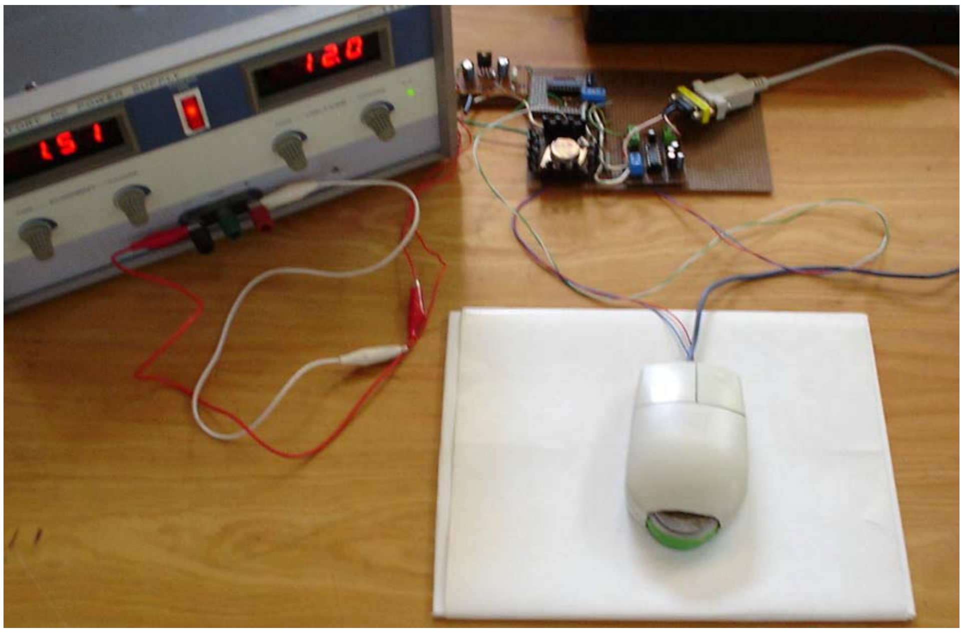


Figure 2.12 Force-Feedback Mouse

2.5 Server Computer

A desktop computer is used as a server side computer in the test bed. It runs the developed software and is connected to the controller unit and video cameras through serial ports. Since the computer has an extra multi-port serial card, it is possible to connect more than two devices to the available serial ports. The computer is also connected to the departmental local area network (LAN) through an Ethernet card, and this LAN is connected around a campus-wide backbone network, METU-NET Backbone. 100 Mbps capacity ULAKNET (Turkish Academic Network) Ethernet connection provides the Internet access of METU from outside the campus [27]. The installed operating system is Microsoft Windows XP Professional. The computer uses a Pentium 4 family CPU 2 GHz speed and 256 MB of RAM.

CHAPTER 3

COMMUNICATION LINK

3.1 Overview

Communication media carry information among all sub-systems of the teleoperation system. Communications between robot's controller unit and server computer, between cameras and server computer, and between force-feedback mouse and client computer are implemented through RS232 connections. On the other side, the communication link between client side and server side is established via the Internet. Internet has an undeniable influence onto the teleoperation system since the communication time delay presents in the transmission of control signals and multimedia. In this section, Internet, time delay as a main drawback of the Internet, and packet loss are described since they are relevant to the design of overall control mechanism. Round-trip time measurements and networking issues are also described. Investigating the communication link plays an important role in determining the control strategy.

3.2 Internet

Internet is a worldwide computer network interconnecting millions of computing devices throughout the world. These computing devices are traditional desktop PCs, servers, PDAs (Personal Digital Assistants), TVs, etc. All of these devices are called *hosts* or *end systems*. End systems are connected together by *communication links*. Different links can transmit data at different rate. The link transmission rate is often called the *bandwidth* of the link, which is typically measured in bits/second. METU network architecture supports data transfer rates of up to 100 Mbit/s. End systems are not usually directly attached to each other via a single communication link. Instead, they are indirectly connected to each other through intermediate switching devices known as *routers*. A router takes a chunk of information and forwards them to other routers or end systems. This chunk of information is called a *packet*. End systems, routers, and other "pieces" of the Internet run *protocols* that control the sending and receiving information within the Internet. TCP (Transmission Control Protocol) and IP (Internet Protocol) are two of the most important protocols in the Internet. The Internet's principal protocols are collectively known as *TCP/IP* [28].

3.3 Internet Time Delay

Today's Internet is a packet-switched network. In packet-switched networks, a communication session's messages use the resources (buffers, bandwidth) on demand, and as a consequence, may have to wait for access to communication link. This waiting means queue. Queuing occurs at each router (node) along the path and causes Queuing Delay, which is one of the important delay types. The others are, Nodal Processing Delay, Transmission Delay and Propagation Delay. Sum of these delays constitute a total nodal delay and shown in Figure 3.1. These delays are described in the following parts.

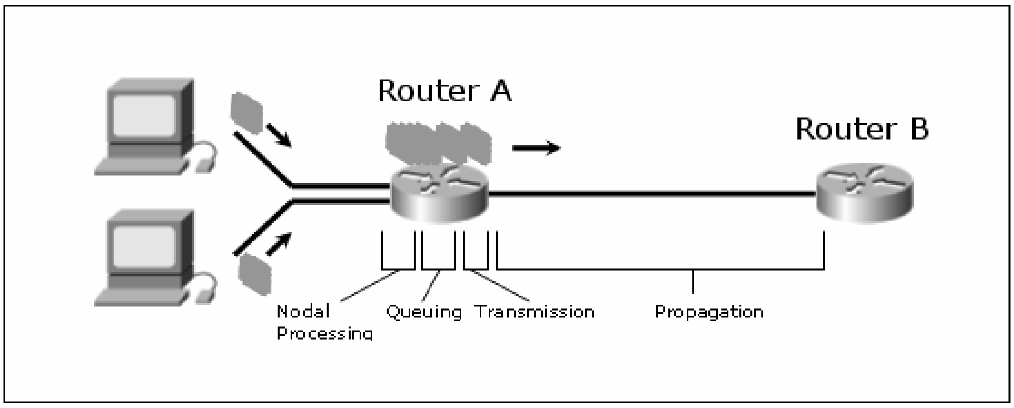


Figure 3.1 The Total Nodal Delay at Router A.

3.3.1 Nodal Delay

Processing Delay: The time required to examine the packet's header and determine where to direct the packet. The processing delay can also include some other factors, such as the time needed to check for bit-level errors in the packet. This type of delay in high speed routers is typically on the order of microseconds.

Queuing Delay: A packet encounters a queuing delay as it waits to be transmitted onto the link. The queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for the transmission across the link. Packets are transmitted in first-in-first-out (FIFO) manner. If there is no packet waiting then the delay is zero. Queuing delays can be on the order of microseconds to milliseconds in practice.

Transmission Delay: This delay is related with the length of the packet (*L* bits) and the transmission rate of the link (*R*) from router A to router B by *R* bits/s. The transmission delay is *L*/*R*. This is the amount of time required to push (transmit) all of the packet's bits into the link. This delay is typically on the order of microseconds to milliseconds in practice.

Propagation Delay: The time required to propagate from the beginning of the link to router B is the propagation delay. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link and is in the range of 2.10^8 m/s to 3.10^8 m/s which is equal to, or a little less than, the speed of light. The propagation delay is the distance between two routers divided by the propagation speed. [28] .

Table 3.1 Delays on Each Route between Two Hosts

Number of Router	Experiment 1	Experiment 2	Experiment 3	Address of the router
1	<1 ms	<1 ms	<1 ms	192.168.7.1
2	56 ms	52 ms	52 ms	85.97.56.1
3	44 ms	52 ms	52 ms	10.20.15.218
4	129 ms	132 ms	130 ms	213.248.75.57
5	137 ms	130 ms	132 ms	213.248.74.13
6	140 ms	145 ms	145 ms	213.248.65.158
7	153 ms	145 ms	145 ms	213.248.72.222
8	151 ms	147 ms	143 ms	62.93.194.229
9	153 ms	145 ms	144 ms	64.125.27.182
10	173 ms	165 ms	168 ms	64.125.27.177
11	233 ms	241 ms	235 ms	64.125.27.165
12	261 ms	266 ms	268 ms	64.125.28.70
13	270 ms	284 ms	273 ms	64.125.28.209
14	310 ms	324 ms	310 ms	64.125.29.49
15	322 ms	496 ms	324 ms	64.125.30.94
16	317 ms	319 ms	331 ms	208.185.92.219

All of these delays constitute the nodal delay, that is, the delay at a single router. End-to-end delay is the sum of nodal delays at all routers between the source host and the destination host. Table 3.1 shows the delays on each route between source host at Ankara and destination host at Amman. There are 15 routers between source and destination discovered in sequence. Each result in experiments shows the delay between the router and source host. These measurements were done making use of *Tracert* tool. Tracert is a route tracing utility that displays a list of near-side router interfaces of the routers along the path between a source host and a destination. The number of the routers may change since the Internet makes its *best effort* to deliver packets in a timely manner. The route may be changed in some cases, because some problems may occur in a router or in the communication link. An alternative route is automatically prepared in advance.

3.3.2 Delay Jitter and Packet Loss

The Internet time delay is not constant. The parameters of all these discussed delays are variable. These parameters are, mainly, number of waiting packets, transmission rate of each link, distance between the routers and the packet size. Occurrences of this variability can be seen in Table 3.1. Packets are transmitted at regular intervals from source host but they arrive at the destination host at irregular intervals. This situation is named as *delay jitter* and also defined as the variation of a metric (e.g. delay) with respect to some reference metric (e.g. average delay or minimum delay).

When the congestion in a communication link increases, it may induce the *packet losses.* A packet may encounter a full queue when arrive at a router. So, the router drops it and the packet is lost. Packet loss is an important factor for measuring the performance of a node and so is delay. TCP has a recovery mechanism for lost or corrupted packets in order to retransmit them. This sometimes results in higher latency but provides reliability. Each TCP packet has a sequence number. The sender waits acknowledgements of the sent packets. If an acknowledgement is not received then TCP retransmits the packet. TCP can fragment the packets into segments, rearrange the order of the segments, and defragment them when all of them arrive at the destination [29]. TCP avoids fragmenting the packets as far as possible. The default maximum size for the segments is 576 bytes (20 bytes for the TCP header and 20 for the IP header, leaving 536 bytes for data). TCP may not fragment the packets up to this size. In this study, the maximum data size sent is 22 bytes. Thus, command packets most probably are not fragmented. If any packet including command is lost or corrupted then TCP will retransmit it. However, the next packet may not lost and arrive at the destination before the previous packet that will be retransmitted. Even though TCP fragment the packets, this type of scenario is possible in transmission for the segments of successive packets. TCP can send all the segments of the next packet before completing the retransmission of previous packet’s lost segment. In these cases, the command order is corrupted.

3.3.3 Round-Trip Time Measurement

Round-Trip Time (RTT) measurement is a widely used method in delay measurements. RTT is elapsed time for a packet to be sent and acknowledged in return. Figure 3.2 and Figure 3.3 shows the RTT statistics obtained by probing a thousand Internet Control Message Protocol (ICMP) Echo Request and Echo Reply packets between Ankara and Tokyo. Each injected packet size was 32 Bytes and Packet Internet Groper (Ping) measurement tool was used. Ping is the simplest of all TCP/IP applications. It sends IP datagram to a specified destination host and measures the round-trip time to receive a response. Ping also records the number of missing packets in terms of percentage of packets that are lost during the measurement. Information on network measurements tools is presented in Appendix A.

Here, the measurement results at two different dates for a connection between Ankara and Tokyo are reported. The difference on behavior of the Internet between Figure 3.2 and Figure 3.3 can easily be noticed. As can be seen in the figures, these data are random and the path between two hosts seems like its characteristics are independent of a unique time delay model. These results and some other results obtained but not mentioned here show that the Internet has a nondeterministic characteristic. The dynamic routing of the Internet makes it impossible for the end-users to select the appropriate route. Furthermore, due to a simple packet processing (e.g., FIFO) at routers, it is difficult to predict the transmission delay of the packet [30] . Khamis et al. [3], Han et al. [21], Xi and Tarn [4] and Elhajj et al. [20] also emphasize that the Internet can not be modeled for prediction. If the Internet time delay would be predicted then it had been used in control system design as a lag.

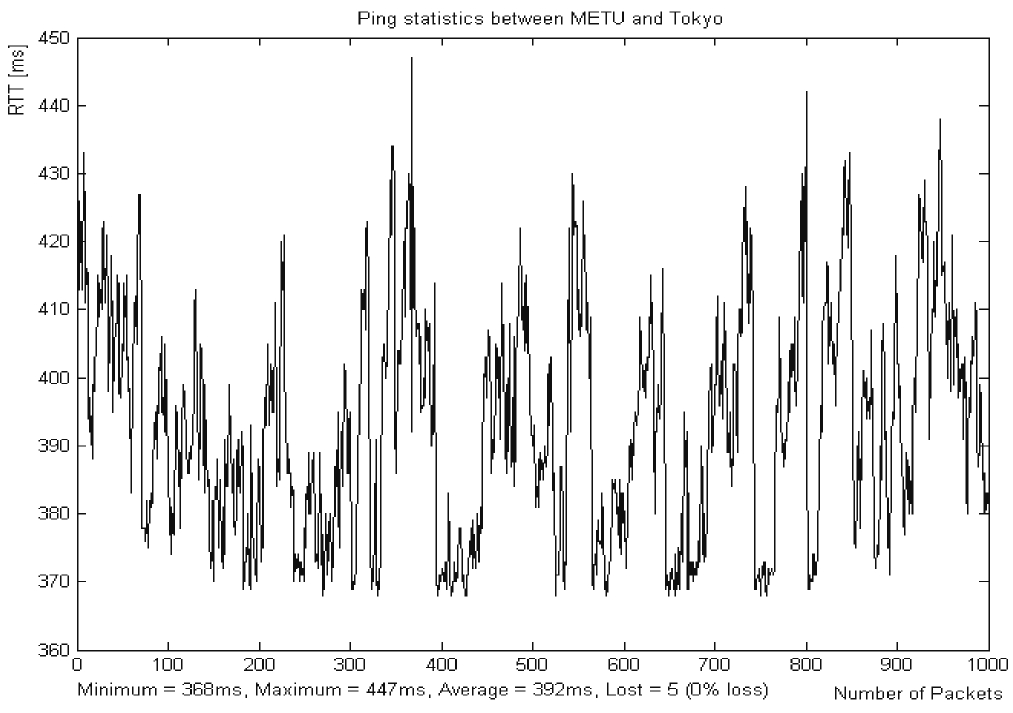


Figure 3.2 RTT Delay between METU and Tokyo on 20 July 2004

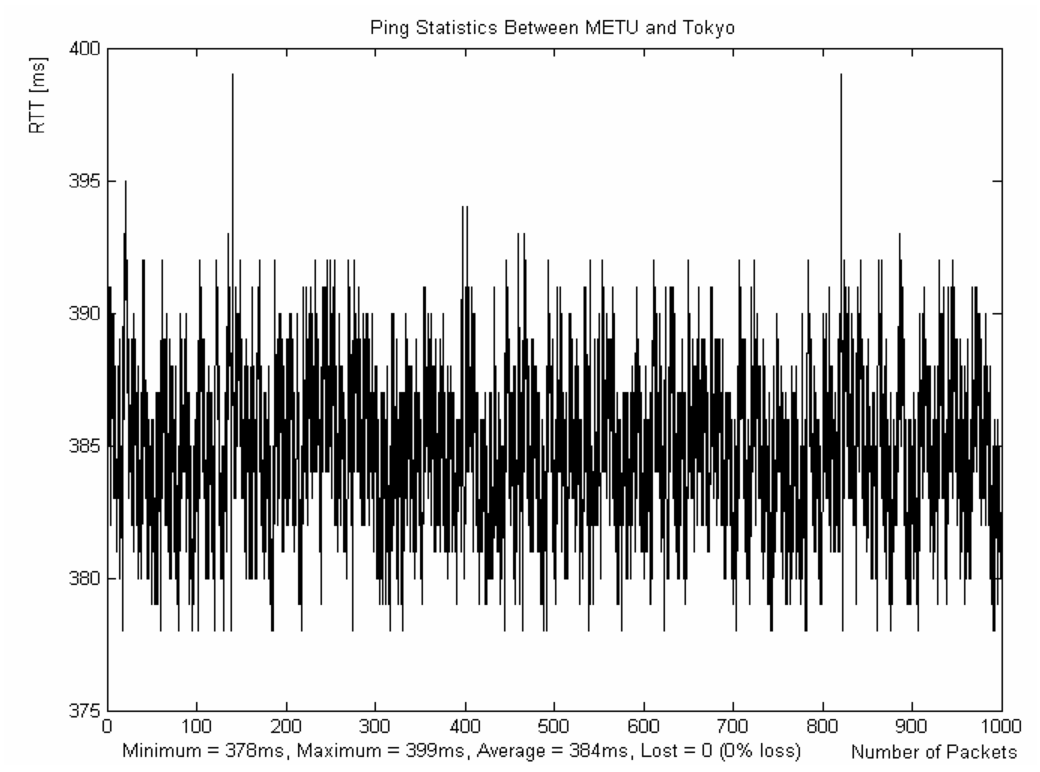


Figure 3.3 RTT Delay between METU and Tokyo on 28 Feb. 2005

Table 3.2 shows the results of several measurements implemented between two local hosts located at the same city and five remote hosts located at different places. One of the local hosts, Host1, is located at Mechanical Engineering Department at Middle East Technical University, and the other local host, Host2, is located at a home in Ankara. Bandwidth offered for Host1 is 100 Mbit/s, and for Host2 it is 256 Kbit/s. Ping utility was used for measurements. The characteristics of these connections show that the Internet has an unpredictable behavior.

Table 3.2 Time Delays between Different Hosts

Host Name	Location	Distance [km]	Average Delay [ms]	Packet Loss Rate	Number of the Routers
Host1	Local	0.5	< 1	0.00	3
Host2	Local	0.05	< 1	0.00	2
Host1	Same City	25	14	0.1	5
Host2	Same City	40	52	0.00	7
Host1	Different City	450	22	0.5	6
Host2	Different City	450	59	0.2	8
Host1	Same Continent	1000	260	0.00	19
Host2	Same Continent	1000	300	0.00	16
Host1	Different Continent	11000	257	0.00	16
Host2	Different Continent	11000	383	0.00	18

The results show that the physical distances between the hosts have not dominant effect on the Internet time delay. It can be seen that number of the routers traversed and supplied bandwidth for hosts are determining factors of the time delay. Since the Internet time delay is affected by the number of the nodes and available bandwidth, it is variable and unpredictable.

The function $y\ (t)\ = 5\ sin\ (0.006\ \pi\ t) + 5$ is used to show the effect of the communication link onto the transmission. This function is sampled between $t_0=0$ and $t_1=1000$ ms for $T=10$ ms. For each sample, $y\ (k*T)$ is sent to the remote host, where $k=0,1,2,3,...$ Two different delay measurements were made between a local computer and two remote hosts, as illustrated in Figure 3.4. According to the first measurement between local computer and Host-1, the average one-way delay is approximately 200 ms. In Figure 3.4; the time lag of Delayed Data-1 with respect to the original data is shown. As for the second measurement made between local computer and Host-2, the average one-way delay is approximately 25 ms, and the time lag of Delayed Data-2 with respect to the original data is also shown in this figure. It can be seen that the test function is not distorted severely in both tests. While the time delays in second measurement are small with respect to the first measurement, it was encountered packet losses which are depicted as broken lines in the figure. These tests also impart the unpredictable behavior of the Internet.

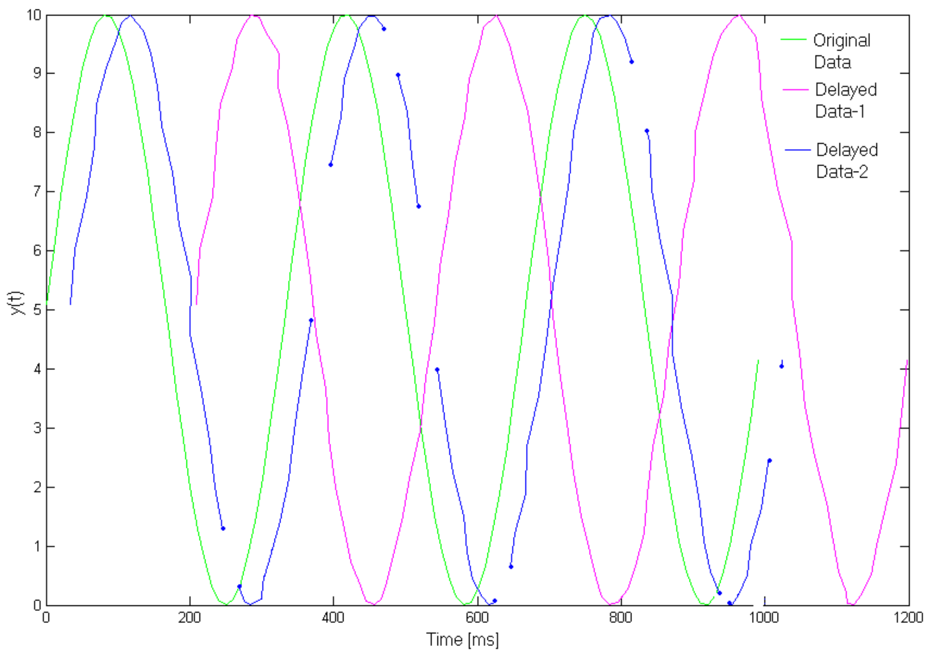


Figure 3.4 Influence of Communication Medium onto the Transmission

3.4 Networking

All computing devices communicating within the Internet use some protocols in order to control the transmission of information. IP and TCP are important protocols in the Internet. “TCP/IP protocol suite is the engine for the Internet and networks worldwide. Its simplicity and power has lead to its becoming the single network protocol of choice in the world today” [31]. In this thesis study, TCP/IP protocol suite is used to provide the communication between client and server sides since TCP is a reliable protocol. When the safety of the operation is taken into account, TCP is preferred although it sometimes makes the transmission slow as considered with another transport protocol, User Datagram Protocol (UDP).

3.4.1 Internet Protocol (IP)

Internet has a layered structure and protocols function on this layers. Network layer is one of them and Network Layer protocol is also called Internet Protocol. This layer is common to all Internet applications. IP provides the transmission of datagrams between hosts. This protocol is responsible of two basic functions: addressing and fragmentation. The datagrams are transmitted through the route according to the addresses carried. Due to packet size limitations in network, datagrams are fragmented and reassembled when necessary.

IP provides a best-effort service to deliver the packets between hosts. But it does not guarantee packet delivery and orderly delivery of packets. Best-effort service can lead to delay, delay jitter, and packet loss. Therefore, IP does not provide reliable data transfer.

3.4.2 Transmission Control Protocol (TCP)

Transport Layer provides the data transfer by transporting an *application*'s data between the client and server sides of the application. TCP operates at this layer. "The most fundamental responsibility of TCP is to extend the IP's delivery service between two end systems to a delivery service between two process running on the end systems" [31] . A process is a running program within a host. The TCP connection between processes in two different hosts is shown in Figure 3.5.

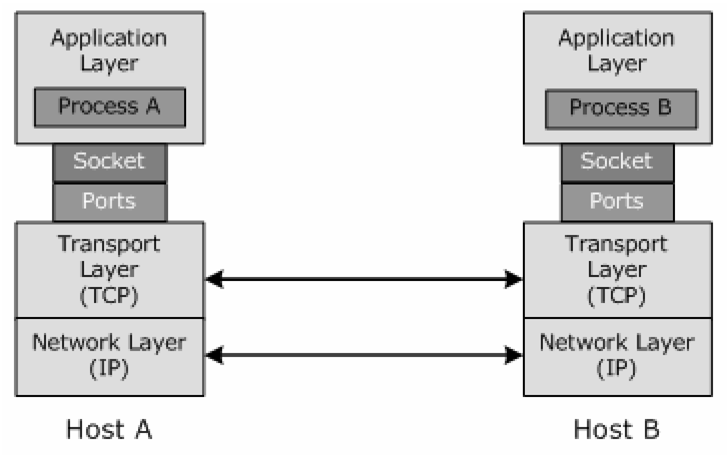


Figure 3.5 TCP Connection between Processes - (Processes A and B Communicate over a TCP Connection Carried by IP Datagrams)

TCP is a connection-oriented protocol that is an end-to-end connection is established before transmission (handshaking) and maintained till the end of the communication. In order to provide the reliable connection service, TCP provides some facilities:

Stream Data Transfer: Data are transferred as continuous stream of bytes between each direction (full duplex). Data are segmented by TCP itself and passed to IP for transmission to the destination.

Reliability: TCP recovers from data that is lost, damaged, duplicated, or delivered out of order that is fragments of a segment which arrive at the destination disorderly. This is achieved by assigning a sequence number to each fragment and source expects an acknowledgement from the destination host. If acknowledgement message is not received within a timeout interval then the packet is retransmitted. Retransmission timeout interval is determined dynamically by TCP according to some procedures [29, 32]. Sequence numbers are used for eliminating the disorder and duplication.

Flow Control: Amount of data sent by the source is determined by the destination indicating this information in the acknowledgement message. This control prevents the overflow in the buffers of destination host.

Multiplexing: TCP provides many addresses and ports within a host to provide the using of the communication facility simultaneously by many processes.

Connections: Each data stream includes some information such as sockets, sequence numbers, etc. This information is called connection and required for reliability and flow control.

3.4.3 Ports, Sockets, and Client/Server Model

Ports and sockets are used to identify the processes communicating between end-systems by providing process, host, and protocol identifiers. *Ports* are logical channels, which connect the processes at each end. They are represented by 16-bit number and processes identify itself to the protocol which will be used through this number. Only one process can engage a port at any time. A *socket* is the interface between application layer and network layer. It can be thought of as a gateway of process to the Internet. A socket is the software and its functions are connecting to ports, requesting connections, listening for connection requests, and accepting connection requests.

A network communication has typically two sides, a client side and a server side. The client program requests a connection, and the server program listens for a connection and accepts the requests. As shown in Figure 3.5, Host A might be the client and Host B might be the server, and vice versa. And also, an application can include a server and a client part, besides; this application can run on the same or on different machines.

3.5. Conclusion

Each Internet connection path has different characteristics because of different bandwidth, distance, number of routers, communication media, and network load. Unfortunately, most of these factors cannot be determined by monitoring the Internet. Besides that, the amount of data transferred from one place to another in a specified amount of time increases as new hosts take part in the Internet. Additionally, growing audio and video transmissions increases network load. During this thesis study, some measurements made in a different months of a year between two hosts imparted that the Internet characteristics can change remarkably on a path.

For Internet-based remote operations, the available network bandwidths for both master and slave sides have an important effect on the performance of the operation. Although there are different bandwidths between the nodes in an end-to-end connection, the available bandwidth offered to end systems has dominant effect on the communication link performance. Especially, live video streams may easily saturate the network if the bandwidth is highly restricted. The remote operation can not performed reliably since the saturation of a network results in loss of control and feedback packets. For instance, using a

56 Kbit/s modem connection is not suitable. If the teleoperation includes visual feedback, higher bandwidth availability must be considered.

The probabilistic type of analysis based on a stochastic model of the Internet is usually not acceptable for some remote control applications [4]. Instead of developing a single model for every connection, there must be developed different models for every connection. The randomness and the need for much more data that can not be obtained in the course of control may lead to the failure of the model. It can not be also guaranteed this model to work for the next months since the Internet grows exponentially.

As considered the delay jitter and packet losses, the control approach to be applied should cope with these disturbing factors. Delivery of the packets out of order should be handled. Taking the communication medium into consideration seriously is important in improving the teleoperation performance.

CHAPTER 4

FEEDBACK SYSTEM

4.1 Overview

The essence of feedback is to modify the input feeding according to the output. When feedback is fed to the controller, control system is designated as closed loop control. In telerobotics case, if control is not fully autonomous, sensory information is fed to the human operator. The human operator evaluates the feedback and makes decisions while generating the input information. The human operator senses, therefore the efficiency of human machine interaction plays an important role on the performance of the remote control. There are two important sensors in remote control in terms of their compliance: *sight* and *touch*. Therefore, visual and force feedback systems are implemented in this thesis study. This chapter deals with the hardware and software parts of these systems.

4.2 Visual Feedback

Vision is the most powerful sense; therefore, it is an essential feedback while the other is force in teleoperations. A vision sensor provides rich information about the object and the environment. Vision sensor is incorporated with the feedback loop in control system. The control loop is closed when visual feedback is fed to the system in some tasks like tracking a moving object with a robot hand. Position and/or feature based information sensed by vision mechanism plays an important role in feeding this information to the controller.

However, remote control system loop is not closed when the visual feedback has no effect on the controller. The reason behind it is that the visual data is fed to the human operator. In most of the remote handling applications, this data perceived by operator is raw data. In other words, the data is not processed automatically to extract information in terms of predetermined objectives. The operator evaluates the data, which is mostly still images, video streams or graphical models and he or she generates commands.

Some teleoperation system architectures were designed according to the utilization of vision. This approach is directly related to their control scheme. If the system including vision is fully-autonomous or semi-autonomous which necessitates the supervisory control, then robot vision methods are applied to extract information from visual data. The other systems like non-autonomous or semi-autonomous make use of the vision facility to feed the information received from the scene. Visual information in teleoperation has six important elements; time delay, frame rate, size of visual field, visual viewpoint, sharpness, image resolution. Especially the time delay, which is unavoidable and serious, is the main problem in teleoperation [6].

In this study, visual feedback is provided in the form of video streams flowing from the server side to the client side continuously. A graphical model of the robot is also used to feed the robot’s current position using the generated commands on the client side. It supplies a supplementary feedback accompanying with the video feedback. These utilized feedback mechanisms will be described in details in the following sections.

4.2.1 Video Feedback System

Video feedback system is composed of two CCD cameras, a video server and developed software. The architecture is described schematically in Figure 4.1. These two video cameras are connected to the server PC via serial ports and also connected to the video server. Their pan-tilt, zoom and focus values are controlled by computer on a remote site by means of developed software. Video server is connected to the network and allows at most four different video streams to be viewed through its own Web server. The video display application gets the video images from this Web server. Cameras, video server and developed software are described in detail in following sections.

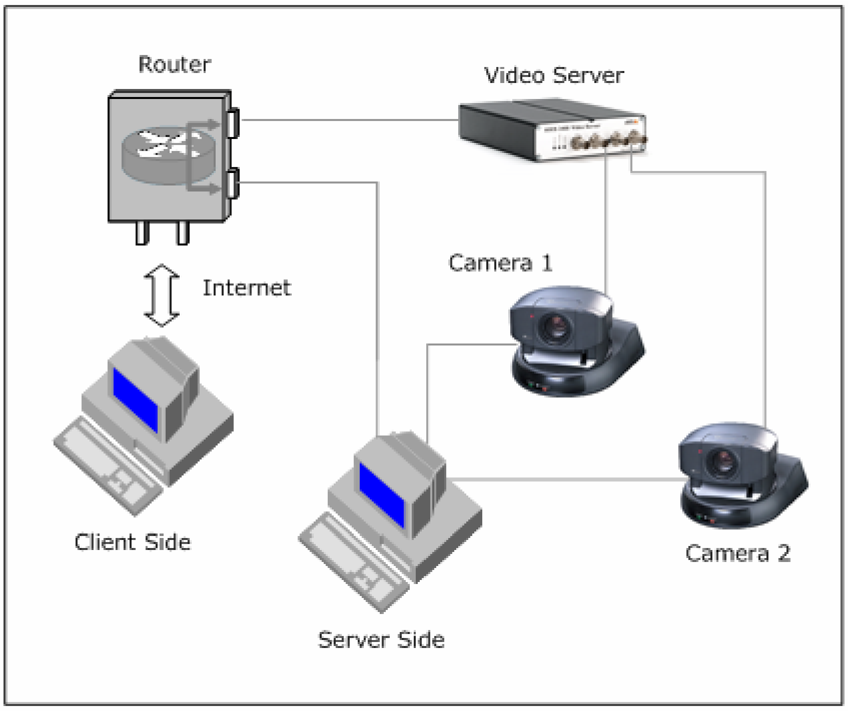


Figure 4.1 Visual Feedback Hardware

4.2.1.1 Video Camera

Two color video cameras are used to capture video from the robot and environment. These video cameras are Sony EVI-D30 pan/tilt CCD camera with highly sophisticated image processing technology which enables target objects to be recognized. These auto tracing and motion detection features are not required in this project. Video signal is produced using NTSC video standard and frames are displayed at 30 frames-per-second (fps). Cameras are connected directly to the video server through video cables. Detailed information about this camera is presented in Appendix C.

The camera head can move from right to left or left to right (pan) and from up to down or down to up (tilt). It can move 249 degrees for pan operation and 88 degrees for tilt operation. These ranges are shown in Figure C.1. The pan and tilt features of these cameras allow the user to fix the visual viewpoint. Their focus and zoom features enable to set the sharpness and size of visual field (angle of view). It is also possible to control the focus and zoom manually or set to auto mode.

Video cameras are controlled by the computer connecting them as shown in Figure 4.2. This figure shows one by one control using serial interface ports of the computer. By this connection, the address of the camera can be identified by the port number or the address assignable to each camera. Cameras are connected to the computer's serial port via VISCA cables. These cables provide the connection described in Figure 4.2. Detailed information is presented in Appendix D.

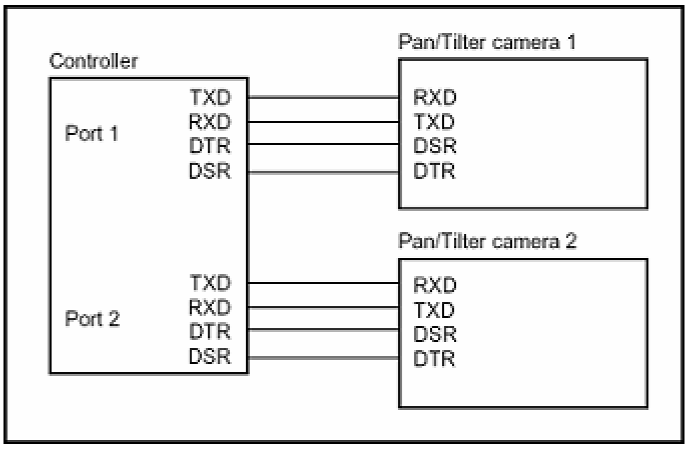


Figure 4.2 Video Camera Connections to the Serial Ports of Computer

The communication specifications are RS-232C, 9600 bit-per-second (bps), 8 bits data, 1 start bit, 1 stop bit and non parity. The rate at which data are transmitted over an RS-232 cable is measured in bps. 8 bits data include the commands, start and stop bits are used for transmission control and parity bit is used to control whether the sum of data bits in the current set is even or odd.

Communication protocol is Video System Control Architecture (VISCA) protocol. It is a network protocol designed by Sony Corp. to interface a wide variety of video equipment to computer. Under VISCA, up to 7 EVI-D30 can be connected to one controller using RS-232C communication. Two important specification of this protocol are described here. One of them is *communication from controller*; communication is started by header which comprises computer's address and camera's address followed by message and ended by terminator code signifying the end of the communication. The message part comprises communication mode (2 bytes), category code (2 bytes) and parameters. Communication mode specifies the category of the command and, category code specifies the category of the command is applicable. The maximum length of the message is 14 bytes. The other is *commands*; a command which is sent from the controller to the camera is comprised in the message part of the sent data. The commands are roughly classified into several functions such as controlling camera, inquiring the information of the camera and the ones for various purposes [33]. A command message packet is described in Figure 4.3. This sample packet size is 9 bytes and each segment is represented by hexadecimal values. Before sending a packet to the camera, every hexadecimal value is converted to byte.

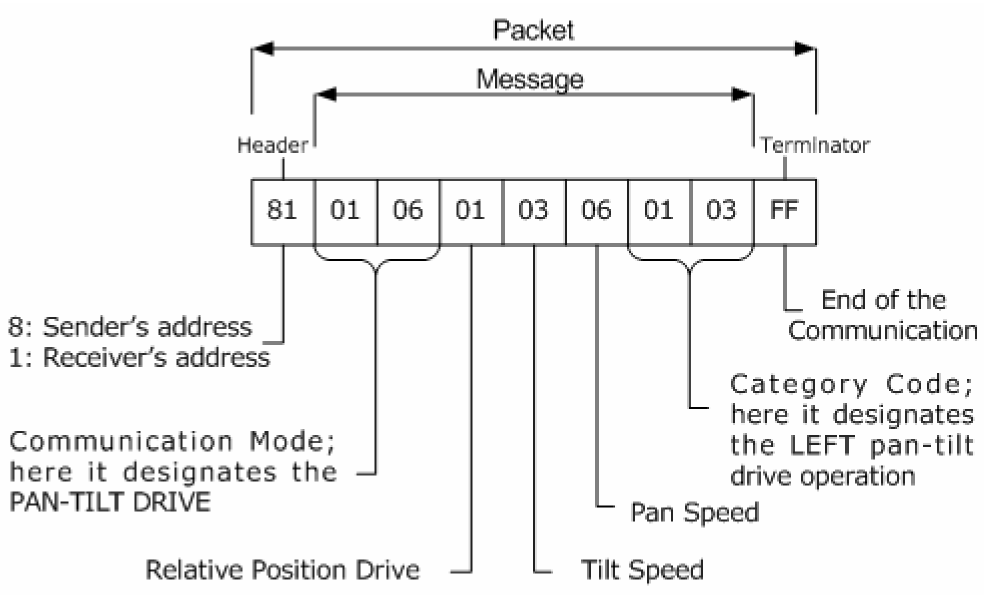


Figure 4.3 Pan-Tilt Drive Command Message Packet.

