This is to certify that the

thesis entitled

REAL-TIME ROBOT CONTROL OVER THE
INTERNET WITH FORCE REFLECTION

presented by

Imad Elhajj

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Eng

Major professor

Date 7/15/99

REAL-TIME ROBOT CONTROL OVER THE INTERNET WITH FORCE REFLECTION

BY

IMAD HANNA ELHAJJ

A THESIS

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

1999

ABSTRACT

Real-Time Robot Control over the Internet with Force Reflection

By

Imad Hanna Elhajj

The use of the Internet is not limited anymore to the transmission of data. In the past few years many successful attempts have been made to use the Internet as a command transmission media; where control can be sent to remote systems and feedback can be obtained via the Internet. But with this media comes several limitations; delay, lost packets and disconnection. All of these limitations may cause instability in the system especially if the system loop is closed via the Internet.

All the previous work done that addresses these problems assumed several conditions; for example, time delay is constant or has an upper bound, control is not in real-time. A new real-time control approach is presented that deals with these limitations and difficulties without any assumptions made regarding the time delay. The approach is based on Event Based Control, which was implemented on a mobile robot over the Internet. The commands sent to the robot are velocity and the feedback is force based on the environment. It will be shown that this approach results in a stable system. In addition, a new force feedback generation method is used, which would give the operator a specific perception about the physical environment that the robot is in.

To my *great* and *loving* Family.....

ACKNOWLEDGEMENTS

I would like to acknowledge all the help from Dr. Ning Xi. Without him, this would not have been possible. And I would like to thank Mr. Keith Jones for all the help with the document formatting.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the past decade the Internet have experienced orders of magnitude growth. This growth was not only characterized by the number of networks and terminals connected to it, but also by the several application that it was used for. It is not only a data transmission media, but also a place to buy, sell, learn, explore and "control".

Control over the Internet can be characterized by several properties and takes different forms. Regardless of the system used all real-time controlled systems over the Internet share many similar difficulties and advantages. A major characteristic and limitation of tele-operation is delay. Data Packets transmitted between 2 points will be delayed. This delay, although not desired, does not constitute a major problem for data being passed. But when we start considering the network (Internet) as an action super-highway, via which we send control commands, time delay becomes an important factor in the stability and efficiency of the system.

We present here a new approach for tele-operation with force feedback that overcomes the problems associated with delay and several other tele-operation problems.

In what follows We will give the motivation behind using the Internet for tele-operation, then we will discuss the difficulties and problems associated with this media. After that we will cover some of the existing methods that deal with these problems and point out their limitations.

1.1 Applications and Motivations

The applicability of a certain technology is a main factor in its advance. When it comes to tele-operation, several applications motivated the advance of this field. Among the major ones we have:

1. Tele-medicine: The attractive thing about this application is the ability to treat

patients by experts that are not in the same city or even the same country. The ultimate objective is to have systems with very high reliability to be able to trust them with human life. Several experiments and improvements have been done with these systems [1][2].

2. Tele-manufacturing: The ability to use sophisticated and expensive manufacturing facilities by several users around the world is very cost effective [5].

3. Exploration (sea and space): This application got very popular during the NASA Mars Pathfinder Mission [3]. There is no need to mention the various advantages for this application and the importance of tele-operation for space and deep sea exploration [6][8], especially in the safety and cost reduction aspects.

4. Hazard material manipulation: One of the first noticed application of tele-operation is handling hazard material; such as, radioactive or explosive objects [4].

5. Tele-operated games: Several tele-operated games can be found on the Internet; for example, in Germany there is an interactive railroad that can be operated remotely.

It was shown in [20]-[21]that the operators performance can be improved in these systems by having force feedback. Force feedback is simply sending back from the slave side force or velocity information that can be felt by the operator, in which case the operator will have a better understanding of the environment. When this is done, the operator is said to be kinethetically coupled to the environment, and the teleoperator is said to be controlled bilaterally.

All of these tele-operation applications, including force feedback, can be done using any kind of communication media. But what interests us is the use of the

Internet as the link between the operator and the robot. The use of the Internet is motivated by many reasons; in comparison to a dedicated link, the Internet is much cheaper, more publicly available and does not require any special hardware. However, with all these advantages comes several disadvantages which will be discussed in the next section.

1.2 Limitations and Difficulties

As any communication link, the Internet introduces delay in the system. This delay can be due to several reasons; propagation, congestion, or being low on resources. For a communication link between the ground station and a space telerobot in low earth orbit delay is expected to be as long as 2-8 sec [7]-[11]. Same amount of delay can be expected for deep sea operations. As for the Internet, the delay is unpredictable, in the sense that no upper bound can be specified and it could have a very complex stochastic model. Several attempts have been made to model delay over the Internet, and several complex models have been derived [37]. In [37] a wavelet-based model[1] was derived. This delay was shown to be self-similar [38], in a statistical sense.

In addition, packets sent over the Internet can be lost and can arrive out of order, that is why we used TCP/IP protocol for communication, which insures the arrival of all packets in order. More severely, Internet connections can be lost, that is why several design precautions had to be taken. These design issues will be discussed in more details in subsequent chapters.

All these can render the system unstable. Especially time delay; where it was shown in [7]-[17] [20]-[24] that delays on the order of a tenth of a second can destabilize the teleoperator. And when force feedback is being used the system is stabilized only when the bandwidth was severely reduced. In addition to instability, de-synchronization can occur between the master and slave.

[1] Wavelet analysis is most used for statistically self-similar processes.

Several approaches have been taken to solve the above mentioned problems, and those will be the topic of sections 1.3 and 1.4.

1.3 Overview of Existing Methods and Approaches

What follows is a summary of the different methods and techniques used to overcome the problems faced in tele-operation in general. The different approaches will be presented here and then section 1.4 will cover their weaknesses. The approaches are divided into 2 groups. One of them concentrates on the *robotics side* by introducing techniques and force/image feedback with prediction. The other concentrates on the *communication side* by using new communication protocols or modifying the existing ones. What was noted is no approach combined both sides in the solution.

1.3.1 Robotics Approach

Most of the approaches in this category relate to control schemes and force/image feedback with prediction. Some of them are even a combination of several approaches.

Control

The problem is handled by changes in the controller modeling/design and control algorithms. Several design techniques are used, some of which apply to the specific system used and others that are more general.

In [18][19] an observer for linear feedback control with delays is designed. [7][8] use *shared compliant control*; where the control task is shared by both the human operator and the autonomous compliant control of the remote robot system. As for [9], time and position de-synchronization was used and in [13] Virtual Internal Model (VIM) was the approach. [15] used design method based on the $\mathcal{H}_\infty$-optimal control and μ-synthesis framework. The work of Anderson and Spong in [20] and [21] was a good

starting point for our research, they introduced a controller that was derived from modeling the network as a loss-less transmission line. Another method was based on stochastic control theory and a separation property [17].

Sensory Feedback

This includes force/torque feedback and image feedback. Either of these fed back elements can be either real or virtual. So some of these techniques actually use prediction.

The most common feedback in tele-operation is video, where cameras capture the task space and then these images are sent back to the operator [7][8][10][11][12]. The other sensory feedback is usually force or torque, where the feeling of the robot is sent back to the operator [20]-[24]. Although these two techniques usually use real values, sometimes some kind of prediction is used in parallel. For example, the position of the manipulator can be predicted using a simulation program and it can then be displayed with the real image so that the operator can compare [11][12][16]. Predicted force can also be used where the simulated value can be sent to the master or just displayed on the monitor [16].

1.3.2 Communication Approach

The approaches in this category try mainly to change the underlying communication protocol or at least modify it. That is why these methods are difficult to implement and test, since in order to do that we have to modify the routing algorithm and communication protocol used by all the different components in our communication system. The thing to note is that non tried to develop a method that does not need major system modification; for example, an application level solution, where the routine that is handling the delay problem is running at the application level.

One of the proposals was to design a rate-based flow control, where the rate at

which the user is allowed to transmit is determined by the switches on the VC[2] between the source and destination [34][35]. Another proposal was used dynamic congestion control and error recovery algorithm over a heterogeneous Internet [36].

1.4 Limitations of Existing Solutions

Most of the methods used, share some disadvantages that will be discussed here. By far the most common limiting assumption, is taking delay as having an upper bound. By upper bound we mean taking a value for the delay beyond which the system would either become unstable or has to stop operating. For example, they might assume delay in their analysis and implementation not to be more than 3 sec, and if it happens that the delay actually exceeds 3 sec the system either becomes unstable or has to stop operating. Another limitation is not analyzing or doing testing with random time delays. Both of these assumptions are quit important when the Internet in involved, since no upper bound can be obtained and the delays are proven to be "random" [37][38]. Moreover; some of the proposed solutions require a lot of computing power which might in itself be a delaying factor. In addition; in some cases the remote system is very complex and in others the approach causes more network traffic, as in video feedback. As for the predictive methods, they require a lot of computation and they do not work well with uncertainties in the environment. As for the communication proposed fixes, they are difficult to implement and test since they require major changes in the existing systems.

1.5 Previous Internet Tele-operation and Their Limitations

In this section we introduce the previous work done specifically in Internet tele-operation, and then we discuss their limitations. Several Internet based robots have

[2]Virtual Circuit

been developed and studied, we present here some examples that cover most of the categories. Before we go on, we would like to clarify a misconception when Internet robots are considered. Many researchers consider Internet robots to be the ones which are web based, meaning the ones that are controlled via a web page. Many discussions have taken place lately[3], and the general trend was to consider Internet robots as being the ones controlled over the Internet in any fashion and not only via a web page.

Tarn and Brady [39] implemented a semi-autonomous telerobot, where the tele-operator supplies a trajectory for the robot to execute. Then the operator only intervenes in case of unexpected circumstances. Pai [40] made an expensive experimental facility, the ACME, available via the Internet, where a user can conduct customized measurements and built accurate models from anywhere. Stein [41] had introduced a Puma manipulator that can be controlled to paint over the Internet. Simmons [42] has an autonomous mobile robot (Xavier) that can be commanded to go to different locations and do different tasks. Other web robots have been introduced, and currently the NASA Space Telerobotics Program web site[4] lists more than 20 of them. These robots perform different tasks, ranging from manipulating objects and navigation to manipulating camera view.

All these systems have several limitations and differ from ours in several aspects. First and most important is that these systems do not have real-time feedback. Meaning that the only feedback the operator has is visual, where either video or sensory information is sent back to be displayed. This is why these systems can be considered to be open loop control systems. Where as in our system, the loop is closed once we include force feedback, and it is considered now a closed loop control system. Moreover, the above examples are either autonomous or tele-programmed, which means the operator gives a high level command for the robot and then the robot executes

[3]IEEE International Conference on Robotics and Automation, Internet Robotics Workshop.
[4]http://ranier.oact.hq.nasa.gov/telerobotics_page/realrobots.html

at a different time. Then the operator can either wait or check later on and see the status of the operation. As for our case, we have real-time control and feedback, where the commands are executed once received and feedback sent directly. In none of the cases that we examined did we find a kinesthetically coupled operator with the environment, which would increase the sense of telepresence. Another limitation is assumptions concerning time delay, no work was found dealing with real-time control with force feedback in the presence of random time delay.

1.6 Outline

We outline here the remaining parts of the thesis:

1. Chapter 2 examines the Internet characteristics when viewed as a real-time control link. We study the Internet behavior and what it causes in a feedback system. We look at the "randomness" of time delay and the instability that it causes.

2. Chapter 3 gives an introduction to the event-based control, which uses an event as a reference and not time. Then the modeling and stability analysis of our system is presented, where the general structure of the system and the dynamics equations of each part is given. In addition, the stability of the system is proven using a non-time based control approach.

3. Chapter 4 provides the details of the implementation and experimentation. In this chapter we give the details of the hardware and software used in the experiments, then a detailed discussion about the different results obtained is presented.

4. Chapter 5 is the conclusion and a brief overview of future work and improvements.

CHAPTER 2

THE INTERNET: REAL-TIME CONTROL MEDIA

The use of the Internet as a communication link for Tele-robotics is not surprising regardless of its very weaknesses. The Internet can connect any two machines anywhere in the world or in space, that is why it is a natural media for real-time tele-operation. But once we start talking about real-time control there are several conditions to insure the stability and robustness of the system. The issue here is that the Internet does not satisfy all of these conditions. That is why in the next section we will look at the properties of the Internet from a real-time control perspective and examine their effect on the system.

2.1 Characteristics of Internet Communication

First we will discuss the services that the Internet provide for tele-operation. Considering the communication protocol TCP/IP the Internet can provide a *reliable* communication link. By reliable we mean a link that does not loss packets nor rearrange them. Many protocols do not insure this, but when TCP/IP is used that means we are given the packets in order and no packets are lost (as long as we are not disconnected).

What the Internet does not provide is a permanent link, since the connection might be lost at any time. And no matter how good your local area network is, once your on the Internet you are subject to all kinds of risks. You might be disconnected at any time and you might face indefinite random amount of delay; there is no upper bound on delay over the Internet. So the inter-arrival times of packets can not be estimated, for an example of this inter-arrival time Fig.2.1 gives a graph of inter-arrival times of groups of packets on the August 1989 Bellcore ethernet trace [37][38]. From this figure we can have an idea about the randomness of the delay.

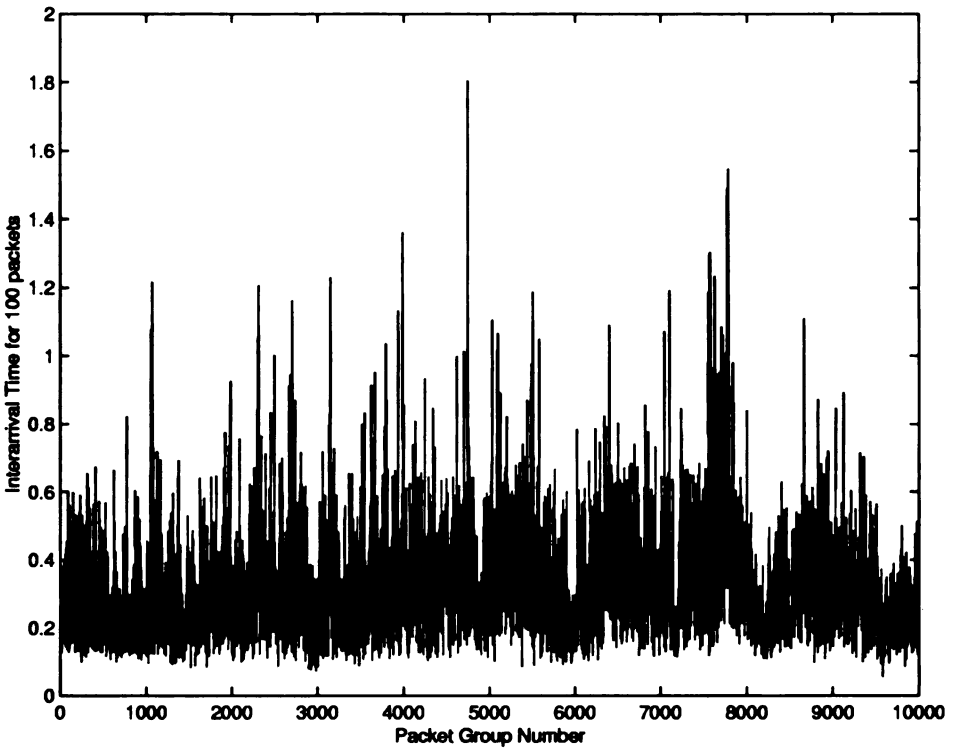


Figure 2.1: Inter-arrival times of groups of packets on the August 1989 Bellcore ethernet trace [37][38].

As for the specific network that we experimented on; we had two main communication setups. Either a station in the Robotics and Automation lab at Michigan State University was used to control the robot, or a machine in the Robot Control lab, at the department of mechanical and automation engineering at the Chinese university of Hong Kong[1], was used. Fig.2.2 and Fig.2.3 give a plot of the round trip time delay between the robot and the teleoperating stations at MSU and in Hong Kong respectively. As for Fig.2.5, it gives the frequency of operation; that is, the number of times per second that the robot receives a command and the operator feels force feedback during actual real-time control. From these plots we can see that the delay is *random* and can not be easily predicted; therefore, any assumption made about the time delay will be quit limiting.

Fig.2.3 and Fig.2.4 give plots of the round trip delay in ms during different days.

