

DEVELOPMENT OF INTERNET-BASED TELEROBOTICS

(PEMBANGUNAN TELEROBOTIK BERASASKAN INTERNET)

ASSOC. PROF. DR. ROSBI MAMAT

PROF. DR. SHAMSUDIN H.M.AMIN

ASSOC. PROF. ZAMANI ZAIN

DR. ZAHARUDIN MOHAMED

LIM CHENG SIONG

MD. FAUZI ZAKARIA

NORHAYATI A. MAJID

**PUSAT PENGURUSAN PENYELIDIKAN
UNIVERSITI TEKNOLOGI MALAYSIA**

ABSTRACT

The internet is a very fast evolving new technology, allowing people to electronically connect places that are thousands of miles apart. The internet serves mainly for information exchange. However since the launch of the first robots on the Internet in 1994, an enormous effort has been undertaken by hundreds of researchers to push this technology. Quite a number of designs and applications of the internet-based telerobotic system have been implemented and launched on the internet. Some of the telerobotic systems are designed for blocks manipulation, paint painting and gardening. There has been much effort taken by UTM to be one of the main players in this area for the last few years. A lot of improvement has been achieved in the internet-based telerobotics since the first project launched in 1994. Thus, the project is carried out to achieve the objectives as: to study the latest finding in the internet-based telerobotics especially the problems faced as well as the solutions; to study the existing project especially the problems faced; to identify the appropriate technology to improve the existing project; to design a new telerobotic system; and, to implement the new telerobotic system. Based on the findings, the task-oriented telerobotic system has been developed. In the system developed, the user need only to specify the task to be done by the robot and then the system will plan the path of the robot movement to complete the task. The system is found to be more user friendly, reliable, the safety of the working objects and the robot are well protected.

ABSTRAK

Internet ialah teknologi baru yang berkembang dengan cepat. Internet membolehkan manusia berhubung secara elektronik dengan tempat yang terletak beribu-ribu meter jauh. Fungsi utama Internet adalah untuk pertukaran informasi. Namun sejak perlanjaran robot pertama di Internet pada tahun 1994, pelbagai usaha telah diambil oleh ahli-ahli penyelidikan untuk memajukan teknologi ini. Dengan itu, pelbagai rekaan dan applikasi sistem telerobot yang berasaskan Internet telah dimajukan dan dilancarkan di Internet. Sebahagian daripada sistem telerobot direka untuk blok manipulasi, mencat dan berkebun. UTM telah berusaha giat sejak beberapa tahun lalu untuk menjadi salah satu daripada pelopor terkemuka dalam bidang ini. Semenjak perlanjaran projek pertama pada tahun 1994, pelbagai kamajuan telah dicapai dalam bidang telerobot yang berasaskan Internet. Memandangkan ini, projek ini telah dilaksanakan untuk mencapai objektif-objektif seperti berikut: untuk mengkaji penemuan terbaru terutamanya masalah-masalah dan penyelesaiannya yang dihadapi dalam bidang telerobot yang berasaskan Internet; untuk mengkaji masalah yang dihadapi dalam projek sedia ada; untuk mengenal pasti teknologi bersesuaian bagi memperbaiki projek sedia ada; untuk mereka senibina sistem telerobot baru; dan, untuk memajukan rekaan senibina sistem telerobot baru. Berdasarkan maklumat yang ditemui, sistem telerobot yang berdasarkan tugas telah dibangunkan. Dalam sistem yang dibangunkan, pengguna hanya perlu memberitahu sistem tugas yang perlu dilaksanakan oleh robot dan seterusnya sistem akan merancangkan laluan untuk pergerakan robot bagi menyelesaikan tugas berkenaan. Sistem tersebut didapati lebih ramah pengguna, boleh diharap, keselamatan objek kerja dan robot adalah terjamin.

CONTENTS

CHAPTER	SUBJECT	PAGE
	TITLE PAGE	i
	ABSTRACT	ii
	ABSTRAK	iii
	CONTENTS	iv
	LIST OF TABLES	ix
	LIST OF FIGURES	xi
	LIST OF SYMBOLS AND ABBREVIATIONS	xvii
	LIST OF APPENDIXES	xviii
CHAPTER 1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Project Background and Motivation	2
	1.3 Objectives and Scope of Project	3
	1.4 Research Methodology	4
	1.5 Layout of Report	5
CHAPTER 2	LITERATURE REVIEWS	6
	2.1 Introduction	6

2.2	Australia's Telerobot on the Web	7
2.3	Carnegie Science Center (CSC)	
	Telerobot	11
2.4	Robotoy	12
2.5	Tele-Garden	14
2.6	Painting on the World Wide Web: The PumaPaint Project	16
2.7	Natural Communication with Robot	19
2.8	Prototype of UTM Web-based Telerobotic System	21
3.7	Summary	23
CHAPTER 3	IMPORTANT FACTORS IN DEVELOPMENT OF INTERNET- BASED TELEROBOTICS	24
3.1	Introduction	24
3.2	Easy to Operate	24
3.3	Reliability	26
3.4	Response Time	26
3.5	Human Factors	28
3.6	Interface Design	29
3.7	Summary	30
CHAPTER 4	TERMS DEFINITION AND SPECIFICATION OF UTM TELEROBOTIC SYSTEM	31
4.1	Terms Definition	31
4.2	UTM Telerobotic System Setup and Application Programs' User Interface	36