4.2.1.2 Video Server

The term "video server" refers to a network-attached server for video that is connected to a computer network like a LAN. A video server can deliver live video, automatically or on request, to a browser. A video server on the network offers a tremendous range of monitoring and surveillance capabilities by distributing live video anywhere with a network connection. Video servers digitize analog video sources and distribute digital video over an IP network turning analog cameras into *network* cameras [34].

The selected hardware is Axis 2400+ Video Server manufactured by Axis Communications. Figure 4.4 illustrates the video server hardware mechanism. The analog video received from video cameras is converted to the digital video by means of *Image Digitizer*. The images from the digitized video are compressed to JPEG still images in the *Compression* chip. The *CPU* performs the communication with network and processes the actions related with serial port, alarm, relay, drivers, etc. The *Ethernet* connection enables a direct connection to the network. Detailed information about Axis video server is presented in Appendix E.

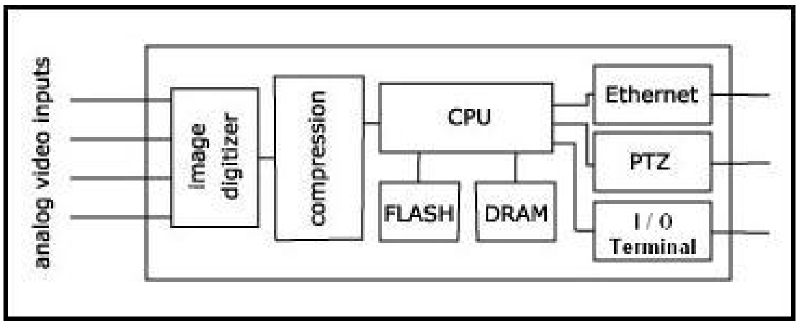


Figure 4.4 Video Server Hardware

This video server has its own IP address, making it accessible from any authorized PC on the network. A network video server can be configured to provide the entire Internet community with access to its images via a Web site, or alternatively to provide restricted viewing access to a limited number of authorized people. All settings are done through this IP address accessing a provided web page. Real-time video can also be viewed through this web page. Video server is configured with TCP/IP in a manner to transmit the video over the Internet. Setting another communication protocol like UDP is not allowed in this server. Transfer of video using TCP/IP is independent of transmission type like standard telephone modem, Ethernet, cellular phone modem, etc. Nevertheless, the elapsed time to transmit the video depends on the *size* of the video and the *bandwidth* available for the transmission.

Image size, compression, frame rate and complexity of the image are determining factors for the video size. *Image size* which is also known as frame size is directly associated with resolution and color. Resolution designates the number of the pixels in a frame, e.g., 176x112, 352x240, and 704x480. In this project, 352x240 is selected as frame size. The less resolution is not suitable to perform the remote operation safely. Image quality depends on the level of *compression*. High compression yields smaller files at the expense of image quality, while low compression results in larger files, but maintains image quality. Utilizing the compression chip in the video server, JPEG standard is used to reduce the still image size and Motion JPEG standard is used to reduce the number of images. Video server offers five different compression levels, and medium level compression is selected. Thus, while video size is decreased, the quality of video is kept in adequate level for safe operation. *Frame rate* is the number of the images transmitted per second. As described in previous section, the cameras are configured with NTSC standard, and therefore the frame rate is 30 fps. Thus, each frame is transmitted to the video server in every 33.3 milliseconds. If frame rate was larger than this rate, images would be transmitted for smaller periods. However, while the more frame rate is the more refresh rate at the client side, growing frame rate increases the bandwidth usage resulting in congestion. The last factor is *complexity*; file size of a compressed image depends on the actual content of the image. More objects in an image results in growing the size. On the other hand, bandwidth is another determining factor for video transmission. Bandwidth is the amount of data that can be sent from one computer to another through a telecommunications medium in a certain amount of time. For digital communication, bandwidth and transmission speed are used interchangeably. Transmission speed is measured in bps. To transmit one byte, two extra bits are needed for control. This means that 10 bits are required to transmit one byte. For instance, when resolution of a captured image size is chosen as 352x240 and medium level compression is applied, image size becomes approximately 7kB. Using formula given in (4.1), for 30 fps, the necessary bandwidth is obtained as 2.1Mbit/s. While some transmission types like 10-100Mbit Ethernet architectures can meet this demand, some of them like ISDN telephone modem, Cable TV modem can not.

$$\text{Bandwidth (kbit/s)} = \text{File size (kB)} \times \text{Frame rate (fps)} \times 10 \tag{4.1}$$

Axis 2400+ Video Server provides to choose color or black and white displays. To choose the black and white display decreases the image size, so, this option is preferred in this project. This video server also provides time display which is optional. If this option is selected, each frame is time stamped on digitizing stage by video server. Hence, the time stamp is a part of any frame hereafter. In this study, this facility is utilized to extract the time information from time stamp on client side. Using this time information every frame is recognized programmatically. Extraction of the time information is described in Chapter 6.

4.2.1.3 Developed Software

The developed software for visual feedback consists of displaying video and controlling camera software. Real-time video is fed to the operator and both cameras are controlled over the Internet. The developed program accesses the video server and displays the transmitted video images by this video server, continuously. When needed pan, tilt, zoom and focus operations can be performed through the buttons presents at the video display window. This software is developed using Microsoft Visual Basic .NET programming language. Figure 4.5 shows the video feedback flow for displaying the video images and message flow for camera control. The detailed information for this software is discussed in Chapter 6.

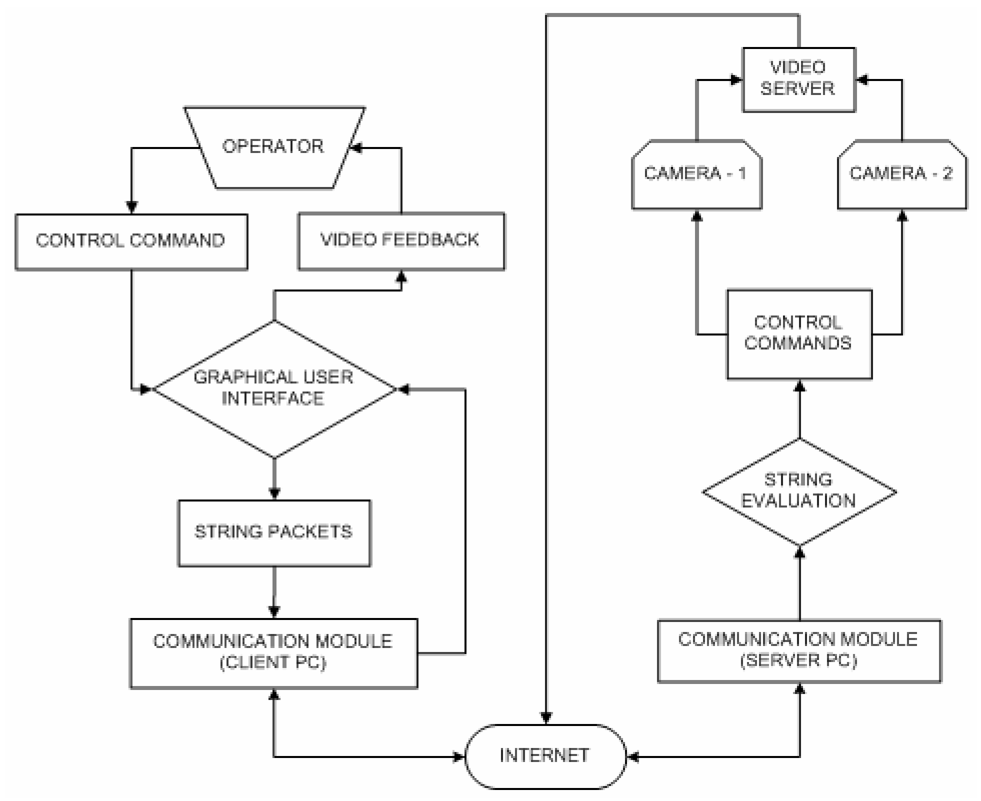


Figure 4.5 Video Feedback and Camera Control Scheme

4.2.2 Virtual Reality Based Simulation Feedback

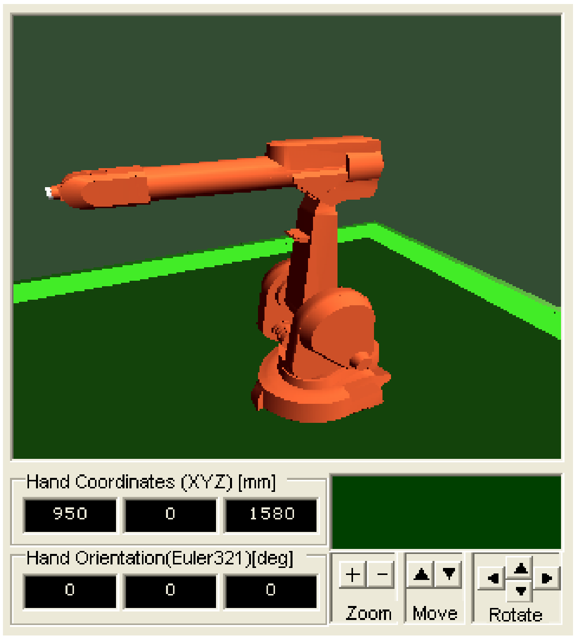
Telepresence is concerned with the study of systems that enable a person to perform human functions in an environment that is physically remote from the operator. This virtual environment allows user to visualize, manipulate and interact with graphical models. Computer generated visual, auditory, force or other sensory outputs to the human user can be mixed with the sensor based models of the real world to generate a virtual environment within the computer. This virtual environment may be a CAD model, a scientific simulation, or a view into a database [35]. An example would be a pilot in a simulator which was actually controlling a real airplane far away, and providing to the pilot visual and other sensory feedback as if the pilot were actually in the cockpit looking out the windscreen and feeling the turbulence. Virtual reality has a wide application area and it is also widely used in remote control [36].

In this study, a graphical model of the ABB IRB2000 robot is used in order to provide a supplementary feedback to the operator. All robot movements are animated using the generated commands. The commands sent to the server are used to move the graphical model. This simulation is not performed like an off-line programming but commands are carried out to move the graphical model after sending the commands. The command is utilized immediately to perform the execution in simulation environment. Therefore, the human operator can prepare for the next operation.

This simulation environment is developed modifying a previous work which was implemented by Anas Abidi in his thesis study, "Man Machine Interface Software Development for an Industrial Robot" [24]. In this thesis, the developed software provides a three dimensional graphics and animation of the robot and its environment. It allows adding custom designed objects to the simulation environment. Another important feature of this software is collision detection capability between the graphically simulated objects. This software was developed by Microsoft Visual Basic 6.0 and OpenGL, which is an Application Programming Interface (API) for graphical applications, was used for visual simulation.

The software developed by Abidi is converted to Visual Basic. NET language. One of the existing classes is excluded from the new software and the existing OpenGL and some functions are changed since it is not compatible with the .NET Framework. Instead of existing OpenGL, Tao.OpenGL developed by Tao Framework Team [37] is used in this project. Most of the unrelated GUI objects and related functions are not included for new software. Developed software is a User Control object which is added to the project as a component like an ActiveX control. A user control represents a link between the user and the program, and it can provide or process data, accept user input, perform any number of other functions that connect the user and the application. This user control can be added any project easily.

The graphical interface of developed user control is shown in Figure 4.6. The Cartesian coordinates generated by operator are inputted to the user control. If there is no collision, the movement is performed. If there is a collision, an error message appears to warn the operator.



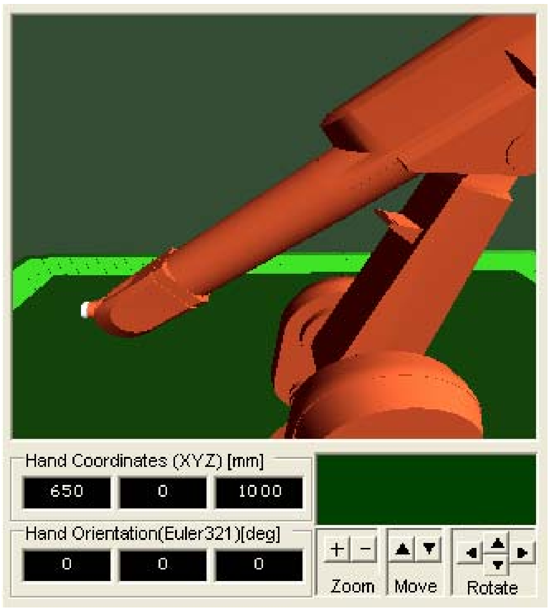


Figure 4.6 Graphical Views from the Simulation Display

4.3 Force Feedback

Haptics is the science of incorporating the sense of touch and control into computer applications through kinesthetic or tactile feedback. Tactile feedback provides information on the contact surface geometry, the surface texture of the object and slippage. Kinesthetic feedback provides information on contact forces, surface compliance, and weight. By using special input/output devices—called haptic devices—with a haptically enabled application, users can feel and manipulate virtual three-dimensional objects. The type of feedback used to convey the sense of touch is determined by the type of haptic device being used [38].

Various type force-feedback devices are available for different applications like master-slave control applications, rehabilitation for patients, surgical training, assistance for visually impaired people, gaming, etc. It has been opened up a wealth of opportunities for academic research and commercial developments, from haptic feedback systems to aid blind persons' exploration of virtual environments, through applications in aerospace and surgery, to a revitalization of the ceramics industry [39].

Force feedback in a master-slave teleoperator means that the net force vector imposed by the slave on the environment is reflected back and imposed by the master on the operator's hand [40]. The human operator sends commands, such as position and velocity, to the slave robot at the remote site using force-feedback master device, such as robotic arm, joystick or mouse. The sensory information sensed in the remote environment (real or virtual) is sent to operator side. The haptic information may be force, temperature, radiation, distance to obstacles, etc. This information is produced again as force using force generating devices, such as force-feedback mouse, force-feedback joystick or other haptic devices, and this force is applied to the operator.

In this study, force-feedback mouse is not used to reflect the sensory information to the operator. It is used in the evaluation of feedback of control signal. When the acknowledgement feedback of any command is not received it exerts force in accordance with the applied control strategy. Force feedback provides physical constraints to the operator resisting his or her movements and so, safety and efficiency of the teleoperation are increased. Therefore, the performance of the remote handling applications is enhanced.

4.3.1 Force-Feedback Mouse

A force-feedback mouse is designed in order to implement a haptic system for this thesis study. There are many haptic devices that can be found on the market and force-feedback mouse is one of them. There is just one commercial product of force-feedback mouse (Figure 4.7) on the market. Logitech Wingman force-feedback mouse provides tactile information to the user. The user controls it like a conventional mouse but feels the graphical changes when the pointer moves over a particular area on the screen. Feel sensations are generated by an electro-mechanical mechanism that imparts physical forces upon the mouse handle with respect to the mouse pad surface. The mouse is physically attached to the pad and does not allow for indexing of the mouse by picking it up and replacing it on the pad. It has a small work area and the maximum generated force is 1N. Programming development tool kit is limited to graphical applications, and it does not support Visual Basic .NET.

Figure 4.7 Logitech Wingman Force-Feedback Mouse

There are some novel designs and studies on force-feedback mouse in literature. The *Moose* was designed as a haptic interface consisting of hardware and software interfaces in order to provide the blind people to interact with the graphical user interfaces [41]. It is a powered mouse, giving the user the ability to feel the graphical objects under the mouse pointer. This mouse is used to navigate the screen like a conventional mouse; but by reflecting forces (produced by its motors) back to the user. Some studies [42-44] suggest that for a given task, use of a force-feedback mouse in cooperation with the graphical user interfaces improves performance. Results indicate that the haptic effects do not improve user's performance in terms of task completion time. However, the number of errors made is significantly reduced.

It is decided to design a force-feedback mouse instead of purchasing it, since the developed software gives flexibility to control the device. Moreover, it provides the necessary properties such as higher amount of force and easily extendable mouse pad area. This study is the first reported application of using the force-feedback mouse in teleoperation over the Internet.

It is aimed to restrict the operator's movements using this device. An electromagnet is placed into a standard conventional mouse. This electromagnet is driven by a digital to analog driver card which changes the voltage value. Electromagnetic force acting on the ferromagnetic mouse pad is changed by means of change of voltage. Server computer and microcontroller are communicated via interface software written for this application to send the control commands to the microcontroller. Thus, exerted force is adjusted by the computer to make the operator feel the resistance.

When the electromagnet is not energized, the mouse acts normally, with the same low friction and only a slight increase in mass. Energizing the electromagnet attracts the mouse towards a ferromagnetic mouse pad, increasing the normal force, and thus the frictional force, between the mouse and the mouse pad.

Following sections present the implementation of the hardware part of the design. The elements constituting the hardware are described separately in the following sections. Mouse body, electromagnet, electronic circuit and mouse pad constitutes the force-feedback mouse. The designed force-feedback mouse is shown in Figure 4.8.

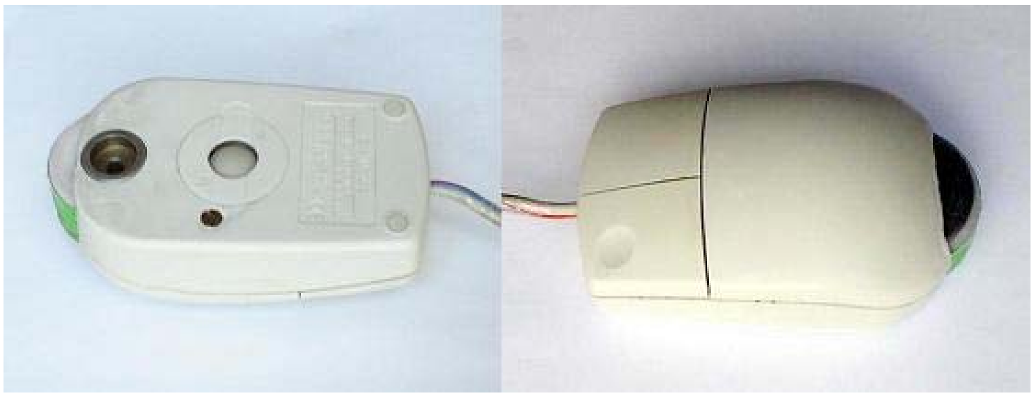


Figure 4.8 Force-Feedback Mouse

4.3.1.1 Mouse Body and Mouse Pad

The body of the force-feedback mouse is not produced but an existing mouse is modified. The modification just includes changing the location of some original parts of the mouse to locate the electromagnet, and changing the mouse body to fix the electromagnet. Mouse pad is simply a steel plate which is covered by fabric. Mouse body is shown in Figure 4.9.

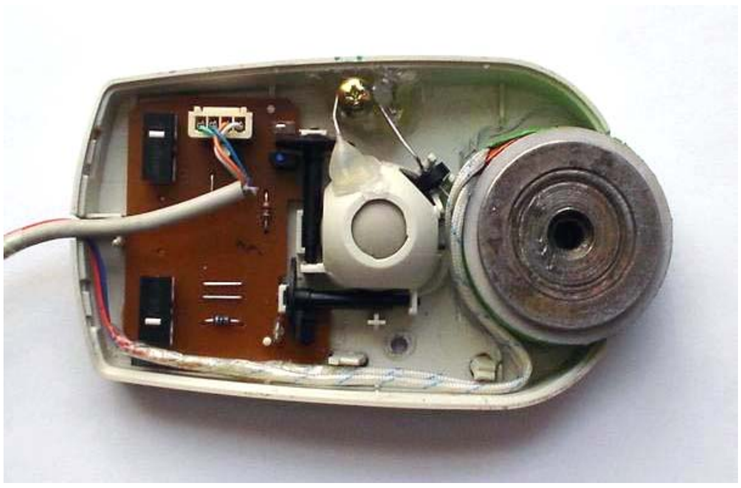


Figure 4.9 The Interior Part of the Mouse Body

4.3.1.2 Electromagnet

An electromagnet is a device in which magnetism is produced by an electric current, Figure 4.10. A strong field can be produced if an insulated wire is wrapped around a soft iron or ferromagnetic core and a current passed through the wire. When a current is passed through the coil; the electromagnet became magnetized and when the current is stopped the coil is de-magnetized. The strength of the magnetic field produced by such an electromagnet depends on the number of coils of wire, the magnitude of the current, and the magnetic permeability of the core material; a strong field can be produced from a small current if a large number of turns of wire are used.

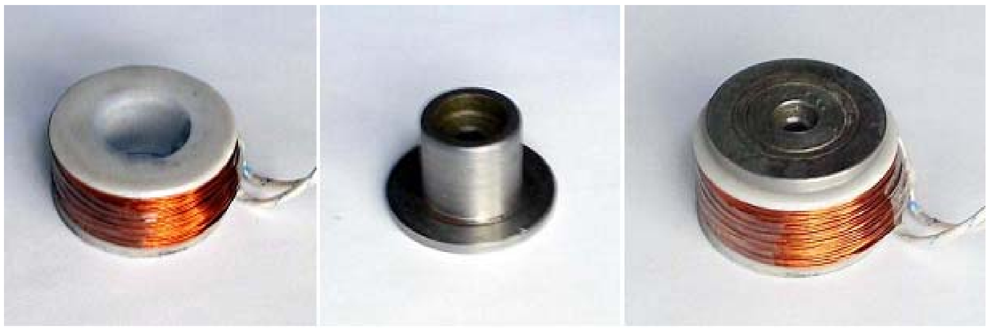


Figure 4.10 Electromagnet

The maximum force exerted by the electromagnet is determined as 50N to provide the sufficient resistance to the operator. In order to find the approximate force value exerted by the electromagnet, the following formula is used:

$$F \cong \frac{B^2 A}{2\mu_0} \tag{4.2}$$

where B is the Magnetic Flux Density, [Tesla], also known as Magnetic Induction, A is the area of the electromagnet contacting the plate, [m^2], and μ_0 is Permeability of Free Space and its value is $4.\pi.10^{-7}$, [Tesla. meter/Ampere].

Magnetic induction is the flux (flow of a magnetic field) per unit area normal to the direction of the magnetic path. A magnetic induction formula is:

$$B = \frac{\mu_0 N I}{L_g} \tag{4.3}$$

where N is Number of Turns in the coil, I is Current [Ampere], and L_g is the length of the path of the central flux line in the air-gap [m]. Air-gap is a low permeability gap in the flux path of a magnetic circuit.

The number of the turns in the coil is calculated by using the following formula [45]:

$$N = \frac{h - 2 \cdot t_p}{d_w} \cdot \frac{r_2 - r_1 - t_p}{d_w} \cdot f \tag{4.4}$$

where t_p is the bobbin thickness, [m], h is the bobbin height, [m], r_1 is the inner radius of the bobbin, [m], r_2 is the outer radius of the bobbin, [m], d_w is the diameter of the wire, [m], and f is the space factor. Type of the designed electromagnet's coil is bobbin-wound coil. It consists of wire wound into a specially prepared bobbin. The bobbin is used to wrap the wire around the core metal orderly and keep to the winding, Figure 4.11. Material of this bobbin is polyamide.

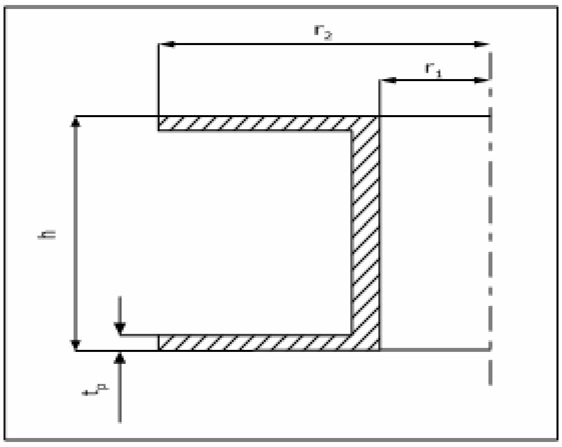


Figure 4.11 Bobbin

Wire of the wound is enamel insulated copper wire. If heating of the coil is considered as an important criterion, the following formula [45] is used to calculate the final temperature rise of the coil:

$$\theta_f = \frac{\rho}{2 \cdot k \cdot f \cdot t} \cdot \left(\frac{N \cdot I}{h} \right)^2 \tag{4.5}$$

where ρ is the resistivity of the wire, [ohm.m], k is the heat-dissipation coefficient, [W/m^2.°C], and t is the gross coil thickness, [m]. This formula is used to estimate the number of turns for a given limit temperature value.

Table 4.1 Parameters of the Electromagnetic System

Parameter	Value
F	50N
N	275
d_w	0.71mm
t_p	1mm
h	17mm
r_1	10mm
r_2	20mm
L_g	0.7mm
μ_0	4. π.10^{-7} Tesla. meter/Ampere
V	12V

The calculated parameters for the electromagnet are shown in Table 4.1. The electromagnet is tested for different voltage values and the drawn current values are measured. The measured current values corresponding to the supplied voltage are shown graphically in Figure 4.12. In this experiment, a paper about 1mm thickness was placed between electromagnet and steel plate. It is also observed that changing the distance between electromagnet and ferromagnetic pad changes the current drawn by electromagnet. Increase in distance results in decrease in exerted force.

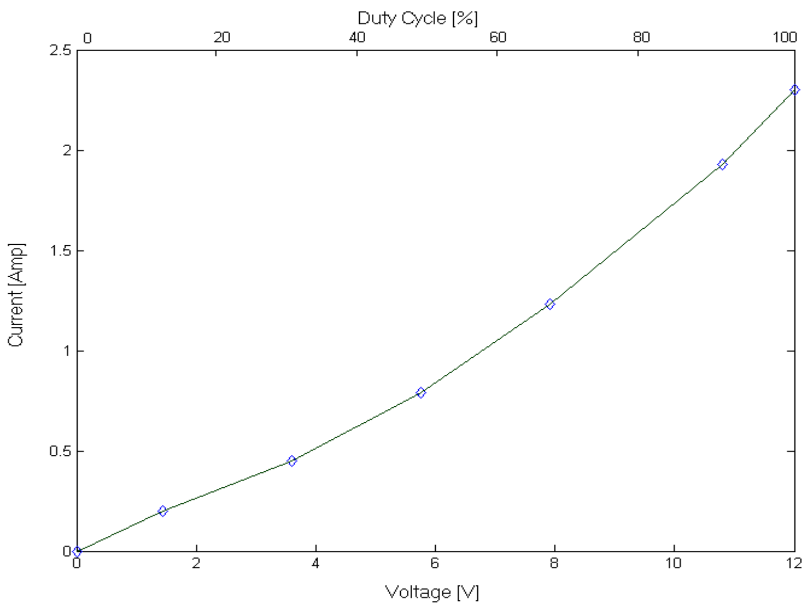


Figure 4.12 Voltage-Current Relationship of the Electromagnet

4.3.2 Driver Circuit of the Electromagnet

The designed electronic circuit is responsible for energizing the coil. This electromagnet driver circuit mainly constitutes of a microcontroller communicating with the computer, coil driver circuit connected to the PWM output of the microcontroller, a serial interface for PC and a voltage regulator. The designed mouse driver card and circuit schematic are given in Figure 4.13 and Figure 4.14, respectively.

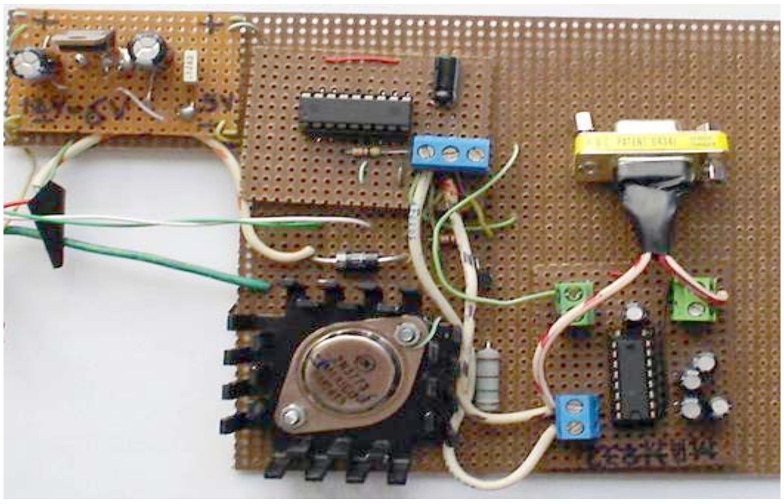


Figure 4.13 Mouse Driver Card

The driver circuit receives the signals indicating force levels from computer by means of the microcontroller. The force level information is evaluated and used to determine the duty cycle of the Pulse Width Modulation (PWM) signal which is basically digital waveform. PWM output of the microcontroller is connected to a coil driver circuit. This circuit comprises a general purpose transistor and power transistor. The function of both transistors is switching. These two transistors are connected together so that the transistor switched by the microcontroller switches the power transistor. Power transistor allows the current to flow through the electromagnet. The components constituting the whole driver circuit are described in the following sections.

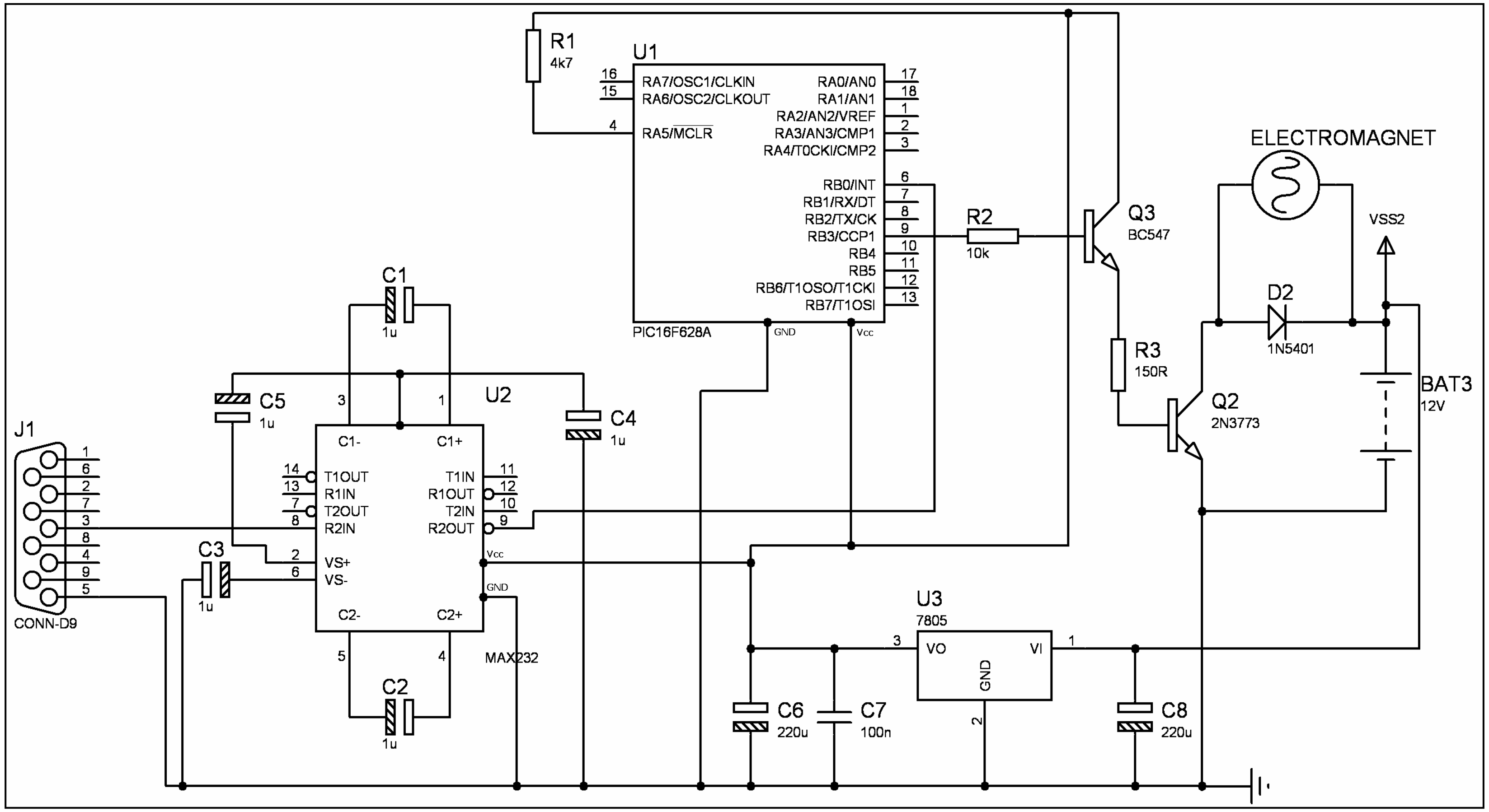


Figure 4.14 Driver Circuit of Electromagnet

4.3.2.1 Microcontroller

A microcontroller is a cheap single-chip computer and typically contains all the memory and I/O interfaces needed. Its most important feature is to store, to erase and run a program easily. Microcontrollers are general purpose devices and a user can create a wide variety of designs thanks to their facilitating features.

PIC16F628 microcontroller is one of the *Microchip Technology*'s PICmicro microcontrollers. Availability, documentation facility, price, programming easiness and some hardware capabilities are the main criteria for selecting this PICMicro microcontroller. Detailed information about this microcontroller can be found in the datasheet [46].

The aim of using this microcontroller is mainly to make use of its communication and PWM facilities. PIC16F628 plays an important role in driving the coil. The information sent from the created graphical user interface passes through the serial interface, MAX232, then it is transmitted to the PIC16F628 and this information is processed by the program which is stored in PIC16F628 to produce PWM signals. These signals drive the coil driver circuit.

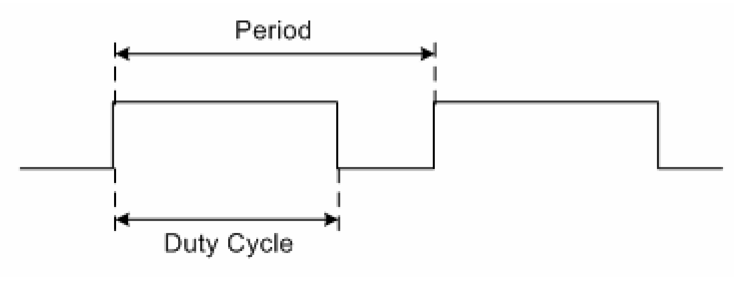


Figure 4.15 PWM Output

PWM is technique providing a logic '1' and logic '0' for a controlled period of time, Figure 4.15. By adjusting the duty cycle of the signal (modulating the width of the pulse), the average voltage can be varied. When needed to adjust the voltage level, duty cycle value is changed between 0 and 100 percent. The supply voltage of this electromagnet is maximum 12V and it corresponds to 100% duty cycle. If duty cycle value is set to 50% then 6V is supplied to the device. The output voltage is not directly generated from the microcontroller but it is generated from another circuit driven through PWM signal. For this application, generated PWM signal is 1 kHz. Two different PWM signal outputs are shown in Figure 4.16 .The duty cycle values are 90% and 50%. Thus, corresponding average voltage values supplied to the electromagnet are 10.8V and 6V.

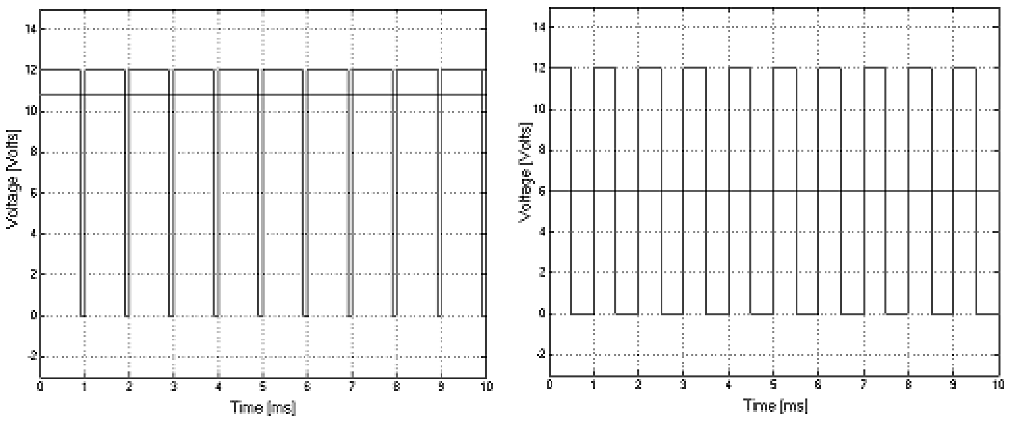


Figure 4.16 PWM Signal Outputs

PIC16F628 has PWM hardware and this facility enables this microcontroller to carry on another process while PWM signal is being produced. In other words, microcontroller does not wait for completion of other processes. This feature allows receiving serial data from computer at any time without interruption. Formulas for period and duty cycle and the related information can be found in PIC16F628's datasheet [46].