[1]distance between East Lansing, where the robot was, and Hong Kong is about 7824 miles

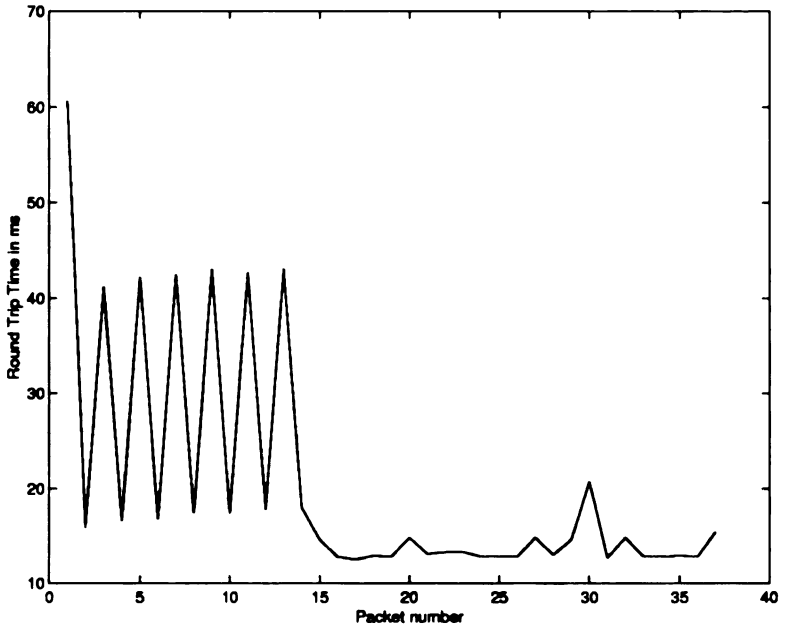


Figure 2.2: Round trip delay in ms for packets transmitted between the robot and MSU station.

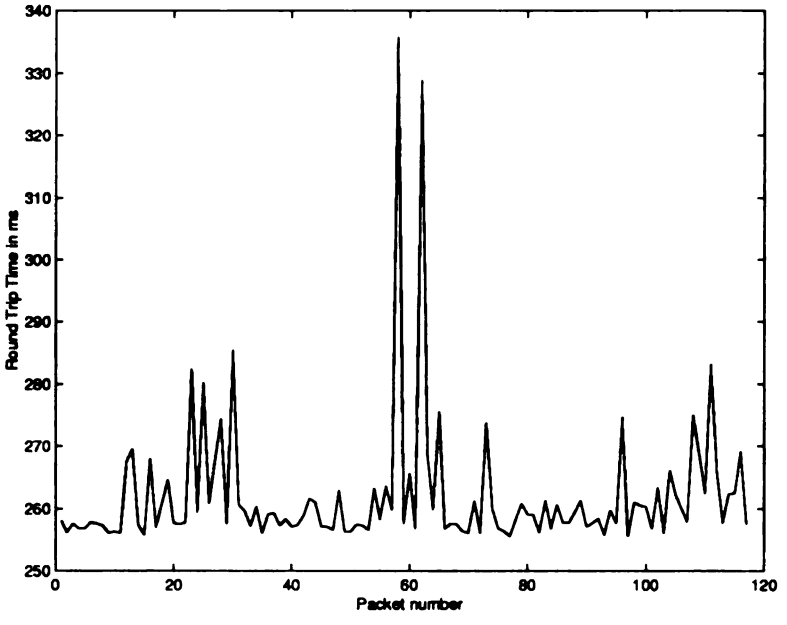


Figure 2.3: Round trip delay in ms for packets transmitted between the robot and Hong Kong station.

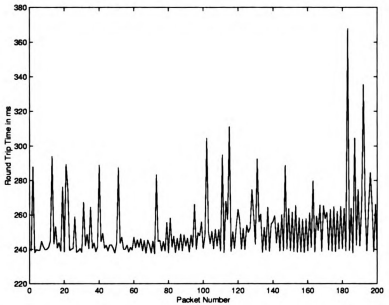


Figure 2.4: Round trip delay in ms for packets transmitted between the robot and Hong Kong station.

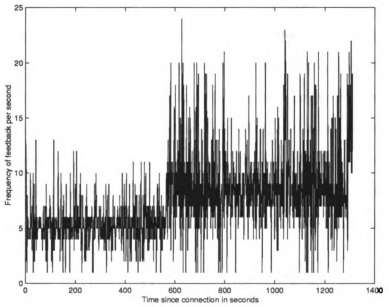


Figure 2.5: The frequency of control and feedback signals between the robot and Hong Kong station.

From these figures we can note that these delays change according to several factors; especially which time of day the experiment is being done, what day and what time of the year. Meaning that these delay can not be predicted, which makes the real time control over the Internet more difficult. This existence and randomness of delay in Internet communication has major effect on the real time control, and more so when force feedback is involved.

2.2 Effects of Time Delay on Real-Time Control

The main effects of these communication characteristics are, instability and desynchronization. In 1966, the first work dealing with force feedback with time delay was done (Ferrell 1966). It was shown that delays on the order of a tenth of a second were shown to destabilize the teleoperator. Keeping in mind that the delays we are getting and might expect could range from 100ms to a few seconds, we could expect that teleoperation systems via the Internet to be unstable unless a fix is found. As for de-synchronization, it is the most intuitive effect of delay, since the operator would have no idea about the current robot status. With visual feedback, by the time the operator sees an image it would be an *old* image of the robot. And taking decisions based on that would be risky and would cause the system to get to a completely different state. So while the operator thinks that the robot is in a certain location it would have already moved to a new one, causing by that the two to be de-synchronized in time and space. But these issues have no influence on the performance of our system. The reason for this immunity to delay is the use of non-time based control, which will be introduced in the next chapter.

The stability of our system will be proven in chapter3. In addition, chapter3 will introduce non-time based control, and will cover the detailed modeling of the different components of our system, including the robot and the operator.

CHAPTER 3

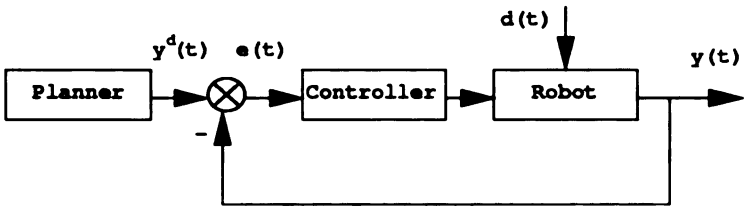
NON-TIME REFERENCED PLANNING AND CONTROL FOR INTERNET-BASED REAL-TIME TELE-OPERATION

In this chapter we introduce non-time based planning and control, and our system with its different components. We start in section3.1 by introducing the event based approach for control and planning, then section3.2 gives a general structure of the system with the detailed modeling of all the components. Then in section3.3 we will compare our model to other models used, specifically the one used in[20]. In section3.4 we analyze the system and discuss the event-based reference that was used, and we prove the stability of our approach.

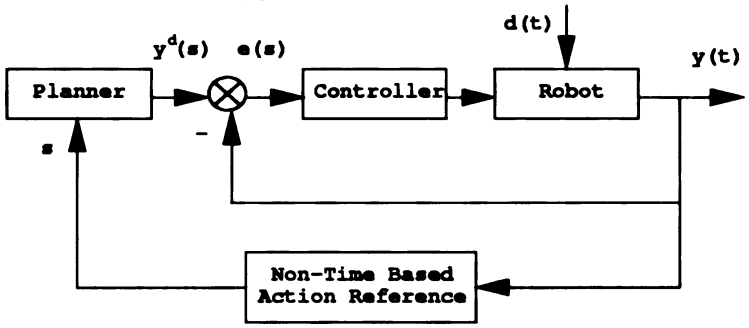
3.1 Non-Time Based Real-Time Control

The delay that exists in communication links have several effects on the stability and synchronization of the system. But this delay is resulting from the use of time as our reference variable; therefore, if we were able to use in our system a non-time based reference it would become immune to delay. This suitable action or motion non-time reference variable, is called event. The use of event-based planning and control provides the tele-operation system the capability to cope with uncertainties and delays in real time, without the need to replan or re-synchronize [25]-[27].

In traditional control systems the dynamics of the system is modeled by differential equations in which the free variable is the time variable t. And usually the trajectory is a function of time, but if we do not constraint the trajectory by time we would allow the system to be at any point at any time. So the general idea is to model the system and the trajectory with parametric equations. The parameter is called *motion reference or action reference*, and usually denoted by s [25]-[27]. The planning and



(a) traditional control



(b) non–time based control

Figure 3.1: The comparison between traditional time-based and event-based planning and control.

control of the traditional time-based and the event-based schemes is shown in Fig.3.1.

The Action Reference block in Fig.3.1 acts like a clock for the system. It is a map from the output or state of the robotic system y to a scalar variable s, and the robot plan as well as the operator commands can be described as functions of this variable. This reference is usually taken to be a physical quantity; distance, position. In our case we will use a different approach, we will use an imaginary reference because in real time teleoperation we do not have a predefined path or trajectory to follow.

This new approach, which was first introduced in [43], has been used in many studies and several robotic applications[26], and have been proven to be very efficient in dealing with uncertainties and delay. But this is the first time that this approach will used for real-time control with force feedback.

3.2 Modeling of The System

We include here the details of the modeling of the system. We will include in Fig.3.2 a general block diagram of a traditional tele-operation system, and then in Fig.3.3 we

present a detailed schematic diagram of our system. The modeling of the blocks in Fig.3.2 and Fig.3.3 will be discussed in details and each term is explained in Table3.1.

As Fig.3.2 shows, the general tele-operation system consists of a Human operator, master[1], communication network, Slave[2], and the environment. The same applies to our system as shown in Fig.3.3 but with some modifications with the connections.

Now we will model each block in our system giving the details of each variable and the equations that relate them. The thing to note is that all the models and dynamic equations are in terms of our event s, and that we are controlling a mobile robot with 3 degrees of freedom, which makes our variables vectors with 3 terms.

1. Human Operator: This was the hardest part to model, but eventually we assumed a spring-like behavior which was shown in [29] and used in several places in the literature [28]. So inspite of all the internal complexity the wrist has a spring-like effect. In addition, the human can compensate for certain machine instabilities making the coupled human-machine system stable [30]-[33]. So once the operator feels a force he will generate a new joystick position according to the following:

$$X_m(s) = \frac{F_h(s-1)}{K_h} \tag{3.1}$$

where K_h is a constant and X_m and F_h are:

$$X_m(s) = \begin{bmatrix} X_{mx}(s) \\ X_{my}(s) \\ X_{m\theta}(s) \end{bmatrix} \quad F_h(s) = \begin{bmatrix} F_{hx}(s) \\ F_{hy}(s) \\ F_{h\theta}(s) \end{bmatrix} \tag{3.2}$$

As table3.1 indicates $F_h(s)$ is the applied force, which means the force that the

[1]Usually the master is a joystick.
[2]Usually the slave is a robot.

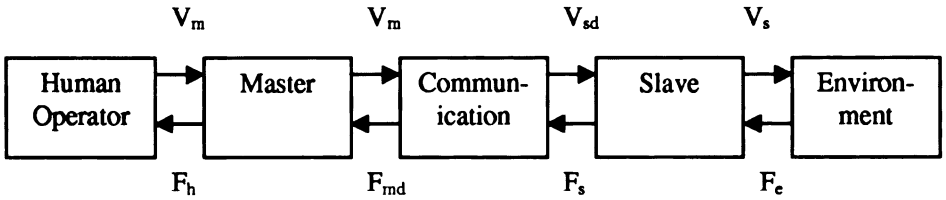


Figure 3.2: Block diagram of a traditional tele-operation system.

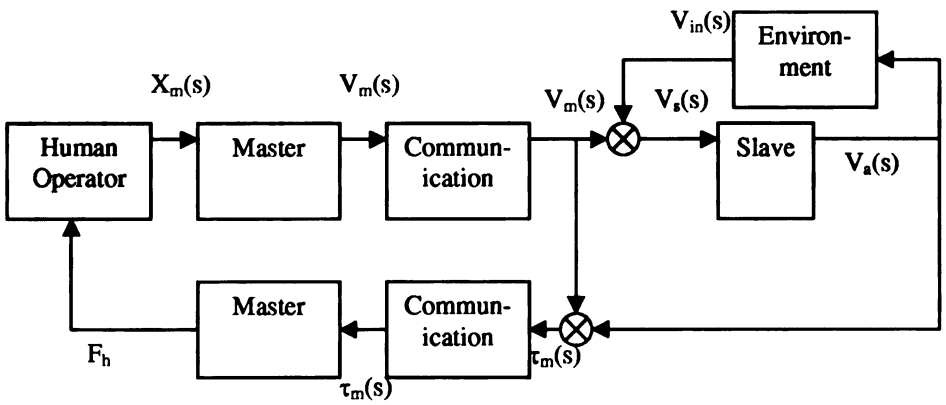


Figure 3.3: Detailed block diagram of our tele-operation system.

Block	Traditional Variables	Our Variables
Human Operator	F_h: applied force	F_h: applied force X_m: position of Joystick
Master	V_m: velocity desired	V_m: velocity desired
Communication Link	F_{md}: desired force V_{sd}: desired velocity	
Slave	V_s: velocity F_s: force	V_a: Actual velocity of robot τ_m: desired force
Environment	F_e: contact force	V_{in}: virtual contact effect V_s: velocity set for the robot

Table 3.1: The Explanation of the various variables in the Tele-operation systems.

operator feels (Newton's third law). Thus, the x and y components are due to the force fed back and due to the additional force required to get the joystick to the new location. Since we do not feed back force in the θ direction, this component is just a result of getting the joystick to the next location. As we see in eq.3.1 $X_m(s)$ is related to $F_h(s-1)$, so $X_m(s)$ at event s is generated by the previous force at event $s-1$. This results in having an event-based system where each event is triggered by the previous one. Several other issues related to human behavior where taken into account in the design and these will be discussed in chapter4.

2. Master (Joystick):The dynamics of the joystick are described in the following equation:

$$M_m \dot{V}_{mm}(s+1) = F_h(s) + \tau_m(s) \tag{3.3}$$

where M_m is the mass of the joystick handle, V_{mm} is velocity of joystick movement, F_h is as described before and τ_m is the feedback from the robot and which would be the force played by the joystick. The result from this dynamics is the joystick getting to a new position $X_m(s+1)$, from this position the desired velocity V_m is derived according to:

$$V_m(s) = K_m X_m(s) \tag{3.4}$$

where K_m is a scaling constant, $X_m(s)$ as before and $V_m(s)$ is the desired velocity

of the robot. The different vectors are:

$$V_m(s) = \begin{bmatrix} V_{mx}(s) \\ V_{my}(s) \\ V_{m\theta}(s) \end{bmatrix} \quad \tau_m(s) = \begin{bmatrix} \tau_{mx}(s) \\ \tau_{my}(s) \\ 0 \end{bmatrix} \tag{3.5}$$

From 3.5 we see that $\tau_{m\theta}(s) = 0$, since we do not feedback force in the rotational direction.

3. Communication Block (Internet): Resulting from event based control, the communication link is simply a delay element that plays no role in the modeling of the system. We assume that the Internet is simply supplying the link and in case this connection is lost, the system will simply stop awaiting the connection. Since the advance of time does not affect our system and only the advance of the event s will, when the connection is lost the system will still be stable and would resume action after the connection is back. This makes the system very robust since no initialization or synchronization is required.