	4.3	System Architecture	44
	4.4	Summary	48
CHAPTER 5		VIRTUAL WORKING AREA CONSTRUCTION, COMMAND PRE- PROCESSOR AND TASK PRE- PROCESSOR	49
	5.1	Introduction	49
	5.2	Virtual Working Area and Working Objects Construction	50
	5.3	Command Pre-processor	52
	5.4	Intelligent Parser	66
	5.5	Task Pre-Processor	67
	5.6	Client-server Connection Manager	69
	5.7	FTP Server	71
	5.8	Summary	72
CHAPTER 6		HIGH LEVEL COMMAND TO LOW LEVEL COMMAND TRANSLATION	73
	6.1	Introduction	73
	6.2	Task Planner	73
	6.3	Robot Path Planner	76
	6.4	Vision Sub-system	86
	6.5	Summary	93
CHAPTER 7		SAFETY, RELIABILITY AND ACCURACY DESIGN OF THE TELEROBOTIC SYSTEM	94

7.1	Introduction	94
7.2	Robot Selection and Task Definition	97
7.3	Working Object Definition	97
7.4	Work Cell Design	99
7.5	Working Area Definition	100
7.6	Distance Between Objects Definition	104
7.7	Gripper with New Fingers Design	105
7.8	System Self-calibration	107
7.9	Working Object Exception Handling	108
7.10	Client-server Exception Handling	109
7.11	Log File and Error Listing	111
7.12	Summary	115
CHAPTER 8	SYSTEM TESTING, RESULT ANALYSIS AND SYSTEM ARCHITECTURE COMPARISON	116
8.1	Introduction	116
8.2	Command Pre-processor Testing	117
8.3	Task Analysis: Single Object Moved	117
8.4	Task Analysis: Single Object Moved and Rotated	122
8.5	Task Analysis: Two Objects Moved and Rotated	128
8.6	Output Accuracy Analysis	137
8.7	Exception Handling	138
8.8	Platform Testing	138
8.9	Local Area Network (LAN) Testing	139

8.10	System Architecture Comparison	139
8.11	Summary	141
CHAPTER 9	CONCLUSION	142
9.1	Introduction	142
9.2	Objectives Achievement	144
9.3	Contribution	146
9.4	Recommendations and Future Work	148
	REFERENCES	
	APPENDIXES	
	APPENDIX A	157- 161
	APPENDIX B	162- 213

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1:	Comparison table for the system of Australia's telerobot on the web	8
Table 2.2:	Comparison table for CSC Telerobot system	11
Table 2.3:	Comparison table for robotoy system	13
Table 2.4:	Comparison table for tele-garden system	15
Table 2.5:	Comparison table for PumaPaint project	17
Table 2.6:	Comparison table for the prototype of UTM web-based telerobotic system	22
Table 4.1:	Comparison between robot-oriented system and task-oriented robotic system	35
Table 4.2:	Motor and the corresponding joint	38
Table 5.1:	Words to symbols and values conversion	62
Table 5.2:	URL strings	71
Table 6.1:	Details of the working objects at the remote site	75
Table 6.2:	Details of the virtual working objects send by the telerobotic client program	75
Table 6.3:	Differences between the information kept in the Table 6.1 and Table 6.2	76
Table 6.4:	Motors' unit encoder count at soft home	83
Table 6.5:	Conversion table	84
Table 6.6:	Motors' unit encoder count at reference point	84
Table 7.1:	Motors position for the gripper model defined	108
Table 7.2:	Working object exception handling	109
Table 8.1:	Output analysis for standard gripper	137

Table 8.2: Output analysis for new fingers design

138

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1:	Research methodology	5
Figure 2.1:	The telerobot system for Australia's telerobot on the web	8
Figure 2.2:	The actual site for Australia's telerobot on the web	9
Figure 2.3:	Operator interface for Australia's telerobot on the web	9
Figure 2.4:	Usher - to specify a new position of the gripper, drag these lines to specify x, y, z, spin, and tilt of the new position. Select "move" from the pop-up menu to let the robot move to the new position	10
Figure 2.5:	Two dimensional wireframe views of the robot included in an earlier interface, circles indicate rotational joints. These images were clickable.	10
Figure 2.6:	Coordinate system for CSC Telerobot	12
Figure 2.7:	Part of the operator interface for CSC Telerobot	12
Figure 2.8:	Robotoy system	13
Figure 2.9:	Part of operator interface for robotoy project	14
Figure 2.10:	The actual site for tele-garden project	15
Figure 2.11:	Part of the operator interface for tele-garden project	16
Figure 2.12:	The actual site for PumaPaint project	17
Figure 2.13:	GUI with added features to increase the fidelity of the virtual canvas	18

Figure 2.14:	Painted text and images	18
Figure 2.15:	Overview of the system	20
Figure 2.16:	Example of dialogue	21
Figure 2.17:	System architecture for the prototype of UTM web-based telerobotic system	22
Figure 2.18:	Rhino robot and the actual site	22
Figure 2.19:	UTM Telerobot web-based interface	23
Figure 4.1:	System concepts	32
Figure 4.2:	Sub-systems of a system	32
Figure 4.3:	Rhino robot	37
Figure 4.4:	Dimension of Rhino robot	38
Figure 4.5:	Motor and the corresponding joint	39
Figure 4.6:	Front view of the robot work cell	39
Figure 4.7:	Top view of the robot work cell	40
Figure 4.8:	Sample of working objects	40
Figure 4.9:	User interface of the UTM telerobotic server program	42
Figure 4.10:	User interface of the UTM telerobotic client program	43
Figure 4.11:	User interface of the UTM telerobotic client program with TCP/IP messaging facilities	44
Figure 4.12:	Local and remote sites of the UTM telerobotic system	46
Figure 4.13:	UTM telerobotic client program architecture	47
Figure 4.14:	UTM telerobotic server program architecture	47
Figure 4.15:	Block diagram of the task control sub-system (closed-loop)	48
Figure 5.1:	Virtual working area with the image of the top view of the working area	51
Figure 5.2:	Coordinate for the top right-hand corner	52
Figure 5.3:	Popup menu 1	64
Figure 5.4:	Popup menu 2	65
Figure 5.5:	Linear equations for the sides	65