Output pulse is not totally a square wave generated from the microcontroller. In order to obtain a smooth square wave output and to prevent the overshoot, an RC filter can be used. It consists of a resistance and a capacitor. After performing some simulations and experiments for different capacitor values, it is observed that capacitor causes to delay in transmitting the electric signal. This filter is not used in this circuit to avoid the delay. Since the signal is used just for switching, its little deviation has not a noticeable effect on driving the electromagnet.

The MAX232 [47, 48] integrated circuit is used between computer and microcontroller in order to do the necessary voltage level conversion. On the other hand using a voltage regulator a fixed 5V output voltage is supplied to the circuit components.

4.3.2.2 Coil Driver Circuit

This circuit is driven by the PIC16F628 and it drives the electromagnet. As it shown in Figure 4.17, it consists of two different transistors, a powerful resistance and a diode. When a certain amount of current is applied to the base of these type transistors, the transistor allows a larger current to flow through from its collector to its emitter. The current flows during the duty cycle of the PWM signal. When the PWM signal reaches to low voltage level, the transistor switches off the collector current.

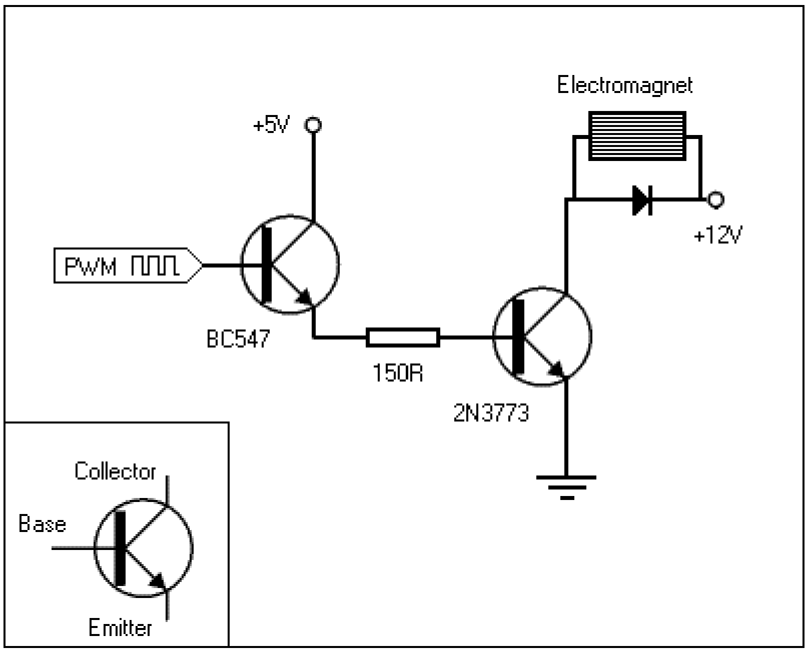


Figure 4.17 Coil Driver Circuit Schematic

Maximum current sourced by any I/O pin of PIC16F628 is 25mA and maximum voltage value is 5V. These values are not enough to drive this type of electromagnet which needs maximum 2.4A and 12V. Considering the heating problem, 2N3773 power transistor is selected to drive the electromagnet directly. These types of transistors are used in power switching circuits such as relay or solenoid drivers, DC to DC converters or inverters. The other transistor is a common type of transistor, BC547. It is used in general purpose switching and amplification. BC547 is driven by the PIC16F628 and it drives the 2N3773. This transistor is used to protect the microcontroller from high current.

4.3.3 Software Implementation

This section presents the implementation of the software part of the design. Two different computer programs are written. One of the programs is added to the graphical user interface and written in Visual Basic.NET. The calculated force value is sent to this program procedure and this value is scaled according to the pre-determined scale value. The scaled value is varied from 1 to 16. When scale value is assigned, it is sent to the microcontroller by means of a communication control object, Microsoft MSComm Control. This control object accompanies with Microsoft Visual Studio and it provides serial communication. The data can be transmitted or received through serial port.

The other computer program is created for microcontroller using PicBasic Pro Compiler programming language from microEngineering Labs, Inc. The program is compiled and converted to assembly format in computer by means of a packet program. The assembly file is written to the memory of PIC16F628 via a packet program and a programming card. This program allows the generation of sixteen different force values varying from 0 to 50N. The duty cycle value is calculated by evaluating the incoming data from computer. This value is written to the PWM related registers of microcontroller. The width of pulses instantly changes when force level information is transmitted from the client application.

4.4 Conclusion

In this chapter, development of a visual feedback system and design of a force-feedback mouse is discussed. The visual feedback system provides the operator continuous transmission of the video images from the robot environment. The operator can control the cameras over the Internet to change the viewpoint of the cameras or to adjust them.

The force-feedback mouse and its driver circuit are described in this chapter. In command generation, the exerted force guides the operator in accordance with the command generation. This force-feedback mouse can be used for many applications easily. The steel pad can be enlarged whenever desired.

CHAPTER 5

INTERNET CONTROL ARCHITECTURE

5.1 Introduction

Controlling a teleoperator, which is needed to meet the performance requirements of the remote operation over the Internet, can be a big challenge. Stability and synchronization are the main criteria for the performance of the teleoperation. Time delays in the control loop inevitably degrade the system performance. These time delays are mainly introduced as random delay in the communication channel between server and client side, computational delay within the robot's controller, and delay arising from the human operator's reaction time.

In order to cope with the time delay Sheridan [49] introduced the Supervisory Control. The human operator communicates a goal over the delayed communication channel to the telerobot. The goal is executed by the remote (slave) control system which presents at the remote site and does not include the human operator within the control loop. The human operator supervises the operation within the supervisory loop by monitoring and iteratively updating the program. This type of control requires the telerobot having a degree of autonomous control. Time delay in the supervisory loop is acceptable so long as

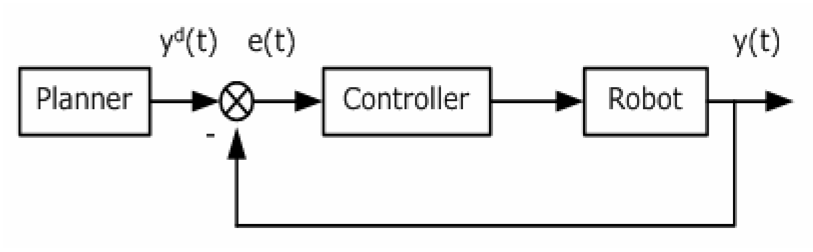
1) the delay is smaller than the time for task execution,
2) the subgoal is a conveniently large 'bite' of the total task,
3) the unpredictable aspects of the remote environment are not changing too rapidly (i.e., the disturbance bandwidth is low), and

4) the subordinate automatic system is trustworthy.

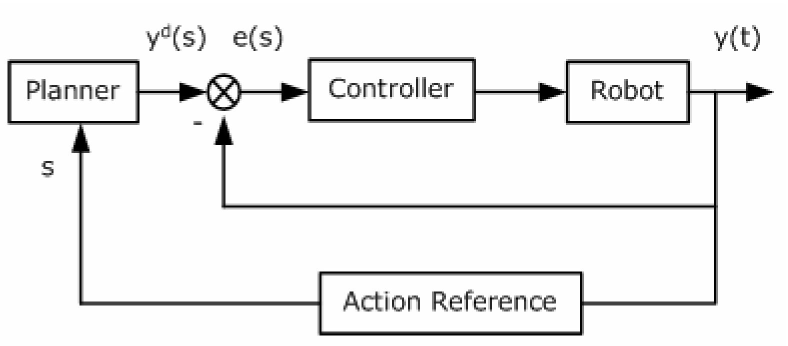
As it appears from this control method, the remote control system must have autonomy [49].

Many teleoperation systems may not have autonomy on the remote site on account of the operation type. Teleoperator may follow the inputs sent by the human operator via the Internet in real time. This type of operation is implemented by performing actions not transferring information. Performing action requires the synchronization of actions, i.e., synchronization between human operator's action such as generation of a command, and evaluation of feedback(s) sent from the robot site and execution of these commands by the remote robotic system. The performance of the Internet affects the synchronization as well as the stability. To cope with this problem a non-time referenced action control scheme was developed by Xi and Tarn [4].

Instead of the traditional control methods using the time as action reference, non-time based action reference is used to deal with the random time delay. This new type of action reference which is based on the event-based planning and control theory called *event* and not directly related to time. Each entities of the system is referred same event and maintain the synchronization regardless of the time delay since the event is based on the sensory output of the robot except for the time. Figure 5.1 depicts the comparison between traditional control and non-time based control; the action reference is usually denoted by s. In traditional control, the task is planned in advance and is a function of time. Planner feeds the input to the system according to the predefined plan. "The real challenge is to develop a planning and control scheme that is able to detect and recognize both discrete and continuous events and adjust and modify the original plan at a high rate (same as the feedback control loop) to recover from errors or unwanted situations and eventually achieve superior performance" [1]. In event-based control, a new action reference variable related to the system output is determined and used in planning. This action reference variable is independent of time. The input given to the system is referenced to the action reference variable which is computed by Action Reference.



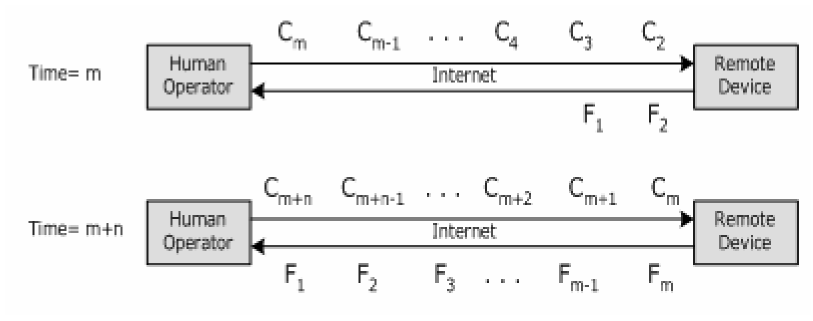
(a) Traditional Planning and Control



(b) Event-Based Planning and Control [4]

Figure 5.1 Traditional Control and Event-based Control Schemes

As for the teleoperation over the Internet, the path is not defined in the beginning but it is generated in real time. The human operator makes decisions and generates commands based on the feedback coming from the robot environment. The human operator functions as a planner. It is necessary to be established a reliable and sufficient feedback system between the human operator and the robot environment. The Internet performance affects the efficiency of flow of control signal and feedback signal. Time delay, buffering effect and packet loss cause the desynchronization between the human operator and the robot. As it can be seen in Figure 5.2, time based control results in many packets to be flowing in the Internet. When the human operator generates the command, C_{m+n}, the command generated previously, C_m, would have being executed, and a feedback signal which belongs to a more previous command would have been arrived at the client side. Until the feedback of the command which is generated at time $m+n$ arrives at the client side, the human operator will not been informed about the most up-to-date status of the remote device.



Time= m
Human Operator
Remote Device
Internet
C_m C_{m-1} . . . C_4 C_3 C_2
F_1 F_2
Time= m+n
Human Operator
Remote Device
Internet
C_{m+n} C_{m+n-1} . . . C_{m+2} C_{m+1} C_m
F_1 F_2 F_3 . . . F_{m-1} F_m

Figure 5.2 Buffering Effect of the Internet Time Delay

Event-based control introduces a solution to overcome the disturbing effects of the communication line. The human operator generates the commands in coordination with each event. Figure 5.3 shows the command generation scheme to eliminate the described effects. The generation and transmission of the commands are not dependent on time. Any command will not be transmitted until the feedback of the last command arrives at the client side. As shown in Figure 5.3, the command C_{n+1} is sent after receiving the feedback of previous command. At this moment, event n finishes and event $n+1$ starts. After command C_{n+1} arrives at the remote device, it is executed and its feedback is sent to the operator after execution. When the human operator receives the information about the most up-to-date status of the remote device, F_{n+1}, the next command, C_{n+2} is transmitted. According to this scheme, each command corresponds to an event.

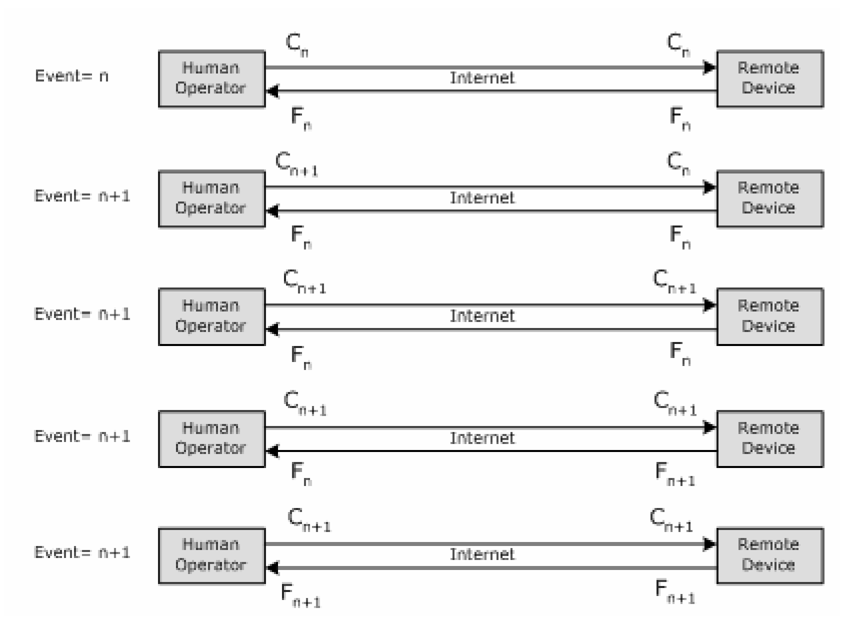


Figure 5.3 Elimination of Buffering Effect through Event-Based Control Scheme

Concerning the stability of the teleoperation system under the event-based referencing, the following theorem was proven in [4]:

> **Theorem 1.** *If the original robot dynamic system (without remote human/autonomous controller) is stable with time t as its action reference; and the new non-time action reference, s= Π(y) is a (monotone increasing) non-decreasing function of time t, then the system is stable with respect to the new action reference s.*

When the event-based approach is used in teleoperation system, the Internet time delay will have no effect on the stability of the system since event-based control is independent of the time if the robot system is stable. The only assumption is needed that the robot system is stable with time t as its action reference. According to the *Theorem 1*, each event has to follow the previous one without disordering the sequence. Sending the commands to the robot in a monotone increasing way in time is necessary for the stability. It is clear that sending a command to the robot before sending the previous command will destabilize the system. Therefore, the command numbering has to increase and the order has to be kept.

In this thesis study, the most proper reference variable for the event is determined as the distance traveled. So, each position coordinate, that is command, corresponds to an event, and also, each event corresponds to a movement of the robot. A command packet sent from the client to the server includes the information for an event and a feedback packet sent from the server to the client includes an acknowledgement for this event.

5.2 Developed Control Methods

In order to achieve the control of Internet-based teleoperation, two control methods are developed and tested. It is aimed to reach the higher performance in remote control. These methods mainly deal with the command generation, command handling, and feedback handling.

The developed methods are named Simple Event-Based Method and Sliding Event-Based Method. These proposed methods and how they are implemented are described in the following sections.

5.2.1 Simple Event-Based Method

This method uses the following technique: a command packet is sent from client (master side) and then waited for an acknowledgement from the server (slave side) before sending the next packet. Figure 5.4 shows this technique. While waiting for the acknowledgement, it is not allowed generating a new command and the operator's hand movement is also restricted via force-feedback mouse. This method implies directly implementation of the even-based control method. The implementation of this technique is described in the following paragraphs.

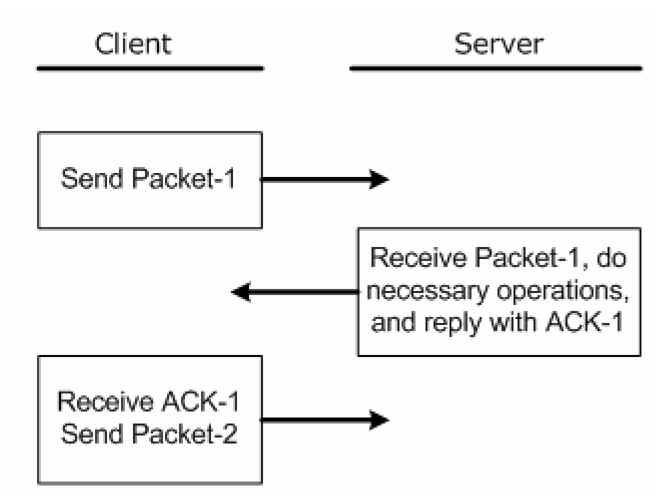

Client
Server
Send Packet-1
Receive Packet-1, do necessary operations, and reply with ACK-1
Receive ACK-1
Send Packet-2

Figure 5.4 Simple Event-Based Method

Commands are generated by mouse as increments or position coordinates in X and Y directions as described in Chapter 6. As shown in Figure 5.5(a), each command packet includes a header, X Coordinate, Y Coordinate, and the Sequence Number. When command generation is started, a packet is sent to the server side. When the packet is sent, it is not allowed generating a new command and also displacing the location of the force-feedback mouse. Each command arriving at the server side is sent to the robot and acknowledged if the operation is successful. When an acknowledgement is received, a feedback packet is prepared and sent to the client immediately. This feedback packet structure can be seen in Figure 5.5(b). The feedback packet consists of a header, the Sequence Number and the Second value which indicates the execution time.

_SEM	X Coordinate	Y Coordinate	Sequence Number

(a) A Command Packet Structure

_SEW	Sequence Number	Second

(b) A Feedback Packet Structure

Figure 5.5 Packet Structures

As soon as a feedback packet arrives at the client, previous and last acknowledged packet numbers are compared. The comparison between these numbers indicates whether the packets arrive successively or not. If there is missing packet the path generator generates the missing packet. The missing command is executed before executing the last command.

Arrival of the acknowledgement feedback is not sufficient to generate the next command. To maneuver the robot, the human operator must see the robot's most current situation corresponding to the execution of the last command. Feeding the images recorded on the execution time to the human operator is necessary for a reliable operation. Synchronization of the feedbacks was implemented to provide the reliability. These feedbacks are the acknowledgement of a command and the video stream from the robot's environment. Synchronization of the feedbacks is described in Section 5.3.

5.2.2 Sliding Event-Based Method

This method uses the following technique: all packets within a group, whose size is pre-determined, can be sent to the client side without receiving an acknowledgement. The server must acknowledge each packet received. At the moment the client receives the acknowledgement for any command, the packet group can be slid one packet to the right. Figure 5.6 shows this technique. As the number of the packets which are not acknowledged increases the operator's hand movement is also restricted via force-feedback mouse. If the number of the packets, which are sent and unacknowledged yet, reaches the packet group size it is not allowed generating a new command.

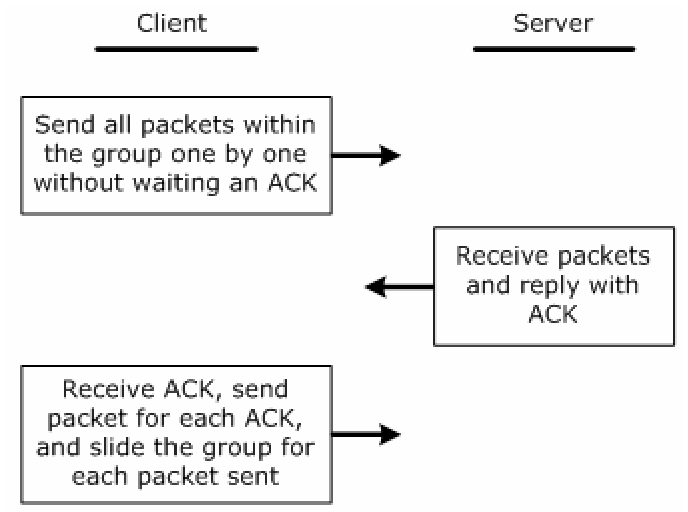

Client
Server
Send all packets within the group one by one without waiting an ACK
Receive packets and reply with ACK
Receive ACK, send packet for each ACK, and slide the group for each packet sent

Figure 5.6 Sliding Event-Based Method

Using this method, it is aimed to make the ratio of RTT to the number of packets constant. Determination of number of the packets (group size) is described in Section 5.4. For example, if RTT is 100 ms then the number of the packets is determined as 2, or, if RTT is 250 ms then the number of the packets is determined as 5. Since the human operator generates the commands as the acknowledgement messages arrive, this mechanism provides to be received the acknowledgements for the same period even RTT changes at certain levels. For example, when operating the remote device while RTT is between 200 and 250 ms, the number of the packets is 4, and the period between the packets is 50 ms, theoretically. If RTT rises to a value between 250 and 300 ms then the number of the packets is 5, and the period between the packets is 50 ms. Thus, the human operator is not affected by the change of RTT. On the other hand, using this technique, command generation flexibility is provided enabling to generate the commands within a range instead of sending a command and waiting for its acknowledgement to generate the next command. The implementation of this technique is described in the following paragraphs.

Commands are generated by mouse using the Drawing Board as position coordinates instead of increments in XY plane. The pointer on the board corresponds to the tip point of the robot. Clicking somewhere on the board and dragging the pointer causes to be moved the robot to this point. When clicked on Drawing Board and the pointer dragged on it, the position coordinates are calculated and sent to the server. Command packet structure and feedback packet structure are the same as in Simple Event-Based Method.

Before the generation of the command, the packet group size is determined. Group size indicates the maximum number of commands which can be sent to the remote site without receiving an acknowledgment. If an acknowledgement is not received after starting to generation, then the number of the commands that can be allowed generating, that is command quota, decreases by one for each generated command. The command quota changes during the operation. The event group slides as commands are acknowledged and generated. It decreases for each unacknowledged command and increases for each acknowledgement. The human operator can generate commands up to quota. The faster the mouse movement the more the commands are generated. The force exerted by force-feedback mouse is proportional to the number of sent and unacknowledged packets in a group. If all packets are sent and none of them are acknowledged, then maximum force is exerted. While approaching from lower limit to the upper limit of group size, the human operator can feel the restriction gradually. If the operator can move the mouse at a certain speed, the control can continue without interrupt and restriction. This allows the operator continuous remote control. The human operator views the robot's movements via video feedback and generates commands at the same time complying with the feedback from force-feedback mouse. Thus, the hand-eye coordination is implemented to some extent. There is also a level indicator in the developed client application to inform the user how many packets flow in the Internet. During the command generation, adaptation of the human operator command traffic results in a synchronization between the human operator and the robot to some extent. As improved the hand-eye coordination of the human operator, the synchronization capability will increase.

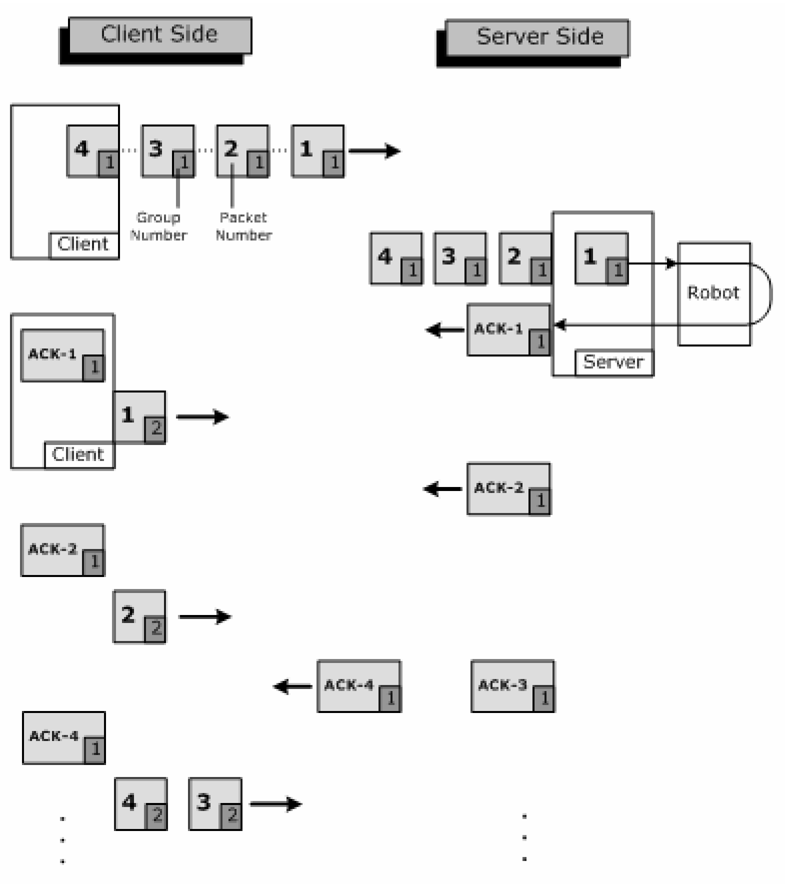


Figure 5.7 Packet Handling Mechanism

The command handling mechanism can be seen in Figure 5.7. As soon as a feedback packet arrives at the client, previous and last acknowledged packet numbers are compared. The difference between these numbers indicates how many commands will be generated. It is expected that the difference must be one, so, just one command is generated and sent. It almost always works in this way. However, since a router may discard some arriving packets due to congestion, a packet may arrive at the destination before the packet which is sent previously. In this case, if the packet numbered *n+2* arrives at the client side instead of the expected packet numbered *n+1*, then it is allowed generating two commands. Hereafter, the packet numbered *n+1* is ignored when arrives at the client.

5.3 Event-Synchronization

"An event-synchronized system is one in which all the signals in the system are always referencing events which are within a certain tolerance range of each other" [5]. In the light of this definition, the acknowledgement of the control signal is synchronized with video feedback. Since each command corresponds to an event, execution information of a command coming from the remote side also belongs to that event. Therefore, the feedback belonging to the most current state of the robot has to be synchronized with the acknowledgement message. The overall event-synchronization scheme is shown in Figure 5.8. Event-synchronization approach is directly used with the Simple-Event Based Control Method. This approach is not directly used with the Sliding Event-Based Control Method. The size of video image is larger than the control signal, thus it takes more time to transmit. Waiting for the arrival of the video image is not acceptable while acknowledgement arrives at the client site. If an image does not arrive at the client in a determined time interval according to the related acknowledgement, this image is waited for until it arrives at the client site.

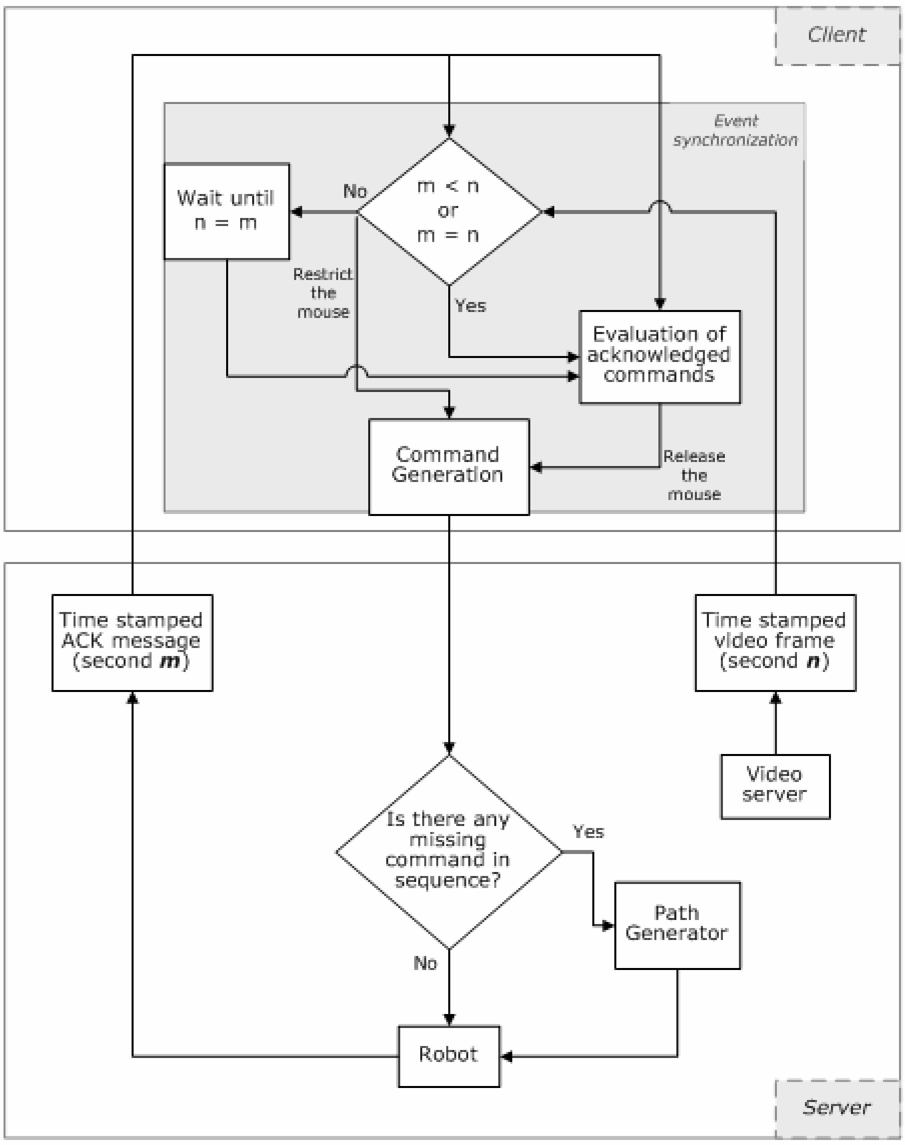


Figure 5.8 The Event-Synchronization Scheme

To synchronize the feedbacks, their agreement in execution of a command must be provided. If execution of any command is considered, since the acknowledgement received from the robot and the recorded video image of this execution are concurrent, the time information is used at the client side to synchronize them. Therefore, second information was added each feedback packet. Also, a time stamp was added to every frame by video server as a part of image. This time stamp includes the date, hour, minute, and second information. Second value can not obtained directly from the video but it is applied some image processing techniques to extract the second. This process is described in Chapter 6. Clocks of the video server and the server computer were synchronized. Since it was not possible to add any information to the video images before sending to the client side, the second information was used to determine which image belongs to which event. Therefore, the numbers obtained from second information are not related with time but they are just numbers indicating an event stamp.

The second value extracted from the video image and the second value obtained from feedback packet are compared. Matching of seconds indicates that the video image recorded at the execution time of the acknowledged command is being seen on video display. If the second value obtained from feedback packet is smaller than or equal to the other second value, then it is understood that the video image is being seen or was seen when the corresponding feedback packet arrives at the client. So, the result of this comparison indicates that the next command(s) can be generated. If the second value obtained from feedback packet is larger than the other second value, this means that the video image does not arrive at the client at that moment, although the feedback packet arrives. In this case, program continuously checks the second value of the video images and waits until the second values are equal to each other. It is allowed generating the next command(s) after matching of the seconds.

5.4 Determination of the Group Size

A formula is developed to determine the number of packets within a group of packets. Since the reason behind using the group of packets in the Sliding Event-Based Technique is to cope with the Internet time delay, it would be convenient to utilize the time delay to determine the group size. In the developed teleoperation system, the time delays occurring are shown in Figure

5.9. The total delay is the elapsed time for a command to be generated and processed its acknowledgement in return.

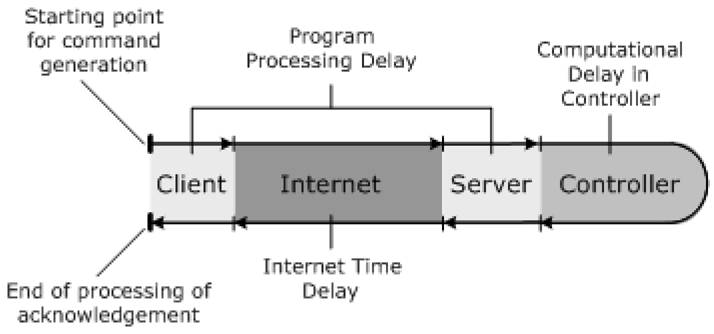

Starting point for command generation
Program Processing Delay
Computational Delay In Controller
Client
Internet
Server
Controller
End of processing of acknowledgement
Internet Time Delay

Figure 5.9 Total Time Delay in the Teleoperation System

The client and the server application add time delay while they process the incoming messages until the relevant job is done. In order to determine the time delay resulting from the program processing, program processing delays were measured. Since the simultaneously running sub-applications at the client program are more than the ones at the server program, the client program runs a little bit slowly relative to the server program. Therefore, instead of measuring the time delay occurs at the server program, the measurement for the client program was used. The measurement started at the moment of starting the command generation and ended after sending the prepared command packet. The measurement data is shown at Figure 5.10. This measurement was done for one way (generating and sending) and the time delay on the other way (receiving and evaluating) is assumed as it equals to the result of this measurement. As a consequence, the mean value of the measurement data was multiplied by four and it is assumed as constant since it is a few milliseconds. The reason behind multiplying the mean delay by four is the assumption of which the program processing delay at the server program is equal to the delay of the client program. Thus, this result was used for the calculation of the total time delay.

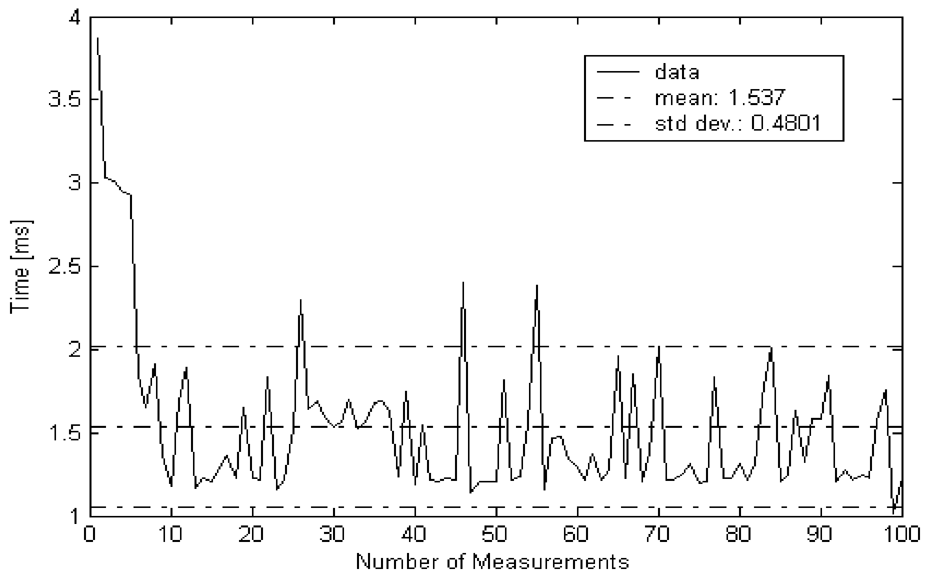


Figure 5.10 Processing Time Measurement for Client Application

The Internet time delay is the most dominant delay in determination of the total time delay. Its variability results in the change of group size. The Internet time delay, RTT, was measured as it is described in Chapter 3. Thus, RTT was directly used for calculation.

The last delay is the computational delay within the robot's controller. When a message is sent to the controller, it sends acknowledgement message in return. The elapsed time for a message to be started and received acknowledgement was measured. The measurement data is graphically shown in Figure 5.11. This delay was also assumed as constant since the standard deviation is so small and used directly for the calculation of the total time delay.