4. Environment: In our system we interact with the environment in a new way. We do not have contact to generate force, we actually use the different sensors around the robot to detect objects. Based on our distance to the objects in our way, we generate a *virtual force.* This is done by having a function of distance $f(d)$[3] that would give us a velocity value $V_{in}(s)$ to subtract from the desired velocity $V_m(s)$.

$$V_s(s) = V_m(s) - V_{in}(s) \tag{3.6}$$

where $V_s(s)$ is the velocity set to the robot, $V_{in}(s)$ is the effect of the environment

[3]Details of this function are included in section4.1.2.

and $V_m(s)$ as before. So now the desired velocity that the robot gets is less than the actual one, this slowing down will be used to generate the virtual force feedback. Keep in mind that we do not generate force in the rotational direction that is why $V_{in\theta}(s) = 0$ and $V_{s\theta}(s) = V_{m\theta}(s)$, as seen in eq.3.7.

$$V_s(s) = \begin{bmatrix} V_{sx}(s) \\ V_{sy}(s) \\ V_{m\theta}(s) \end{bmatrix} \quad V_{in}(s) = \begin{bmatrix} V_{inx}(s) \\ V_{iny}(s) \\ 0 \end{bmatrix} \tag{3.7}$$

5. Slave (Mobile robot): Once the robot receives $V_s(s)$ it will be commanded to move with that velocity but would actually have $V_a(s)$ as its velocity. Then the robot will calculate the velocity tracking error with respect to original desired velocity $V_m(s)$ and send that back to the master to be played as the virtual force due to the environment. So $\tau_m(s)$ and the dynamics of the robot are:

$$\tau_m(s) = K_\tau(V_m(s) - V_a(s)) \tag{3.8}$$

$$M_s\dot{V}_a(s) = F_e(s) + \tau_s(s) \tag{3.9}$$

$$\tau_s(s) = -\gamma V_a(s) + KV_{err}(s) - \alpha_f F_e(s) \tag{3.10}$$

$$V_{err}(s) = V_s(s) - V_a(s) \tag{3.11}$$

$$V_a(s) = \begin{bmatrix} V_{ax}(s) \\ V_{ay}(s) \\ V_{m\theta}(s) \end{bmatrix} \tag{3.12}$$

K_τ, γ, K and α are constants, M_s is mass of robot, F_e is the actual environment

forces if any and usually assumed very small. τ_s and V_{err} are robot internal controller terms. Eq.3.8 shows us that what the operator is feeling is actually the velocity tracking error, which could be a result of getting close to an object, this implies that from the direction and magnitude of the force we can know where and how far are the obstacles to the robot. The important point here is that the operator will still feel force in case of actual force being applied to the robot during contact. In the actual contact case $V_a(s)$ would decrease by that increasing $\tau_m(s)$. Moreover; we take $V_{a\theta}(s) = V_{m\theta}(s)$ since we have $\tau_{m\theta} = 0$.

3.3 Comparison with Other Models and Approaches

It is very difficult to compare our model to the other approaches used in teleoperation in general. Because the approach we use is quit different and many of the presented systems are not even modeled. Moreover, our system has more complex parts that deal with obstacle avoidance, which will be discussed in more details later. There are some common methods used to model and analyze teleoperation systems, the most popular ones are networks, wave variables and dynamic equations. The model and analysis we examine here is based on network modeling and was used by Anderson in [20] and [21].

Anderson bases his analysis on the fact that for the system to be stable, it should be passive. Using the analogy between mechanical and electrical systems he represented a teleoperator as a network, as shown in Fig.3.4. In this model, the master, communication block, and slave are represented by two-port, as for the operator and environment they are represented by one-port. An n-port is characterized by the relationship between effort F (force, voltage), and flow v (velocity, current). This relationship for one-port is specified by its impedance $Z(s)$ according to:

$$F(s) = Z(s)v(s) \tag{3.13}$$

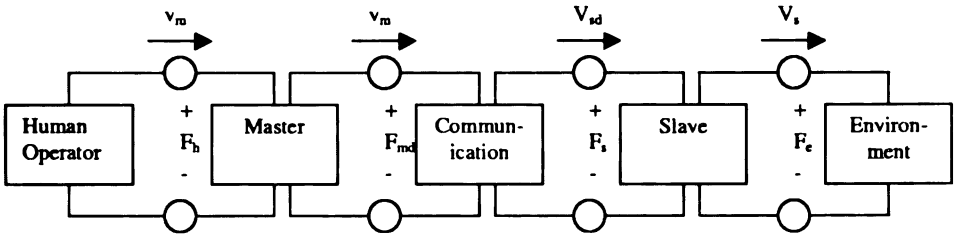


Figure 3.4: Network representation of teleoperator, the different terms are as explained in table3.1.

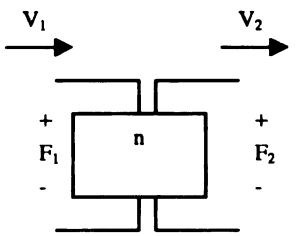


Figure 3.5: Transformer.

where $F(s)$, $v(s)$ are the Laplace transforms of $F(t)$ and $v(t)$. As for the two-port, the relationship is:

$$\begin{bmatrix} F_1(s) \\ -v_2(s) \end{bmatrix} = \begin{bmatrix} h_{11}(s)h_{21}(s) \\ h_{12}(s)h_{22}(s) \end{bmatrix} \begin{bmatrix} v_1(s) \\ F_2(s) \end{bmatrix} = H(s) \begin{bmatrix} v_1(s) \\ F_2(s) \end{bmatrix} \tag{3.14}$$

where $H(s)$ is the *hybrid matrix* and F_1, F_2, v_1 and v_2 are as defined in Fig.3.5.

For the system to be stable, its different components have to be passive. That is, they can dissipate energy but can not increase the total energy of the system they are part of. This is why Anderson modeled all the components of the system using passive circuit elements. And then to achieve a stable system under time delay, he choose a control law that would make the characteristics of the communication block identical to a two-port lossless transmission line, which is a passive element. The derived control law for the communication circuit is:

$$F_{md}(t) = F_s(t-T) - v_{sd}(t-T) + v_m(t) \quad (3.15)$$

$$v_{sd} = v_m(t-T) - F_s(t) + F_{md}(t-T) \quad (3.16)$$

where T is delay, and the other terms are explained in table3.1. Comparing this approach and model to ours, we find several differences. The stability analysis in this case relies on the specific model used, whereas in our case the stability analysis is done independent of the model. Moreover, the model here assumes same amount of delay in both directions of the communication; in our model we do not make any assumptions regarding the delay. In addition, the human model here is assumed to be passive but in our case it is not.

3.4 Stability Analysis

Several methods have been used to analyze the stability of tele-operation systems, scattering theory [20][21], wave variables [22]. The main concept used in their analysis is based on proving that the system is passive. A passive system can not create energy, and thus from a control point of view a passive system can not go unstable [20]. Our approach is slightly different, and the first difference is our use of an event based control. Concerning the stability of the system under event-based referencing, the following theorem was proven in [25]-[27]:

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

The only assumption we need to make is that the robot is a stable system, which means that the original robot dynamic system (without remote human operator) is asymptotically stable with t as its action reference. This would allow us to use theorem1 and prove the (asymptotical) stability of our system with respect to the new action reference s, simply by proving that the new non-time action reference is (monotone increasing) nondecreasing function of time t. The advantage of this approach is that stability is proven independent of the human model or the statistics of time-delay. Meaning, the system would be stable for random time delay and regardless of the operator.

The major difficulty of selecting s was the many uncertainties in the system. The main uncertainties were, the path (trajectory) and the environment. Usually the event s is taken to be a function of the physical output; for example, the distance to the origin, the angle, the absolute position. But, in tele-operation, usually non of these parameters is defined. In our case the selection of the event will be clear after we explain the flow of commands and feedback on our system. And this is the first time this specific approach have been used for event-based tele-operation.

This is how the control and feedback commands are communicated; the operator places the joystick in a certain position that corresponds to velocity vector, the behavior of the joystick is similar to the gas pedal in the car. This vector is sent to the sensing unit on the robot. The Sensing unit scans the environment and based on the position of obstacles the velocity is reduced and sent to the robot motors to be executed. The motors will execute the command, after that the actual velocity is calculated. Then the actual velocity is subtracted from the original velocity required by the operator, and this difference is sent back to the joystick motor to be played as force. The important thing in this operation is that non of these step can be executed out of order, each event is triggered but the end of the previous one. And what should be clear here is that during all this time, although the operator could move the joy-

stick but his commands will not be sampled by the joystick until feedback is received and played. This might seem to cause instability or de-synchronization, but it does not. On the contrary, every force the operator feels, he can be sure that it is the most recent one and that is this the current status of the robot, so he will be always synchronized with the robot. In addition, he can be sure that his new command is the next one to be executed and no other commands are pending. So for example, the operator sends a series of commands $\{V_1, V_2, \cdots, V_n\}$, V_1 will be sent to the robot and feedback will be awaited. Until feedback is received, no other command will be sent, so $\{V_2, \cdots, V_n\}$ will actually be discarded by the system. When feedback is sensed, the operator would assume that its the result of $\{V_1, V_2, \cdots, V_n\}$ but actually its only the result of V_1. Inspite of this all the system components are synchronized and stable, because no commands and no feedback is flowing in the system. Now the operator based on his feeling can take a decision and send a new set of velocities, and the whole sequence repeats. So the control and feedback speed varies depending on the time-delay and the speed of the different machines used but will have no effect on the stability of the system. Based on this control algorithm a natural selection of the event variable s was some kind of a counter. In our case, it is taken to be the number of forces felt by the operator since the connection was established. This way the operator would know when the reference increments and the robot would know the exact s since it corresponds to the number of commands executed till now.

The operator might not know the exact value of s but this is not required since knowing when it is incrementing would be enough to inform the operator that the system is advancing to the next step. Its obvious that s is nondecreasing function of time t, since at any time the number of forces felt or commands executed can not decrease. We can run a new command or feel a new force but we can not *unfeel* or *undo* an execution.

This approach constitutes a general method to designing and implementing event based tele-operation systems. And based on this design the system stability, synchronization and robustness is guaranteed, since the proof given above is not based on a particular model or system. So our frame work applies to any tele-operation system, regardless of the robot used.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTATION

The implementation of the project was mainly done in house. We have not seen in the literature a similar implementation, with the specific hardware used. In the next section we present the details of the system structure and the implementation details and difficulties. After that we present the experimental results of our work.

4.1 System Implementation

We can divide the implementation into two parts; hardware and software. In hardware, we will detail the different systems used and their specification. In addition, we will describe their connection and communication. As for software, the main structure and the different modules will be examined. Then we will explain how obstacle avoidance was achieved and how the virtual force was generated.

4.1.1 Hardware

The diversity of the hardware used caused some problems in the integration of all the parts. As we see in Fig.4.1 and table4.1, the system consists of different operating systems and different configurations. So the problem of interconnection had to be studied carefully. The system consists of:

1. Joystick: Programmable force feedback joystick, which is used to obtain commands and play force. Joystick is connected to the local PC with WIN98 operating system.

2. Local PC: Used to control the joystick and interface to it, and to communicate over the Internet with the mobile robot.

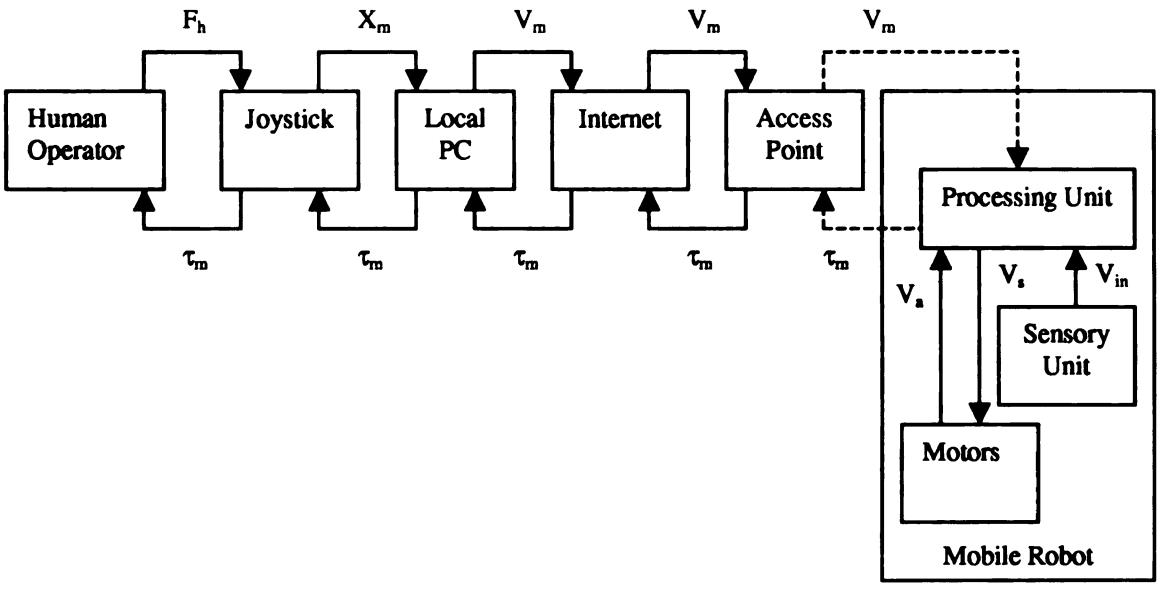


Figure 4.1: Hardware structure of system.

3. Access point: Makes up the link between the Internet and the mobile robot via wireless communication.

4. Mobile Robot: A mobile robot that is equipped with several sensors (Infrared, Ultra sonic and Bumper), which are used to detect and avoid obstacles. The robot will execute the velocity commands depending on the surrounding environment and then calculates the velocity error and sends that back to the joystick.

5. Camera: The robot has a camera mounted on top to allow visual feedback to the operator also through the Internet.

4.1.2 Software

The development of the software took a considerable amount of time because of the complexity of the systems and their diversity. The main issue was the ability to have two different operating systems, Linux and WIN98, communicating together. Here came the role of the communication protocol TCP/IP, which can be understood by

Device	Specification
Joystick	Microsoft SideWinder Force Feedback Pro. 3 degrees of freedom
Local PC	PentiumII 300MHZ, 128MB, WIN98 Interfaces to the joystick through the game port and to the Internet through an ethernet card.
Access Point	Proxim RangeLan2 radio access point Connects the robot to the Internet
Mobile Robot Called *ARONE*	Nomadic XR4000, 2 PCs on board Pentium 233MHZ, Linux Interfaces to the Internet through wireless ethernet card Equipped with 3 kinds of sensors, Infrared, Ultra sonic and Bumper
Camera	Sony EVI-D30 camera Connected to the robot through a frame grabber

Table 4.1: Specifications of the system hardware.

both operating systems. So the Internet was not only acting as a communication link but also as a translator. The software developed can be divided into three main parts; motion server, camera server and client. Motion and camera servers run on the robot and where developed under C++; where as the client, runs on the local machine and was written in Microsoft visual C++.

1. Motion Server: In Internet communication usually we have 2 kinds of processes, servers and clients. Servers job is to wait for a client to connect to it and request a service. In our case the service is moving the robot and getting back feedback. As seen in Fig.4.2, this is mainly what the motion server does. Except that the server does not execute the request blindly, it first checks the sensors and based on that a decision is made according to the flow chart in Fig.4.3 that will be explained in more details in the next section.