Figure 5.6:	Natural language listing	67
Figure 5.7:	Object rotated 90°, 180° and 270°	69
Figure 5.8:	Client-server connection process	71
Figure 6.1:	Robot soft home	82
Figure 6.2:	Working area and robot-based coordinate systems	83
Figure 6.3:	Top view of the robot	83
Figure 6.4:	Side view of the robot	84
Figure 6.5:	Robot physical configuration at reference point	85
Figure 6.6:	Robot commands	85
Figure 6.7:	Image captured	88
Figure 6.8:	Image segmented	88
Figure 6.9:	Image smoothed	89
Figure 6.10:	Image binarized	89
Figure 6.11:	Image opened	90
Figure 6.12:	Image closed	90
Figure 6.13:	Model defined in the ModelFinder Control	91
Figure 6.14:	Boxes are drawn at the working objects recognized	91
Figure 6.15:	Model defined in the ModelFinder Control	92
Figure 6.16:	A box is drawn at the part of the gripper recognized	93
Figure 7.1:	Dimension of gripper opening	98
Figure 7.2:	Gripper and cube	99
Figure 7.3:	Robot work envelope	100
Figure 7.4:	Robot work cell	100
Figure 7.5:	Optimum height of the working area	102
Figure 7.6:	Working area dimension calculation	102
Figure 7.7:	Virtual working area	103
Figure 7.8:	Workable, viewable areas comparison	103
Figure 7.9:	The maximum outer length of the robot finger	104
Figure 7.10:	The area cleared for the gripper rotation	105
Figure 7.11:	The minimum distance between the working objects required	105

Figure 7.12:	Types of gripper available for Rhino robot	106
Figure 7.13:	New fingers design	106
Figure 7.14:	Image offset to calibration the system	108
Figure 7.15:	Example of the events recorded in the log file	114
Figure 7.16:	Events recorded in the error listing	115
Figure 8.1:	Task planned	119
Figure 8.2:	Motor F is moved (left) followed motor E (right)	119
Figure 8.3:	Motor D is moved (left) followed by motor B (right)	120
Figure 8.4:	Motor A is moved (left) followed by motor D (right)	120
Figure 8.5:	Motor E is moved (left) followed by motor F (right)	120
Figure 8.6:	Motor E is moved (left) followed by motor B (right)	121
Figure 8.7:	Motor D is moved (left) followed by motor A (right)	121
Figure 8.8:	Motor B is moved (left) followed by motor D (right)	121
Figure 8.9:	Motor E is moved	122
Figure 8.10:	Task completed	122
Figure 8.11:	Task planned	124
Figure 8.12:	Motor F is moved (left) followed by motor E (right)	124
Figure 8.13:	Motor D is moved (left) followed by motor B (right)	125
Figure 8.14:	Motor A is moved (left) followed by motor D (right)	125
Figure 8.15:	Motor E is moved (left) followed by motor F (right)	125
Figure 8.16:	Motor E is moved	126
Figure 8.17:	Motor B is moved	126

Figure 8.18:	Motor D is moved (left) followed by motor A (right)	126
Figure 8.19:	Motor B is moved (left) followed by motor D (right)	127
Figure 8.20:	Motor E is moved	127
Figure 8.21:	Task completed	128
Figure 8.22:	Task planned	131
Figure 8.23:	Motor F is moved (left) followed by motor E (right)	131
Figure 8.24:	Motor D is moved (left) followed by motor B (right)	132
Figure 8.25:	Motor A is moved (left) followed by motor D (right)	132
Figure 8.26:	Motor E is moved (left) followed by motor F (right)	132
Figure 8.27:	Motor E is moved (left) followed by motor B (right)	133
Figure 8.28:	Motor D is moved (left) followed by motor A (right)	133
Figure 8.29:	Motor B is moved (left) followed by motor D (right)	133
Figure 8.30:	Motor E is moved (left) followed by motor F (right)	134
Figure 8.31:	Motor E is moved (left) followed by motor D (right)	134
Figure 8.32:	Motor B is moved (left) followed by motor A (right)	134
Figure 8.33:	Motor D is moved (left) followed by motor E (right)	135
Figure 8.34:	Motor F is moved (left) followed by motor E (right)	135
Figure 8.35:	Motor B is moved (left) followed by motor D (right)	135

Figure 8.36:	Motor A is moved (left) followed by motor B (right)	136
Figure 8.37:	Motor D is moved (left) followed by motor E (right)	136
Figure 8.38:	Task completed	136

LIST OF SYMBOLS AND ABBREVIATIONS

2D	-	Two dimensional
3D	-	Three dimensional
CD-ROM	-	Compact Disc-Read Only Memory
CSC	-	Carnegie Science Center
DOF	-	Degree of freedom
GUI	-	Graphical User Interface
IP	-	Internet Protocol
ISDN	-	Integrated Services Digital Network
ISP	-	Internet Service Provider
JARA Award	-	Japan Robot Association Award
JPEG	-	Joint Photographic Experts Group
kb	-	Kilobit
kbps	-	kilobits per second
MIL	-	Matrox Imaging Library
MIME	-	Multipurpose Internet Mail Extensions
NASA	-	National Aeronautics and Space Administration
PC	-	Personal computer
PSTN	-	Public Switched Telephone Network
RAM	-	Random access memory
ROVs		Remotely Operated Vehicles
URL	-	Uniform Resource Locator
UTM	-	Universiti Teknologi Malaysia
VC++	-	Visual C++
WWW		World Wide Web