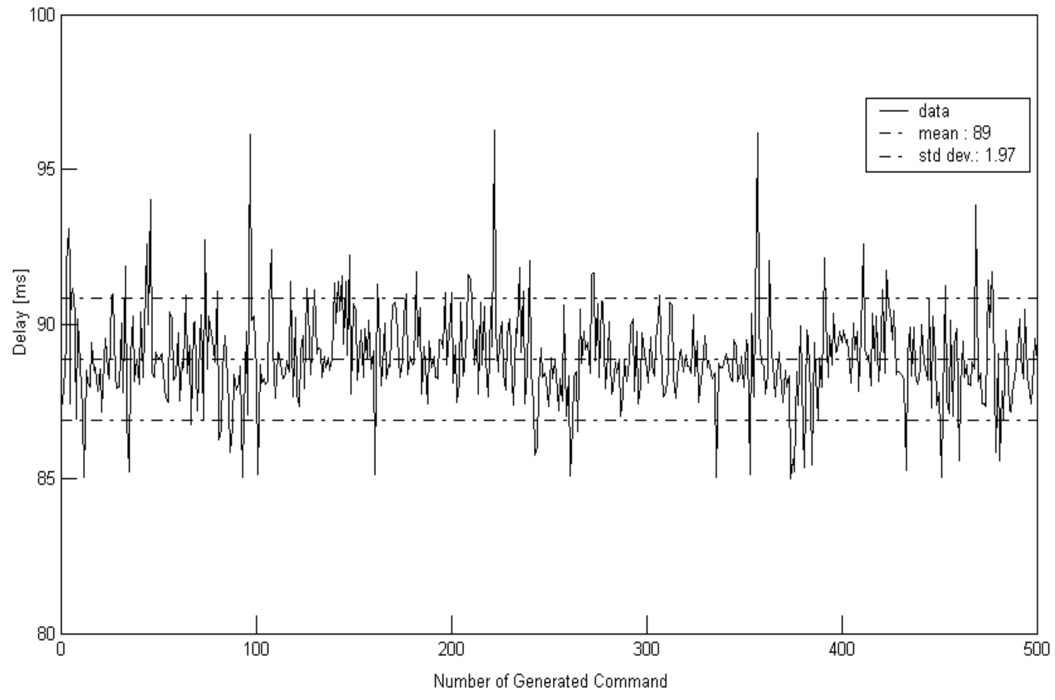


Figure 5.11 Computational Delay of ABB IRB2000 Robot's Controller

According to the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) recommendations, if end-to-end delays can be kept below 150 milliseconds for one-way transmission, the application will experience essentially transparent interactivity [50]. This recommended value is also used for two-way transmission. Thus, it was assumed that if the total time delay is below 150 ms this delay does not disturb the operator. RTT is variable and all other delays are constant. RTT limit was found when the constant delay was subtracted from 150 ms. The constant delay value was 100 ms, thus the limit for RTT was calculated as 50 ms. It means that the RTT value larger than 50 ms would disturb the operator. This value is accepted as the determining factor for the group size. For example, if RTT is measured as 250 ms then the number of the packets in a group is 5.

5.5 Path Generator

In order to deal with the disorder of packet arrival a path generator was developed in server application. This path generator generates the missing packets (simply position data) using a curve fitting technique. Although it can extrapolate the position data, it was just used for interpolating the data. The path generator can not generate the same position coordinates but it generates the position coordinates approximately.

To estimate the missing coordinate values between the last acknowledged command and the last incoming command. For this purpose, polynomial interpolation method was used. The general formula for nth-order polynomial is:

$$f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \tag{5.1}$$

After determining the nth-order polynomial that fits n+1 data points, it is easily computed intermediate values. The method is used to implement the curve fitting is the Newton's Divided-Difference Interpolating Polynomials. The general form of the polynomials is:

$$f_n(x) = b_0 + b_1(x - x_0) + ... + b_n(x - x_0)(x - x_1) ... (x - x_{n-1}) \tag{5.2}$$

where the coefficients are:

$b_0 = f(x_0)$; $b_1 = f[x_1, x_0]$; $b_2 = f[x_2, x_1, x_0]$ ...
$b_n = [x_n, x_{n-1}, ..., x_1, x_0]$
The bracketed function evaluations are finite divided differences. For example,

$$f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{5.3}$$

is the first divided difference, the *n*th divided difference is:

$$f[x_n, x_{n-1}, ..., x_1, x_0] = \frac{f[x_n, x_{n-1}, ..., x_1] - f[x_{n-1}, x_{n-2}, ..., x_0]}{x_n - x_0} \quad (5.4)$$

Since mouse movements are considered, fourth-order interpolation is enough for estimation. Thus, fourth-order interpolation function is:

$$f_4(x)= f(x_0) +(x - x_0) f[x_1, x_0] + (x - x_0) (x - x_1) f[x_2, x_1, x_0] + (x - x_0) (x - x_1) (x - x_2) f[x_3, x_2, x_1, x_0] + (x - x_0) (x - x_1) (x - x_2) (x - x_3) f[x_4, x_3, x_2, x_1, x_0] \quad (5.5)$$

Fourth-order interpolation needs five data points. The first four data consist of previous acknowledged commands and fifth datum is the last incoming command. The data points to be estimated are between fourth and fifth data points. The last four data points are stored continuously. The number of the data points to be estimated is limited to two. That is, if there are more than two lost packets the path generation is not carried out. In this case, the lost packets are demanded from client side automatically. The last acknowledged sequence number is sent to the client side. Since all transmitted packets are stored in client side, all packets between current sequence number and last acknowledged sequence number are retransmitted sequentially. Although some commands generated before arriving lost information at the client side, these packets are ignored in the server side. The path generation mechanism is described as follows:

- If the last incoming packet (P_{n+k}) and the previous acknowledged packet (P_n) are consecutive ($k=1$) then the command is executed.

- If the difference ($k=2, 3, 4,…$) between packet numbers is larger than "1" (if there is missing packet) then :

    - If the difference > 3 (2=limit value)

[There are more missing commands than limit value]

- Path generation is not carried out
- Retransmission of command(s) is demanded

  - o If difference <=3
    - ▪ If difference=1
      - The missing command is generated and executed
      - Acknowledgement message is sent

    - ▪ If difference=2
      - The first missing command is generated and executed
      - Acknowledgement message is sent

      - The second missing command is generated and executed
      - Acknowledgement message is sent

    - ▪ The last command is executed and ACK message is sent

- If this difference is less than "0" (*k=-1, -2,* …) [arrived in a wrong order]
  - The packet (P_n) arriving after the next packet (P_{n+k}) is ignored.

The commands have to send to the robot sequentially since the event-based approach provides the stability. The path generation guarantees that the commands are sent to the robot sequentially even if the packets arrive at the server disorderly.

5.6 Path Smoothing

Data smoothing facility is offered for generated commands. The idea of using the data smoothing is to smooth the path which is formed by the operator's commands. If operator needs to smooth the generated position values, s/he can choose the smoothing option while generating commands. This facility is optional since it may cause loss of information. The human operator can start or stop smoothing at any stage of the command generation.

A smoothing algorithm might be applied to filter the data if the position data is noisy due to tremor in hand, mouse related problems, undesired fluctuations, etc. The smoothing process attempts to estimate the average of the distribution of each position data. The estimation is based on a specified number of neighboring position data. In order to smooth the data, Moving Average Filtering method is used in this study.

Moving Average Filtering method is widely used in noise filtering, and it does not require any parameter like exponential smoothing methods. A moving-average algorithm simply takes the average of neighboring data points. The average value defines the new point. As shown in Figure 5.12, this algorithm requires a span of m data points which include a central data point, X_n, and neighboring points, points from X_{n-3} to X_{n+3} except for X_n. The span slides for each data point, which corresponds to the center of the span, in the data array consecutively. The calculated average of these m points is the new smoothed value, X'_{n-3}. The next calculation is done for the points between X_{n-2} and X_{n+4} where the central data point is X_{n+1}. The calculated values are not used recursively in the smoothing.

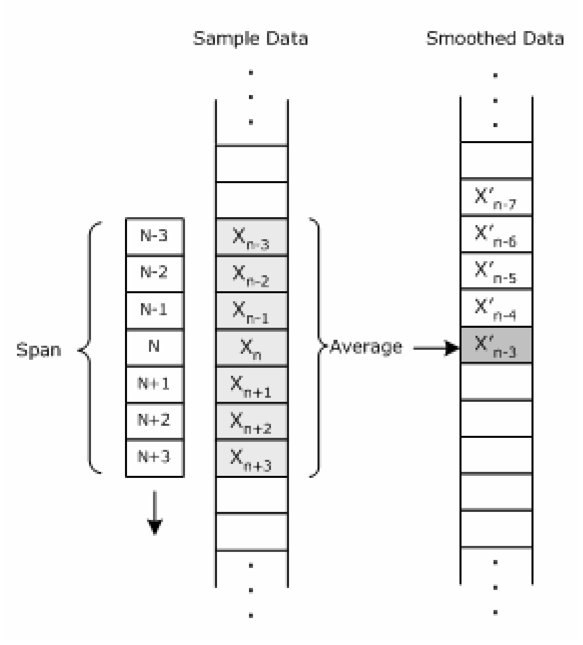


Figure 5.12 Moving Average Filter Algorithm

The Moving-Average Filtering difference equation for the average of *m* samples of data sequence, X_i, is represented as

$$X'_{n-m+1} = \frac{1}{m} \sum_{i=n-m+1}^{n} X_i \tag{5.6}$$

where X'_{n-m+1} is the smoothed value for the *i*th data point.

In this study, the moving window length (span size) is determined as 3. Since the smoothness increases when span size is increased, the smoothed data become insensitive to local changes. Four experiments were conducted to determine the span size. As shown in Figure 5.13, the smoothed data is more sensitive to changes, and it is obvious that the smoothed data more similar to original data when span size is chosen as 3. In these experiments, the absolute error between the original data and smoothed data in X direction is shown in Figure 5.14.

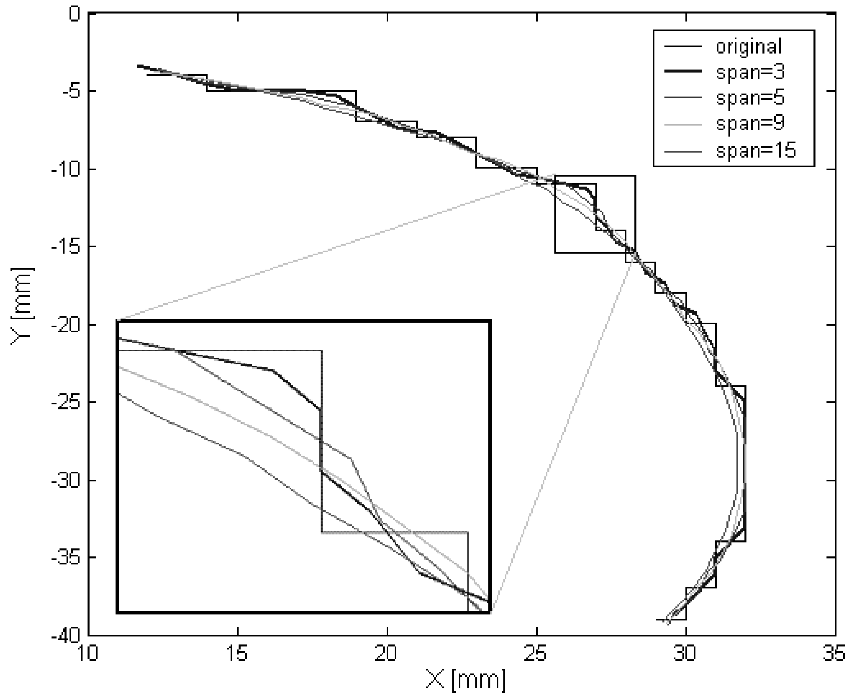


Figure 5.13 Moving-Average Filter Method Results for Different Span Sizes

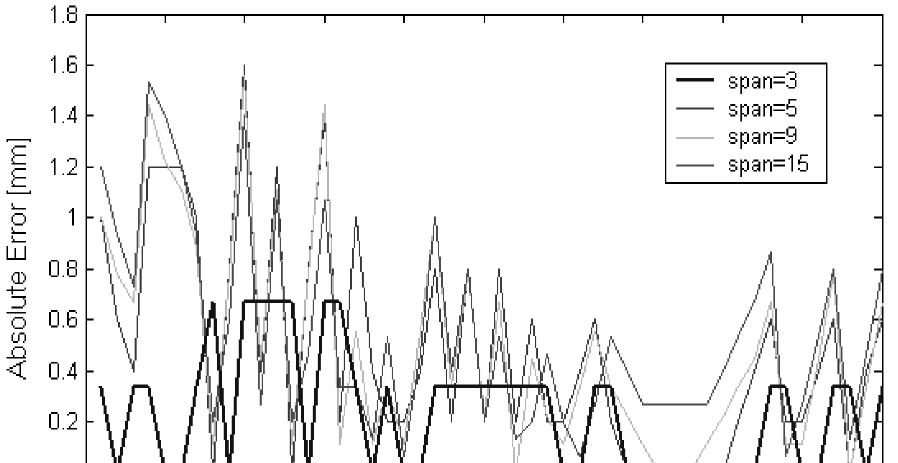


Figure 5.14 Absolute Errors in X Direction for Smoothing Operation

If smoothing operation is performed while controlling the remote robot, the commands are used in smoothing calculation immediately, and then the average values are sent to the robot. In order to test the smoothing operation, two experiments were conducted. Two circles were drawn on a paper previously and the robot tracked the lines with equipped pen to the end-effector. Figure 5.15 shows two drawings of the robot where the drawing on the left was done without smoothing. The drawing carried out on the second experiment shows the result of smoothing. Figure 5.16 shows the data of this drawing where the left one depicts raw data, and the right one depicts the smoothed data.

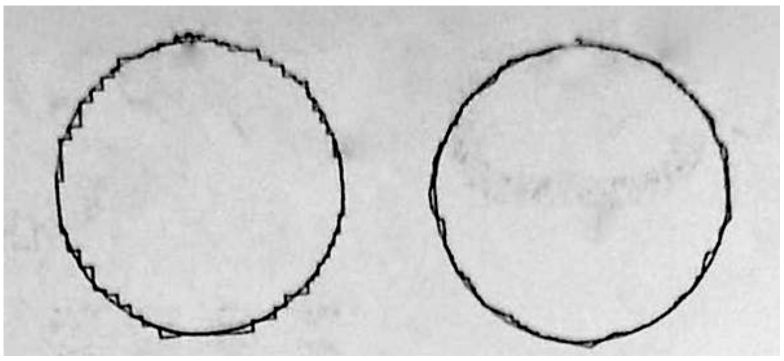


Figure 5.15 Two Drawing Experiment Results; without Smoothing and with Smoothing

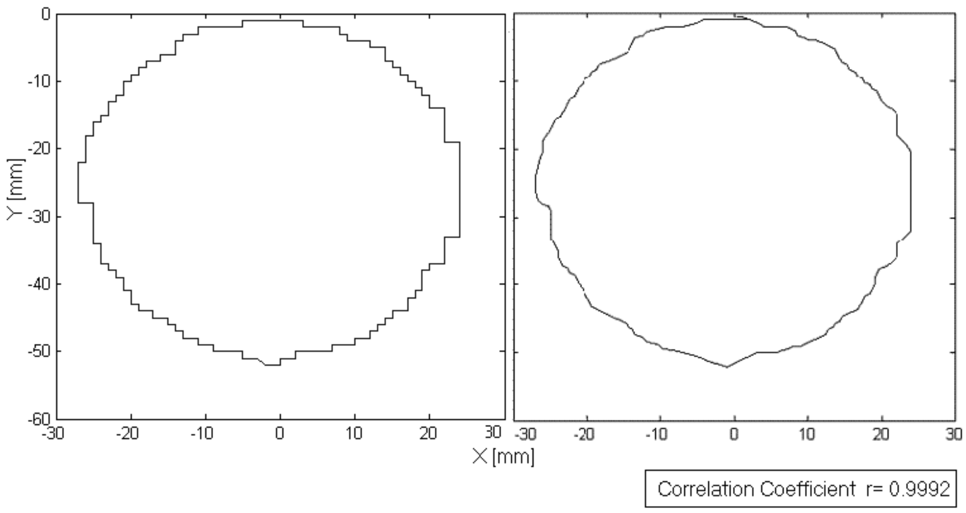


Figure 5.16 Raw and Smoothed Data Graphics

The results show that the using path generator of span size 3 reduces the fluctuations on the path. Chosen span size is the minimum to be chosen in Moving-Average Filtering method. As seen in Figure 5.14, the absolute error results indicate that the minimum loss of information is obtained by using 3 as span size. The path created by the robot is improved as shown in Figure 5.16.

5.7 Experimental Implementation and Results

In order to test the teleoperation system with the proposed control methods some experiments are conducted. It is aimed that the robot to follow a sinusoidal path drawn previously, Figure 5.17. The function of this path is *y(x) =25.sin (0,025.π.x).* Before the operation, this function is graphed by the robot and then the robot takes the start position (x=0). These operations are just implemented for tests. Both control methods are tested for the same path.

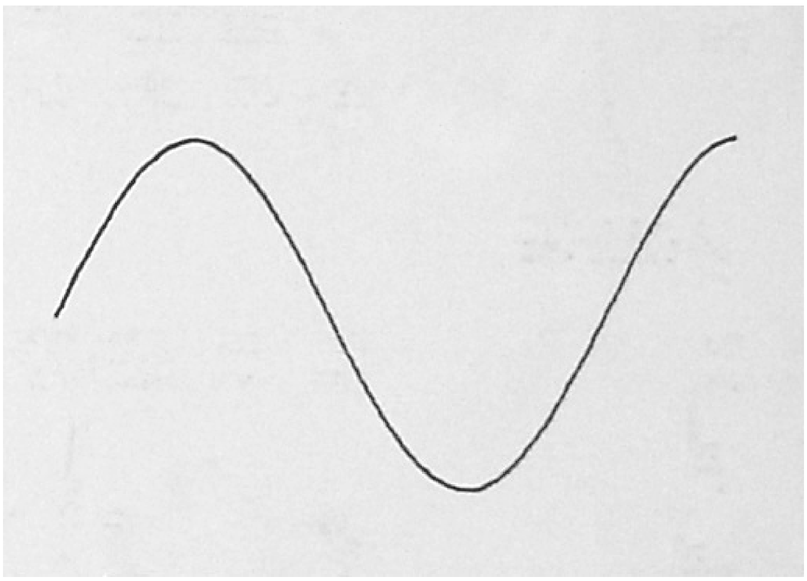


Figure 5.17 The Path Followed by the Robot

Client and server applications are run on the same machine. All commands are sent subjected to the round-trip time delays. RTT is continuously measured during the operation between a remote site and the local host. When a command is generated, it is waited for the instantaneously measured RTT value and then sent to the server application. As for video feedback, the bandwidth allocated to the link between video server and local host is limited by using a bandwidth limiter software. This resulted in delay in the transmission of the video images.

Two different case studies are implemented for both control methods. The following sections describe these case studies. During the tests, generated position coordinates are stored to compare the data constituting the sinusoidal path. For this reason, the correlation between these data is investigated using Pearson's correlation coefficient (*r*). Pearson's correlation coefficient is a measure of the strength of the association between two variables. In order to study the relationship between these two data, the scatter plot is drawn. The scatter plot is used to check whether the data to be related following a straight line. The nearer the scatter of points is to a straight line, the higher the strength of association between the variables. If the relationship is linear then Pearson's correlation coefficient can be calculated. A correlation coefficient is a number between -1 and 1. If there is no relationship between two data sets the correlation coefficient is 0 or very low. If there is positive or negative correlation between data sets the correlation coefficient is ±1 or near the ±1. The Pearson's correlation coefficient is written:

$$r = \frac{\sum_{i=1}^{n}(A_i - \overline{A})\ (B_i - \overline{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \overline{A})^2\ \sum_{i}^{n}(B_i - \overline{B})^2}} \tag{5.7}$$

where *A* and *B* are data. $\overline{A}$ and $\overline{B}$ are the means of *A* and *B*.

5.7.1 Case 1

In this case Sliding Event-Based Control Method is tested. The average Internet time delay is 210 ms, Figure F.1. The resolution value is 0.25 mm. Result of the drawing is shown in Figure 5.18. The scatter plot for original path data and drawn path data is shown in Figure 5.19. The correlation coefficient between both data is also given in Figure 5.19. Figure 5.20 illustrates the error between both data. The error is indicated as the deviation of the drawn data according to original data. This deviation is the shortest distance between generated data and original path curve. The operation is completed in 426 seconds, Figure 5.21.

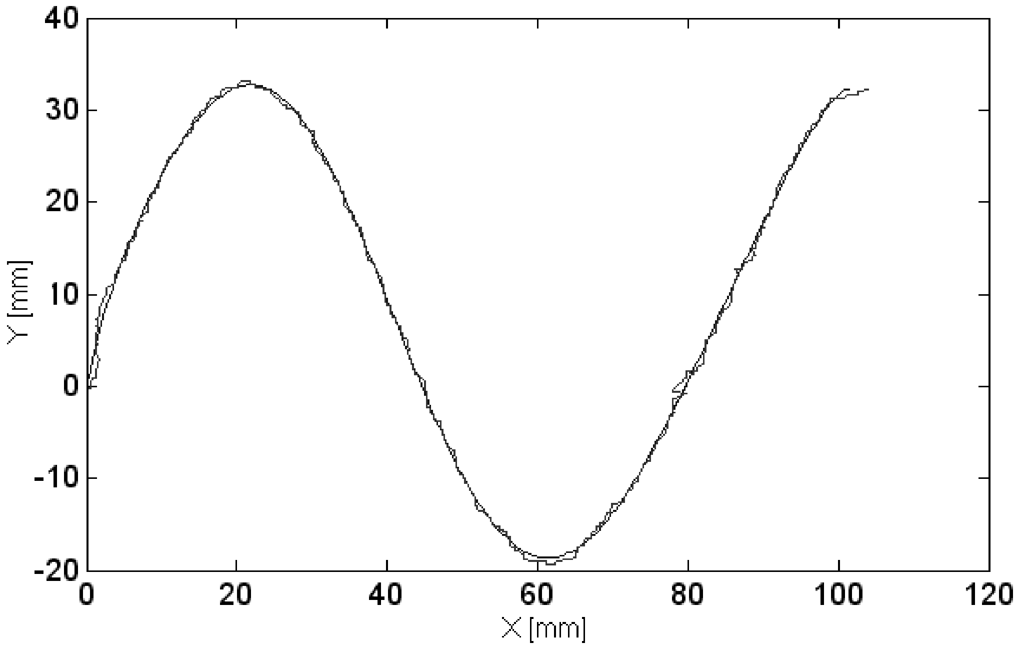


Figure 5.18 Drawing Result

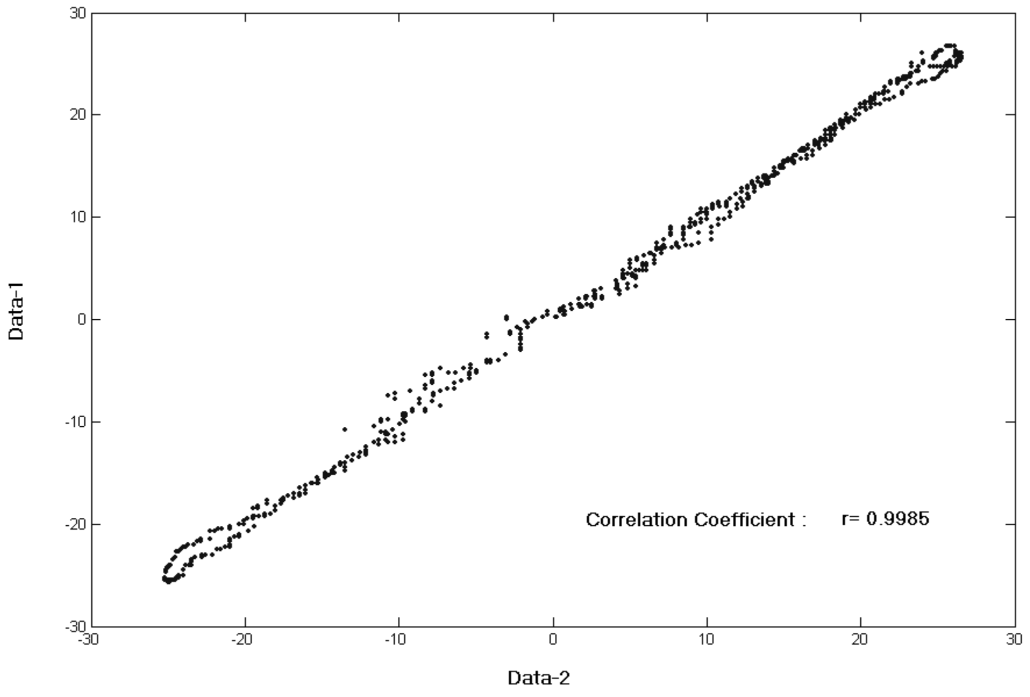


Figure 5.19 Scatter Plot and Correlation Coefficient between Data

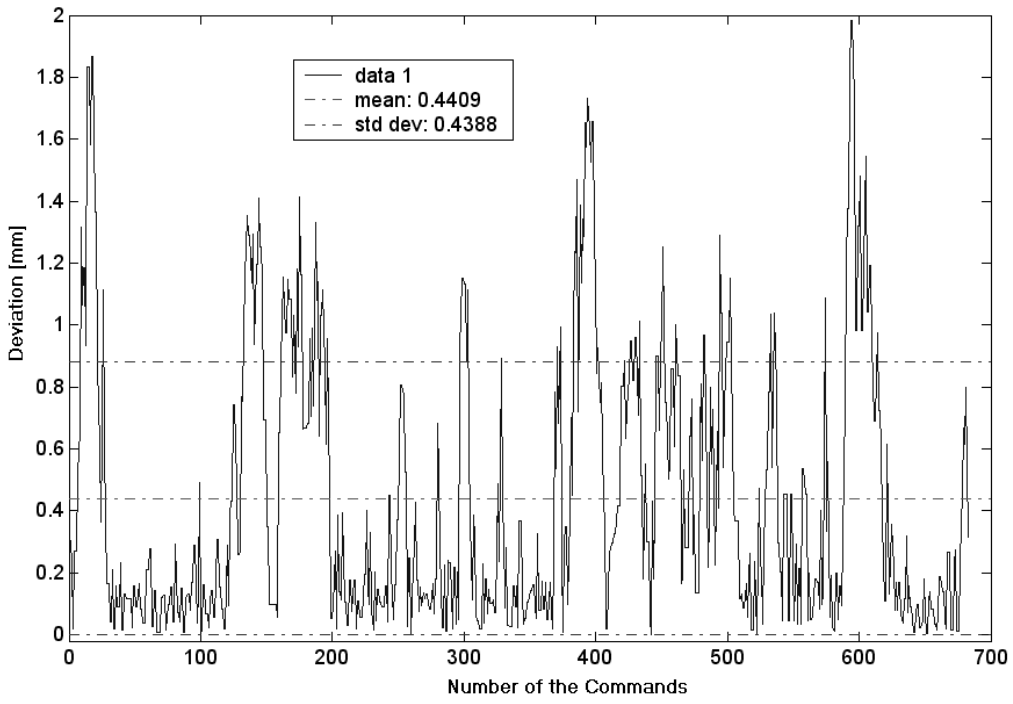


Figure 5.20 Deviation from Original Path

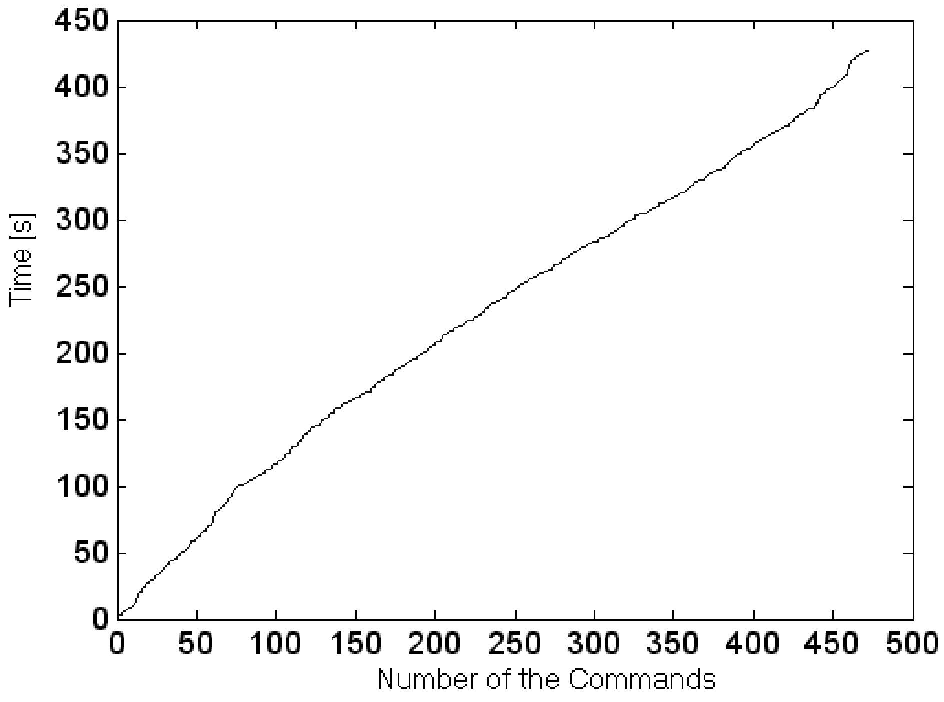


Figure 5.21 Elapsed Time to Follow the Path

5.7.2 Case 2

In this case Sliding Event-Based Control Method is tested. The average Internet time delay is 460 ms, Figure F.3. The resolution value is 0.25 mm. Result of the drawing is shown in Figure 5.22. The scatter plot for original path data and drawn path data is shown in Figure 5.23. The correlation coefficient between both data is also given in Figure 5.23. Figure 5.24 illustrates the error between both data. The error is indicated as the deviation of the drawn data according to original data. This deviation is the shortest distance between generated data and original path curve. The operation is completed in 404 seconds, Figure 5.25.

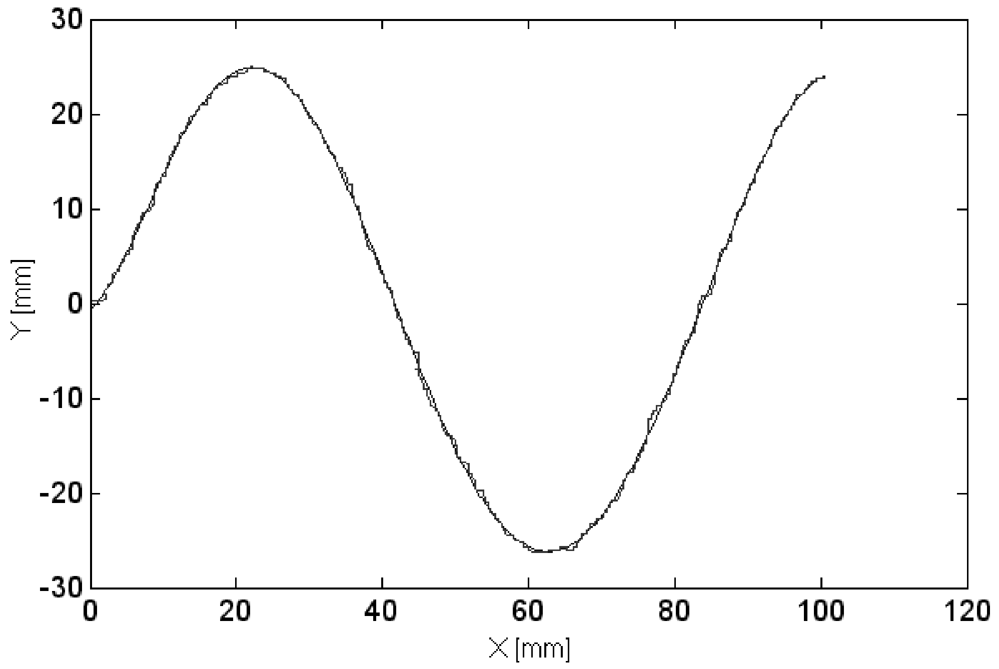


Figure 5.22 Drawing Result

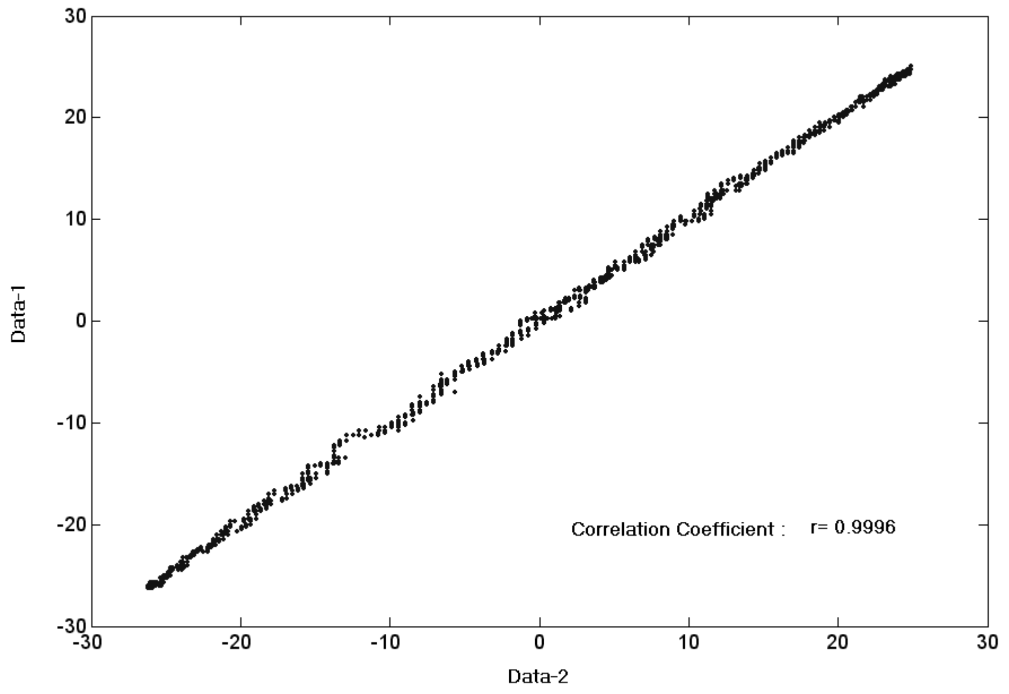


Figure 5.23 Scatter Plot and Correlation Coefficient between Data

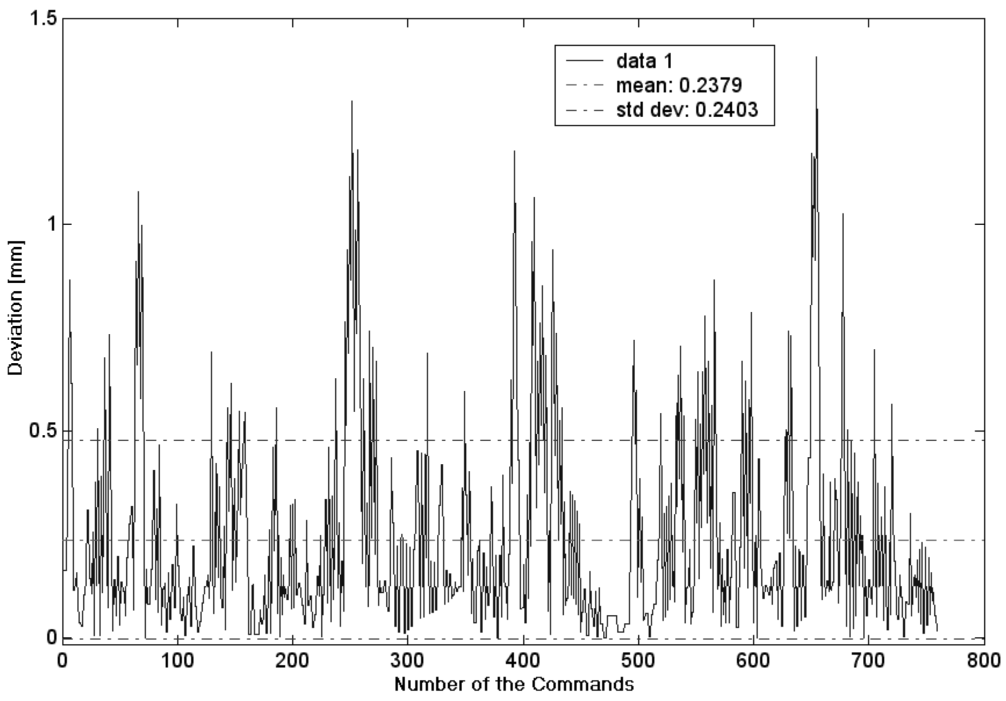


Figure 5.24 Deviation from Original Path

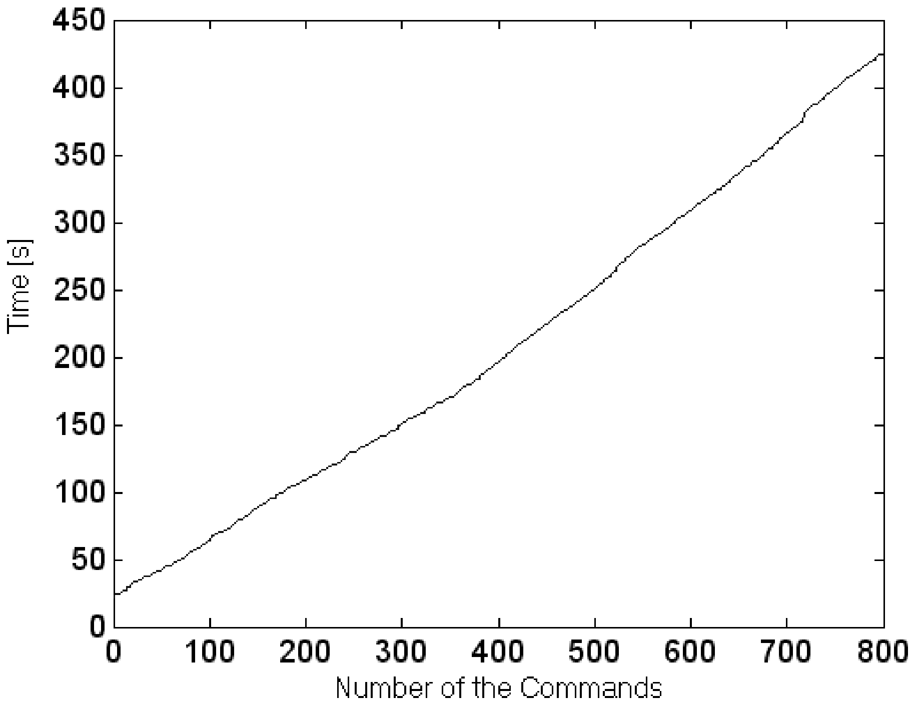


Figure 5.25 Elapsed Time to Follow the Path

5.7.3 Case 3

In this case Simple Event-Based Control Method is tested. The average Internet time delay is 210 ms, Figure F.5. The resolution value is 0.25 mm. Result of the drawing is shown in Figure 5.26. The scatter plot for original path data and drawn path data is shown in Figure 5.27. The correlation coefficient between both data is also given in Figure 5.27. Figure 5.28 illustrates the error between both data. The error is indicated as the deviation of the drawn data according to original data. This deviation is the shortest distance between generated data and original path curve. The operation is completed in 667 seconds, Figure 5.29.