   After the velocity to be set is decided, it is sent to the motors for execution. Then the server checks the actual velocity of the robot and subtracts that from the original desired one V_m and sends it back to the client as force feedback. To

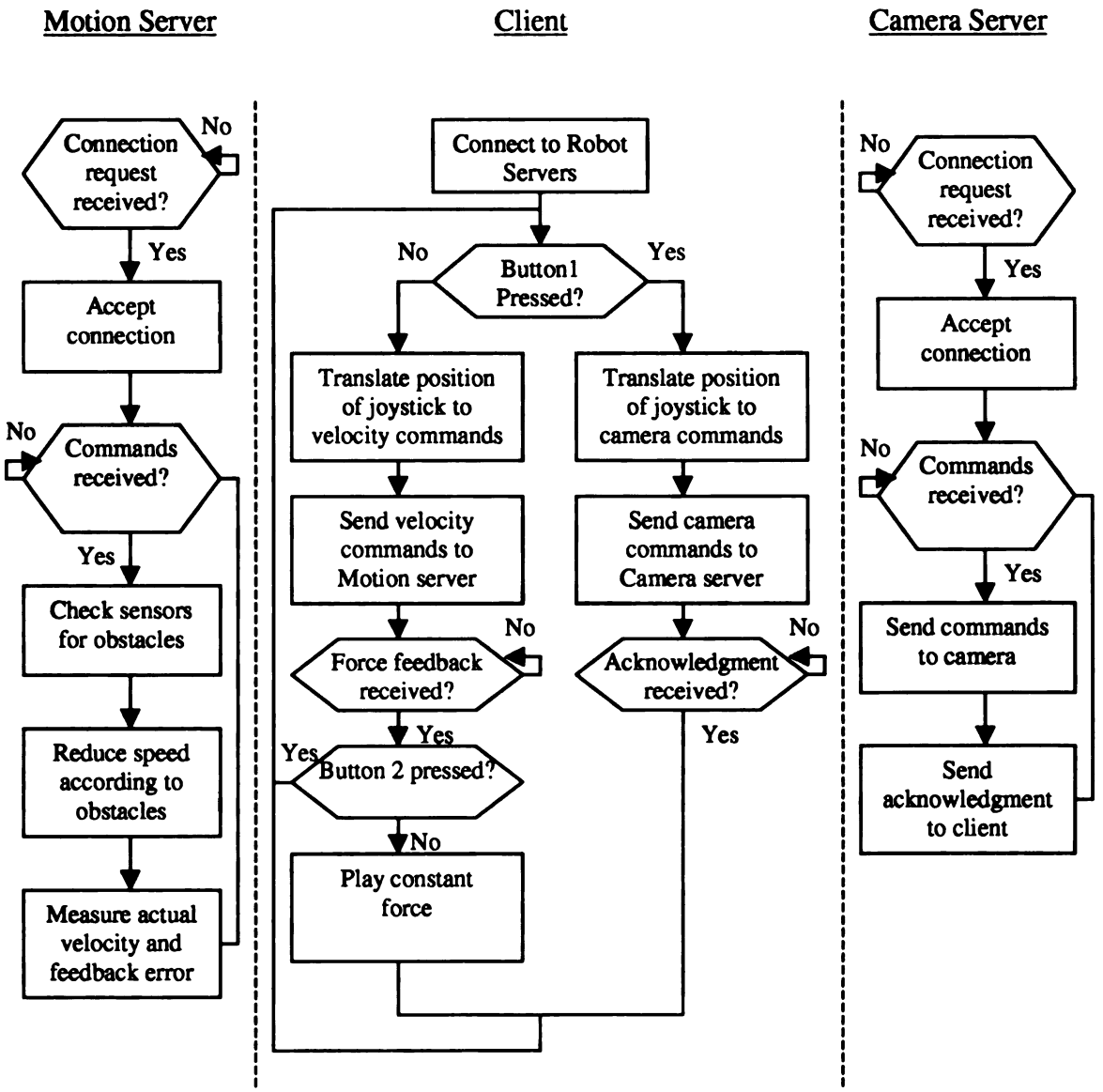


Figure 4.2: Motion server, Client, and Camera server general flow chart.

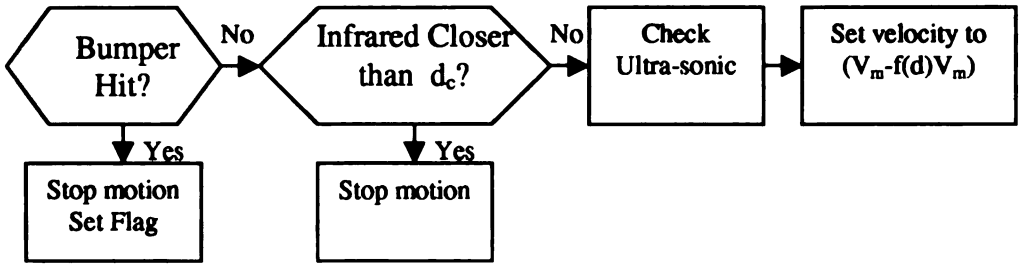


Figure 4.3: Flow chart for deciding velocity based on sensors.

over come the problem of disconnection, the server would execute the velocities for $250ms$; after which, if no new command is received, it would time out and stop moving waiting for the next command.

2. Camera Server: This server is much simpler than the motion server, its job is to simply interface to the camera. When the client sends camera commands (pan, tilt and zoom), the camera server sends these commands to the camera through the serial port and sends back to the client a confirmation that it has received the request. During all of this, even when the client is not requesting camera motion, a process would be running in the background sampling the camera as fast as possible and displaying these images on a web page[1] to be used as visual feedback for the operator. The image is updated on the web page at a frequency of 1 frame per second.

3. Client: This is the program that runs on the local machine and communicates with both servers and the joystick. The client sends commands to either of the servers based on one of the buttons, whether it is pressed or not. If the button is not pressed the joystick position will be translated into velocity commands and sent to the motion server. On the other hand, if the button is pressed, the position of the joystick is translated into pan and tilt commands for the camera and 2 other buttons specify whether zoom in or out is requested. In addition, based on which state the client is in, it will either send the force command back to the joystick once feedback is received or just wait for acknowledgment. The communication with the joystick is done with DirectX technology, and with the servers its done over the Internet. The client has a few other convenient options, for example force effect can be turned off, and disconnection can be done with one button. Another client implementation trick was to delay the sampling of the next command, so when feedback is received and force played,

[1] arone.egr.msu.edu

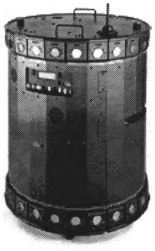


Figure 4.4: The XR4000.

the client would wait $250ms$ and then sample the position of the joystick. This way the operator is given time to feel the force and plan the next step based on that. Moreover; the force played is actually the average of the last 10 forces received, this way the high frequency changes in the force will be filtered giving the operator a smoother feeling of force with no abrupt changes.

4.1.3 Obstacle Avoidance and Virtual Force Generation

Obstacle avoidance was a very important part of the project that affects the behavior of the whole system, that is why we will discuss it in details here. Because of delay we had to make the robot more intelligent, since the operator might be seeing an old image whereas the robot is very close to objects. This might cause a collision, and to avoid this we had to keep the final decision of the velocity set to the robot since the sensory unit are the ones with the most up to date environment information. We will first give a detailed explanation about the sensors available and there distribution on the body of the robot, then we will explain their use for obstacle avoidance and force generation.

As we see from Fig.4.4 the XR4000 is a barrel shaped mobile robot. It is equipped

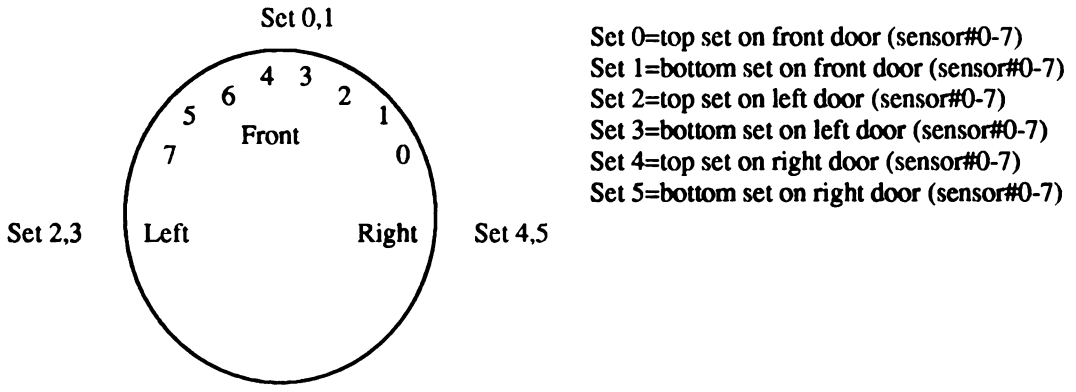


Figure 4.5: A top view of the robot that gives the distribution and numbering of the different sensors.

with 3 kinds of sensors:

1. Tactile Sensors (Bumper): They provide information about the physical contact with the environment. It is hoped that our obstacle avoidance would work, but since nothing is perfect these sensors will be used to detect any actual contact. The Nomad XR4000 has 48 bi-level tactile sensors that surround its top and bottom perimeters. Additionally, it has 4 door bumpers on each door that sense contact between the top and the bottom perimeters. The sensors are divided into 6 sets, 2 per each of the 3 doors on ARONE. Fig.4.5 gives the distribution and numbering of these sensors[44].

2. Infrared Proximity Sensors: Give a range information to nearby objects (typically less than 30 to 50 cm away). The range is determined by emitting infrared energy using high-current LEDs and sensing the amount of returned energy with infrared photodiodes. The thing to note is that the returned energy is a function of the object's reflectivity, that is why these sensors are not very good for the measurement of distances. The robot has 48 of these sensors distributes just like the tactile sensors, in 6 sets and have the same numbering shown in Fig.4.5[44].

3. Sonar Proximity Sensors: These provide range information to objects that are

relatively far away (between 15 and 700 cm away). The distance is calculated by multiplying the speed of sound by the time of flight of a short ultrasonic pulse traveling to and from a nearby object. The robot has 48 of these sensors distributes just like the others, in 6 sets and have the same numbering shown in Fig.4.5[44].

As for obstacle avoidance and force generation, it is accomplished according to the flow chart in Fig.4.3. First all the bumper sensors are checked, which gives 360 degrees coverage around the robot. If any is hit, the motion is stopped and a flag is set and sent back to the joystick where a sudden jerking force is played to tell that a bump has occurred. If non is hit, the infrared sensors, 90 from both sides of the desired motion, are checked. If any, detects an object closer than d_c, which is a predefined programmable critical distance, the motion is stopped. If it is non of these cases, the ultra-sonic sensors are checked. Again we check 90 from both sides of the desired velocity. Based on the one that gives the closest distance d, we set the velocity to V_s, which is given in eq.3.6, where $V_{in} = f(d)V_m$. So the velocity set would be a fraction of the desired one based on the predefined programmable function $f(d)$. $f(d)$ can be any function of distance depending on how conservative or risky you want to be. In our case f(d) is plotted in Fig.4.6. So the robot will slow down linearly when an object is detected between 0.5m and 1m, then it would stop if an object is detected closer than 0.5m. As seen in Fig.4.7, we end up with two regions around the robot, one that would cause the robot to slow down and another to completely stop. Now the force is generated by measuring the actual velocity of the robot and taking the difference between that and the desired and sending it back to the joystick. To avoid the problem of disconnection and long delay, the robot would move for $250ms$ only. After these $250ms$, in case no new command is received, the robot would stop and wait.

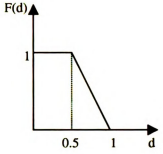


Figure 4.6: The fraction of velocity that is deducted based on the distance d in meters to the closest obstacle.

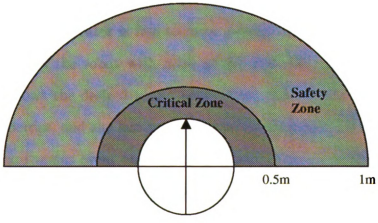


Figure 4.7: The two safety regions around the robot according to the direction of motion.

4.2 Experimental Results

Several experiments where done during the duration of the project, we will include the most important and relevant to our discussion. Two kinds of experiments where done, one is where we control the robot over the same LAN, and another is where the robot was controlled from Hong Kong.

4.2.1 Tele-operation Over the Same LAN

There was 3 scenarios for the control over the same network; first case is normal operation, second case is where we induced 2 seconds of round trip delay and third is when we had simulated random time delay. The results of these three cases are plotted in Fig.4.9, Fig.4.12 and Fig.4.15 consecutively, and will be discussed in more details. But first let us examine the performance of the system in a friendly environment with relatively far obstacle, and let us see how close the robot actually tracks the desired velocity and if it is stable. In this experiment, the operator tries to keep away from obstacles (about $1m$ away), but with no other constrains. Meaning there is no predefined path and the operator is free to move around randomly. Fig.4.8 shows a plot of the operation of the robot with most of the obstacles being more than $1m$ away. The figure shows the desired x and y velocities, then under them we see the actual x and y velocities, and the last row gives the distance detected at each event. All of these plots are versus event and not time, since that is our reference. We can see that we have a very close tracking of the desired velocities, except when the robot gets to obstacles which are closer than a meter, which is what we expect. So when so obstacles are found the system is showing a close tracking performance and is stable.

When the environment becomes more *hostile*, that is having several close objects, the behavior of the system changes. Since now we have obstacle avoidance taking place. In all the following experiments, the operator is not constrained, complete

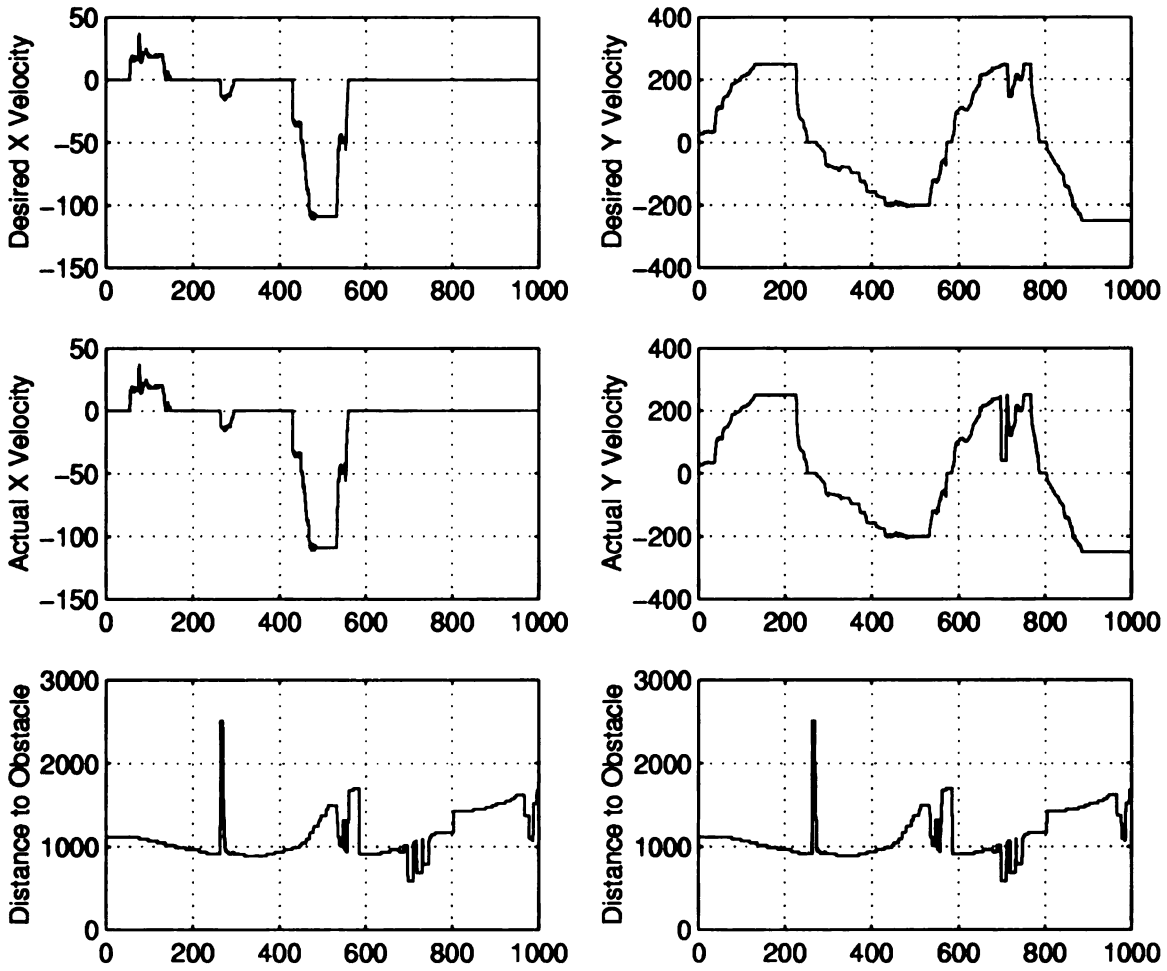


Figure 4.8: The behavior of the system under normal operation with relatively far obstacles.

freedom is given to move around in any direction or fashion and is allowed to get close to obstacle as desired. To understand the system operation under these conditions let us consider Fig.4.9. In the first row we have a plot of time versus s, the event. Its clear that s is nondecreasing function of time, which was very crucial in the proof of the system stability. The other plot in the first row is the desired rotational velocity. The second row is the desired velocities in x and y directions, which we call V_m. The third row displays the actual velocities in the both directions, which we call V_a. Next we give the force that is played by the joystick in both directions, we call this τ_m. The last row displays the same plot, which is the closest distance detected to obstacles in the environment. To get an idea about the performance and behavior of the system let us consider all these plots together. Looking at the actual velocity we see that it is changing according to the distance detected, for example if the distance gets closer the actual velocity decreases, when this happens we see that the force increase, which is what we expect. Thus a close object causes the robot tracking error to increase by that telling the operator that an obstacle exists. Whenever the robot gets to the critical distance of $0.5m$ the actual velocity becomes zero, and this applies to both x and y directions. At the time when the robot gets further away from obstacles the actual velocity starts tracking the desired one again, by that reducing the tracking error and therefore the force. So looking at the sum of the actual velocity and the force plots, they should make up the desired velocity. This is what Fig.4.10 is showing, the first row is the desired velocities and the second row is the sum of the force and the actual velocities. It is clear that the plots are similar although not identical. The difference is due to the fact that the force we play is a filtered version of the tracking error as seen in Fig.4.11. In this figure we show the tracking error and the actual force felt, which is clearly a low pass filtered version of the error. As we mentioned before, this is done so that the force felt is more smooth and no abrupt changes are felt. That is why the plots in Fig.4.10 are not identical, since whenever the tracking

error changes suddenly the force would change slowly, and that is why the differences in these plots are at the points where the actual velocity is directly reduced to zero, by that causing a big sudden increase in the tracking error.