LIST OF APPENDIXES

APPENDIX	TITLE	PAGE
A	Work Cell Dimension	157

CHAPTER 1

INTRODUCTION

1.1 Introduction

The internet is a very fast evolving new technology, allowing people to electronically connect places that are thousands of miles apart. However, up to now, electronic networks serve mainly to exchange and acquire information. The first robot has appeared on the internet in 1994. The project, named Mercury project (Goldberg, K., et al., 2000), was the first system that allowed WWW users to remotely view and alter the real world via telerobotics. Four weeks after that, the second robot, ASEA IR-6 was connected to the internet at the University of Western Australia (Taylor, K. and Dalton, B., 1997). The later robot is still running on the web since the launch. Since the launch of the robots on the internet, an enormous effort has been undertaken by hundreds of researchers to push this technology.

Telerobot is a robot that accepts instruction from a distance, generally from a trained human operator (Fauzi Zakaria, 2000). The technology can be applied in many areas. Nevertheless the current projects are largely experimental and none have been used to provide commercial services. Areas where this technology is thought likely to be useful are (Taylor, K. and Dalton, B., 1997):-

- i) Entertainment. It is apparent from the reaction of people to Australia's Telerobot, and other internet devices that many people consider operating them entertaining. A private company, LunaCorp Inc in

conjunction with Carnegie Mellon University plan to launch the first private lunar mission. The project involves landing a pair of teleoperated robotic vehicles on the Moon's surface (<http://www.ri.cmu.edu/lri/>). Intended customers for the mission include a theme park, television network, commercial sponsors, and scientists.

- ii) Telemanufacturing. There is a large group at University of California Berkeley with a grant of US\$1.3 million developing an Internet accessible, machining service called CyberCut. (<http://CyberCut.berkeley.edu/>).
- iii) Training. Providing access to robots and other expensive equipment for training purposes where purchasing cannot be justified.
- iv) Mining. Teleoperation of underground mining equipment is being practised at some mines and this technique could be used to operate the equipment from any location.
- v) Underwater Remotely Operated Vehicles (ROVs). ROVs are subject to time delays, limited bandwidth, and unstructured environments providing an ideal application for supervisory control. These constraints are in many ways similar to those experienced in internet telerobotics.

1.2 Project Background and Motivation

The prototype for the internet-based fixed arm type telerobotic system had been developed by Fauzi Zakaria (2000). The details of the system are discussed in chapter 2. Some of the problems faced in the project are as below:-

- i) Users are limited to the button-based interface to operate the robot. The disadvantage of the design is that fine movement cannot be well supported;
- ii) No operation guidance for operators in both actual site and operator interface. Some type of guidance such as grid on the working area as well as in the virtual environment should be provided;
- iii) The image captured and presented to the operators should be processed and optimized. This can be done through filtering out the unnecessary image as well as compressing the image before sending it to the operator to reduce the size; and,
- iv) No application defined. The system should support at least one application to make it practical. A good choice of application can make the project interesting and attractive to the internet users.

Since this is UTM's first internet-based fixed arm type telerobotic system, a lot of effort is still needed to improve the system to catch up with the latest technology in internet-based telerobotics. A lot of improvement has been achieved in the internet-based telerobotics since the first launch in 1994. Some of the projects provide the virtual environment operation as well as more complicated application.

1.3 Objectives and Scope of Project

The objectives of the project can be summarized as:-

- i) To study the latest finding in the internet-based telerobotics especially the problems faced as well as the solutions;
- ii) To study the existing project especially the problems faced;
- iii) To identify the appropriate technology to improve the existing project;
- iv) To design a new telerobotic system; and,

- v) To implement the new telerobotic system.

Based on the result of the literature review, the scope of the project has been set to cover mainly on the 3 areas:-

- i) To design a task-oriented telerobotic system based on the existing Rhino robot;
- ii) To support visual servoing in the new telerobotic system; and,
- iii) To support type-written natural language command from the user to control the new telerobotic system.

1.4 Methodology of Research

The methodology in this project can be divided into three stages, which are the theoretical work, implementation and experimental testing of the prototype of telerobotic system. The research methodology is shown in the Figure 1.1. First of all, literature reviews are carried out as well as assessing the current project setup. From the literature review, the problems faced by the other researchers and the solutions are identified. A proper design and plan had been drawn out based on the existing project setup.

The new telerobotic system is built based on the existing Rhino robot. The *UTM telerobotic server program* and *UTM telerobotic client program* are built. The user can choose to control the telerobot through natural language command or through manipulating the virtual objects in the virtual environment. The *UTM telerobotic client program with TCP/IP messaging facilities* is built to facilitate the debugging process of the system when an error occurred. The new telerobotic system is implemented on the local area network (LAN). The new telerobotic system is then tested. The process from implementation until testing of the UTM telerobotic system is simplified in the Figure 1.1. Any error discovered is corrected. The new telerobotic system is improved from time to time. The process of correction and improvement is carried out again and again until the final system is produced.