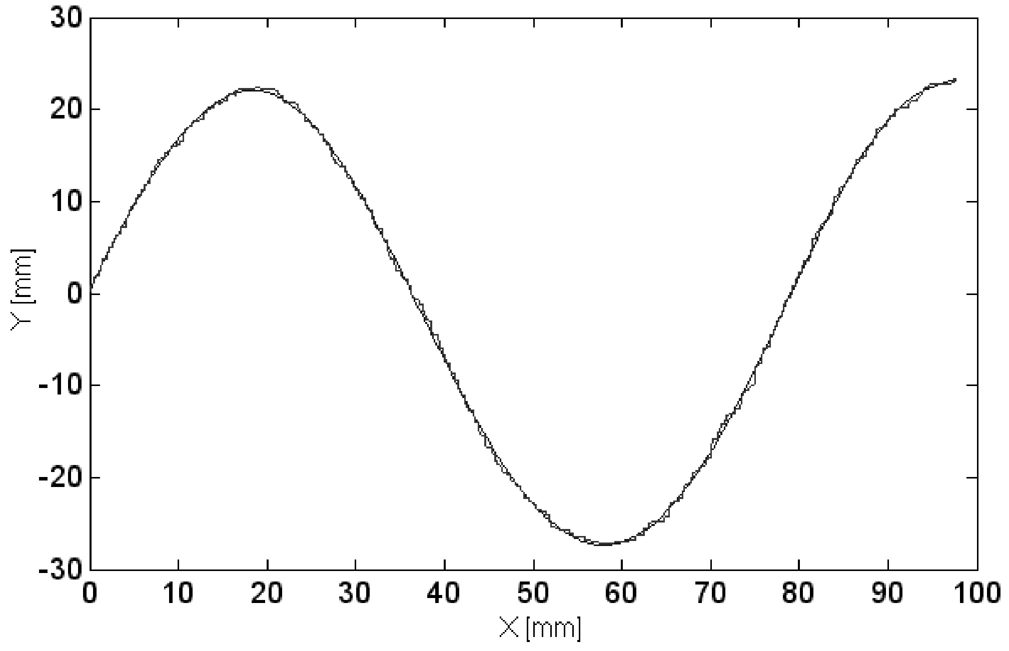


Figure 5.26 Drawing Result

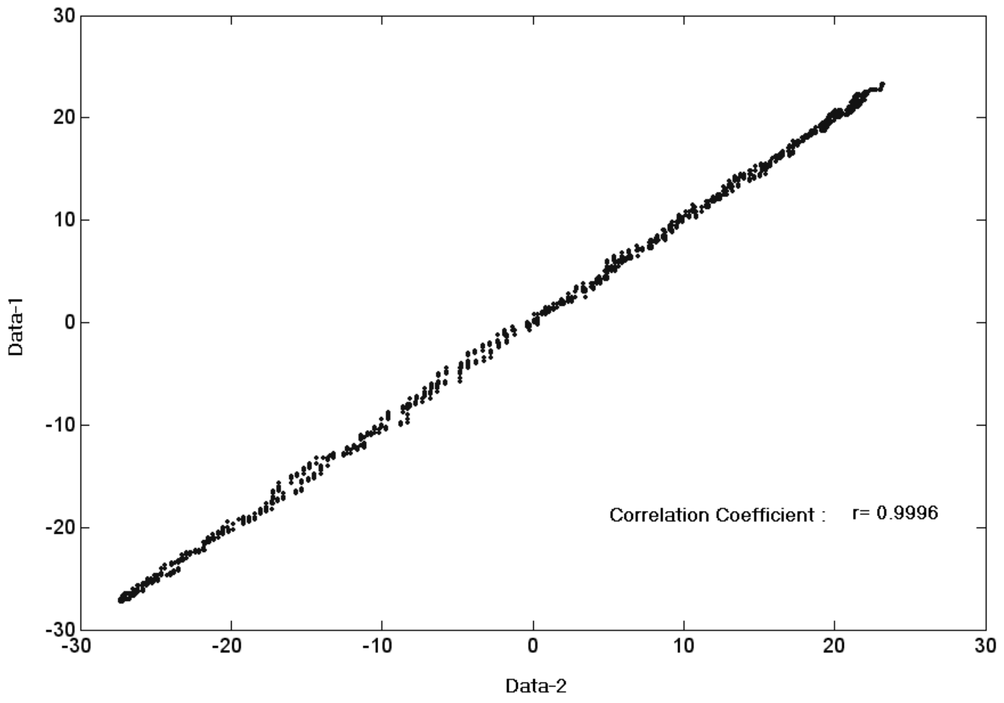


Figure 5.27 Scatter Plot and Correlation Coefficient between Data

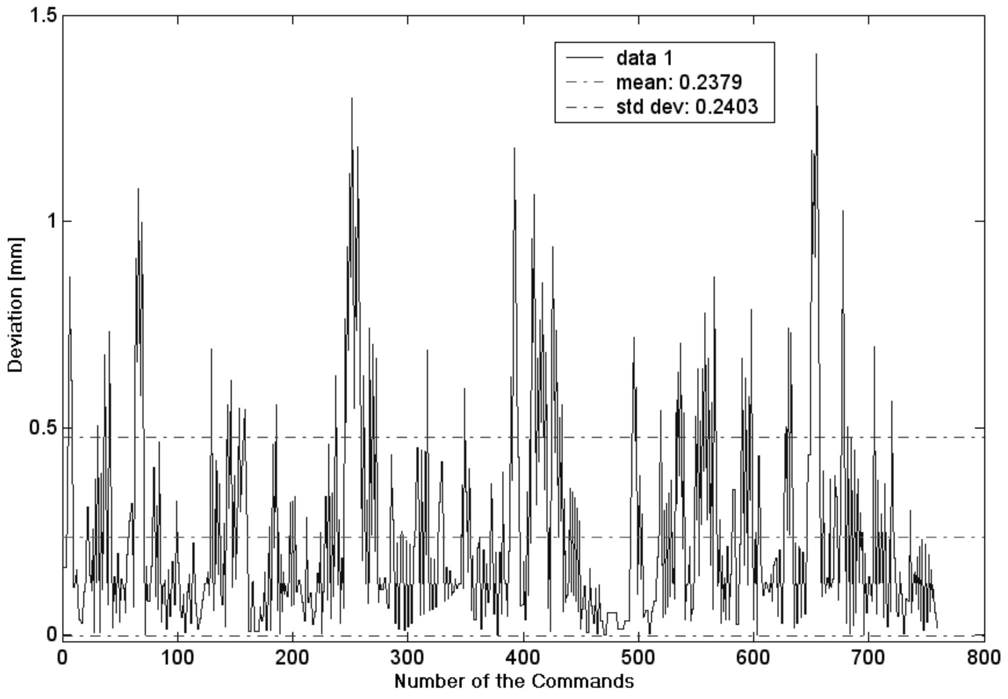


Figure 5.28 Deviation from Original Path

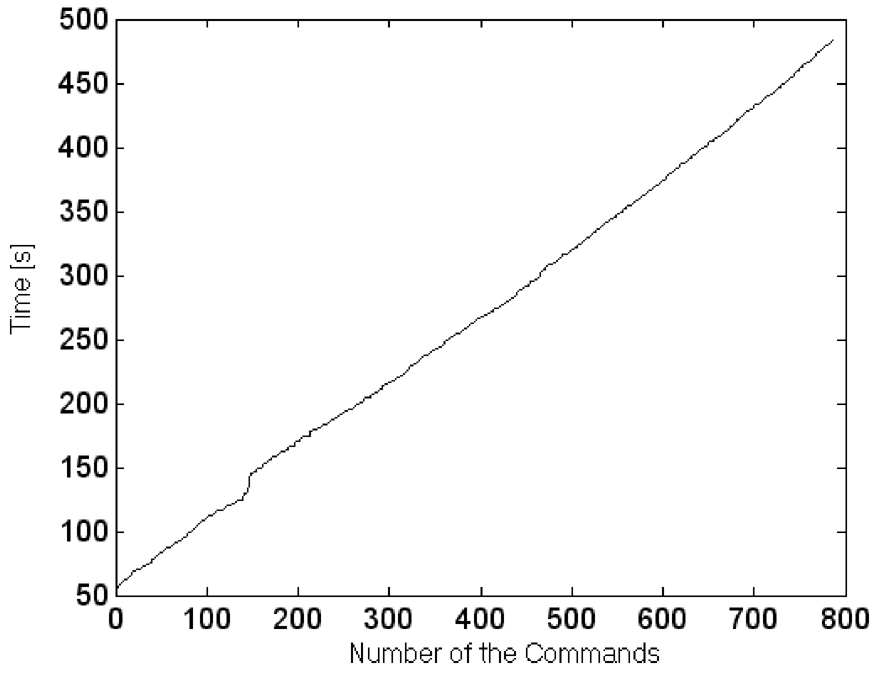


Figure 5.29 Elapsed Time to Follow the Path

5.7.4 Case 4

In this case Simple Event-Based Control Method is tested. The average Internet time delay is 460 ms, Figure F.7. The resolution value is 0.25 mm. Result of the drawing is shown in Figure 5.30. The scatter plot for original path data and drawn path data is shown in Figure 5.31. The correlation coefficient between both data is also given in Figure 5.31. Figure 5.32 illustrates the error between both data. The error is indicated as the deviation of the drawn data according to original data. This deviation is the shortest distance between generated data and original path curve. The operation is completed in 701 seconds, Figure 5.33.

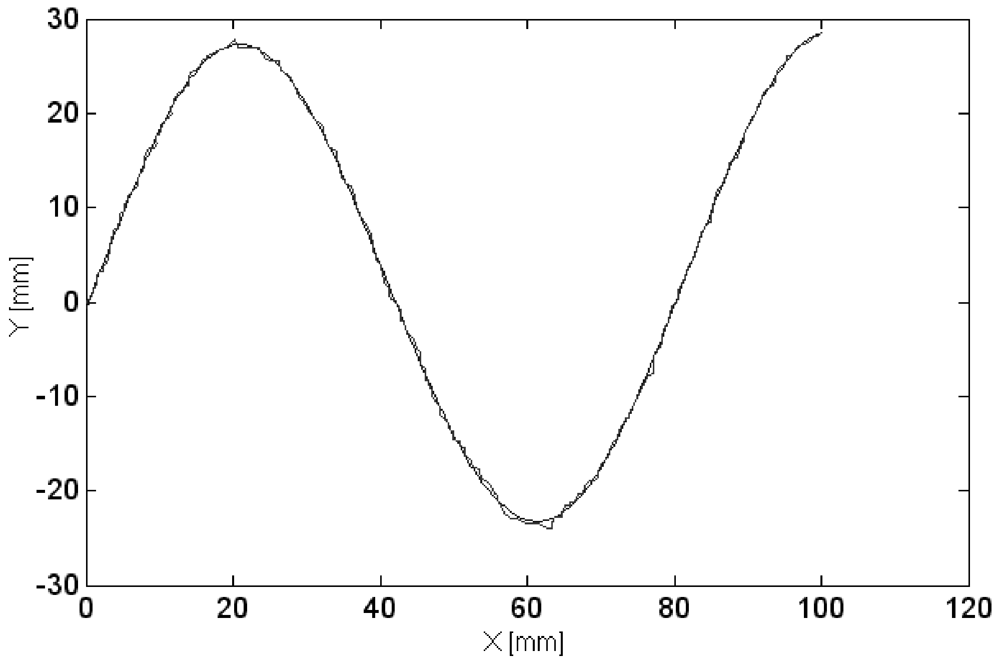


Figure 5.30 Drawing Result

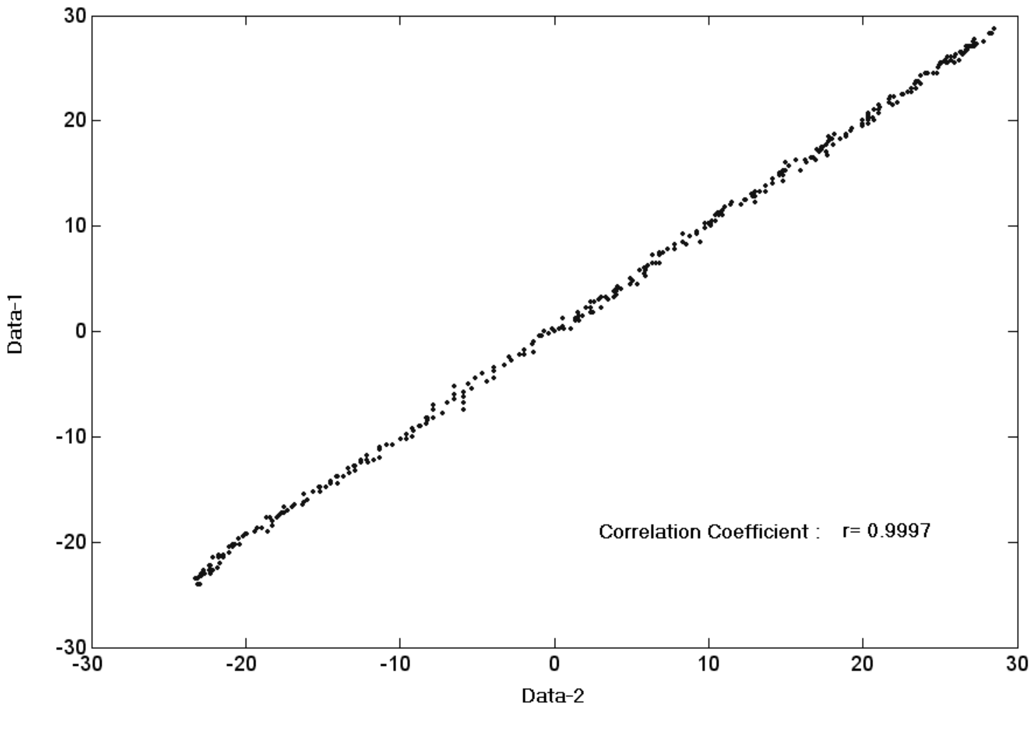


Figure 5.31 Scatter Plot and Correlation Coefficient between Data

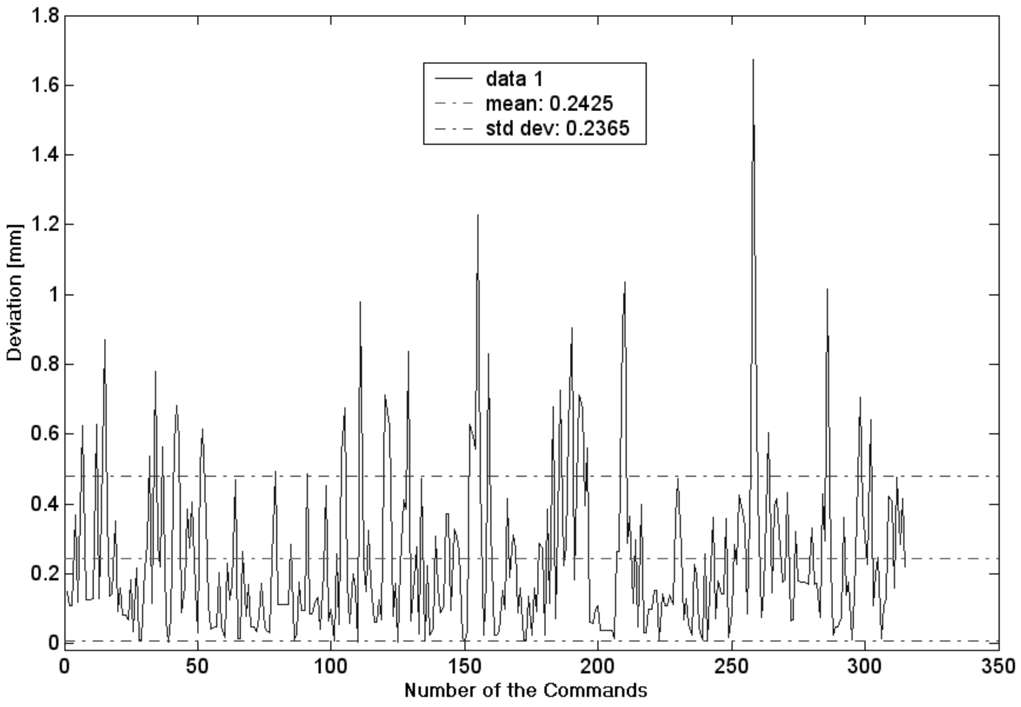


Figure 5.32 Deviation from Original Path

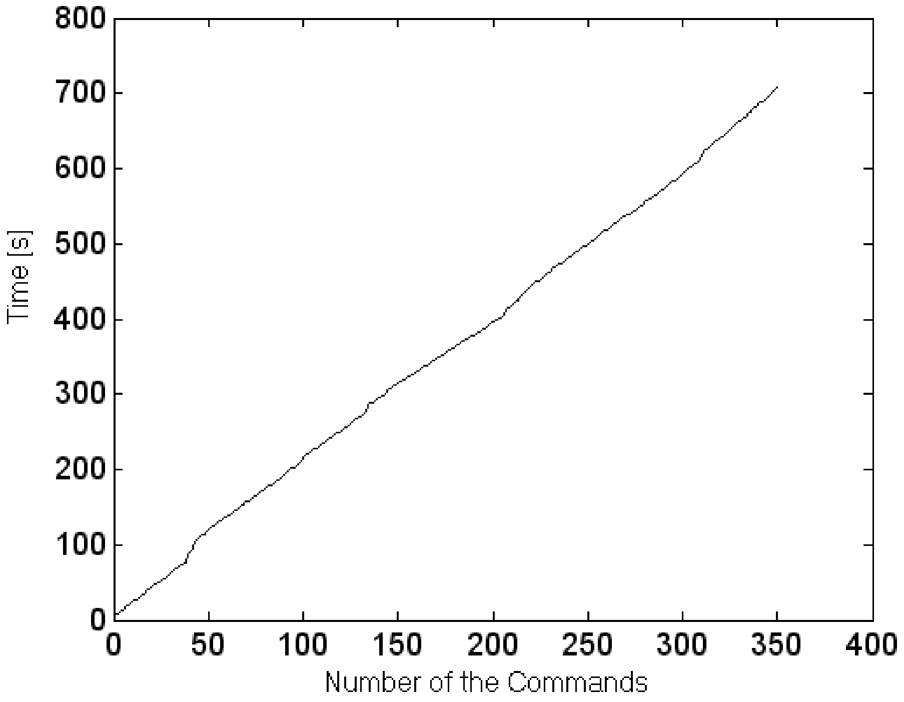


Figure 5.33 Elapsed Time to Follow the Path

5.7.5 Conclusion

In order to test the performance of the teleoperation system, some experiments are carried out. In these experiments, the proposed control methods are tested. The position coordinates are generated by the operator, and they are sent to the robot to follow a path. The video feedback from the drawing environment is fed to the operator back. Also, force-feedback mouse is used to direct the operator’s mouse movements.

Path following accuracy is one of the objects on these experiments. Using event-based control approach, it is not encountered undesired robot movements due to buffering effects of the Internet. Accumulation of the commands in the Internet is not allowed in the Simple Event-Based Control. In the other method, this is possible to some number which is command quota. If the operator obeys the feedbacks of force-feedback mouse, the accumulation risk decreases significantly. Therefore, accumulation of the commands is under control. Examining the presented drawings, the path drawn by the operator is almost the same as the original path. The correlation coefficients reveal this similarity. There are highly significant correlations between the original path data and the drawing data. Since the other object of these experiments is the time, it is possible to obtain more accurate paths through slow and careful command generation. Path smoothing also is used in the operations. As for the operation time, it is observed that the Sliding Event-Based Control method enables to complete the operation faster. According to the implemented experiments, the operations are completed in the Sliding Event-Based Control method 30%-40% faster than the Simple-Event Based method.

It is important to emphasize that the both proposed control methods may not suitable for the same operations. The Sliding Event-Based Control method enables the operator to generate commands according to changing command quota. This flexibility may results in the accumulation of the commands in the Internet. The operator can generate the commands up to the quota without receiving any acknowledgement for previously generated commands. Even though the command generation is restricted with the command quota, the accumulation of the commands may result in undesired movements in sensitive remote operations. Therefore, selection of the method according to the application type is important. For example, undersea applications may be suitable for using the Sliding Event-Based Control method since highly accurate operations are not needed under the water. The Simple Event-Based Control method can be used in aerospace applications since “move and wait” strategy is mostly used in aerospace applications. Also, it can be used in telesurgery due to the need of precise operations.

CHAPTER 6

SOFTWARE DESIGN AND IMPLEMENTATION

This chapter describes the developed software applications. The applications are composed of server and client parts. Server part runs at the robot side and client part runs at the human operator side. The chapter gives information about the programming environment, developed software architecture and functions of the application entities.

6.1 Programming Environment

The software applications are developed by using the Microsoft Visual Basic .NET programming language. Visual Basic .NET is a part of Visual Studio .NET environment which includes other development tools like Visual C# .NET. All Visual Studio development tools share the same Integrated Development Environment (IDE). IDE allows sharing same tools in the creation of mixed language applications. For instance, a 2D charting control object which is developed by using Visual C# .NET is added directly and easily to the Visual Basic .NET project. This facility allows that any component of developed applications can be used in a project to be created by another Visual Studio development tool. Therefore, Visual Studio .NET gives a flexibility to use the developed software applications with other software applications which might be developed in future studies. Visual Studio .NET is built on the top of the .NET Framework which is a programming infrastructure that allows building, running and deploying applications. It simplifies application development and appears as an execution platform and development environment turns the Internet into an operating system. The .NET Framework redistributable package is necessary to run the developed applications in Windows operating system. This package can be downloaded from the Microsoft’s web site [51].

Visual Basic .NET is an Object-Oriented Programming (OOP) language. OOP is a way of thinking about the development of software and is independent of the programming language. OOP provides a better structured, more robust and more easily used code. An object-oriented program consists of a collection of objects, so, everything in Visual Basic .NET can be treated as an object. An object is a software entity of related data and methods. It is a structural unit of the software and provides modular programming facility. The main outcome of the OOP is to simplify the producing quality software.

The developed software applications consist of almost only classes in this study. A class is an object prototype or template used to create new object instances, and it defines the variables common to all objects and how its methods are implemented. Besides the created classes by user, the .NET Framework has a large class library organized within a hierarchy called a *namespace*. At the root of the hierarchy is the *System* namespace. The mostly used object in this thesis is the Windows Form. The Windows Form is the class inherited from the *System.Windows.Forms.Form* namespace. Inheritance is an OOP feature that allows the use of existing classes as bases for creating other classes.

The .NET Framework provides multithreading programming. Each application is divided into many processes and each process consists of many threads. A thread is a basic unit to which an operating system allocates the processor time. All applications running in a computer are divided into many threads and these threads are executed one by one. Each time the processor can run only a program, that means a program’s thread is processed for a definite allocated processor time, then the other program’s thread is processed and so on. A multitasking operating system creates the illusion two or more program are running at the same time. All running applications have a main thread. While developing the software if an extra thread (worker thread) is added, this thread holds a definite CPU time when the program runs. Thus, using worker threads increases the responsiveness of the program, and process the data in order to execute the task at almost the same time. This facility is used in the developing software to increase the performance of the program.

Visual Basic .NET enables creating asynchronous applications in which program tasks execute sequentially. An asynchronous socket is one of these facilities. Using this type of socket and related programming methods for network programming provides that the application is not suspended while waiting for the network operations to complete. Asynchronous client socket and asynchronous server socket are used to make the developed software more responsive.

The objects constitute the software applications and their functions are discussed in the following sections. These applications are client and server applications named as *RoboClient* and *RoboServer*, respectively.

6.2 Software Structure and Interfaces

Software architecture can be divided into two parts: server and client. Both softwares are different and separate applications. These applications are described in the following paragraphs.

Client application is a graphical user interface and composed of modules illustrated in Figure 6.1. The human operator can define application tasks as a sequence of single position commands through Drawing Board or Graphical Joystick Interfaces. Commands are generated by these interfaces and sent to the client through communication class. Communication between client and server is established via a developed communication module using Transmission

Control Protocol. This class handles converting the messages to bytes, sending them, converting the incoming messages to string of characters, and firing an event to notify the application about received message.

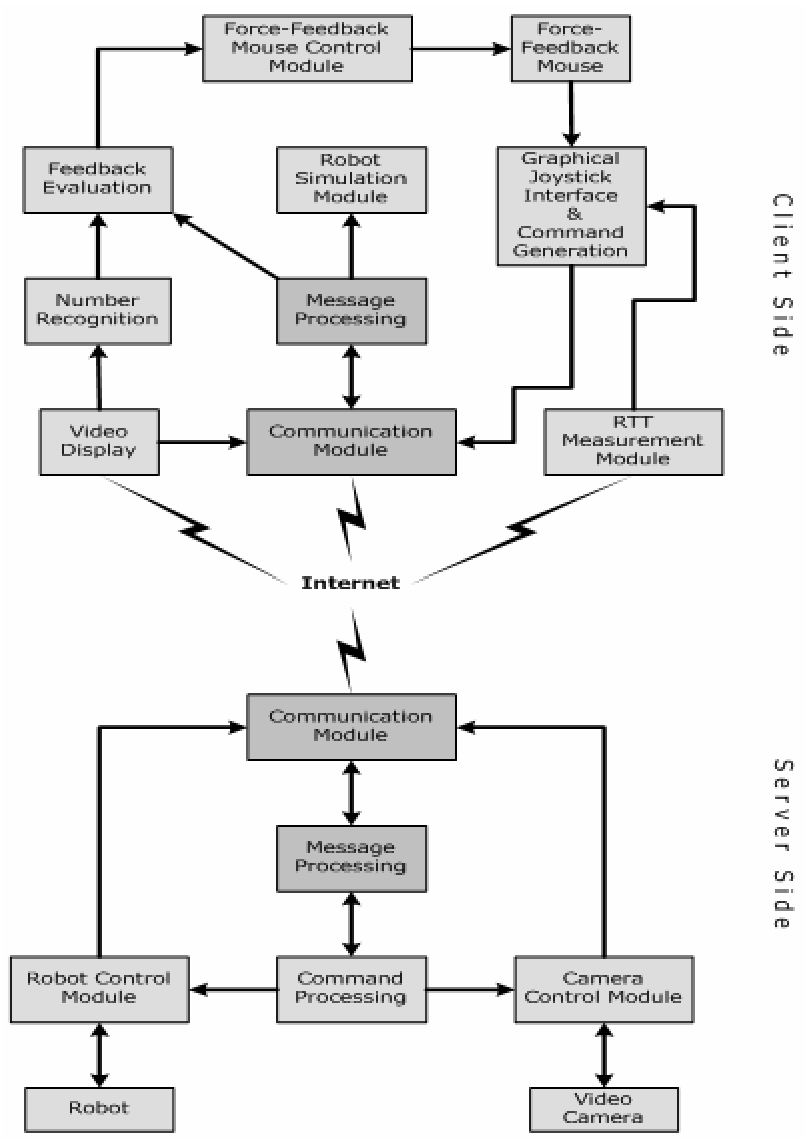


Figure 6.1 Software Architecture

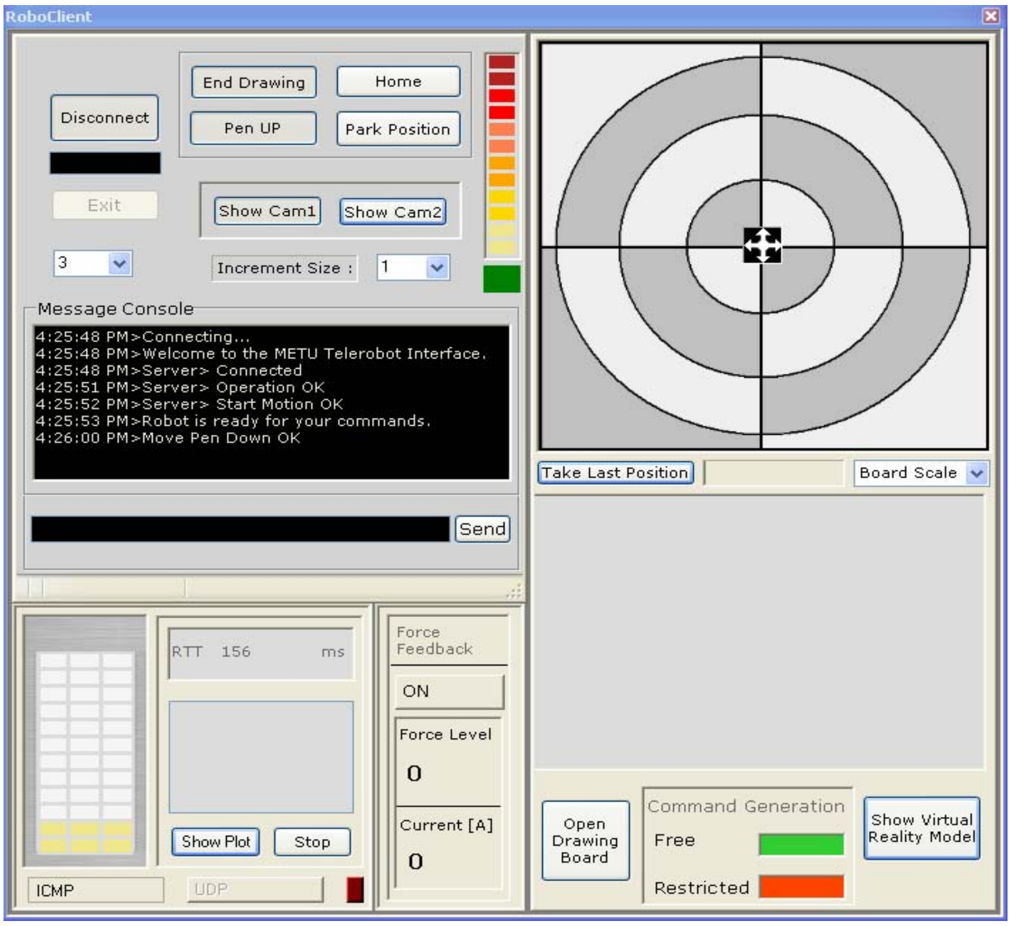
Round-trip time measurement element measures the time delay between client and server side by means of the Ping utility. These RTT values are used in control system. It is also possible to view the dynamic plot of the time delay.

The captured video images taken from the robot environment by two cameras are displayed to the operator in two separate windows on the client side. These windows include some facilities to control the cameras remotely. Video images include the time information which is added to every frame by video server. This time information is important since it designates the execution time of the assigned command. The time information is extracted from the images by applying image processing techniques. The numbers are recognized by the application continuously. The extracted time information is compared with the time information of the positive acknowledgement message sent from the server after a successful execution. Evaluation of this comparison is used in command generation process to stop or to limit the command generation applying force on the operator via force-feedback mouse. A class related with the force-feedback mouse works between the other processes of the software application and the serially connected mouse.

Robot simulation module displays the simulation of executed commands by means of graphical model of the robot. The Cartesian coordinates are sent immediately to the robot as soon as they are generated.

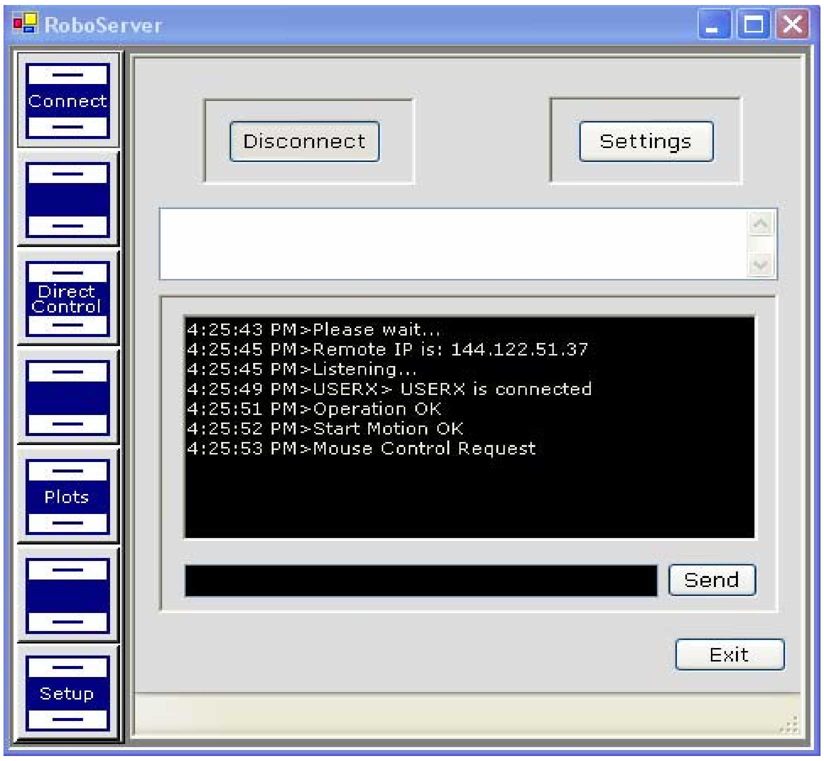
Server application also includes the same communication module. The commands received from the client are transmitted to the robot or cameras. The server application receives commands, executes them, and sends results back to the client. A Path Generation algorithm enables to generate the commands which are lost or not received in spite of the next command is received. The path formed by the human operator sending the position data can be smoothed through Path Smoothing algorithm.

The developed interfaces, *RoboClient* and *RoboServer* are shown in Figure 6.2. When both applications run, these default views are shown. Other developed components can be run through these interfaces. Following sections are related with these interfaces and the other components.



```
4:25:48 PM>Connecting...
4:25:48 PM>Welcome to the METU Telerobot Interface.
4:25:48 PM>Server> Connected
4:25:51 PM>Server> Operation OK
4:25:52 PM>Server> Start Motion OK
4:25:53 PM>Robot is ready for your commands.
4:26:00 PM>Move Pen Down OK
```

RTT 156 ms



```
4:25:43 PM>Please wait...
4:25:45 PM>Remote IP is: 144.122.51.37
4:25:45 PM>Listening...
4:25:49 PM>USERX> USERX is connected
4:25:51 PM>Operation OK
4:25:52 PM>Start Motion OK
4:25:53 PM>Mouse Control Request
```

Figure 6.2 *RoboClient* and *RoboServer* Interfaces

6.2.1 Command Generation

In order to generate the maneuvering commands two different command generation interfaces are developed. The idea of designing two different command generation means is giving the operator options considering the usability of a command generation means.

As stated before, the tasks are executed by the interaction of master and slave systems in some teleoperation systems. This interaction can be described by considering the teleoperation system is an interface between master and slave systems. The overall system is interfaced on one side with a human operator, and on the other side with the environment. In this manner, the command generation interface and force-feedback mouse cooperation represents the master system.

frmBoard can be accepted as the most important form of the developed software since the commands are generated through this form. *frmBoard* includes a Graphical Joystick and Virtual Paper. They can be seen in Figure 6.3. Also, the Drawing Board can be open through this form.

Graphical joystick has a pointer which can be moved by a mouse. Motion inputs are generated while moving the pointer in the joystick area. Each step of the pointer corresponds to a movement of the robot. The movement resolution of the robot can be set by the user in operation time. A combo box control on the *frmBase* allows setting the resolution value. The resolution value is 1mm, by default. That means each increment generated by mouse is multiplied by 1. The range of the resolution changes from 0.25 mm to 5 mm.