As for the second case, where we induce in the system an additional round trip delay of 2 seconds, the results are shown in Fig.4.12. The plots are the same as the ones shown in Fig.4.9, which were explained before. These plots show that the system have a similar behavior to the case where no additional delay exists. The actual velocity tracks the desired one until an obstacle becomes close Then the actual velocity starts deviating from the desired one, by that increasing the tracking error. This increase would increase the force felt. The opposite happens when the robot is getting away from an obstacle, the tracking becomes closer and the error, thus the force, decreases. Again adding the force and actual velocity would give a result similar to the desired velocity as seen in Fig.4.13. The cause of the differences is again the filtering that was mentioned before and seen in Fig.4.14.

The third case is when we simulate a random time delay, that can range from 1 to 4 seconds. The results of one of these experiments is shown in Fig.4.15. The same thing that we saw for the previous two cases is displayed here. The tracking error, that is the force, is a function of the distance detected. The closer the distance the more force is felt and the slower the robot moves, until it comes to a complete stop. Then as the distance increases, the force reduces and the robot follows the commands closer. A plot of the sum of the force and the actual velocity is shown in Fig.4.16. Here too the errors are due to the filtering illustrated in Fig.4.17.

Considering all these cases, we can conclude that the robot is stable as proven before. And more importantly, is that this stability is unaffected by the time delay, which can be constant or random and can take any value. Not only the robot is stable, but also has a very close tracking performance, where the actual velocities track the desired ones very closely as long as there are no obstacles. Once obstacles

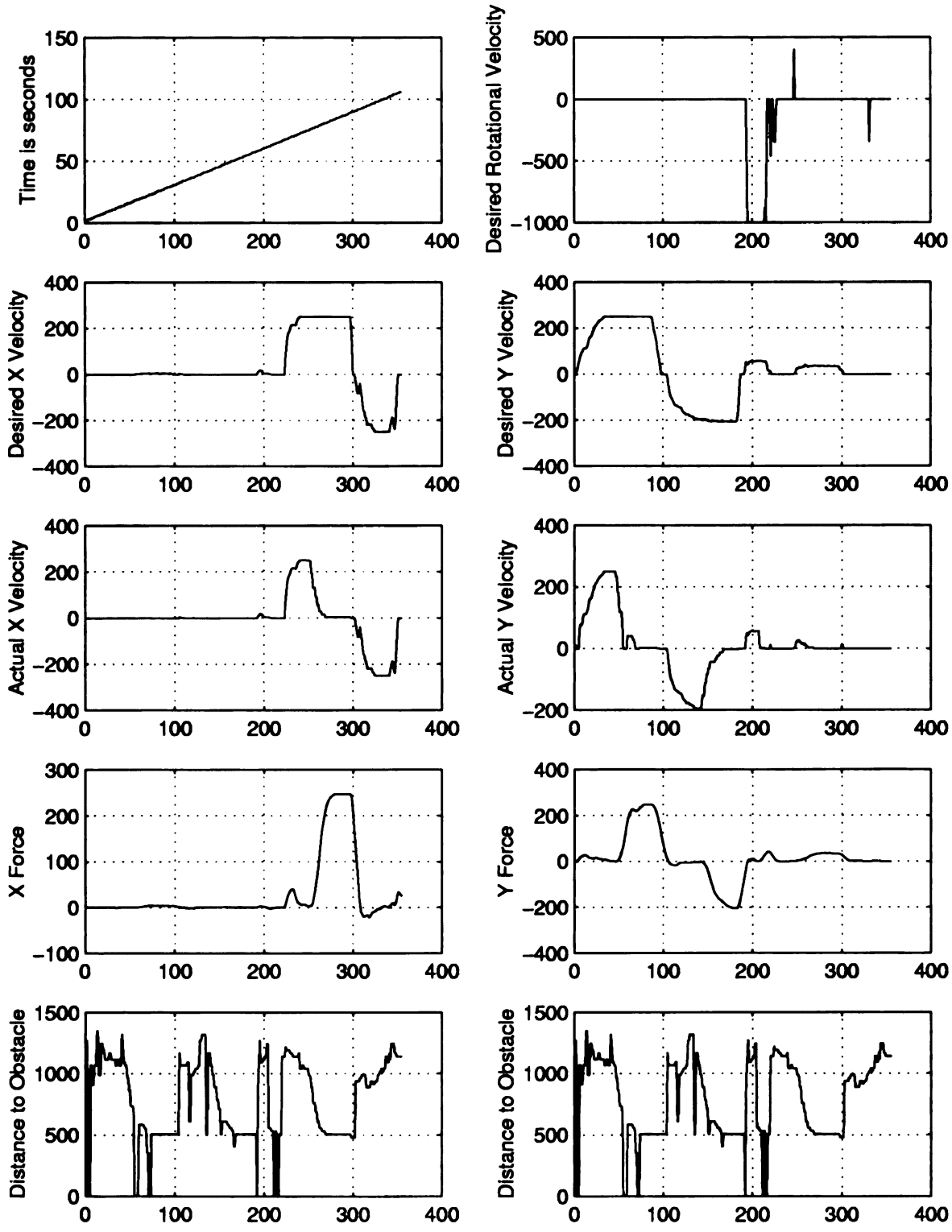


Figure 4.9: The behavior of the system under normal operation.

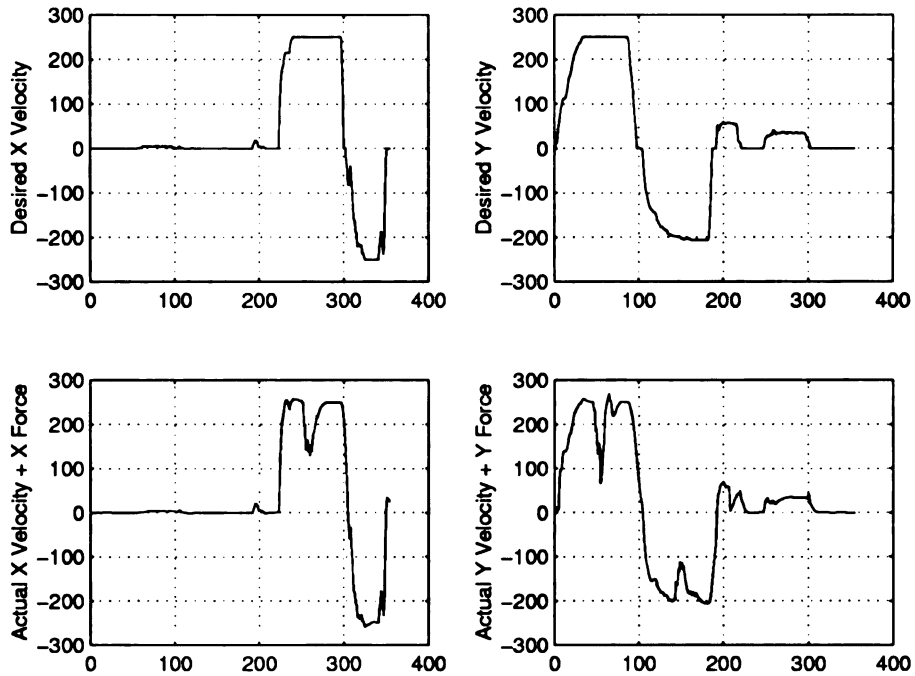


Figure 4.10: Comparison between the desired velocities and the sum of the actual velocities and the force felt.

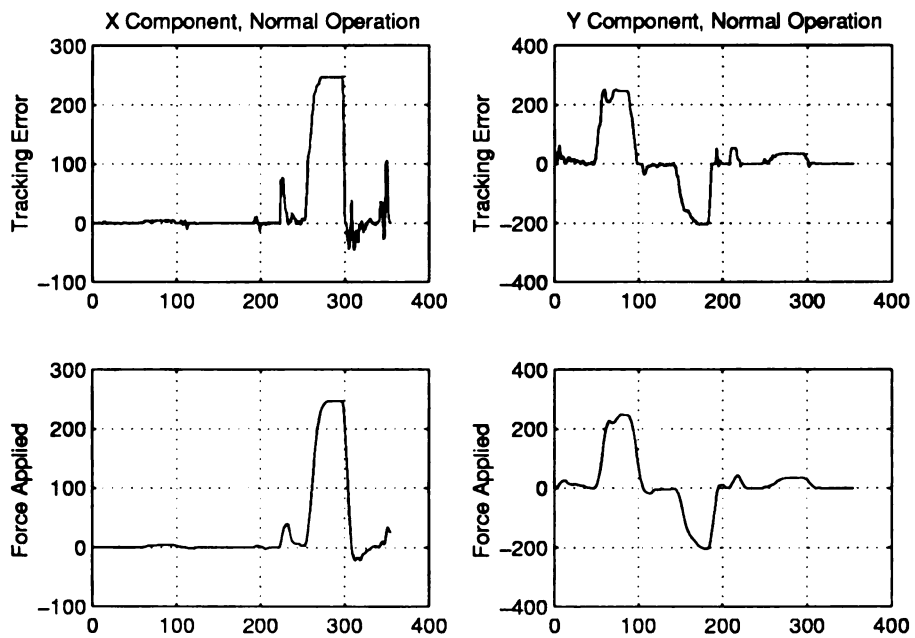


Figure 4.11: Comparison between tracking error and force felt for normal operation.

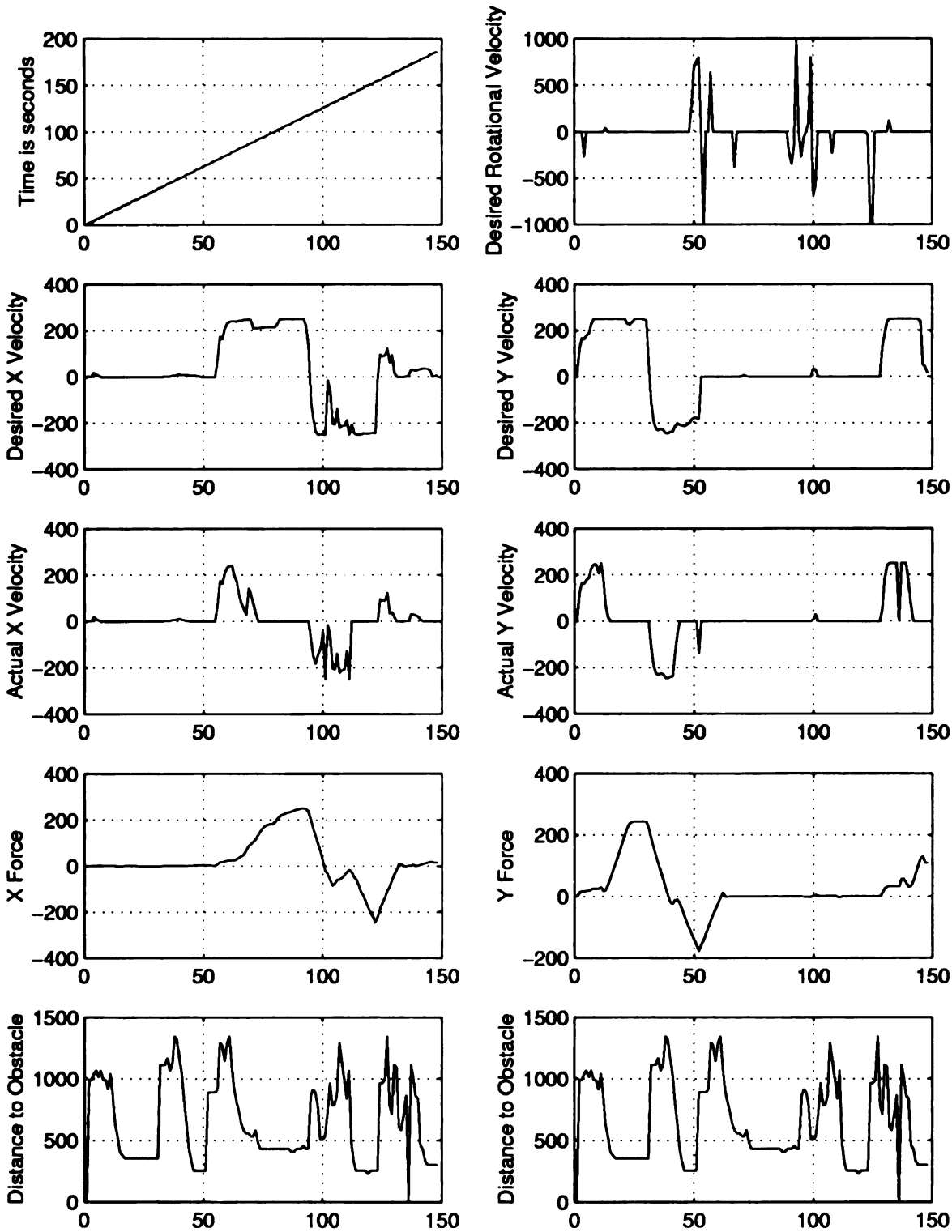


Figure 4.12: The behavior of the system under 2 seconds of round trip delay.

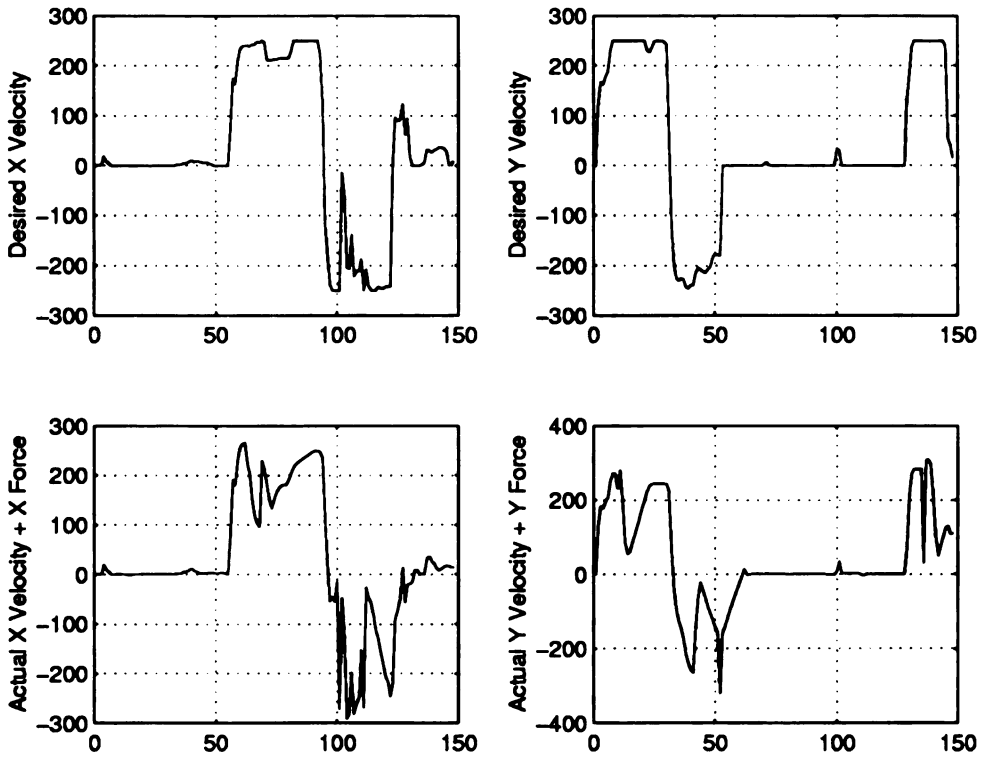


Figure 4.13: Comparison between the desired velocities and the sum of the actual velocities and the force felt.