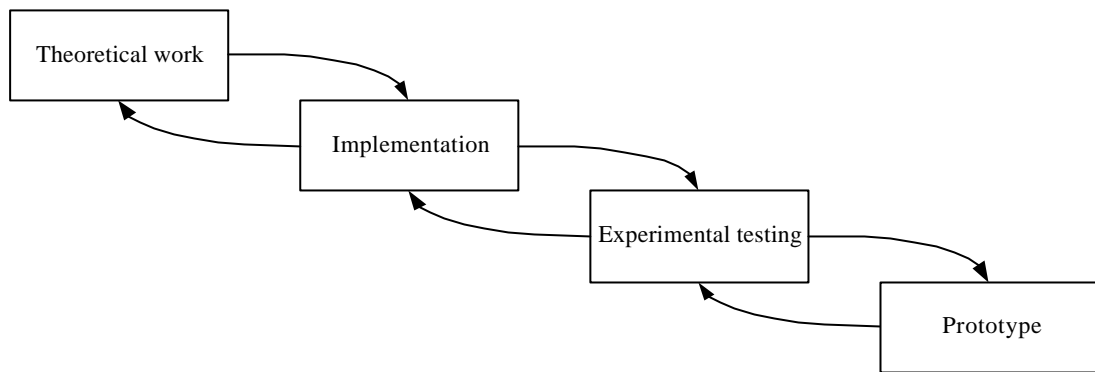


Figure 1.1: Research methodology

1.5 Layout of Report

The Chapter 2 reviews some of the works in telerobotics conducted by other institutes. The Chapter 3 describes the problems faced in the internet-based telerobotics as well as the possible solutions. In the Chapter 4, some of the terms used are discussed in details and the UTM telerobotic system architecture designed is briefly explained. The details of the system design on the vision sub-system, natural language processing, high level command to low level command translation, accuracy and reliability consideration are discussed respectively in Chapter 5, 6 and 7. The result and performance analysis are provided in the Chapter 8. Finally the contribution and the conclusion are given in Chapter 9.

CHAPTER 2

LITERATURE REVIEWS

2.1 Introduction

In this chapter, some of the successful internet-based telerobotic projects developed by the other research groups are reviewed. Since the project to be developed involves fixed arm type robot system, almost all of the projects reviewed are of this type of system except in Section 2.7, which focuses on the use of natural language to communicate with the robot. More studies have been put on the first three projects because of the similarity with the UTM telerobotic system developed.

There is a comparison table given as a summary for every fixed arm type robot system mentioned. There are six items of information given in each of the table, namely institute, application, type of robot, robot/task-oriented, feedback, user guidance and web site. Institute and web site items provide the basic information where the readers can find further details about the project. Application item summarizes the purpose that the system was designed for. Robot/task-oriented item looks at the approaches used to command the robot. The robot-oriented and task-oriented concepts are discussed in detail in the Section 4.1.6. Basically, robot-oriented approach is the method where the user operates the robot based on the movement of the robot. Meanwhile task-oriented approach is the method where the users command the robot by specifying the task to be done by the robot and then the system will plan the movement for it. User guide mentioned here is not referring to the tutorial or help facilities provided to the users. The user guide refers to the

guidance, such as coordinate system, provided to assist the users to complete the task. Finally, feedback item refers to the type of the information provided by the telerobotic system to the users.

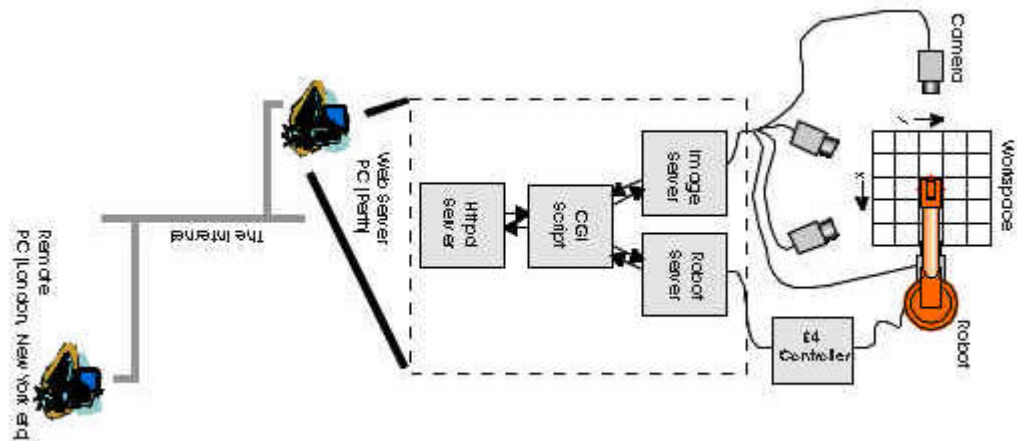
2.2 Australia's Telerobot on the Web (Taylor, K. and Dalton, B., 2000)

This is the longest running web robot on the internet. Perhaps there are as many as 500,000 people have controlled the robots since 1994. The researchers, Taylor, K. and Dalton, B. (2000), won 1996 JARA award for their paper entitled, "Australia's Telerobot on The Web". The users can manipulate wooden blocks on a table in front of the robot through the website.

Currently, the system supports both text base command and usher interface. The usher interface (Friz, H., 1998) as shown in the Figure 2.4, is an augmented reality user interface that allows the users to manipulate the telerobot with the touch of the mouse. According to the paper, a clickable 2D wireframe (shown in Figure 2.5) and clickable images were found in earlier interface. A clickable 2D wireframe of the workspace was used for only 4% of movement requests and 39% of the operators made use of an option to switch off the image of 2D wireframe. Similarly the clickable images are not widely used, and furthermore a more recent innovation of multiple moves is only used for 2.6% of robot movement requests (Taylor, K. and Dalton, B., 1997).