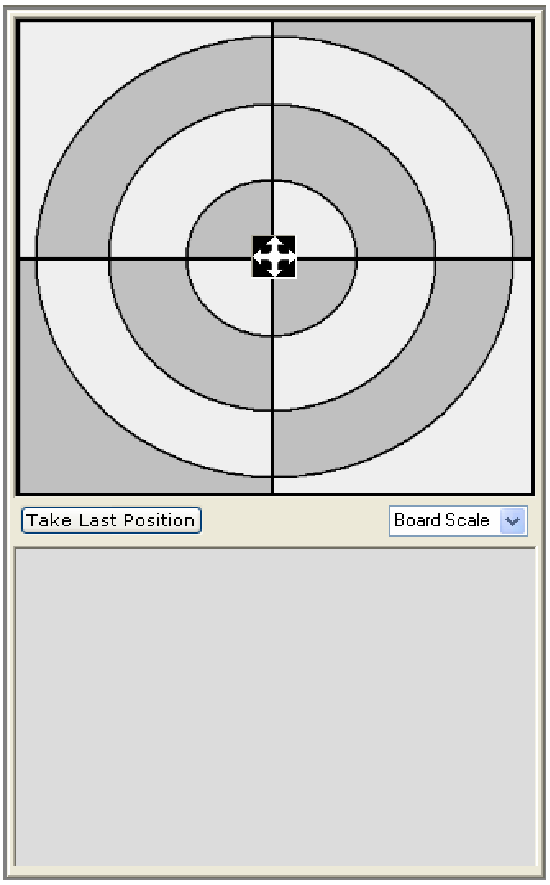


Figure 6.3 Graphical Joystick and Virtual Paper

The commands are increments on X and Y plane and sent to the robot side to transfer them to the robot as position coordinates. The joystick area corresponds to X-Y plane of the robot's workspace. The joystick area is scaled so that all the values generated are in -10 and 10 ranges by default. The range can be changed by the operator. Farther the distance of the pointer to the center of the joystick area, the larger step sizes are generated. At the farthest point on the joystick area, generated step size is ±10, by default.

The pointer is moved by mouse holding down the left button of the mouse. When the left button is released the pointer returns the center of the joystick area and increments are initialized immediately. Graphical Joystick application allows the operator to move the pointer to a desired position without generating any command by holding down the right button of the mouse. Command generation can be started from this point. While moving the pointer to generate increments, if the pointer slips unintentionally, a button presents at the *frmBoard* provides to take the pointer last position.

The Drawing Board is another command generation means. Whenever the user clicks somewhere on the Drawing Board the tip point of the robot moves this coordinate. The position of the pointer corresponds to the position of the tip point. The commands are not increments as in Graphical Joystick; instead, they are position coordinates. The resolution value can be set as described for Graphical Joystick. The Drawing Board is shown in Figure 6.4.

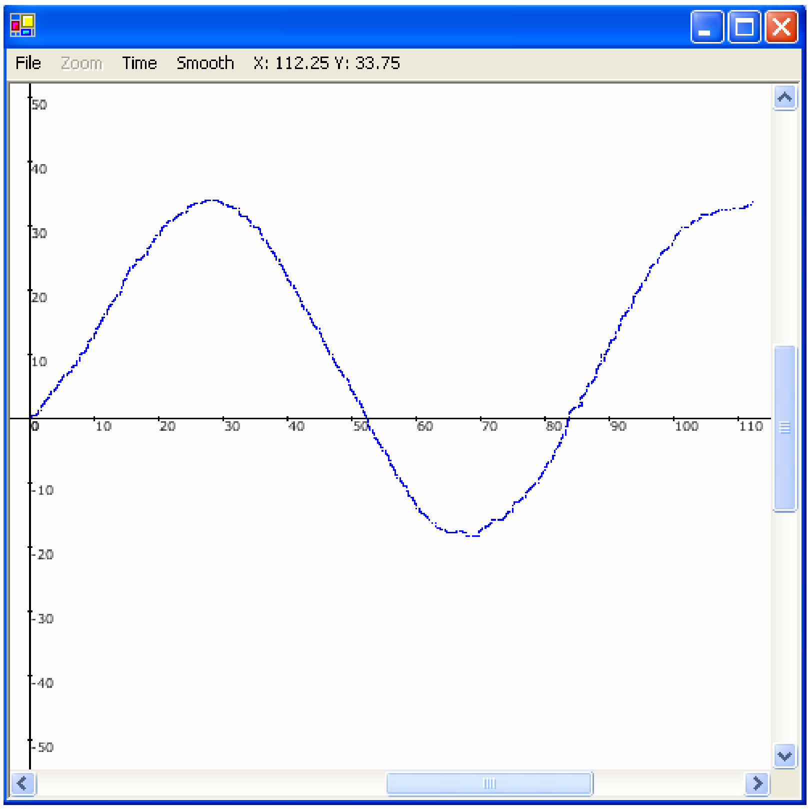


Figure 6.4 Drawing Board

6.2.2 Virtual Paper

Virtual Paper component of the *frmBoard* shows the operator movement of the mouse. The path generated by mouse is sketched on it. This path is obviously the path followed by the robot. Instead of drawing the position of the robot's tip point by using acknowledged commands, they are drawn immediately while they are being generated. Virtual Paper helps the human operator prepare for the next movement.

6.2.3 Delay Measurement

The measurement operation is implemented by *frmMeasurement* which is a form object. *frmMeasurement* is shown in Figure 6.5. This form includes a user control, *NetMeasure*. *NetMeasure* is designed to measure the RTT between client and server side. When a communication is established between client and server this control starts to measure RTT time. The measured value is held by a variable which can be accessed by *frmBoard* form.

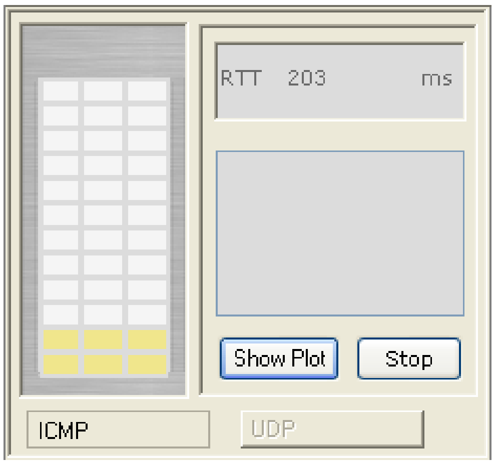


Figure 6.5 *frmMeasurement* Form Object

NetMeasure control includes two user controls: *Vumetre* and *ScatterChart*. *Vumetre* is a graphical level indicator application used to display the operator RTT value. Displaying the RTT on level indicator, the human operator can be informed intuitively about the Internet time delay. The RTT is simply scaled by 100 and this scale can be changed easily in the program. *ScatterChart*, which is developed by Mr. Paul M. Seabury, provides 2D charting ability. If it is needed, the measurements can be viewed by this chart which has dynamically refreshed view, Figure 6.6. RTT measurement is implemented in two ways: using Ping utility or probing UDP packets. The way of measurement is optional. In case of UDP based measurement, a reflector application must have been launched on the *RoboServer* application before starting the measurement. Simply, UDP packets are sent to the server side sequentially giving each packet a sequence number. They are reflected immediately to the client side by means of *frmUDP*. The elapsed time is calculated for each acknowledged packet. In order to implement the ICMP based measurement *CPinger* class is used. When an ICMP packet is sent the timer starts and elapsed time is held until the packet is acknowledged. The measured RTT value is displayed to the operator, scaled for using in level indicator, and held in an array so as to plot when needed.

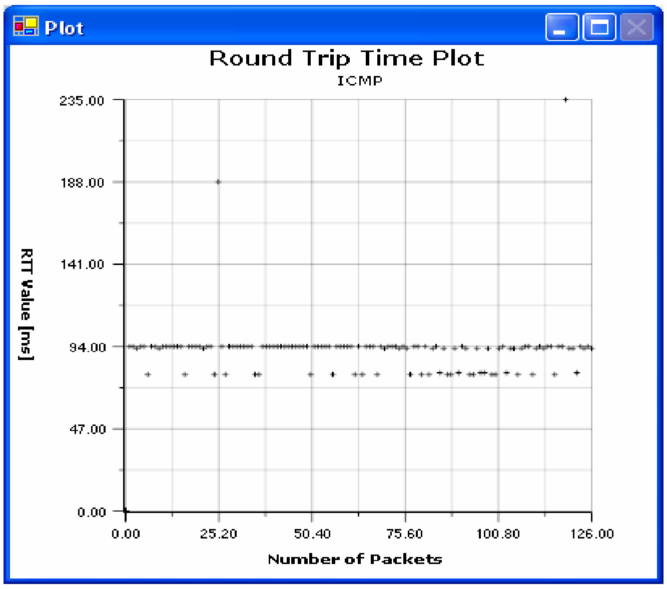


Figure 6.6 Round-Trip Time Plot Form Object

6.2.4 Communication

The communication is established between client and server through *nCom* class. *nCom* provides both client and server functions, so it can be used alternatively. Windows Sockets are used to establish the communication. “The *System.Net.Sockets* namespace provides a managed implementation of the Windows Sockets (Winsock) interface for developers who need to tightly control access to the network. Windows Sockets are based on the UNIX sockets implementation in the Berkeley Software Distribution (BSD, release 4.3) from the University of California at Berkeley” [52]. The Socket class provides many functions for network communication.

As discussed in Chapter 3, the communication protocol is TCP. Since TCP is connection-oriented protocol, before transmitting the data, the connection must be opened between client and server. Therefore, *RoboServer* starts to listen to the connection when pressed a button on its *frmBase* form. *frmBase* is an important form of the software since it is a base for other objects of the developed software. Both client and server sides include a form named *frmBase*, Figure 6.7. They are not identical but they function similarly. *nCom* is inherited to *frmBase* and so, *frmBase* is used to call the procedures of *nCom* when needed to send or receive data. The connection is checked when client attempt to communicate. If server listens to the connection, it is allowed to pass the next step to establish the communication. If server is not ready, operator is warned and application is not allowed to continue. When connection is shut down by remote side, the other side is warned and connection is also shut down automatically on this side.

The data are sent converting them into sequence of bytes. All data coming from remote side are processed in *frmBase*. When data arrives, they are converted to string of characters before processing it. These conversions are carried out by *nCom* object. Each arrival of the data causes the program to call a procedure to process the data immediately.

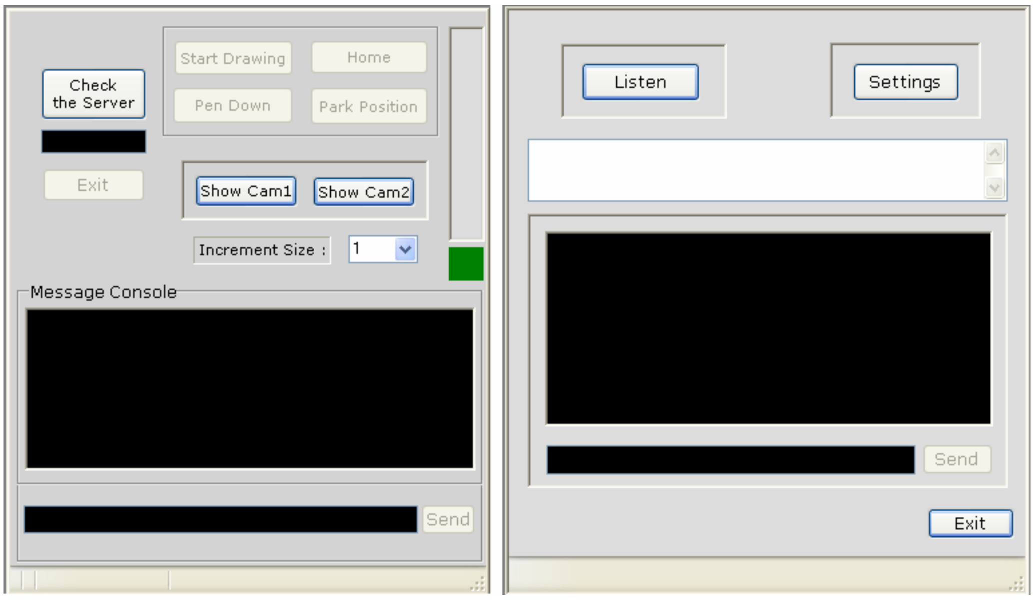


Figure 6.7 *frmBase* of the Client Interface and *frmBase* of the Server Interface

frmBase allows the user sending chat messages to the remote application and receiving chat messages. Incoming messages are shown in Message Console. Connection status and the robot's condition are also shown by Message Console.

6.2.5 Force-Feedback

frmFFM is designed to establish the communication between force-feedback mouse and the client application. The communication is established using the Microsoft Comm Control object which exists in the Visual Studio environment. It provides serial communication by allowing the transmission and reception of data through a serial port. If on/off button, which is on the *frmFFM*, is not pressed to open the communication, messages are not sent to the microcontroller. This allows the operator to use an ordinary mouse.

frmFFM also informs the user about the force level. The functions in the *frmFFM* can be called by other forms to send the voltage level value to the microcontroller. The voltage value is to drive the electromagnet is between 0 and 12 volts. How much voltage will be produced is calculated through the force level value sent from the *frmFFM*. The *frmFFM* form object is shown in Figure 6.8.

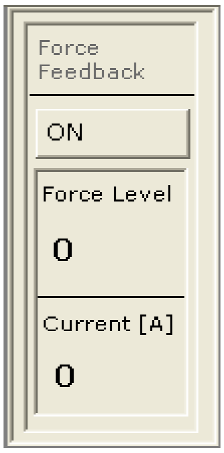


Figure 6.8 *frmFFM* Form Object

6.2.6 Video Display and Camera Control

There are two forms related with video feedback, *frmCam1* and *frmCam2*. Both forms are designed for displaying the video images and sending control commands for pan-tilt, zoom and focus operations to the server application. They are identical applications except for little differences related with settings for the camera being used. Differing from *frmCam2*, *frmCam1* includes the number recognition feature. Therefore, one of these applications, *frmCam1*, is discussed. Both applications are not shown when the client application is launched. They can be opened and hidden through the buttons on *frmBase*.

The video feedback application has two parts: client and server. Client part is a graphical user interface and the screen-shots of *frmCam1* and *frmCam2*, are shown in Figure 6.9. Video streams from Axis video server can be viewed using Axis Camera Control ActiveX component. The Axis Camera Control object can be used for showing live images, to capture and save the images, and to capture, save and play audio from camera. This control is just used for viewing the live video. As for server part, a class, *cCam12*, is designed to handle the control operations for both cameras.

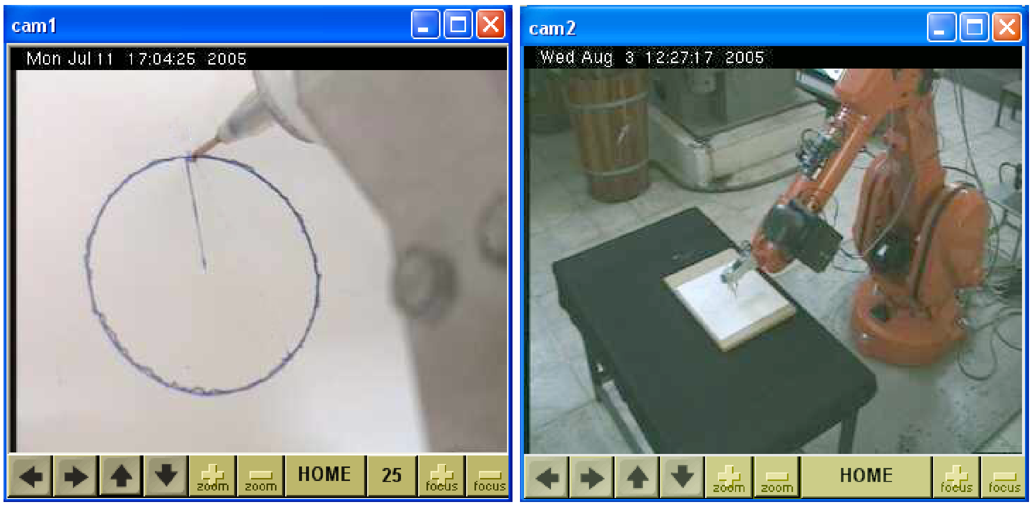


Figure 6.9 GUI's of Video Displays (frmCam1 & frmCam2)

Pan-Tilt commands can be generated in two ways: clicking the arrow buttons, which is straightforward, or clicking somewhere on image. When user clicks a button, the related string of characters is sent to the server site to carry out the corresponding operation. When clicked on image, the point clicked is assigned as the middle point of the image. Displacements between current coordinate and assigned coordinate are calculated. As far as displacements are calculated, the related string of characters and displacements are sent to the server. All control commands has a unique string of characters. For instance, "_LE1" characters represent the pan operation towards left for Camera-1. The amount of motion is determined in server part in design-time, so, every button click corresponds to a certain amount of movement. Strings of characters are sent to the server through developed *nCom* communication module.

The commands coming from client are received through *nCom* module in server part. They are evaluated to run the related function. For instance, *LEFT1* function includes the command for pan operation towards left. When *LEFT1* function is called, byte sequence is constructed and sent to the Camera-1 through serial port communication class. This byte sequence is formed according to VISCA protocol as described in Chapter 4. It can be noticed that there is not a displacement or position instruction. The command packet just includes the pan-tilt velocities. All camera functions are operated for a certain pre-determined interval. After sending the command to the camera, function waits for this time interval. The camera moves for this time interval until the stop command is sent through *CamSTOP1* function. Time dependent operation gives flexibility to the control since time interval can be changed easily.

When the operator clicks the somewhere on the image in video display, the distances between this point and middle point are calculated as horizontal and vertical displacements. When these displacements are received *RelativeM1* procedure is implemented. The displacements are extracted from the incoming packet, and sequence of bytes is prepared using this information. Relative motion is carried out by Camera-1 after sending the command packet. This motion is not carried out as in the other pan/tilt operations. The camera moves to the position specified in the command packet, and stops automatically.

6.2.7 Number Recognition

To obtain the record time of a video image some image processing techniques are applied to video images. When video camera signals are digitized by video server, a time stamp is added to the image. The fact that time stamp is a part of image makes the use of image processing necessary. An instance of the time stamp is shown in Figure 6.10. In order to extract the second number, it must be recognized by computer. This is achieved by finding the locations of digits, extracting the images of figures in these digits, and comparing them with each element of figure set.

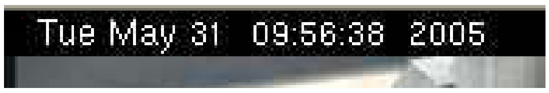

Tue May 31 09:56:38 2005

Figure 6.10 Sample of a Time Stamp Added by the Video Server

The locations of the digits are not stationary. Change of the preceding character widths before clock digits shifts the locations of all digits. Everyday the locations change and the clock also changes every second. Since the distance between a colon sign and second digits is fixed, the first colon sign was chosen as a base location.

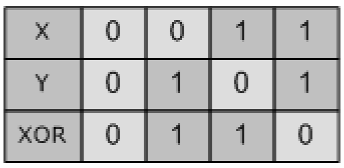
When *frmCam1* is opened, a rectangle region which starts from the left edge of the video image is captured as a bitmap image. The bitmap image is converted to the binary image, which consists of only 1 and 0 values, by applying thresholding. Thresholding is one of the most commonly used image segmentation methods. An intensity value is defined as threshold value and intensity values of the each pixels of the whole image are compared with threshold value. The pixels having greater value than this threshold are accepted as the object's pixels and assigned to "1". The other pixels having less value than the threshold value are accepted as the background's pixels and assigned to "0". This operation can be formulated as:

$$I_T(i, j) = \begin{cases} 1 & \text{if } I(i, j) \leq T \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where I_T (i, j) is the intensity values of the thresholded image (binary image), I(i, j) is intensity values of the original image, T is the threshold value, and (i, j) is the pixel coordinates.

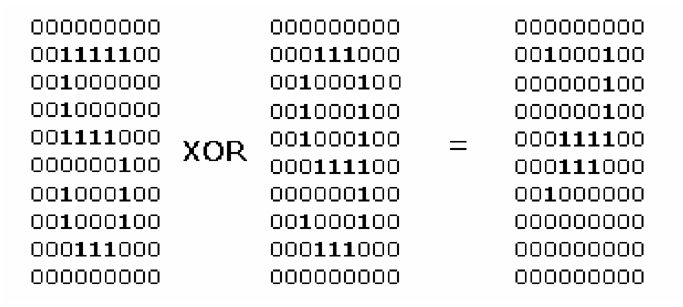
On the other hand, all figures appear in the time stamp are captured and converted to binary images. These all binary images are held in a text file. When *frmCam1* is initialized, this text file is read and assigned to a 3D matrix, named *Nmbr* matrix. *Nmbr* matrix holds the binary values of the numbers from 0 to 9 and is permanently used for number recognition.

A template is designed to find the location of the colon in this binary image. This template consists of all pixels in the neighborhood of pixels of the colon sign in the time stamp. This template is searched in the binary image. When the template is matched with a region, the location of the colon sign is obtained. Since the distance between the colon and first digit of the second is calculated in advance, locations of first and second digit are obtained directly. After determining the digit locations, program starts to capture the digits of second. They immediately are assigned to matrices as binary images. It is tried to match the values in these matrices with values of each figure in the *Nmbr* matrix every 5 milliseconds. The matching operation is simply XOR operation. XOR operation is shown in Figure 6.11, and Figure 6.12 shows an implementation of XOR operation between two matrices. If the resulting matrix does not include any non-zero value, matching operation is achieved. Thus, one digit for the second is recognized. The other digit is recognized in the same way.



X	0	0	1	1
Y	0	1	0	1
XOR	0	1	1	0

Figure 6.11 XOR Operation between X and Y



```
000000000          000000000          000000000
001111100          000111000          001000100
001000000          001000100          000000100
001000000          001000100          000000100
001111000   XOR    001000100    =     000111100
000000100          000111100          000111000
001000100          000000100          001000000
001000100          001000100          000000000
000111000          000111000          000000000
000000000          000000000          000000000
```

Figure 6.12 XOR Operation Between 5 and 9 Figures

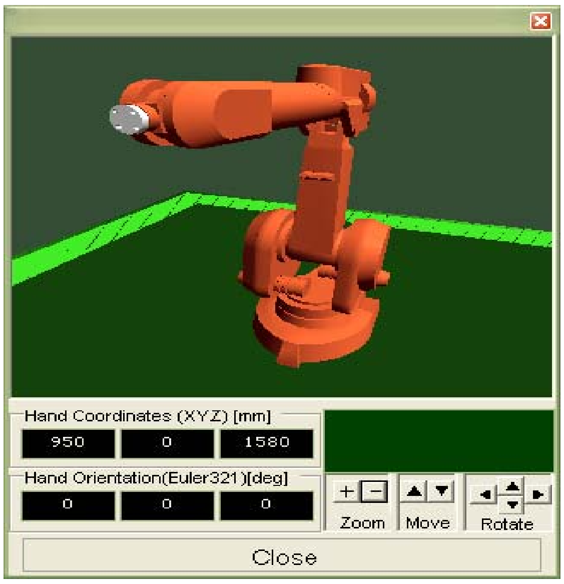
6.2.8 Simulation of the Robot

As described in Chapter 4, in order to obtain a virtual reality based feedback, a component, *SimABB*, is designed modifying a software application developed in the earlier thesis study submitted by Anas Abidi [24]. All classes belong to that study and graphical models are used directly, and some classes are used after necessary modifications. *SimABB* is a user control which is a set of functions that can be executed and its file extension is “dll” (Dynamic Link Library). SimABB.dll also uses some dll files; therefore, seven more dll’s are added to software automatically when *SimABB* control is added. Using SimABB.dll necessitates that all graphical models are added to project.

Graphical model of ABB IRB2000 Robot and its environment were developed using a CAD program and were converted to object files. These files are read by *cObjReader* class when program starts and are converted into type variables. The type variables are inputted into the classes *cVSimulation* and *cModel*. In the class *cVSimulation* the position and orientation of each solid model is set by using the appropriate OpenGL commands.

In the run phase of the software, the graphical user interface of the software can take inputs from the other parts of software. When acknowledgement of a command arrives at the client application, this command is also simulated inputting the coordinates of the robot's tip position to *SimABB*. Joint angles for each joint are obtained from inverse kinematics solution of the robot through *cAbbKinematics* class. This class provides the solution for forward and inverse kinematics for ABB IRB2000 robot. After the calculation of angle values of joints, these values are used to move the graphical models of each link. Before moving the graphical model of the robot, the position and orientation of all the links of the visual robot and its platform are outputted to *cCollideResult* class. The class *cCollideResult* checks for collision between the robot and its environment or the robot with itself. After doing the check, the class sends a message to the *cVSimulation* class whether there is a collision or not. If there is no collision *cVSimulation* updates the scene (by changing the joint angles' values) of the robot and exports it to the GUI view window. If there is collision *cVSimulation* class sends an error message to the GUI to warn the user of possible collision, and the class exports an unchanged scene of the robot to GUI view window. More and detailed information about software can be found in Abidi's study [24].

frmVR is developed to utilize the *SimABB*. This form object holds the *SimABB* user control, initializes *SimABB*, and closes it. *frmVR* is shown in Figure 6.13.



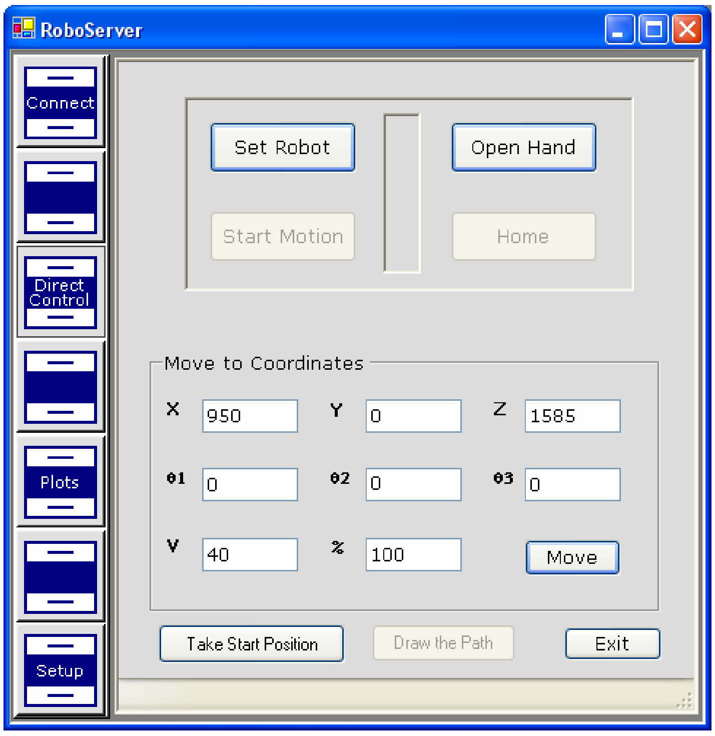
Hand Coordinates (XYZ) [mm]: 950 0 1580

Hand Orientation(Euler321)[deg]: 0 0 0

Figure 6.13 *frmVR* Form Object

6.2.9 Direct Control of the Robot

RoboServer allows maneuvering the robot directly through *frmCom*. *frmCom* is shown in Figure 6.14. This is an extra application and is not directly related with teleoperation. The secondary operations such as moving the robot home position, opening or closing the gripper, and drawing the test paths are executed through this form.



Move to Coordinates

X	950	Y	0	Z	1585
θ1	0	θ2	0	θ3	0
V	40	%	100		

Figure 6.14 *frmCom* Form Object

6.3 Working Mechanism of the Program

All data transferred between the client and server sides are prepared as packets which are string of characters. Some packets consist of just the identification characters like "_RI1". Number of the identification characters is constant and four for every packet. For example, when the packet including "_RI1" is received by the server application, the first four characters are evaluated to find the matching case amongst all cases. The corresponding case leads to calling the *RIGHT1* function of *cCam12* class. This function sends the commands to the Camera-1 in order to be moved to the right.

The other packets include some additional information as in the packet sent when clicked on the image in the video display with mouse. Horizontal and vertical displacements and separating character are added to the packet. Figure 6.15 shows the structure of this packet. When this packet is received by the server application, the case corresponding to the "_RL1" expression leads to extracting the information and calling the *RelativeM1* function of *cCam12* class. The separator character, "V" is found firstly, and then horizontal and vertical values are extracted. These values are sent to the Camera-1 to carry out the motion through *RelativeM1* function.

_RL1	Horizontal Displacement	V	Vertical Displacement

Figure 6.15 A Packet Structure

In command traffic, it is possible to arrive two or more packets at the server one right after another. This sometimes causes to call a procedure before finishing its job. If this procedure is responsible for sending commands to and receiving acknowledgements from the robot then calling this procedure before finishing its job may result in problems. In order to overcome this problem, each command arriving is queued if there is a command waiting. The procedure is called as soon as it finishes its job. Handling the commands in this way is necessary for a safe operation.

After connection is established between client and server side, in order to start to drawing, the human operator has to press on “Start Drawing” button. Pressing on this button leads to calling a procedure in *frmBase*. A string of characters, “_MCR”, are sent to the server side calling the *Send* procedure of the *nCom* class. When a packet arrives at the server side, *nCom* object immediately calls a procedure in *frmBase*. This procedure calls the *DataProcess* routine passing the string of characters to it. The task corresponding to “_MCR” data is performed calling the *Initialize_Control* function. This function is one of the functions of *cRobot* class. *cRobot* class wraps all telegram commands as functions. This class was developed for previous studies by Anas Abidi and Ali Osman Boyaci. After setting the robot for operation, a message notifying that the robot is ready is sent to the client. This message is shown in Message Console and leads to enabling the other control buttons and Graphical Joystick or Drawing Board to be used. When pressed on Pen Down button, the packet of “_ZCD” is sent to the server. This message leads the robot to move along Z axis to the fixed coordinate. At this point, the tip of the pen touches to the paper. Hence, the drawing operation can be started.

The pen is moved on XY plane using Graphical Joystick. The pointer of the joystick is dragged via mouse in the restricted joystick area. Center of the joystick area is the origin and the displacement of the pen is the distance between this origin and the center of the pointer. Limits of the joystick area are limits in the increments of the robot. In order to generate the position coordinates continuously, the pointer must be displaced instead of stopping on a point since the joystick is designed in the way that it does not generate a command while the pointer stopping. When the pointer moves to a point, X and Y coordinates of this point are calculated and scaled to Board Scale. A command packet including these coordinates, resolution value and event related information are prepared as string of characters and sent to the server.

If it is needed to use the Drawing Board, a button is pressed to close the Graphical Joystick and to open the Drawing Board. The pointer of the board is dragged via mouse pressing the left button of the mouse. In order to generate the command it is necessary to drag the pointer while the left button is pressed. Clicking somewhere on the board does not cause to generate a command. The reason behind this is to prevent the undesirable command generation due to accidental clicks on the board. The limits of the drawing board is corresponds to the working area of the robot. The scale between the Drawing Board and the real paper is 105/100. Position coordinates can be generated between +200 and -200 ranges for X and Y directions.

The position coordinates are multiplied by the chosen resolution value before sending them. A command packet including these values and sequence number information are prepared as a string of characters and sent to the server. At the server side, all information is extracted from this packet, the robot's new coordinates are calculated, and the robot is maneuvered to the new position. As soon as the positive acknowledgement is received from the robot, a feedback packet including sequence number and acknowledgement time as second is prepared as string of characters and sent to the client. After being received feedback packet by client application, all information is extracted from this packet, the variables relevant to event-based control are changed, and comparison between the second value extracted from video images and the second value extracted from the feedback packet is made. The result of this comparison and event related variables are used for the next command generation.

CHAPTER 7

SUMMARY AND CONCLUSION

7.1 Summary

Teleoperation technology implies interaction between the human operator and remote device. The human operator communicates commands, and the remote device executes this commands. The operator accepts sensory information transmitted from the remote site to explore the remote environment, plan tasks, verify that tasks are completed, and create plans to resolve problems. The developed teleoperation system allows remote users to control the remote robot through the Internet.

The communication link between client (master) and server (slave) sites is the Internet since it is the cheapest way for communication and also opens to public. In order to design an efficient control system the characteristics of the Internet is investigated. Many measurements are made to identify the behavior of the Internet. Results of these measurements show that the Internet causes time delay in the transmission of data. Furthermore, time delay is variable due to unpredictable behavior of the Internet. Beyond a certain delay, remote control of a telerobot may become challenging task prone to error; therefore, the operation may become unreliable. On account of the delay, many commands sent from operator will be flowing within the Internet instead of arriving at the remote site on time. This effect is known as buffering effect and decreases the synchronization of the human operator action with the robot reaction. The buffering effect of the Internet leads to sending commands supposing either commands can not be sent or the device can not accept commands since any information is not fed back to the operator for a while. This may result in undesirable or excessive movements of the remote device when the commands are received by the device. Packet losses and disconnection are the other difficulties in transmission of data. These major limitations over the Internet destabilize a telerobotic system. In determining control strategy for teleoperation, taking the communication link into account is inevitable. This will significantly increase the safety and efficiency of the teleoperation system.

The ultimate goal of this thesis is to overcome these difficulties developing interface software between human operator and the robot via Visual Basic.NET programming language. The sub-goals are as follows:

Improving the stability on the Internet-based teleoperation through event-based control approach:

Tasks and actions of a robotic system are synchronized and controlled according to given action reference. While the action reference is time in traditional control systems, it is sensor-based measurement and task in event-based control systems, independent of time.

Robot's control will be stable using event-based control approach implementing (1) each task of the robot will be assigned as an event and (2) each event will be acknowledged. In order to use the event-based approach two different methods are developed. One of the methods implies directly implementation of the even-based control method. In this method, each command is accepted an event. The command is sent to the remote site, it is executed by the robot and finally its positive acknowledgement message is sent to the human operator. It is not allowed to generate any new command via force-feedback mouse until the last acknowledgement arrives at the human operator. When the acknowledgement message is received and it is allowed to generate a new command, a new event takes place. This implies that whole operation is divided into events. Although events are independent of time, each event follows the previous event in an increasing order in the time.

The other method is sliding event-based control method. While the event-based approach is kept, it is tried to give flexibility to the human operator in command generation. In this approach, the generated commands belong to an event group. Group size indicates the maximum number of events which can be sent to the remote site without receiving an acknowledgment. If an acknowledgement is not received after starting to generation, then the number of the commands that can be allowed generating decreases by one for each generated command. The allowed number of the commands for generation changes during the operation. The event group slides as commands are acknowledged and generated. It decreases for sent but unacknowledged command and increases for each acknowledgement. Group size is determined according to the delay in the system. While other delays in the system are almost constant, variability of the Internet time delay is the determining factor for the group size. Internet time delay is measured during the remote control, and group size is calculated proportional to the RTT. It is obvious that the speed of the mouse and RTT change the allowed number of commands for generation. Change of the quota is displayed to the user through a graphical level indicator on the graphical user interface. Furthermore, the force-feedback mouse exerts force in accordance with this quota.

Improving synchronization capability:

Synchronization implies happening at the same time or agreeing in time, speeds, and also in feedbacks. Improving the synchronization between the human operator and the robot means improving the performance of the remote control. Two different approaches are presented since two different control approaches are proposed.

In the first control approach, command generation mechanism works according to the events. The control signal feedback coming from the remote site as acknowledgement of the command represents the most up-to-date situation of the robot. On the other hand, visual feedback also fed to the human operator back. In the synchronization sense, it is expected the video image should belongs to this most up-to-date situation. In order to guarantee the synchronization of the feedbacks, it is aimed to couple the acknowledgement feedback of a command and video image belonging to the execution time of this command. When event-synchronization is obtained by coupling these feedbacks, it is allowed to generate a new command.

The second control approach is not directly suitable for event-synchronization due to its nature. The size of video image is larger than the control signal, thus it takes more time to transmit. Instead of waiting the video image when the related acknowledgement is arrived, it is checked whether the video image is received in a determined time interval. In sliding event-based control method, the human operator is directed according to the command traffic. As the human operator adapted to this traffic, a continuous movement of the mouse can be achieved though time delay occurs. Viewing the video images and moving the mouse by evaluating the applied force and command quota results in a synchronization between the human operator and the robot to some extent. However, the delay in the transmission of the video images restricts the synchronization. As improved the hand-eye coordination of the human operator, the performance will increase.

Providing feedback to the operator via force-feedback mouse:

Force-feedback mouse is a haptic device and it applies force on the human operator. Force-feedback mouse will resist to the operator's movements according to whether a task is performed or not, and the force felt will also change proportional to changing command quota.

A force-feedback mouse is designed and built for this thesis. It is a conventional mouse except for added electromagnet and the electromagnet driver circuit which is interfaced to the computer via serial port. The mouse pad is ferromagnetic pad. According to the information sent from computer, the voltage is changed and electromagnet is energized. The attraction between the electromagnet and the pad results in resistance. The exerted force changes as soon as client program sends information to the mouse.

Setting up visual feedback system to feed video to the operator:

Video feedback is supplied using two cameras and a video server. One of the cameras is mounted on the robot to provide the human operator a suitable viewpoint for drawing operation. The other camera is placed to the robot environment to provide for a panoramic view. These cameras are connected to the server computer via serial ports. They are also connected to a video server which allows transmitting the video streams over the Internet by means of its Ethernet connection and Web server feature. These cameras can be controlled for pan, tilt, zoom and focus operations through developed camera display applications over the Internet.