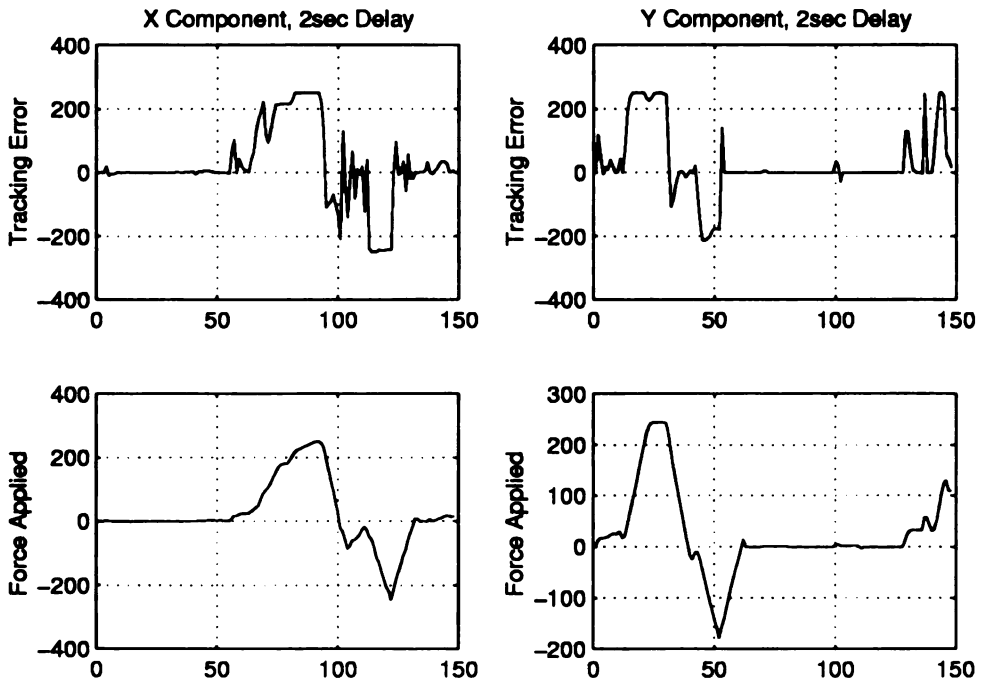


Figure 4.14: Comparison between tracking error and force felt for 2 seconds delay operation.

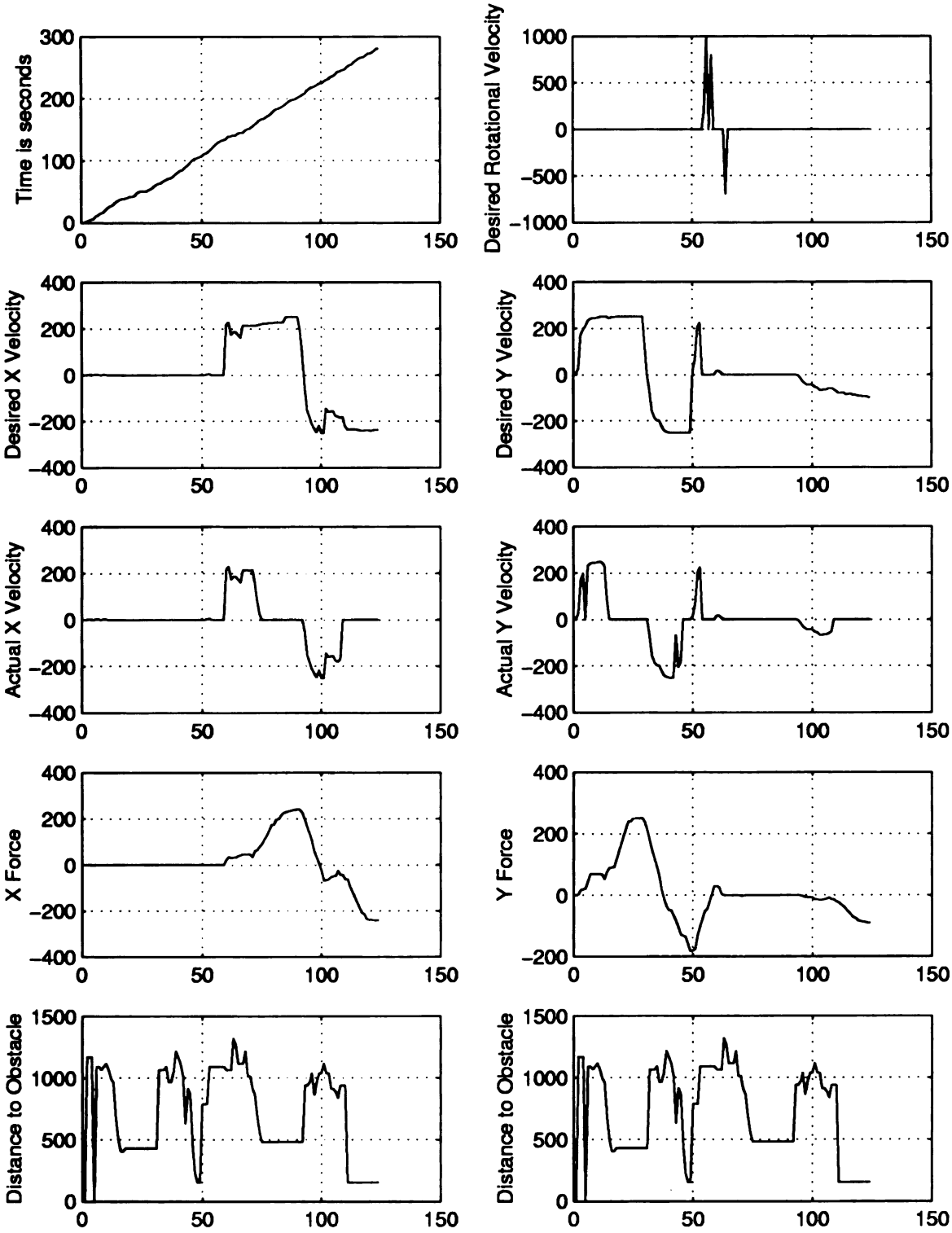


Figure 4.15: The behavior of the system under *random* round trip delay, ranging from 1 to 4 seconds.

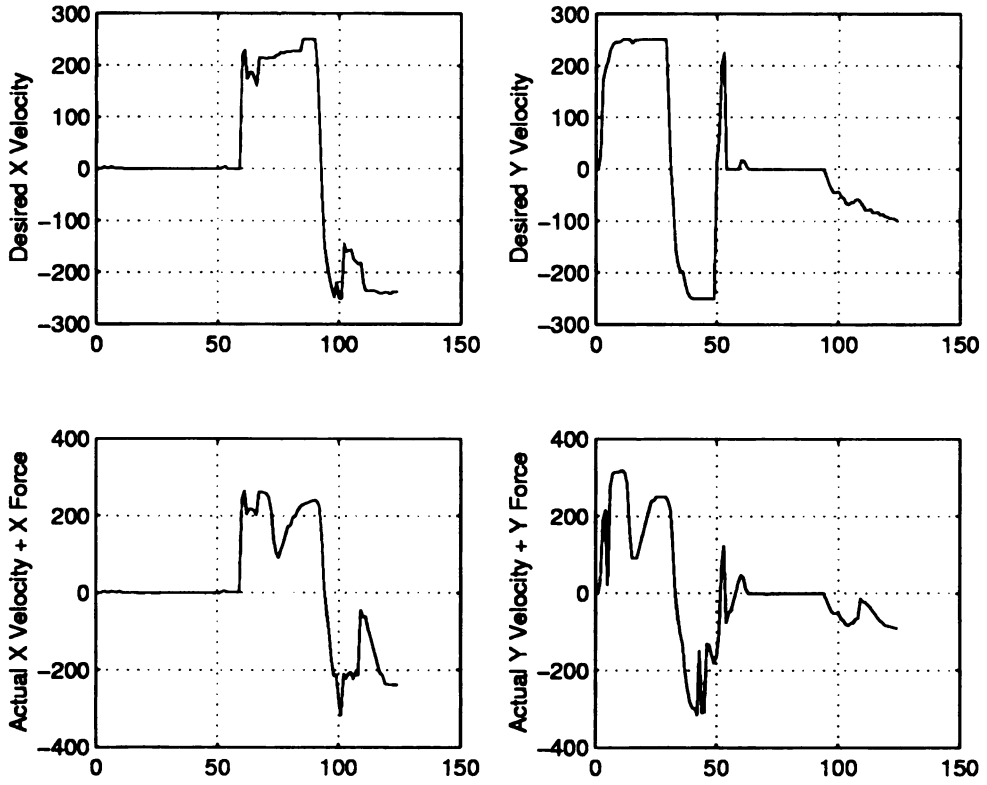


Figure 4.16: Comparison between the desired velocities and the sum of the actual velocities and the force felt.

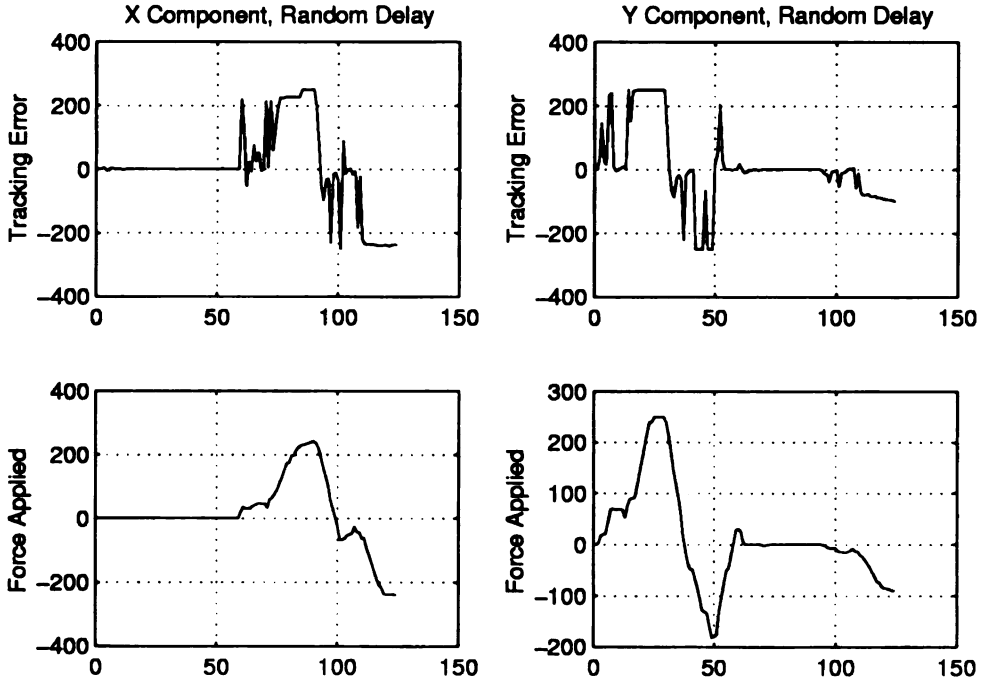


Figure 4.17: Comparison between tracking error and force felt for random delay operation.

are detected, the robot uses its obstacle avoidance algorithm. This algorithm allows it to avoid collision and more interestingly, generate force feedback that would notify the operator of the existence of these obstacles. An important note here is that, since we did not have a predefined path or trajectory and the operator had complete movement freedom, all the things discussed here would apply regardless of the path. This is desirable since in tele-operation we rarely have a pre-known complete idea about the trajectory.

4.2.2 Tele-operation from Hong Kong

As for the experiments done with the *Robot Control Lab* in Hong Kong, we experienced a control frequency[2] between 1.4 and 1.5 events/sec. Some of the results of these experiments are shown in Fig.4.18 and Fig.4.19. These plots are similar to the ones in Fig.4.9, Fig.4.12 and Fig.4.15. In all the following experiments, the operator is not constrained, complete freedom is given to move around in any direction or fashion and is allowed to get close to obstacle as desired.

The first plot was time since the beginning of the connection versus our event s, it is clear that in all the cases s is a nondecreasing function of time. Then we plotted the 3 desired velocities in the x, y and θ directions. Under those, we gave the actual velocities and forces in the x and y directions. Last we gave the distances to the closest obstacles detected. Observing those figure, we see that the forces increase as we get closer to objects and decreases as we get further. Meanwhile, the opposite happens to the actual velocity, where it becomes smaller and smaller than the desired one as we approach an object, and starts approaching the desired velocity as we go away. For all of these cases if we add up actual velocity with force then we should obtain the desired velocity. This is what is shown in Fig.4.20. There we see the desired velocity and under it the sum of the force with the actual velocity. As is expected, the two

[2]Control frequency is the number of times per second that the control commands are sent and the feedback received, this gives us a measure of events occurring per second

are not identical although close. Since we have to keep in mind that the force being played is actually a low pass filtered version of the velocity tracking errors, the only effect this has is smoothing the forces out as seen in Fig.4.21, which gives the tracking error and the force played which simply a filtered signal.

Actually the experiments done with Hong Kong were repeated several times to figure out if the system is affected by the time of day or the network load. To study this we present Fig.4.22. In this figure, which shows the same data as in Fig.4.18, we observe the same tracking and obstacle avoidance behavior as all the previous cases. As for Fig.4.23 we see the similarity between the desired velocity and the sum of the force with the actual velocity. Fig.4.24 simply shows the tracking error and the actual force played just to illustrate the filtering process.

From all the experiments in this section we can conclude that the system's behavior is consistent and similar for all the testing that was done regardless of the time or the network load. In addition, the system is stable under control over the Internet.

4.2.3 Experimental Conclusions

From all the experiments done and discussed previously we can conclude several things regarding our approach. The most important is that the system is consistent and behaves as expected. It is clear that the system works similarly for all cases. Meaning, whether you have constant time delay, random delay, control over the Internet or even no delay, the system is having the same performance. This is a very important advantage since the system is immune to time delay and its variations. In addition, it is obvious that the system is stable and that the actual velocities track the desired ones as long as there are no obstacles. Moreover, the system is executing the obstacle avoidance similarly for all the cases. Again this stability and obstacle avoidance apply in either scenario. Moreover, all of these properties apply irrespective of the path or trajectory taken since in all the experiments the operator had complete freedom of

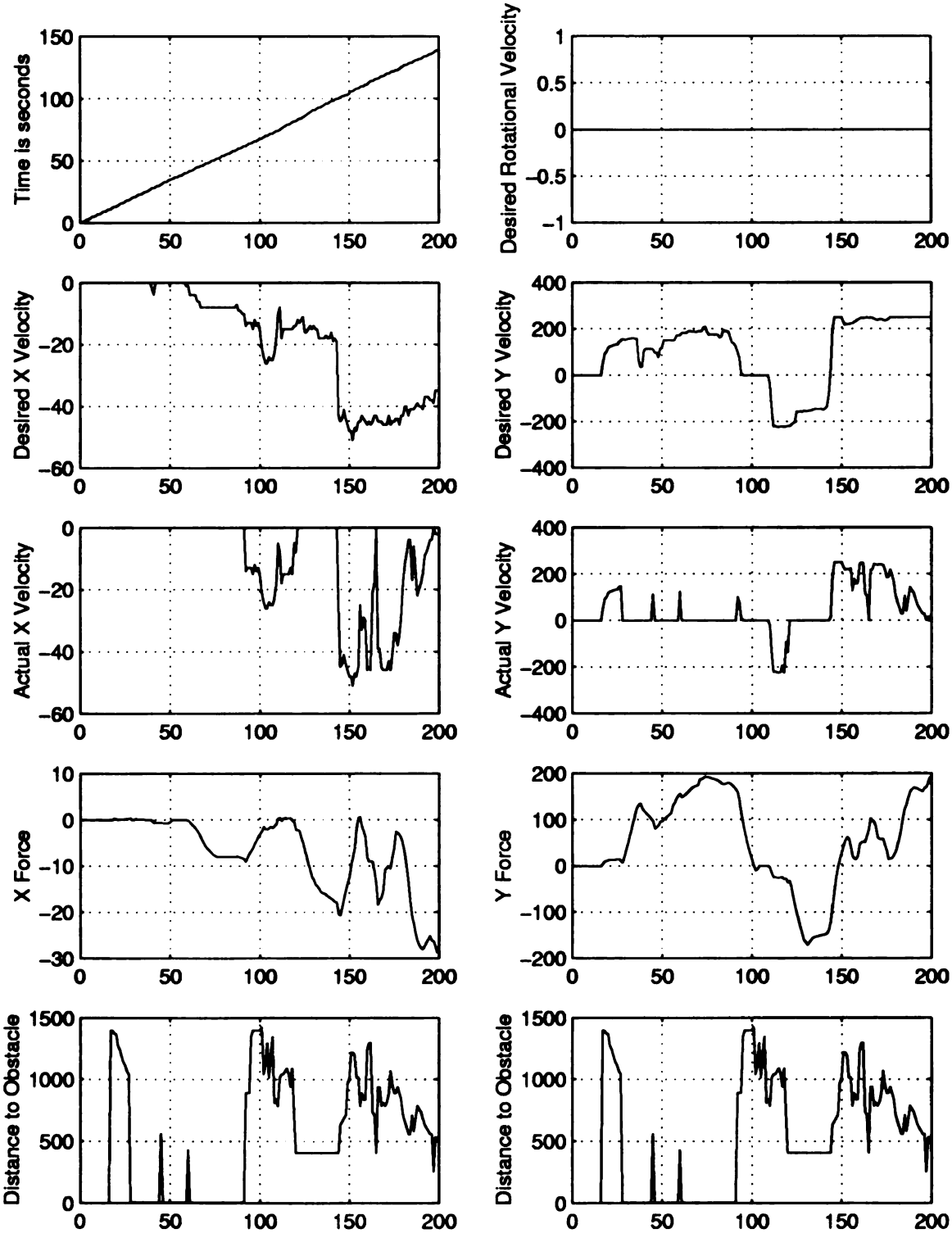


Figure 4.18: The behavior of the system during the control from Hong Kong.