Table 2.1: Comparison table for the system of Australia's telerobot on the web

Institute	University of Western Australia, Australia.
Application	Manipulate wooden blocks on a table in front of the robot.
Type of Robot	ASEA, industrial 6-axis robot.
Robot/Task-oriented	Robot-oriented i) Text base command; ii) Usher; iii) Clickable images (earlier interface only); and, iv) 2D wireframe (earlier interface only).
Feedback	i) Real image; and, ii) Robot status.
User guidance	i) Grid on the table; ii) Known size of wooden blocks; and, iii) Usher.
Web site	telerobot.mech.uwa.edu.au/

**Figure 2.1: The telerobot system for Australia's telerobot on the web**

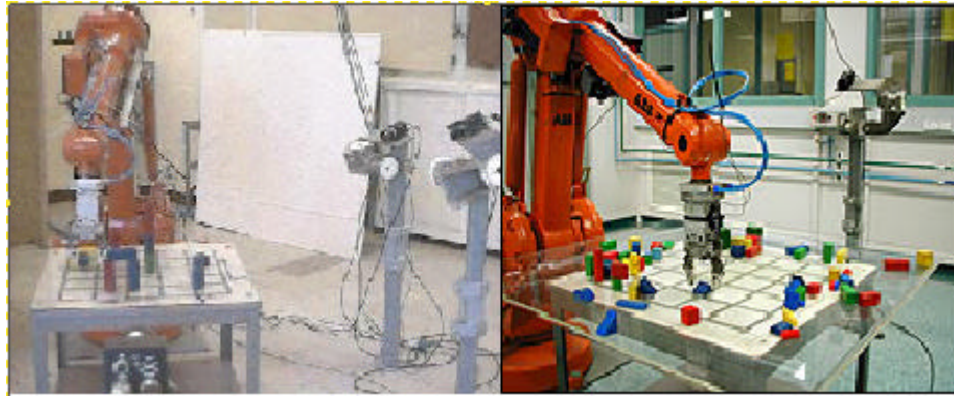


Figure 2.2: The actual site for Australia's telerobot on the web

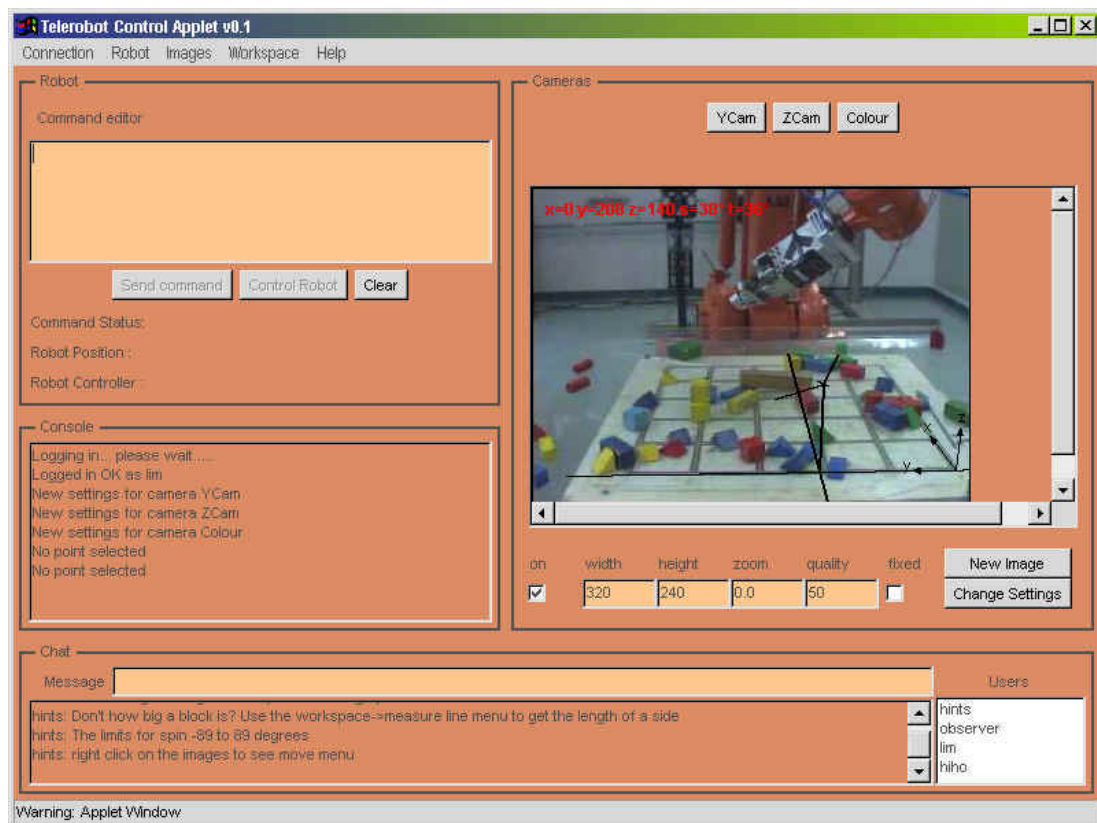


Figure 2.3: Operator interface for Australia's telerobot on the web

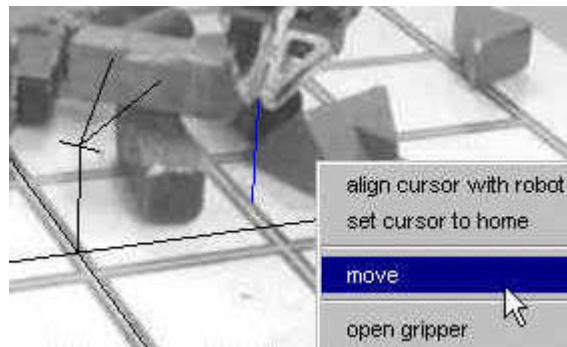


Figure 2.4: Usher - to specify a new position of the gripper, drag these lines to specify x, y, z, spin, and tilt of the new position. Select "move" from the pop-up menu to let the robot move to the new position