Developing interfaces for teleoperation:

As stated before, teleoperation system can be considered as an interface between the robot and the human operator. This interface has two parts, server and client. The system is interfaced on the client side with human operator and on the server side with the robot environment. The interaction is implemented through developed software application. It abstracts the most of the operation. Command generation, command handling, visual feedback system, force feedback system, communication between devices, communication between remote sites, and delay measurements are implemented by developed software application using Visual Basic .NET programming language.

7.2 Conclusions

In this study, a teleoperation system is developed to control an industrial robot over the Internet. The developed teleoperation system involves all the necessary sub-systems for a teleoperation system. In order to implement a teleoperation system, the interaction between a human operator and the remote device is considered. This interaction is in information level instead of energetic interaction, and the communication link between two sites provides the information transmission. The developed teleoperation system is responsible for high-performance interaction. Since the communication link presents the time delay imposed between the operator's actions and the corresponding feedback, the time delay is considered to be the most important attribute in the design. Therefore, the Internet and its effects over the communication are examined. The variability of the delay and the nondeterministic behavior of the Internet are observed.

The visual feedback system is implemented since it is indispensable for a remote operation. The image size and compression level are set to minimum as far as possible to decrease the influence of the delay onto the video transmission. The designed visual feedback system provides different viewpoints using two cameras. By providing the control of the cameras from remote site, the human operator may change the viewpoints. Also a supplementary feedback as a virtual reality based simulation of the robot is supplied. This simulation may also be used when the visual feedback is not possible for a while. Virtual reality feedback is not used but tested since it significantly slows down the running client application due to messy graphical processes.

The programming language based deficiencies also affect the performance of the application which uses the ActiveX controls and Dynamic Link Libraries (DLLs) created outside the .NET environment. Visual Basic .NET converts them to functions to make use of them. This conversion slows down the application as in encountered in this study. The initialization of the virtual reality based simulation display takes more than two minutes since it uses many DLLs.

In order to prevent the destabilizing effect of the time delay on teleoperation system, event-based control method approach is used in control mechanism. Commands are generated as position coordinates by the human operator using graphical interfaces and a force-feedback mouse. Each command and its acknowledgement are assigned to an event. Time-based control necessitates operating depending on the delay in time or disregarding the arrival of the feedback (acknowledgement and/or multimedia) since it takes time. In teleoperation, the cooperation with time delay is not reliable due to characteristics of the Internet. Therefore, two different control methods are implemented using event-based control approach. The human operator is guided in command generation through these methods. In the first method, acknowledgement of the command and video image related to execution of this command are synchronized. In the second method, it is tried to adapt the operator to the Internet congestion. Using force-feedback mouse, the operator is guided depending on the delay of acknowledgement messages.

The designed force-feedback mouse operates successfully and without delay in the change of force levels. It might be necessary to handle the heating of the electromagnet when draw the higher currents.

The path generation is implemented to regenerate the missing commands. This tool also checks the order of packets. The correct order is important to transmit the commands in the monotone increasing way. The path smoothing tool allows regenerating smoother paths. The commands coming from the remote site are not directly sent to the robot but they are smoothed and then transmitted to the robot.

The interface applications are developed allowing the interaction between the robot and human operator through Visual Basic .NET. The facilities provided by this programming language helped to develop better interface applications.

7.3 Recommendations for Future Work

Instead of using a force-feedback mouse and a graphical interface application, it can be used a haptic command generation device which does not require a graphical interface. Thus, command generation efficiency can be improved.

Autonomy can be brought in using sensory information from the remote environment. Using proximity sensor information, the robot can move autonomously for definite conditions. Using robot vision technology, visual servoing can be implemented for task based operations. 3D construction of the remote environment, to some extent, can be considered for changing environments.

Multimedia feedback system can be improved using Real-Time Transport Protocol (RTP) which allows faster transmission of multimedia than TCP. For this purpose, the video server should be removed and image grabbing should be handled in accordance with RTP.

Since the firewalls impeded the communication between the client and server in this study, a professional contribution from computer engineering can be considered about the firewall-friendly software.

New technologies and capabilities are developed for the Internet by Internet2 consortium. As soon as the performance of the today's Internet is developed, Quality of Service technology may be used in the Internet-based teleoperation.

REFERENCES

[1] Hamel, W. R., Sensor-Based Planning and Control in Telerobotics, in Ghosh, B. K., Xi, N., and Tarn, T. J. (eds.), *Control in Robotics and Automation: Sensor-Based Integration*, Academic Press, San Diego, 1999.

[2] Internet Systems Consortium, http://www.isc.org , Last access date: 27.07.2005

[3] Khamis, A. M., Rodriguez, F. J., and Salichs., M. A., Remote Interaction with Mobile Robots, Autonomous Robots 15, pp. 267-281, 2003.

[4] Xi, N. and Tarn, T.J., Stability analysis of non-time referenced Internet-based telerobotic systems, Robotics and Autonomous Systems, 32 (2000), pp. 173–178.

[5] Elhajj, I., Xi, N., Song, B., Yu, M.-M., Lo, W. T., and Liu, Y. H., Transparency and Synchronization in Supermedia Enhanced Internet-Based Teleoperation, Proceedings of the 2002 IEEE International Conference on Robotics & Automation, Washington DC, May 2002.

[6] Ando, N., Lee, J.-H., and Hashimoto, H., A study on Influence of Time Delay in Teleoperation - Quantitative evaluation on time perception and operability of human operator, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'99), Tokyo, Japan, 1999, pp. V-1111-V-1116.

[7] Taylor, K. and Dalton, B., Issues in Internet Telerobotics, FSR97 International Conference on Field and Service Robotics, Canberra, Australia 8-10 Dec. 1997.

[8] The Telelabs Project, http://telerobot.mech.uwa.edu.au, Last access date: 27.07.2005

[9] Puma Paint Project, http://pumapaint.rwu.edu, Last access date: 27.07.2005

[10] Real Robots on the Web, http://ranier.oact.hq.nasa.gov/telerobotics_page /realrobots.html, Last access date: 27.07.2005

[11] Oboe, R., Force-Reflecting Teleoperation Over the Internet: The JBIT Project, Proceedings of the IEEE, Volume: 91, Issue: 3, March 2003; pp. 449 – 462.

[12] Park, J.-H. and Park, J., Real Time Bilateral Control for Internet based Telerobotic System, Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, Nevada, 2003.

[13] Liu, P. X., Meng, M. O.-H., and Yang, S. X., Data Communications for Internet Robots, Autonomous Robots 15, 2003, pp. 213-223.

[14] Boyaci, A.O., Control Of an Industrial Robot via Internet, MSc Thesis submitted to Middle East Technical University Department of Mechanical Engineering, September 2002.

[15] Mirfakhrai, T. and Payandeh, S., A Model for Time-Delays for Teleoperation Over the Internet, Proceedings of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2001, pp. 236- 241.

[16] Brady, K. and Tarn, T. J., Internet–Based Teleoperation, Proceedings of the IEEE International Conference on Robotics & Automation, Seoul, Korea, 2001.

[17] Kosuge, K., Kikuchi, J., and Takeo, K., VISIT: A Teleoperation System via Computer Network, In International Conference on Intelligent Robots and Systems (IROS): Workshop on Web Robots, Victoria, Canada, 1998.

[18] You, S., Wang, T., Eagleson, R., Meng, C., and Zhang, Q., A low-cost internet-based telerobotic system for access to remote laboratories, Artificial Intelligence in Engineering 15(2001), pp. 265-279.

[19] Park, J.-W., Kim, C.-D., and Lee, J.-M., Concurrent Bilateral Teleoperation over the Internet, IEEE International Symposium on Industrial Electronics Proceedings, Pusan, Korea, 2001, pp. 302-307.

[20] Elhajj, I., Xi, N., Fung, W. K., Liu, Y. H., Li, W. J., Kaga, T., and Fukuda, T., Supermedia in Internet-Based Telerobotic Operations, Proceedings of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services: Management of Multimedia on the Internet, Lecture Notes In Computer Science; Vol. 2216, 2001, pp. 359–372.

[21] Han, K.-H., Kim, S., Kim, Y.-J., and Kim, J.-H., Internet Control Architecture for Internet-Based Personal Robot, Autonomous Robots 10, 2001, pp. 135-147.

[22] ABB IRB2000, Product Manual, 1992.

[23] ABB Robotics Products, Computer Link / ADALP 10 / ARAP S3, 3HAA 3911-102, January, 1992 / M92.

[24] Abidi, A., Man Machine Interface Software Development for an Industrial Robot, MSc Thesis submitted to Middle East Technical University Department of Mechanical Engineering, December 2002.

[25] Konukseven, E. I., Moving Part Recognition and Automatic Pick and Place Using an Industrial Robot, PhD Thesis submitted to Middle East Technical University Department of Mechanical Engineering, June 1996.

[26] Jain, R., Kasturi, R., and Schunck, B. G., *Machine Vision*, McGraw-Hill. Inc., New York, 1995.

[27] METU - CC : Network Group > Backbone Network, http://www.cc.metu.edu.tr/index_eng.php?go=ng&sub=backbone, Last Access Date: 27.07.2005.

[28] Kurose, J. F. and Ross, K. W., *Computer Networking: a top down approach featuring the Internet, 2nd Ed.*, Addison-Wesley, Boston, 2003.

[29] Postel, J., Transmission Control Protocol, RFC 793, September 1981.

[30] Fujimoto, K., Ata, S., and Murata, M., Statistical Analysis of Packet Delays in the Internet and Its Application to Playout Control for Streaming Applications, IEICE Trans. On Communications, Vol. E00-B, No. 6, June 2001.

[31] IBM Corp., *TCP/IP Tutorial and Technical Overview*, IBM Redbooks, 2001.

[32] Paxson, V. and M. Allman, Computing TCP's Retransmission Timer, RFC 2988, November 2000.

[33] Sony Corporation, Sony EVI-D30/31 Command List, Version 1.1.

[34] Axis Communications, What is a Video Server?, Axis White Paper, 2002.

[35] Isdale, J., What is Virtual Reality? – A Web-Based Introduction, Version 4, Draft 1 September 1998, http://www.site.uottawa.ca/~petriu/CEG4392-VE-Introd.pdf, Last Access Date: 31.07.2005.

[36] Nguyen, L. A., Bualat, M., Edwards, L. J., Flueckiger, L., Neveu, C., Schwehr, K., Wagner, M. D., and Zbinden, E. Virtual Reality Interfaces for Visualization and Control of Remote Vehicles, Autonomous Robots 11, pp. 59–68, 2001.

[37] The Tao Framework for Mono and .NET, http://www.mono-project.com/Tao, Last Access Date: 31.07.2005.

[38] SensAble Technologies, Inc, 3D Touch SDK, OpenHaptics Toolkit, Programmer's Guide, Version 1.0, 2004.

[39] Stone, J. R., Haptic Feedback: A Potted History, From Telepresence to Virtual Reality, Haptic Human-Computer Interaction Workshop, August 2000.

[40] Sheridan, T. B., Teleoperation, Telerobotics and Telepresence: A Progress Report, Control Engineering Practice, Vol. 3, No. 2, pp. 205-214, 1995.

[41] O'Modhrain, M. S. and Gillespie, B., The Moose: A Haptic User Interface for Blind Persons, http://ccrma.stanford.edu/~sile/papers/www6-paper.html, Last Access Date: 31.07.2005.

[42] Dennerlein J. T., Martin D. B., Hasser C., Force-feedback Improves Performance for Steering and Combined Steering-Targeting Tasks, Proceedings of the Conference of Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, 2000.

[43] Dennerlein, J. T. and Yang, M. C., Haptic Force-Feedback Devices for the Office Computer: Performance and Musculoskeletal Loading Issues, Human Factors, 43(2):278-86, 2001.

[44] Oakley, I., McGee, M., Brewster, S.A. and Gray, P.D. Putting the Feel in 'Look and Feel', In Proceedings of ACM CHI 2000 (The Hague, Netherlands) ACM Press, Addison-Wesley, 2000, pp. 415-422.

[45] Roters, Herbert C., *Electromagnetic Devices*, John Wiley & Sons, Inc., New York, 1941.

[46] Microchip Technology Inc., PIC16F628 Datasheet, DS40300C, 2003.

[47] Maxim Integrated Products, MAX220-MAX249 Datasheet, 19-4323, Rev. 11, 2/03, 2003

[48] Serial Programming: MAX232 Driver Receiver, http://en.wikibooks.org/wiki/Serial_Programming:MAX232_Driver_Receiver, Last Access Date: 31.07.2005.

[49] Sheridan, T. B., Space Teleoperation through Time Delay: Review and Prognosis, IEEE Transactions on Robotics and Automation, Vol. 9, No. 5, October 1983.

[50] International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Recommendation G.114.

[51] Microsoft The .NET Framework version 1.1 redistributable package, http://www.microsoft.com/downloads/details.aspx?FamilyId=262D25E3-F589-4842-8157-034D1E7CF3A3&displaylang=en, Last Access Date: 27.07.2005.

[52] MSDN Library, http://msdn.microsoft.com/library, Last Access Date: 27.07.2005.

[53] ICMP Protocol Overview, http://www.freesoft.org/CIE/Topics/81.htm, Last Access Date: 27.07.2005.

[54] Postel, J., Internet Control Messaging Protocol, RFC 792, September 1981.

[55] Ping, http://www.freesoft.org/CIE/Course/Section3/8.htm, Last Access Date: 27.07.2005.

APPENDIX A

TOOLS FOR NETWORK MEASUREMENT

A.1 ICMP

Among the various network protocols used over the Internet, the Internet Control Message Protocol (ICMP) is used to check the status of the network connection between two specified computers over the network [53]. Documented in RFC 792, ICMP is a required protocol tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation or mis-operation. Since ICMP uses IP, ICMP packet delivery is unreliable, so hosts can not count on receiving ICMP packets for any network problem. ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when a network device does not have the buffering capacity to forward a datagram, and when a network device can direct the host to send traffic on a shorter route [54]. Some of ICMP functions are to:

Announce network errors and announce network congestion. When a router begins buffering too many packets, due to an inability to transmit them as fast as they are being received, it will generate ICMP *Source Quench* messages. Directed at the sender, these messages should cause the rate of packet transmission to be slowed. Generating too many Source Quench messages would cause even more network congestion, so they are used sparingly.

Measure time delays and announce timeouts. ICMP supports an *Echo* function, which just sends a packet on a round--trip between two hosts. Ping, a common network management tool, is based on this feature. Ping will transmit a series of packets, measuring average round--trip times and computing loss percentages. If an IP packet's TTL (Time To Live) field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. *TraceRoute* is a tool which maps network routes by sending packets with small TTL values and watching the ICMP timeout announcements.

A.2 Ping

The most useful software tool for testing Internet operation at the IP level is *Ping*. Ping is the simplest of all TCP/IP applications. It sends IP datagrams to a specified destination host and measures the round trip time to receive a response. Ping uses the ICMP Echo and Echo Reply messages to determine whether a host is reachable. Since ICMP is required in every TCP/IP implementation, hosts do not require a separate server to respond to ping requests [30]. Some ping properties are as follows:

- Ping places a unique sequence number on each packet it transmits, and reports which sequence numbers it receives back. Thus, it can be determined if packets have been dropped, duplicated, or reordered.
- Ping checksums each packet it exchanges. It can be detected some forms of damaged packets.
- Ping places a timestamp in each packet, which is echoed back and can easily be used to compute how long each packet exchange took - the Round Trip Time (RTT).
- Ping reports other ICMP messages that might otherwise get buried in the system software. It reports, for example, if a router is declaring the target host unreachable.

- Some routers may silently discard undeliverable packets. This is especially common over Ethernet, which does not provide link-layer acknowledgments. Therefore, ping may not always provide reasons why packets go unanswered.
- Ping does not declare why a packet was damaged, delayed, or duplicated. It can not declare where this happened either.
- Ping can not give a blow-by-blow description of every host that handled the packet and everything that happened at every step of the way [55].

A.3 TraceRoute

Traceroute sends IP datagrams with low TTL values so that they expire en route to a destination. It uses the resulting ICMP Time Exceeded messages to determine where in the Internet the datagrams expired and pieces together a view of the route to a host.

The Traceroute program is used to determine the route IP datagrams follow through the network. Traceroute is based upon ICMP and UDP. It sends an IP datagram with a TTL of 1 to the destination host. The first router decrements the TTL to 0, discards the datagram and returns an ICMP Time Exceeded message to the source. In this way, the first router in the path is identified. This process is repeated with successively larger TTL values to identify the exact series of routers in the path to the destination host.

Traceroute sends UDP datagrams to the destination host. These datagrams reference a port number outside the standard range. When an ICMP Port Unreachable message is received, the source determines the destination host has been reached [30].

APPENDIX B

CAMERA ATTACHMENTS DRAWINGS

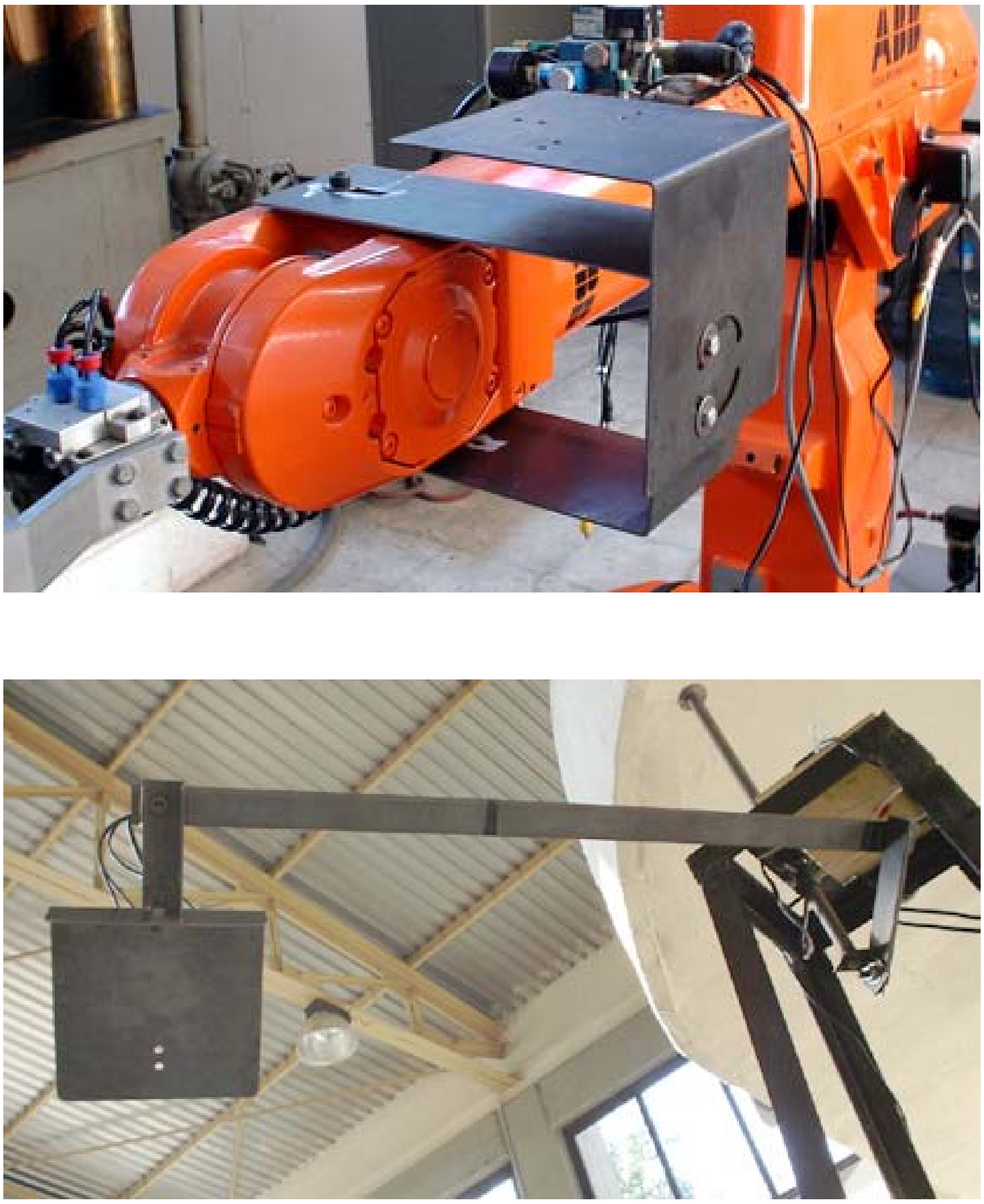
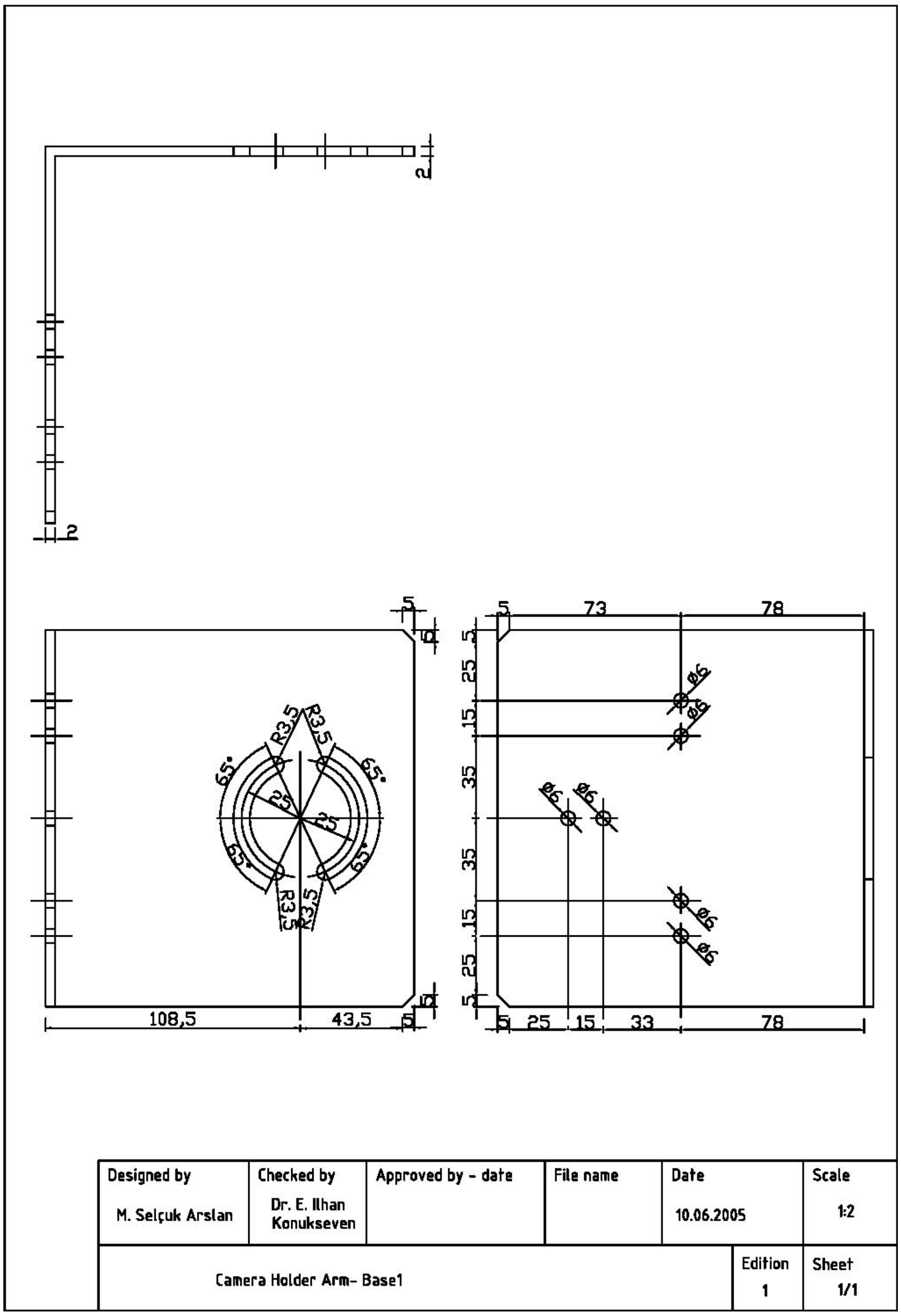


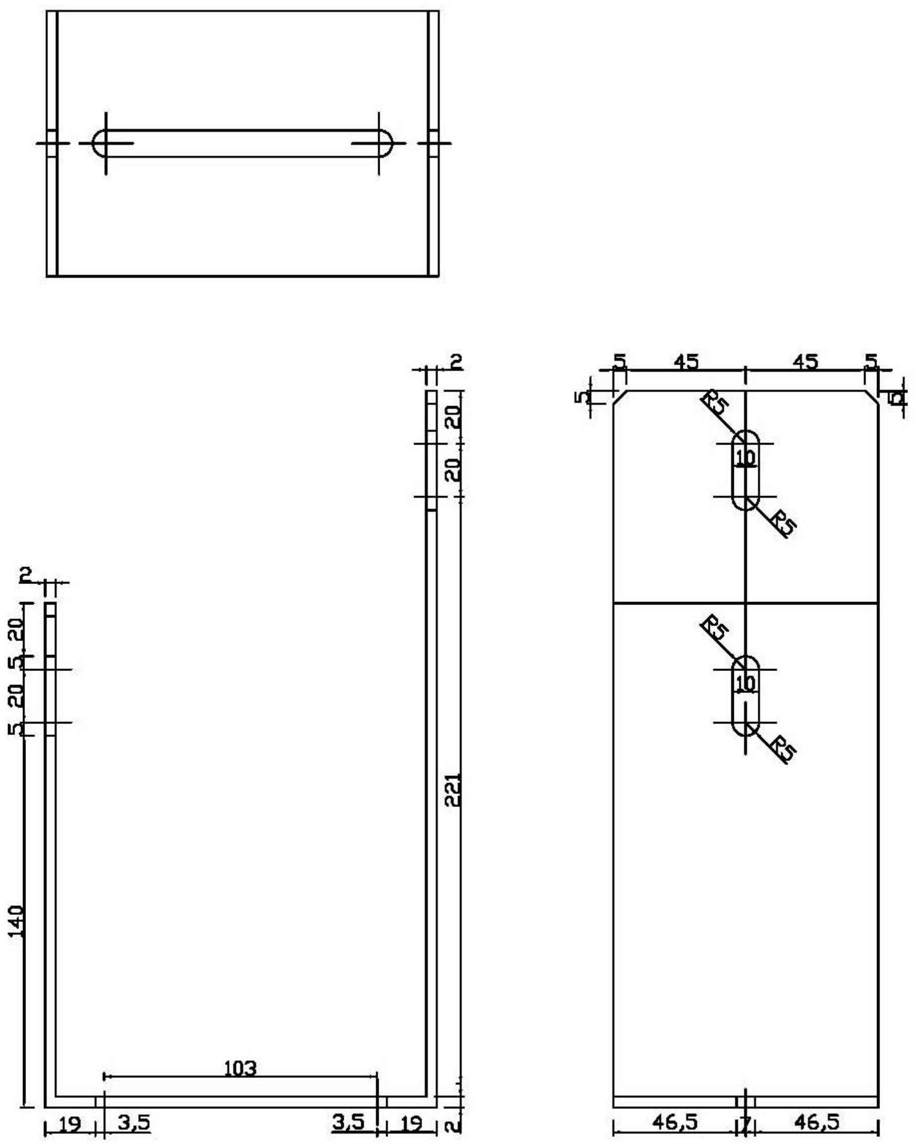
Figure B.1 Mounted Camera Attachments



Designed by	Checked by	Approved by - date	File name	Date	Scale
M. Selçuk Arslan	Dr. E. Ilhan Konukseven			10.06.2005	1:2

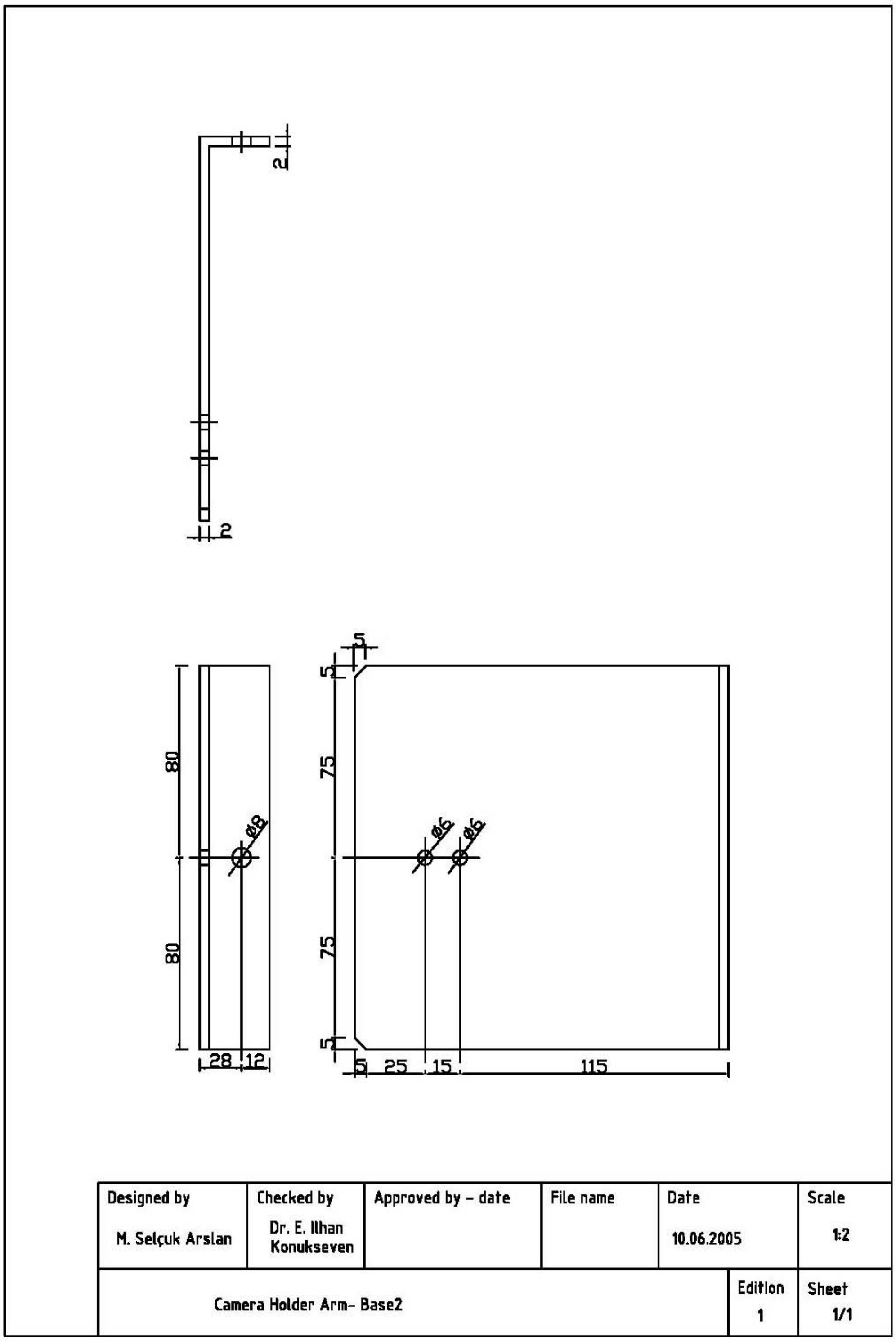
	Edition	Sheet
Camera Holder Arm- Base1	1	1/1

Figure B.2 Camera-1 Holder Base-1 Drawing



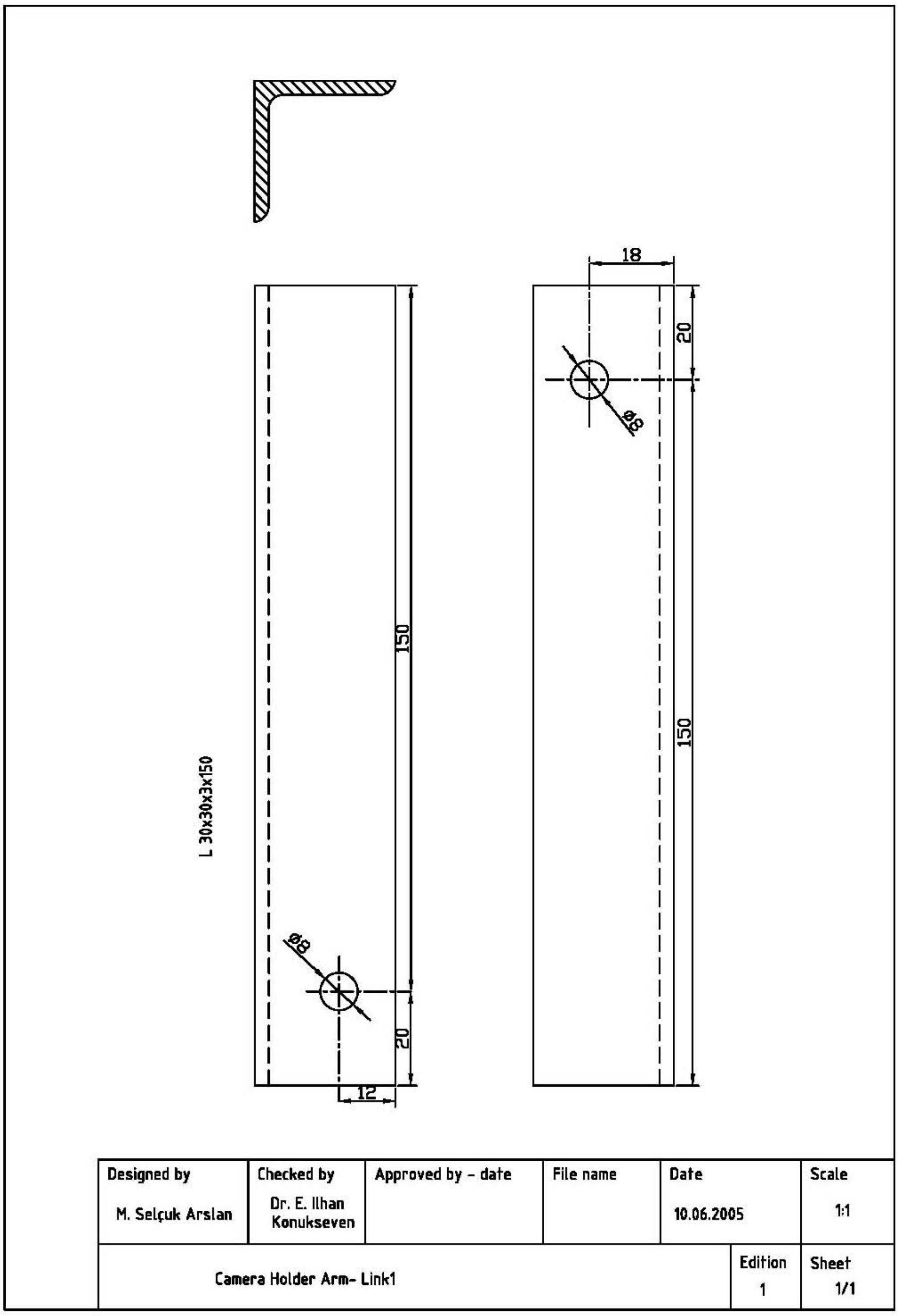
Designed by	Checked by	Approved by - date	File name	Date	Scale
M. Selçuk Arslan	Dr. E. Ilhan Konukseven			10.06.2005	1:2
Camera Holder Arm- Base3				Edition 1	Sheet 1/1

Figure B.3 Camera-1 Holder Base-2 Drawing



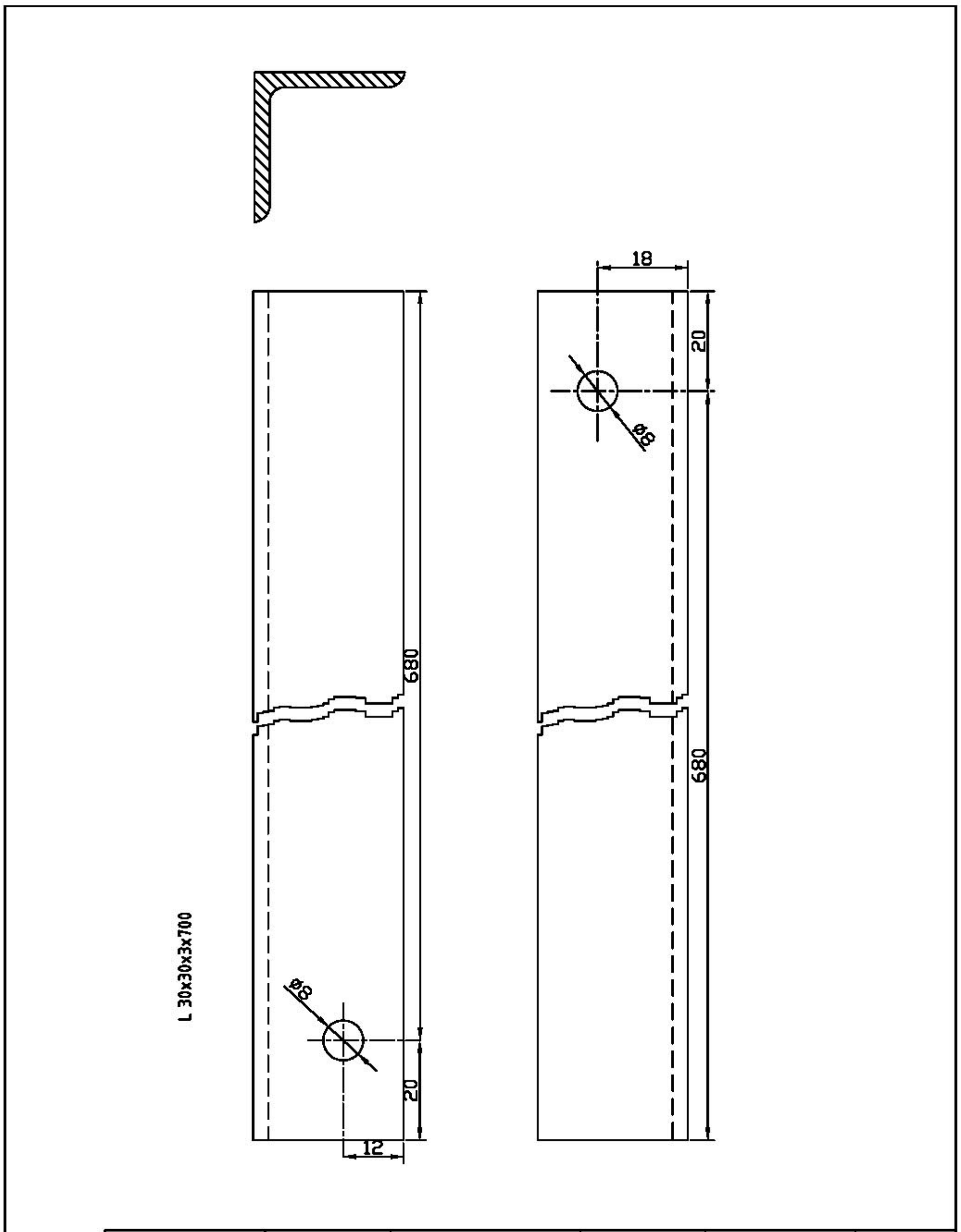
Designed by M. Selçuk Arslan	Checked by Dr. E. Ilhan Konukseven	Approved by - date	File name	Date 10.06.2005	Scale 1:2
Camera Holder Arm- Base2				Edition 1	Sheet 1/1

Figure B.4 Camera-2 Holder Base Drawing



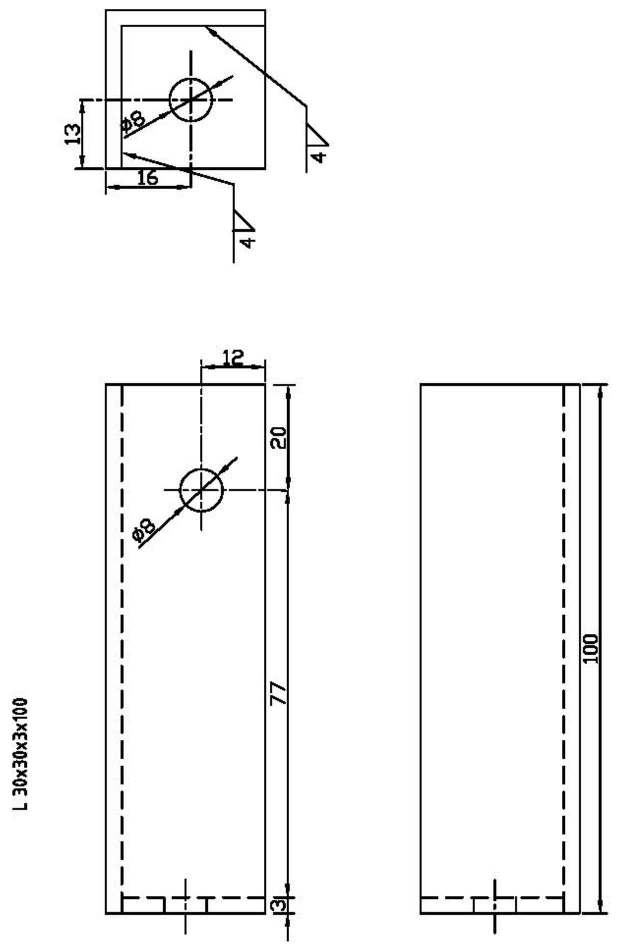
Designed by M. Selçuk Arslan	Checked by Dr. E. Ilhan Konukseven	Approved by - date	File name	Date 10.06.2005	Scale 1:1
Camera Holder Arm- Link1				Edition 1	Sheet 1/1

Figure B.5 Camera-2 Holder Link-1 Drawing



Designed by	Checked by	Approved by - date	File name	Date	Scale
M. Selçuk Arslan	Dr. E. Ilhan Konukseven			10.06.2005	1:1
Camera Holder Arm- Link2				Edition 1	Sheet 1/1

Figure B.6 Camera-2 Holder Link-2 Drawing



Designed by	Checked by	Approved by - date	File name	Date	Scale
M. Selçuk Arslan	Dr. E. Ilhan Konukseven			10.06.2005	1:1
Camera Holder Arm- Link4				Edition 1	Sheet 1/1

Figure B.7 Camera-2 Holder Link-3 Drawing

APPENDIX C

SONY EVI-D30 VIDEO CAMERA

EVI-D30 is a pan/tilt video camera with highly sophisticated image processing technology, which enables target objects to be recognized.