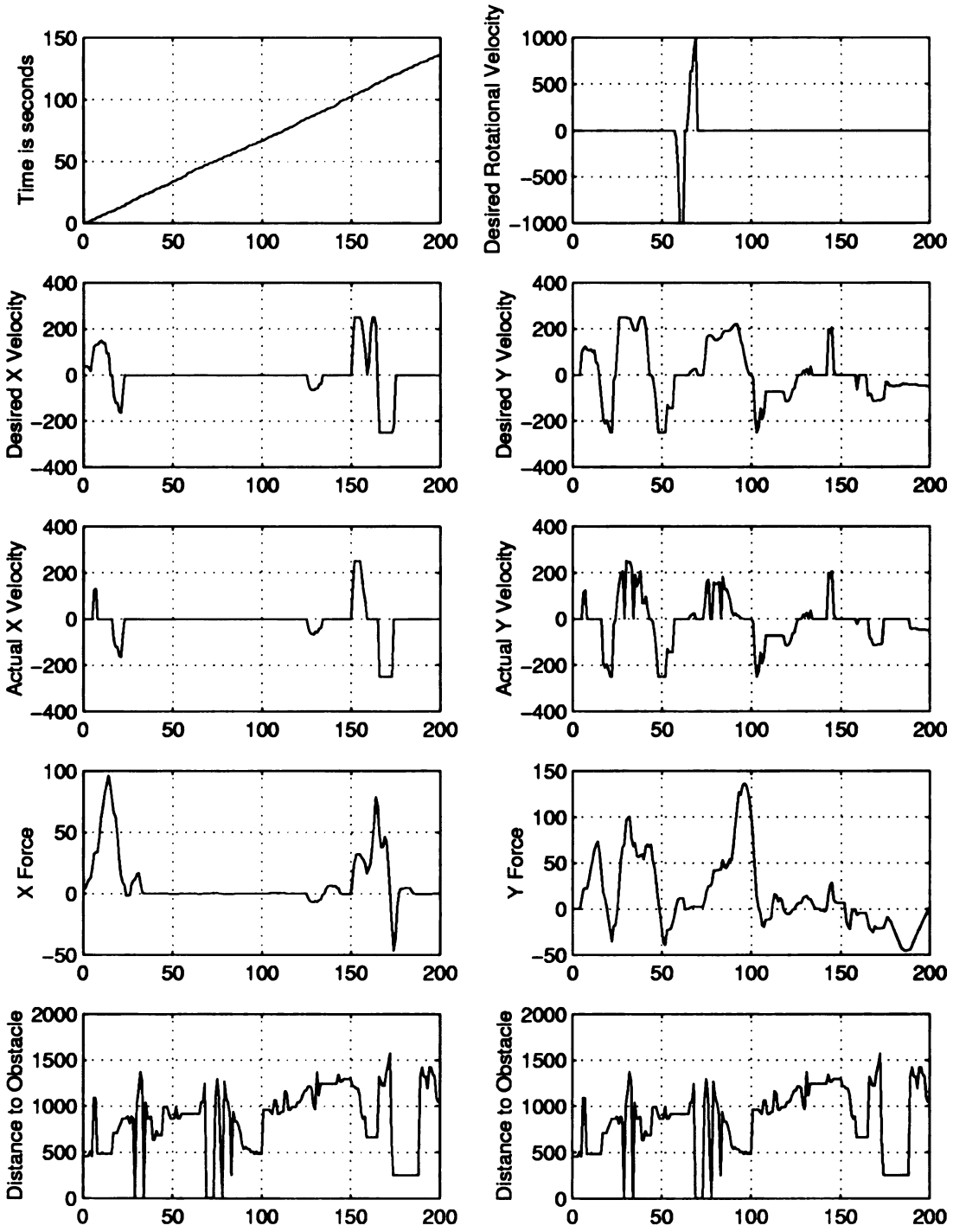


Figure 4.19: The behavior of the system during the control from Hong Kong.

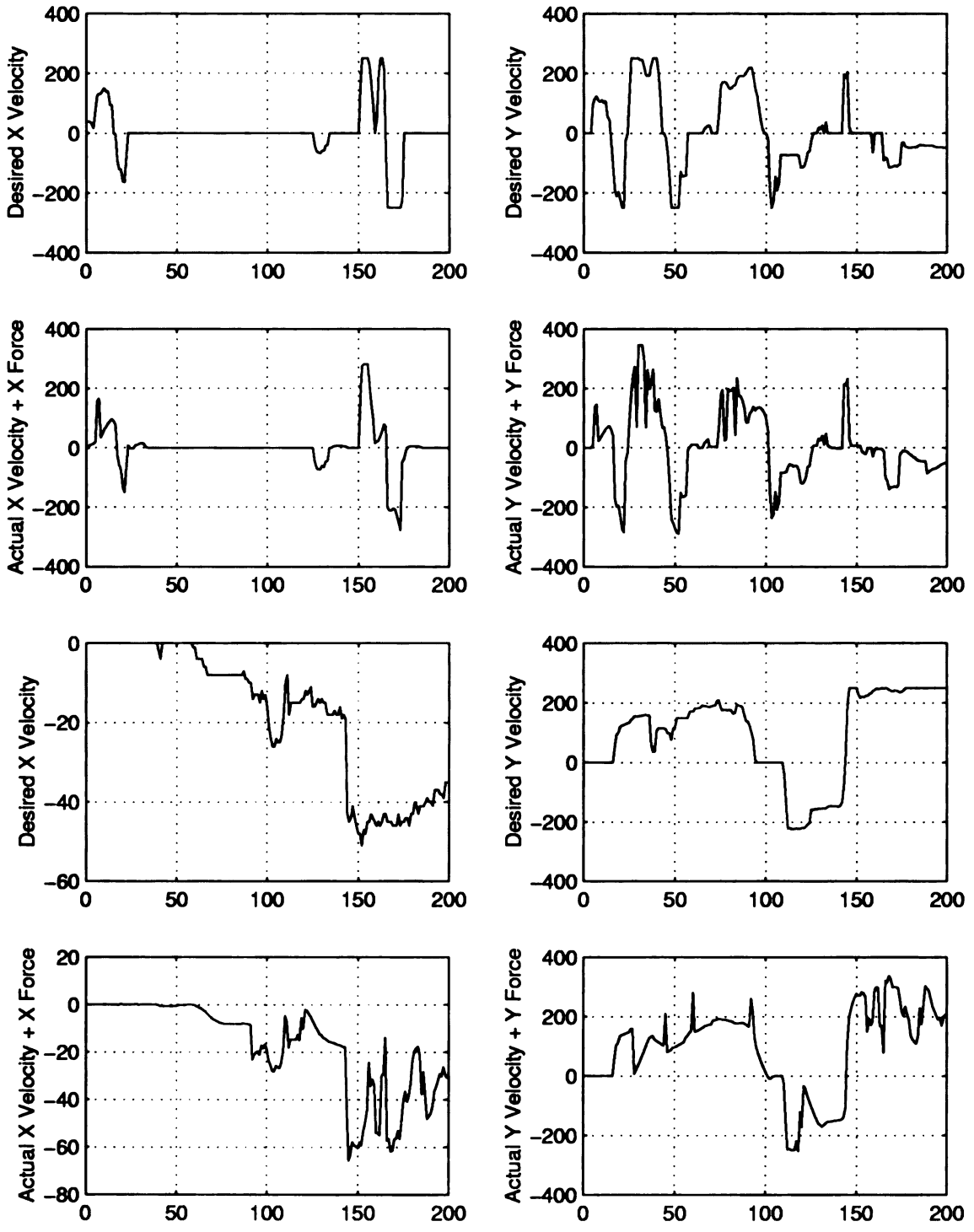


Figure 4.20: Comparison between the desired velocities and the sum of the actual velocities and the force felt.

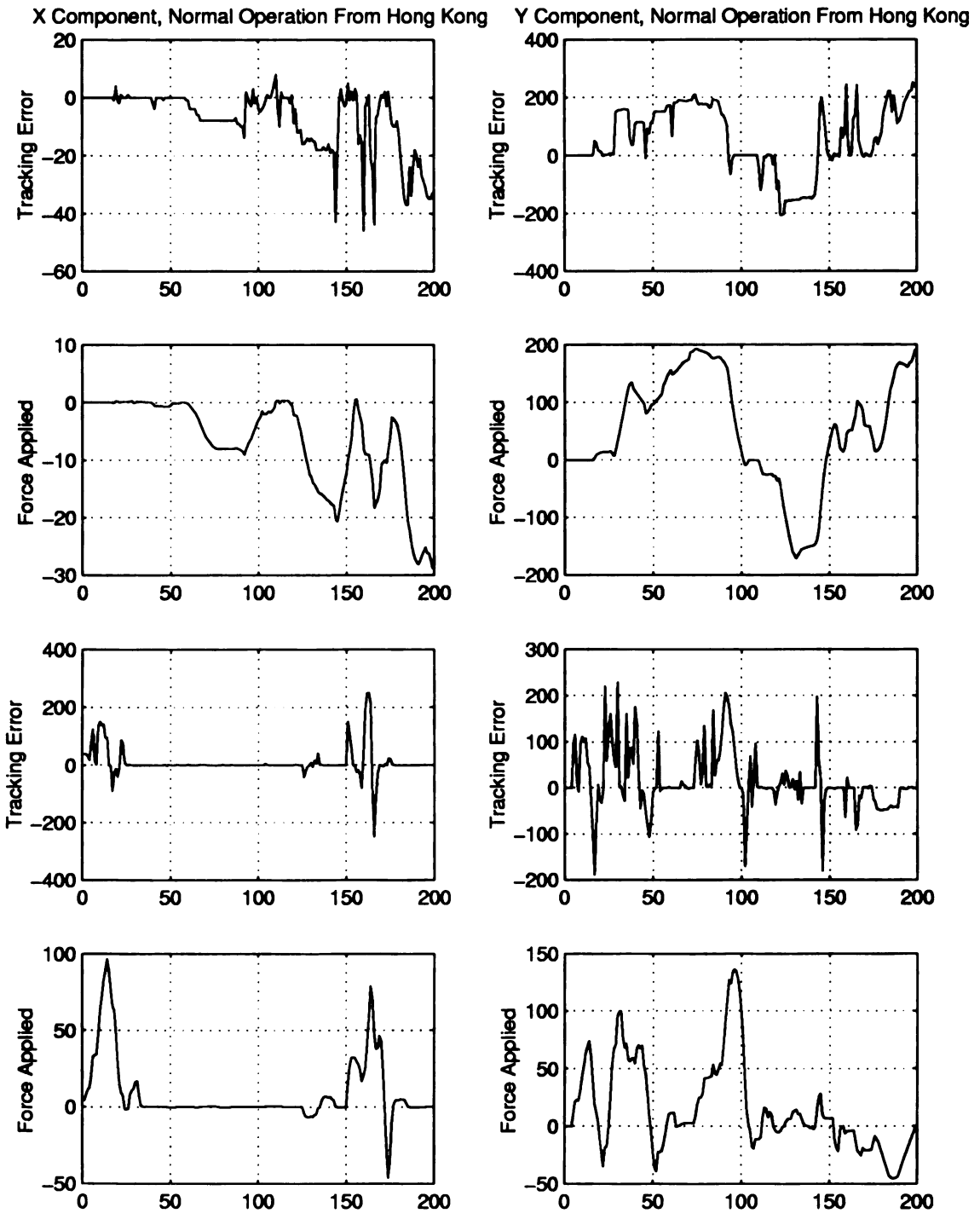


Figure 4.21: Comparison between tracking error and force felt for the two Hong Kong experiments.

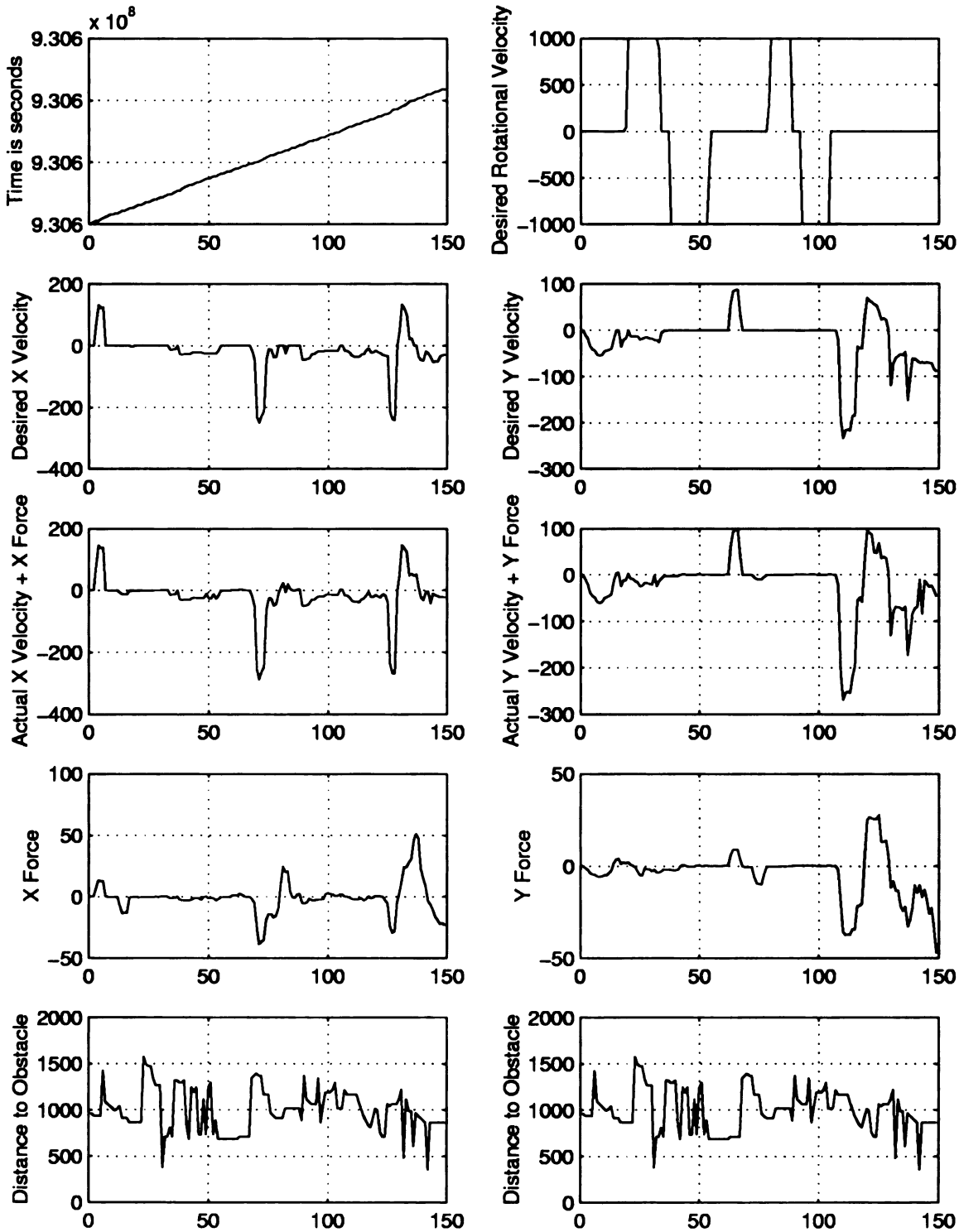


Figure 4.22: The behavior of the system during operation from Hong Kong.