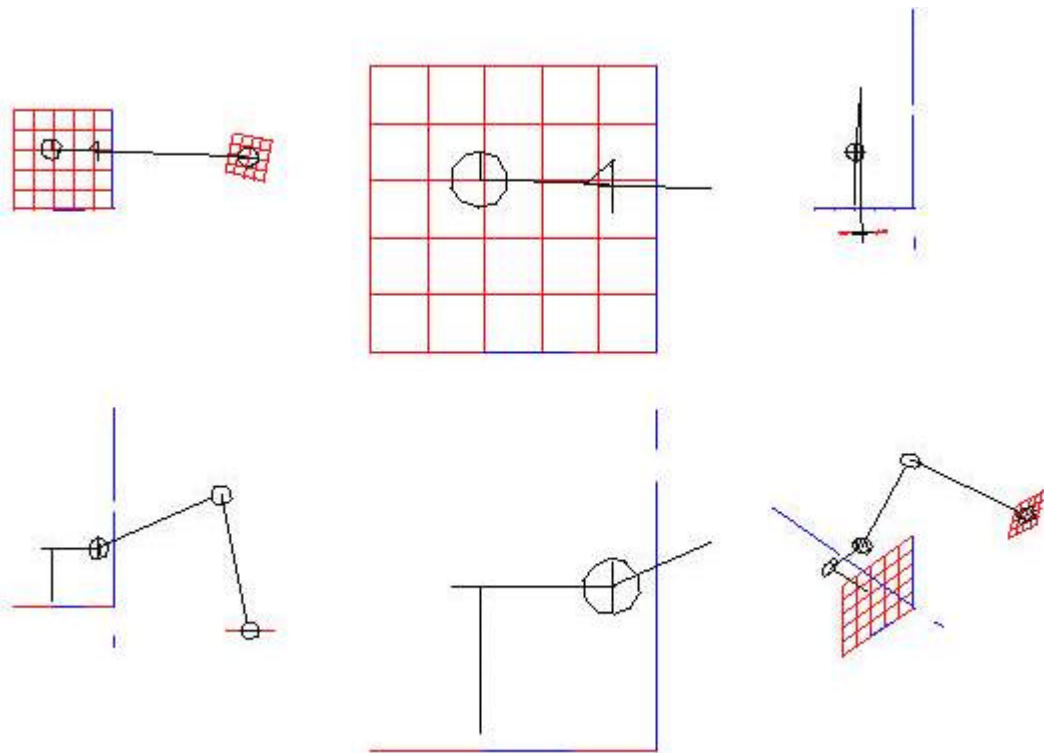


Figure 2.5: Two dimensional wireframe views of the robot included in an earlier interface, circles indicate rotational joints. These images were clickable.

2.3 Carnegie Science Center (CSC) Telerobot

There is not much information can be found on the official web site of the CSC telerobot project. As similar to the Australia's Telerobot project, there are grids provided on the table to guide the operators. There are two real images presented to the users. The user can operate the robot by entering coordinates to go, degrees to lean and spin. Another way to operate the robot is by clicking on the images of the working area presented to the user. The user must click a location on both of the images which represent x-coordinate and y-coordinate of the point where the gripper will be moved to.

Table 2.2: Comparison table for CSC Telerobot system

Institute	Carnegie Science Center, Pittsburgh.
Application	Manipulate wooden blocks on a table in front of the robot.
Type of Robot	6-DOF robot.
Robot/Task-oriented	Robot-oriented i) Text base command; and, ii) Clickable images.
Feedback	i) Real images.
User guidance	i) Grid on the table.
Web site	www.carnegiesciencecenter.org/kids/telerobot.asp

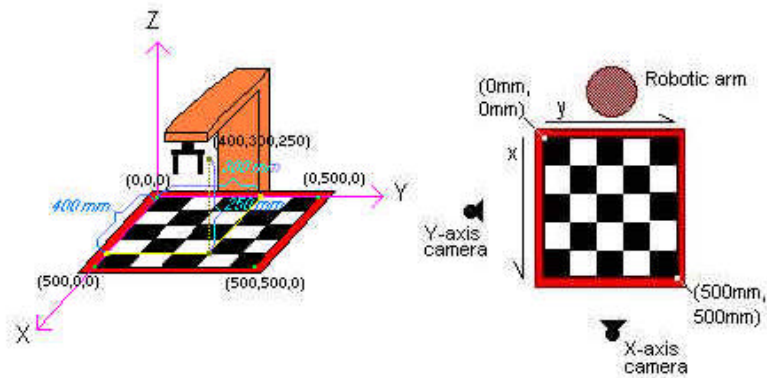
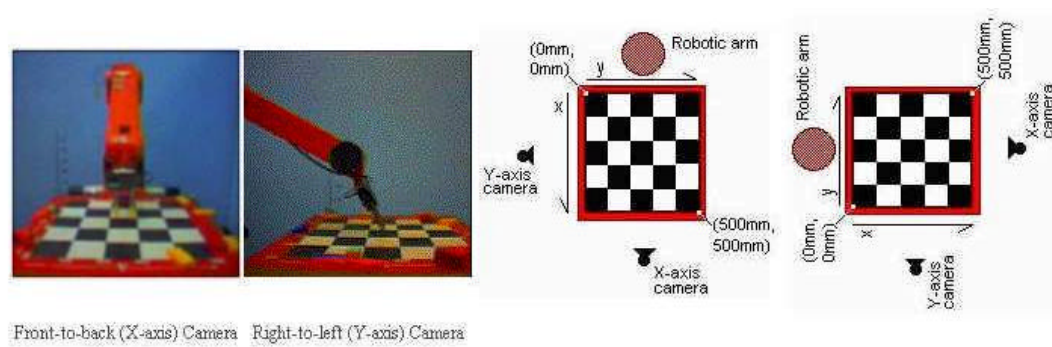


Figure 2.6: Coordinate system for CSC Telerobot



Enter numbers between 0 and 500 for X, Y, and Z.