Features:

High speed, Wide Range Pan/tilter
X12 Optical Zoom, High Speed Auto-Focus Lens
6 Position Preset
Auto Tracking / Motion Detector
RS-232C Serial Control (VISCATM)
IR remote Commander

AT (Auto Tracing) Mode

AT is a function, which continually extracts a subject that the user pre-defines. After picking up pixels of similar color and brightness around the selected subject, EVI-D30 extracts the target by using the subject model based on light reflection and nonlinear camera processing. There are four modes for pre-defining the subject. AT-PAN/TILT function follows the moving subject automatically by controlling the pan & tilt motors without the use of special sensors. AUTO ZOOM function automatically controls the zoom lens to ensure that the size of the subject remains constant. The EVI-D30 employs the auto exposure and advanced backlight compensation systems to ensure that the subject remains bright even in harsh backlight conditions. Because the subject position is known a comparison can be made between its brightness and that of the background and the camera subsequently adjusted to compensate for the conditions.

MD (Motion Detector) Mode

MD basically detects the difference between the initial reference image and the current image. The conventional technique employed in MD uses only the brightness of the video signal. The EVI-D30 uses both the brightness and color, which enables even an object of the same brightness as the background to detect. Changes in light conditions are constantly monitored and the data in the reference image adjusted.

A user can set two rectangular detection areas of any size and in any position of the scene. Once motion is detected within the preselected windows an alarm signal is output, which, for example, can commence the recoding on a VCR. This mode is made even more versatile by the ability to adjust the detection brightness, color and area.

VISCA

EVI-D30 can be controlled by RS-232C serial control using VISCA^{TM}. VISCA™ is an acronym of Video System Control Architecture. It is a network protocol designed to interface a wide variety of video equipment to computer. Under VISCA^{TM}, up to 7 EVI-D30 can be connected to one controller using RS-232C communication. RS-232C parameters are communications speed of 9600 baud, data length of 8 bits, 1 stop bit, and no parity.

Pan/Tilt Range

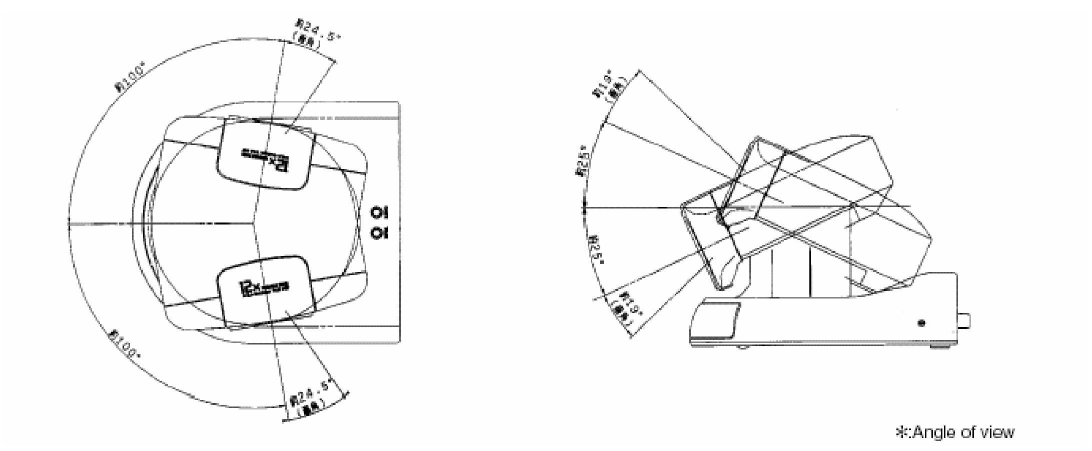


Figure C.1 Pan/Tilt Range

Specifications

System

Video signal:	NTSC Color, EIAJ standards
Picture element:	1/3 inch color CCD
Total picture element number:	Approx. 410,000
Effective picture element number:	Approx. 380,000
Lens:	Electro-motion twelve fold zoom lens f=5.4 to 64.8mm, F1.8 to F2.7
Horizontal angle:	4.4° to 48.8°
Point-blank range:	WIDE end: 10mm TELE end: 800mm
Minimum illumination:	7 lux (F1.8)/with 50IRE
Illumination range:	7 to 100,000 lux
Shutter speed:	1/60 to 1/10,000 (VISCA control)

Gain selector:	Automatic/manual
Horizontal resolution:	NTSC: 460 TV
	PAL: 450 TV
Video S/N:	48 dB
Pan/tilt action:	Horizontal: 100°
	Vertical: 25°

Input/output terminals

Video output:	RCA pin jack (1), 1Vpp, 75 ohm unbalanced
Synchronization:	Negative
S video output:	4 pin mini DIN (1)
Audio output:	RCA pin jack, monaural (1)
Rated output:	327 mV
Output impedance:	Less than 2.2 kilo ohms
Input/output control terminals:	
	RS232C (input: 1, output: 1),
	8 pin mini DIN, 9600bps
Data:	8 bit
Stop bit:	1
Microphone input terminal:	
	Mini jack (monaural) (1) (ø 3.5)
	Rated input 0.775 mV
	DC 3V for low impedance microphone
Input impedance:	More than 10 kilo ohms
Power terminal:	EIAJ type4

General

Input voltage:	DC 12 to 14 V
Power consumption:	11 W

Operating temperature:	0°̣ to 40°̣ (32° to 104°F)	
Storage temperature:	– 20°̣ to 60° (– 4° to 140°F)	
Dimensions Video camera:		
	Approx. 142 x 109 x164 mm	
	(5 5/8 x̣ 4 3/8 x̣ 6 1/2 in.)	(w/h/d)
Remote commander:	Approx.56 x̣ 26 x̣ 210 mm	
	(2 1/4 x̣ 1 1/16 x 3/8 in.)	(w/h/d)
Mass Video camera:	Approx. 1,200 g	(42.3 oz.)
Remote commander:	Approx. 109 g	(3.8 oz)

APPENDIX D

VISCA™/RS-232C CONTROL PROTOCOL

The VISCA protocol is a concept which has been developed by Sony to simplify the combination and networking of various elements from the video industry. For example, VISCA allows up to seven RS232 controllable devices to be addressed with one single address. In order to achieve this, each VISCA compatible device has a VISCA IN and VISCA OUT connector. The VISCA OUT socket can be connected to the VISCA IN of another VISCA device and thus a daisy chain of devices can be implemented. During the last few years, VISCA has become a video industry standard and is used not only by those developing ISDN image transfer systems, but also in video conferencing.

Connection

Camera may be controlled by personal computers (PCs) or workstations by connecting as shown in Figure D.1 or Figure D.2. Figure D.1 shows one by one control using serial interface port of the PCs or workstations. By this connection, the address of the camera can be identified by the port number or the address assignable to the each camera.

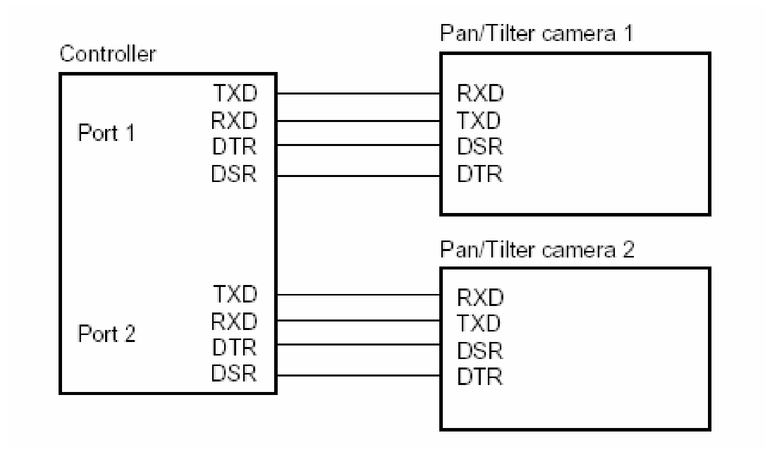


Figure D.1 Direct Connection

Figure D.2 shows schematically connections of camera when daisy chained. For actual connection, refer to information supplied by each model. In this instance, the maximum set on one network is seven and the address can be assigned automatically by the controller. The address of the controller is set to 0 and the camera address will be assigned from 1 to 7 (nearer, the younger address). The interface to the controller is RS-232C, 9600 bps, 8 bits data, 1 start bit, 1 stop bit and non parity.

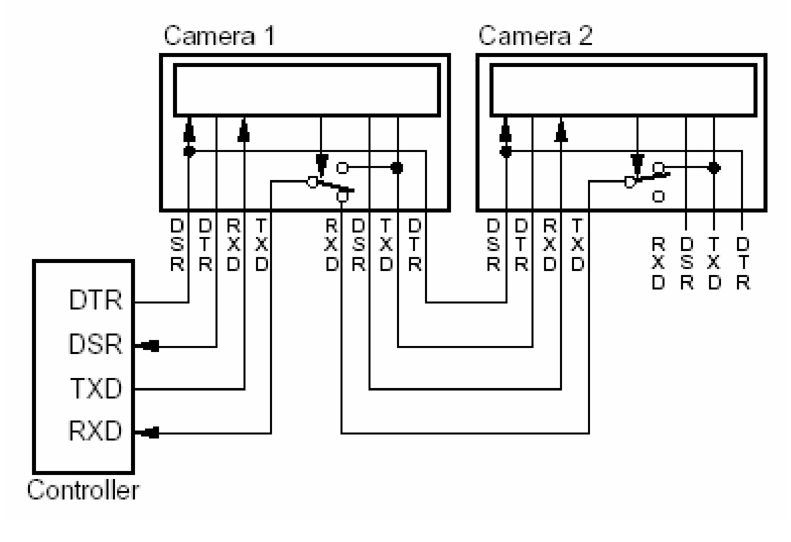


Figure D.2 Daisy Chain Connection

Communication specification (RS-232C)

Communication speed: 9600 bps	Start bit : 1
Stop bit : 1	Data bits : 8
Parity : None	MSB first

Communication protocol

- *Communication from the controller*

Communication is started by header which comprises sender's address and receiver's address followed by message and ended by terminator. The message part comprises communication mode (2 bytes), category code (2 bytes) and parameters.

The maximum length of the message is 14 bytes. The terminator is fixed to FFH and the controller should check the value to terminate communication. The bit 15 should be 0 in the message part.

Header	Com-mode	Category code	Message	Terminator
8x	01	rr	.. ..	FF

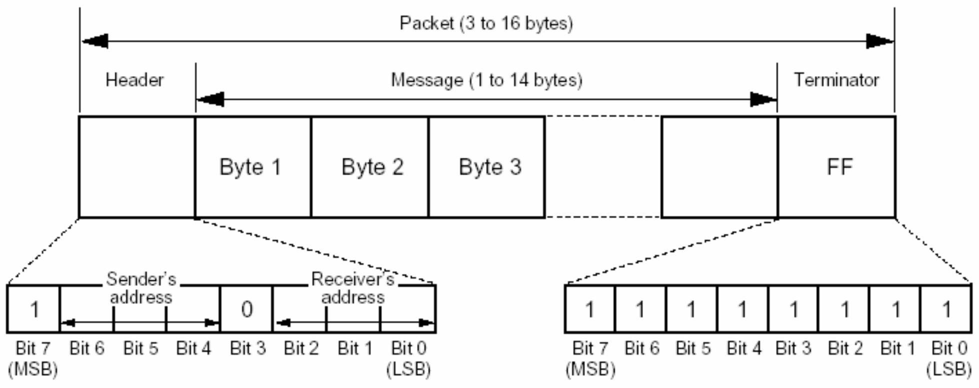


Figure D.3 Packet Structure

Header: Signifies the start of the communication and comprises the sender's address and receiver's address. Since the address of the controller is fixed to 0, header is 8x in which x is the receiver's address. (The value of x should be from 1 to 7) In case of broad cast, the header should be 88H.

Com-mode: Code which specifies the category of command.

Control command: 01H

Information request command: 09H

Net-keeping command: 00H

Category: Code which roughly specifies the category the command is applicable.

Main message: Part between header and terminator. 14 bytes maximum. Comprises command and parameter if any.

Terminator: Code which signifies the end of communication. Fixed to FFH.

- *Commands*

A command which is sent from the controller to the camera is comprised in the message part of the send data. The commands are roughly classified into several functions such as to control camera, to inquire the information of the camera and the ones for various purposes. The controller commands may comprise some parameters as needed.

- *ACK message*

The ACK message is returned to the controller from the camera to acknowledge the command. ACK message comprises the address of the camera (expressed by z and z = address + 8), socket number (y) and terminator. Socket is the memory buffer reserved in the camera and used to store commands. Having this feature enables the camera to execute new commands during former commands being under execution. In case of inquiry commands, the information is returned between the third byte and the terminator. If the commands are inquiry, the camera returns information message immediately, but for the commands, the camera returns ACK message immediately and returns the command completion message when the command is actually executed.

ACK	z0 4y FF
Command completion	z0 5y FF
Information return	z0 50 .... FF

Following message is sent from the camera to the host when power ON.

Address set z0 38 FF

- *Error messages*

If the command is not executable or failed to execute, the camera returns error message instead of the acknowledge message. The error message is expressed by the following codes.

Syntax error	z0 60 02 FF
Command buffer full	z0 60 03 FF
Command cancel	z0 60 04 FF
No sockets	z0 60 05 FF
Command not executable	z0 60 41 FF

- *Broadcast*

Used to command all the set regardless the individual address. In this mode of transmission, the header code is set to 88H.

- *Auto address assignment*

This command is only valid when the camera is connected as shown in Figure D.2. When the camera receives the address assignment command (88 30 01 FF), the first camera sets parameter 01 as the self address and hands over to the next camera by incrementing this parameter. The controller can recognize how many camera cameras are connected on the network by the returned parameter.

APPENDIX E

AXIS 2400+ VIDEO SERVER

The AXIS 2400+ Video Server is high performance video server designed for professional surveillance applications, giving increased memory and performance compared to their predecessors.

Connecting directly to an Ethernet network or modem, the AXIS 2400+ can accommodate up to four analog video streams. Transforming traditional analog video into high-quality digital images, this Axis video server provides a comprehensive single box solution for video transmission over intranet networks, or the Internet.

Axis 2400 4-Port Video Server – Specifications

GENERAL

• Use as a standalone system or added on to existing CCTV systems
• Remote configuration and status using Web based tools
• PPP/Modem support
• Event handling including mail, TCP, HTTP and FTP notification and video upload
• Audio functionality available in combination with the AXIS 2191 Audio Module

HARDWARE AND SYSTEM

• CPU: ETRAX 100LX, 32 bit RISC, 100 MIPS

• Compression chip: ARTPEC-1

• 4 MB Flash memory

• 32 MB SDRAM

• 100baseTX Fast Ethernet or 10baseT Ethernet

• Network protocols: TCP/IP, HTTP, FTP, SMTP, NTP, ARP, DHCP, BOOTP and more

• Based on Linux version 2.4 operating system

CONNECTIONS

Network

• RJ45 connection to 10/100 Mbit Ethernet networks

Video inputs

• The AXIS 2400+ has four BNC composite video inputs with 75/Hi Z termination

• The AXIS 2401+ has a single BNC composite video input with 75/Hi Z termination and one BNC video loop-through port

• Auto sensing for NTSC and PAL

I/O

• A single Terminal Block connector providing four opto-isolated alarm inputs and a single output relay

Serial connectors

• 9 pin D-SUB RS-232 max 115 Kbit/s

• 9 pin D-SUB RS-232 or RS-485/422 half duplex, max 115 Kbit/s

Power

3 alternative power sources:

• External power supply 12V AC, 9.6 VA (PS-D, included)

• 9-20V AC, min 10VA

• 6-30V DC, min 7W

VIDEO

Frame rate

• Image frame rate: up to 25 (PAL)/30 (NTSC) frames/second

• Bandwidth control prevents data saturation on the network

Compression

• Motion-JPEG video, as well as JPEG still images

• 5 levels of compression are available.

The file size of a JPEG compressed image depends on the actual content of the image. Images containing a lot of detail will generate larger files. Image quality is controlled through the level of compression.

High compression yields smaller files at the expense of image quality, while low compression results in larger files, but maintains image quality.

Table D.1 File Sizes Different Compressions and Resolutions

Resolution	Compression Level		
	Low	Medium	High
NTSC, 352x240	10 KB	7 KB	3 KB
NTSC, 704x480	43 KB	28 KB	13 KB
PAL, 352x288	12 KB	8 KB	4 KB
PAL, 704x576	52 KB	34 KB	20 KB

Table D.1 shows average file sizes, derived from real life tests.

Resolution

• QCIF: 176 x 112 (NTSC), 176 x 144 (PAL)

• CIF: 352 x 240 (NTSC), 352 x 288 (PAL)

• 4CIF: 704 x 480 (NTSC), 704 x 576 (PAL)

Video features

• Date and time stamp and text overlay.
Color control (B/W or color)

PAN/TILT/ZOOM

• PTZ support for remote camera control.

• Future support for other PTZ units will be added – check Axis web site for the latest update

CUSTOMER APPLICATIONS

• Compliance with the AXIS HTTP API for applications and systems integration
• Support for shell scripting to allow user defined applications
• Up to 0.5 MB available for storage of customer Web pages and scripts

OPERATING CONDITIONS

• Temp: 40-125°F (5-50°C)
• Humidity: 20-80% RHG

DIMENSIONS AND WEIGHT

• Height: 1.7" (4.2 cm)
• Width: 5.7" (14.5 cm)
• Length: 8.7" (22.0 cm)
• Weight: 1.7 lb. (0.8 kg), excluding power supply

APPENDIX F

RTT AND FORCE LEVEL GRAPHICS OF THE IMPLEMENTED EXPERIMENTS

Case 1

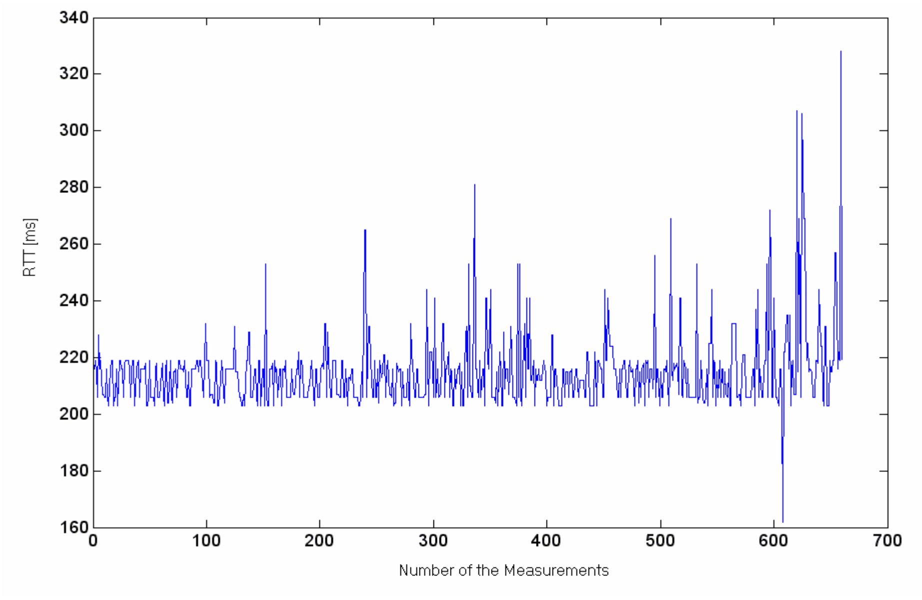


Figure F.1 RTT Measurements during the Operation

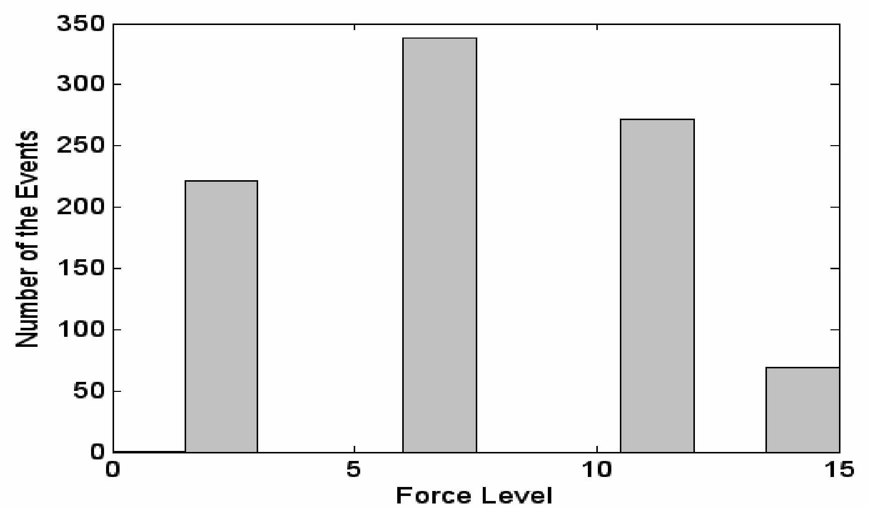


Figure F.2 Forces Exerted during the Operation

Case 2

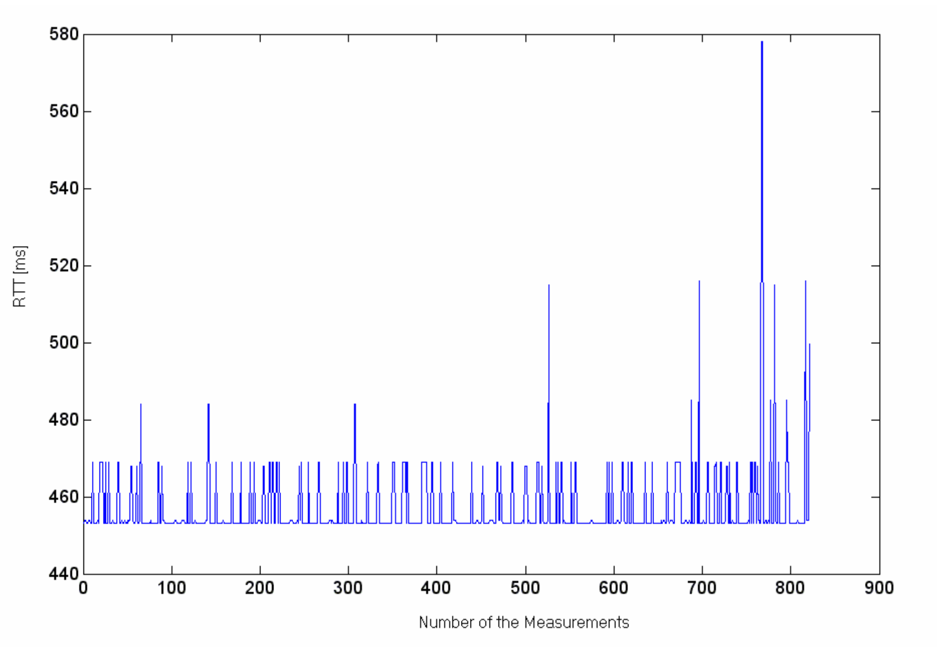


Figure F.3 RTT Measurements during the Operation

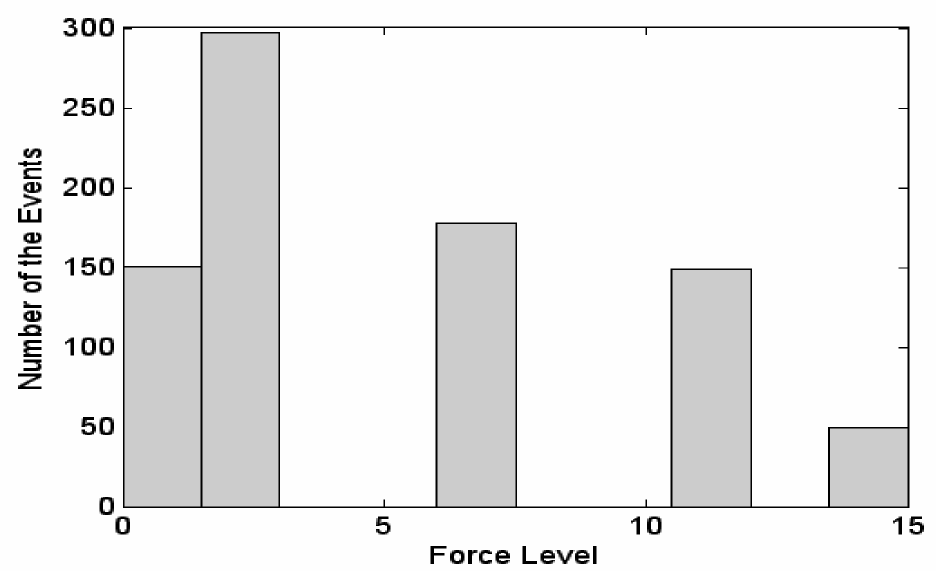


Figure F.4 Forces Exerted during the Operation

Case 3

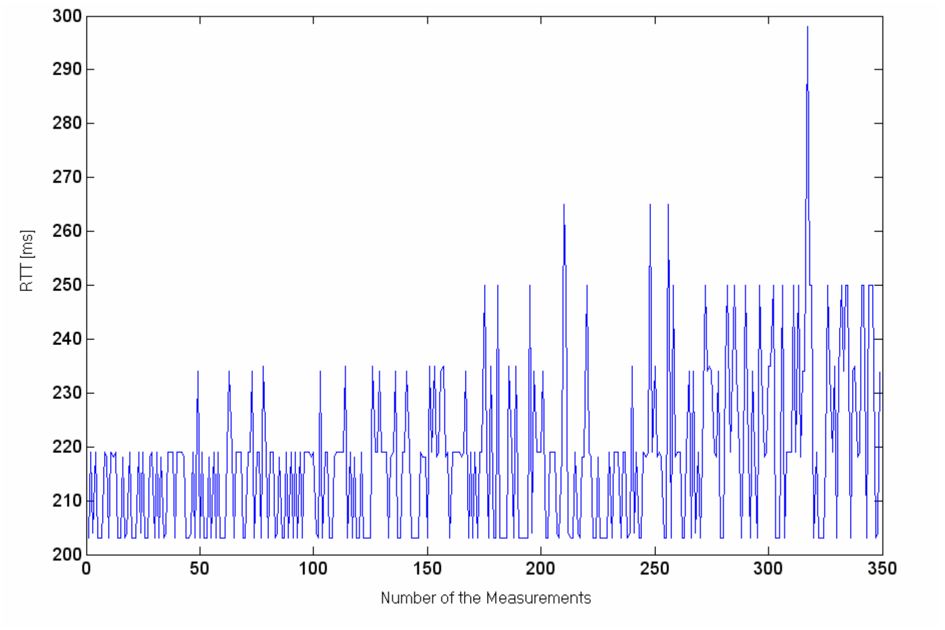


Figure F.5 RTT Measurements during the Operation

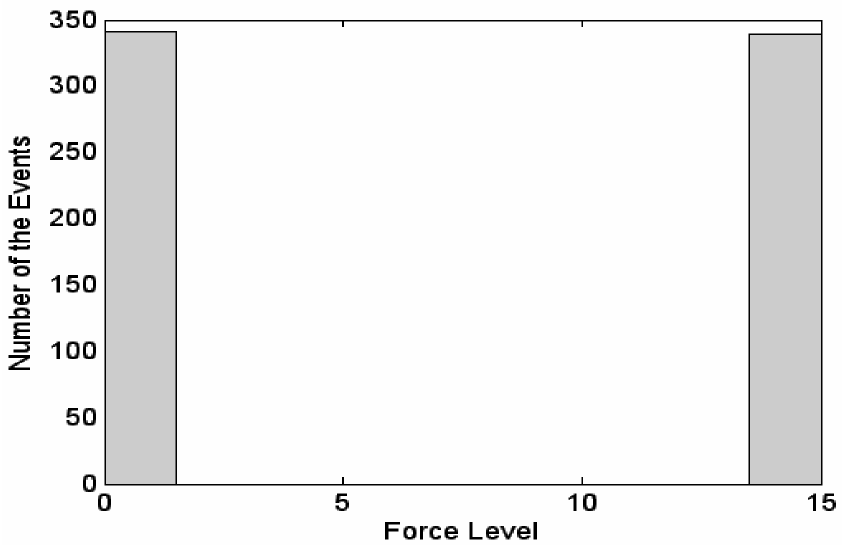


Figure F.6 Forces Exerted during the Operation

Case 4

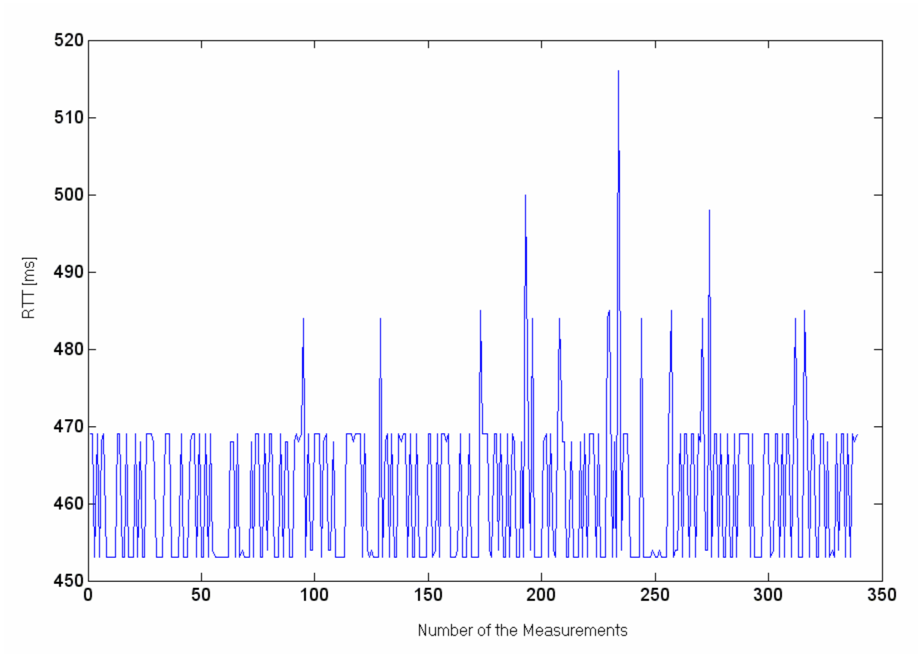


Figure F.7 RTT Measurements during the Operation

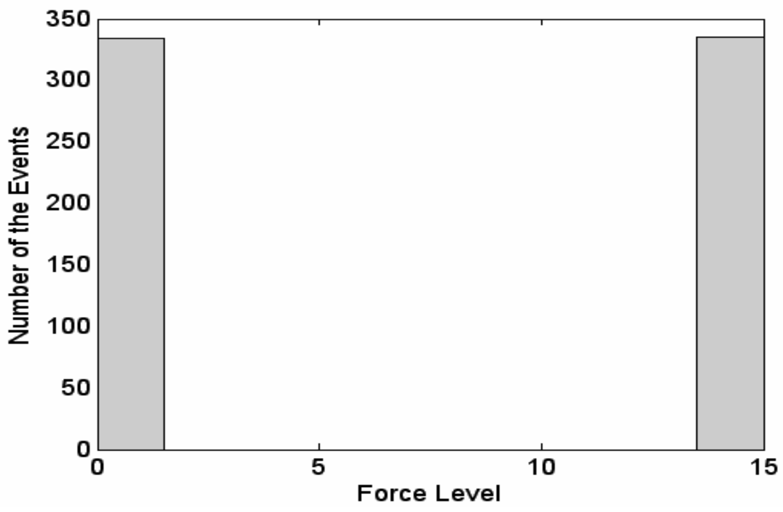


Figure F.8 Forces Exerted during the Operation