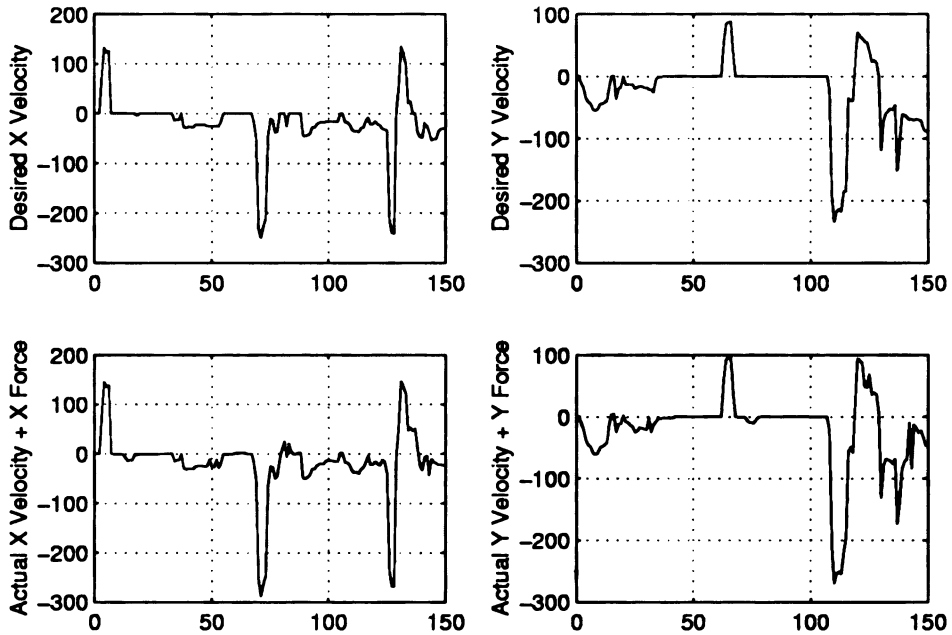


Figure 4.23: Comparison between the desired velocities and the sum of the actual velocities and the force felt.

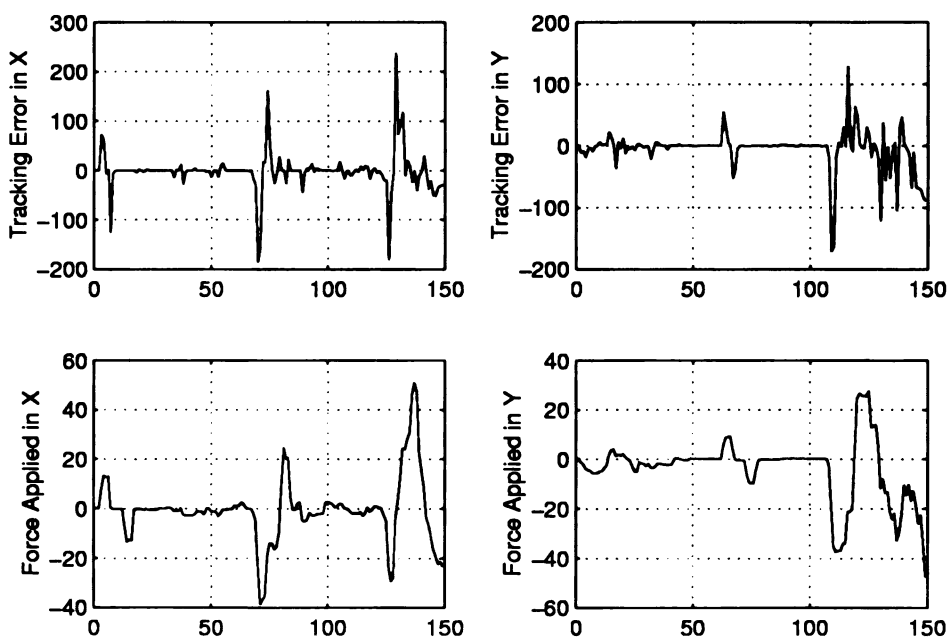


Figure 4.24: Comparison between tracking error and force felt for teleoperation from Hong Kong.

movement. A few notes about the experiments; first, we note that in some cases the actual velocities do not track the desired ones even if the obstacles are far, and therefore a force was generated. This tracking error can be due to several things; such as, a slope, a load or simply robot internal controller tracking errors. Although this is very rare but we have to note that a force can be felt even if there are no obstacle, and this is to our advantage since we want to feel the behavior of the robot and not only the obstacles. We want to know if the robot is going as fast as we want it, even if there are no objects in the way. Another thing, is that these experiments are the only ones we know of where a mobile robot is controlled over the Internet with force feedback in real-time.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter we conclude our work, giving a summary of our results. Then we discuss additional improvements that can be made on our approach, and what future work will be done.

5.1 Conclusion

Teleoperation is a very attractive and quickly growing research field. But this technology faces a lot of difficulties. The most important one, which will always exist, is time delay. This delay is caused by several things, most obviously signal propagation and hardware speed. At first thought this might be seen as a trivial issue not affecting the system performance. But it was shown that this delay would render the system unstable, especially when force feedback, which improves the operator's interaction with the environment, is used.

In this study we presented a new event-based control method, which can be used in any mobile robot, that solves the problem of delay. All the previous studies have certain limitations on the delay; for example, constant or having an upper bound. In our case no assumptions are made regarding the delay. This approach is immune to any kind of time delay, no upper bound is assumed and the delay can be random or fixed; therefore, it is ideal for Internet applications. The approach was demonstrated over the Internet, where no previous work was done having real-time control with force feedback. It was found that this method results in a stable and synchronized system. The whole method is based on having an event s as our reference and not time. Our implementation, does not only work over the Internet but also over any communication network, and can also be used in tele-operation with no feedback.

In addition, we came up with a new force generation method, where the force felt

is actually a function of the distance to obstacles in the environment of the robot. This means the operator would be able to deduce the position of objects in the robot's vicinity simply by moving around and feeling the force. This way we get force feedback with obstacle avoidance built in.

We demonstrated experimentally and theoretically that the system obtained is asymptotically stable. Not only that but also that the system was safe due to the built in obstacle avoidance. In addition, the system is reliable and robust, since a disconnection does not affect the system. And the operation can be resumed once a connection is re-established. Another advantage to our system and approach is that it decreases the task execution time by giving the operator a better *feeling* about the surrounding environment.

5.2 Future Work

Concerning the future work and the improvements on the system, there are many. First we would like to work on synchronizing the visual feedback with the force feedback so that all the components of the system would be at the same event during any time. Not much work have been done in that aspect, where several feedback forms are used in the system and all being synchronized. As an extension of the work presented here; is using the event, not only to control, but also to synchronize different parts of the system. In addition we would like to improve the speed of visual feedback from 1 frame per second to about 4 frames per second. Second, the system should be able to monitor the delay in real time and change some of the control and feedback parameters based on that so we have a faster response, and reduce the task execution time. Another interest, is to include both tele-operation and local control; meaning, the ability of the robot to track a certain predefined path but when unexpected events happen the operator would intervene to alter the path. Then the operator can release the robot to continue with the previous path tracking.

In addition, auto-tracking of certain specified object in real time is of interest. As for the filtering that is done from the error to the force felt, we are in the process of developing a smarter filter which would give a faster feeling of the environment. Maybe this filter could monitor the delay and based on that change the weight given to the different terms. Another extension would be using event based control for manipulators control over the Internet with force reflection.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] D. Kwon, K. Y. Woo, H. S Cho, "Haptic Control of the Master Hand Controller for a Microsurgical Telerobot System", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 1722-1727, May 1999.

[2] M. Tanimoto, F. Arai, T. Fukuda, and M. Negoro, "Force Display Method to Improve Safety in Teleoperation System for Intravascular Neurosurgery", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 1728-1733, May 1999.

[3] P. G. Backes, K. S. Tso, and G. K. Tharp, "Mars Pathfinder Mission Internet-Based Operations Using WITS", Workshop on Internet Robotics IEEE Int. Conf. on Robotics and Automation, pp. 284-291, May 1999.

[4] S. E. Everett, R. V. Dubey, "Model-Based Variable Position Mapping for Telerobotic Assistance in a Cylindrical Environment", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 2197-2202, May 1999.

[5] R. Luo, W. Z. Lee, J. H. Chou, and H. T. Leong, "Tele-Control of Rapid Prototyping Machine Via Internet for Automated Tele-Manufacturing", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 2203-2208, May 1999.

[6] L. Hsu, R. Costa, F. Lizarralde, J. Soares, "Passive Arm Based Dynamic Positioning System for Remotely Operated Underwater Vehicles", IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 407-412, May 1999.

[7] W. Kim, B. Hannaford, and A. Bejczy, "Force-Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay", IEEE Trans. on Robotics and Automation, Vol. 8, April 1992.

[8] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-Based Space Robotics-ROTEX and Its Telerobotic Features", IEEE Trans. on Robotics and Automation, Vol. 9, No. 5, October 1993.

[9] L. Conway, R. Volz, M. Walker, "Teleautonomous Systems: Projecting and Coordinating Intelligent Action at a Distance", IEEE Trans. on Robotics and Automation, Vol. 6, No. 2, April 1990.

[10] G. Hirzinger, K. Landzettel, Ch. Fagerer, "Telerobotics with Large Time Delays-The ROTEX Experience", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Munich, Germany, pp. 571-578, September 1994.

[11] A. Bejczy, W. Kim, S. Venema, "The Phantom Robot: Predictive Displays For Tele-operation with Time Delay", IEEE Int. Conf. on Robotics and Automation, pp. 546-551, Cincinnati, May 1990.

[12] M. mitsuishi, T. Hori, T. Nagao, "Predictive, Augmented and Transformed Information Display for Time Delay Compensation in Tele-handling/Machining", IEEE Int. Conf. on Robotics and Automation, pp 45-52, 1995.

[13] ,M. Otsuka, N. Matsumoto, T. Idogaki, K Kosuge, T. Itoh "Bilateral Telemanipulator System With Communication Time Delay Based on Force-Sum-Driven Virtual Internal Models", IEEE Int. Conf. on Robotics and Automation, pp. 344-350, 1995.

[14] D. Lawrence, "Stability and Transparency in Bilateral Teleoperation", IEEE Trans. on Robotics and Automation, Vol. 9, No. 5, October 1993.

[15] G. Leung, B. Francis, J. Apkarian, "Bilateral Controller for Teleoperators With Time Delay via μ-Synthesis", IEEE Trans. on Robotics and Automation, Vol 11, No. 1, February 1995.

[16] T. Sheridan, "Space Teleoperation Through Time Delay: Review and Prognosis", IEEE Trans. on Robotics and Automation, Vol 9, No. 5, October 1993.

[17] J. Nilsson, B. Bernhardsson, "Stochastic Analysis and Control of Real-Time Systems with Random Time Delays", Automatica, Vol. 34, pp. 57-64, 1998.

[18] J. Klamka, "Observer for Linear Feedback Control of Systems With Distributed delays in Controls and Output", Systems and Control Letters, Vol. 1, 1982.

[19] K. Watanbe, M. Ito, "An Observer for Linear Feedback Control Laws of multivariable Systems with Multiple Delays in Control and Output", Systems and Control Letters, Vol. 1, No. 1, July 1981.

[20] R. Anderson, M. Spong, "Asymptotic Stability for Force Reflecting Teleoperators with Time Delay", The Int. Journal of Robotics Research, Vol. 11, April 1992.

[21] R. Anderson, M. Spong, "Bilateral Control of Teleoperators with Time Delay", IEEE Trans. on Automatic Control, Vol. 34, No. 5, May 1989.

[22] G. Niemeyer, J. Slotine, "Stable Adaptive Teleoperation", IEEE Journal of Oceanic Engineering, Vol 16, No. 1, January 1991.

[23] C. Lawn, B. Hannaford, "Performance testing of Passive Communication and Control in Teleoperation with Time Delay", Proc. IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 776-781, Atlanta, May 1993.

[24] B. Hannaford, W. Kim, "Force Reflection, Shared Control, And Time Delay in Telemanipulation", Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, Cambridge, November 1989.

[25] T. J. Tarn, N. Xi, A. Bejczy, "Path-Based Approach to Integrated Planning and Control for Robotic Systems", Automatica, Vol. 32, No. 12, pp. 1675-1687, 1996.

[26] J. Tan, N. Xi, T. J. Tarn, "Non-Time Based Tracking Controller for Mobile Robots", IEEE Canadian Conf. on Electrical and Computer Engineering, Edmonton Canada, 1999.

[27] N. Xi, T. J. Tarn, "Action Synchronization and Control of Internet Based Telerobotic Systems", IEEE Int. Conf. on Robotics and Automation, Vol.1, pp. 219-224, Detroit, May 1999.

[28] Y. Zheng, "Human-Robot Coordination for Moving Large Objects", Workshop Note, 1997 ICRA.

[29] N. Hogan, "Multivariable Mechanics of The Neuromuscular System", IEEE Eight Annual Conference of the Engineering in Medicine and Biology Society, 1986.

[30] T. Milner, "Human Operator Adaptation to Machine Instability", Advances in Robotics, Mechatronics, and Haptic Interfaces, DSC-Vol. 49, ASME 1993.

[31] E. Todosiev, R. Rose, L. Summers, "Human Performance in Single and Two-Axis Tracking Systems", IEEE Trans. on Human Factors in Electronics, Vol. HFE-8, No. 2, June 1967.

[32] G. Bekey, H. Meissinger, R. Rose, "Mathematical Models of Human Operators in Simple Two-Axis Manual Control Systems", IEEE Trans. on Human Factors in Electronics, September 1965.

[33] H. Tan, M. Srinivasan, B. Eberman, B. Cheng, "Human Factors For the Design of Force-Reflecting Haptic Interfaces", Dynamic Systems and Control, DSC-Vol. 55-1, 1994.

[34] E. Altman, T. Basar, R. Srikant, "Multi-User Rate-Based Flow Control with Action Delays: A Team-Theoretic Approach", Proceedings of the 36^{th} Conference on Decision and Control, San Diego, December 1997.

[35] E. Altman, T. Basar, "Multi-User Rate-Based Flow Control: Distributed Game-theoretic Algorithms", Proceedings of the 36^{th} Conference on Decision and Control, San Diego, December 1997.

[36] U. Madhow, "Dynamic Congestion Control and Error Recovery Over a Heterogeneous Internet", Proceedings of the 36^{th} Conference on Decision and Control, San Diego, December 1997.

[37] R. Riedi, M. Course, V. Ribeiro, R. Baraniuk, "A Multifractal Wavelet Model with Application to TCP Network Traffic", IEEE Trans. on Information Theory (Special Issue on Multiscale Signal Analysis and Modeling), pp. 992-1018, April 1999.

[38] W. Leland, M. Taqqu, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", IEEE/ACM Trans. on Networking, Vol. 2, No. 1, February 1994.

[39] K. Brady, T. J. Tarn, "Internet-Based Remote Teleoperation", Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 1998.

[40] D. Pai, "ACME, A Telerobotic Measurement Facility for Reality-Based Modelling on the Internet", IROS Workshop on Robots on the Web, Canada, 1998.

[41] M. Stein, "Painting on the World Wide Web: The PumaPaint Project", IROS Workshop on Robots on the Web, Canada, 1998.

[42] R. Simmons, "Xavier: An Autonomous Mobile Robot on the Web", IROS Workshop on Robots on the Web, Canada, 1998.

[43] N. Xi, "Event-Based Planning and Control for Robotic Systems", Doctoral Dissertation, Washington University, December 1993.

[44] Nomad XRDEV Software Manual, Release 1.0, Nomadic Technologies, Inc., 1999.