X position Y position Z position

Spin Lean

Gripper open ☐

Move the robot Quit

Figure 2.7: Part of the operator interface for CSC Telerobot

2.4 Robotoy

This is another internet-based telerobotic project from Australia. The robot used in this project is very similar to the robot currently being used in UTM project. Robotoy is a Mitsubishi Micro-Robot. It is a RM-101 model and is intended for educational use only. Therefore it has neither the precision nor the robustness which

can be provided by an industrial robot. The Figure 2.8 shows how the robotoy system is put together.

Table 2.3: Comparison table for robotoy system

Institute	University of Wollongong, Australia.
Application	Manipulate wooden blocks on a table in front of the robot.
Type of Robot	Mitsubishi Micro-Robot, educational 6-axis robot.
Robot/Task-oriented	Robot-oriented i) Text base command.
Feedback	i) Real image.
User guidance	Not available.
Web site	robotoy.elec.uow.edu.au/

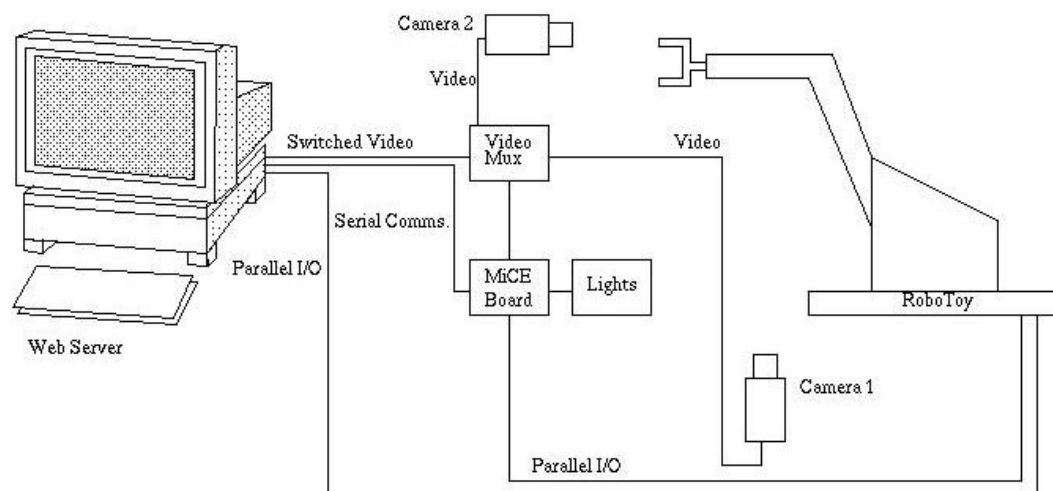


Figure 2.8: Robotoy system

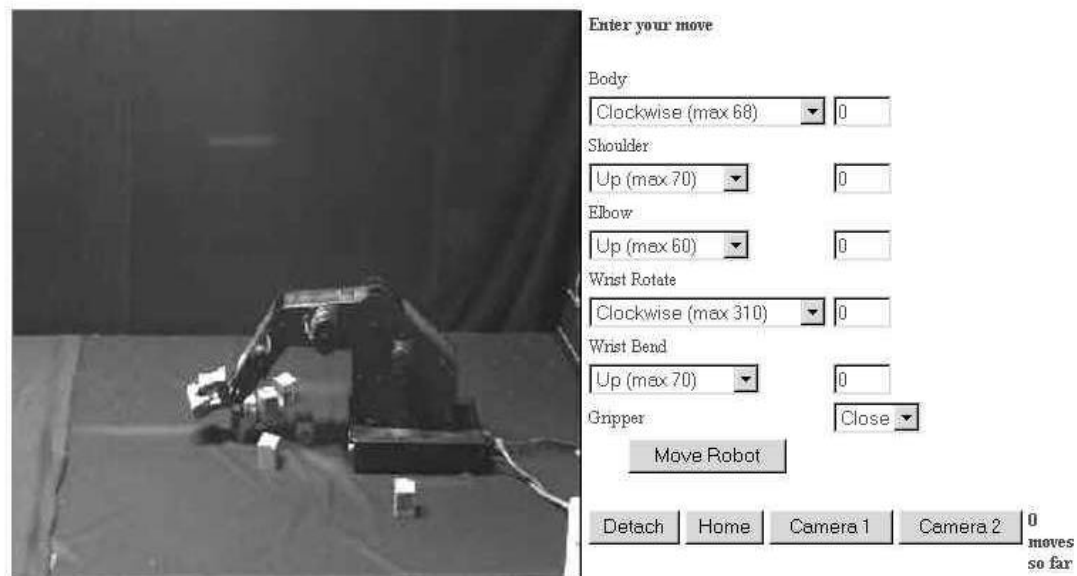


Figure 2.9: Part of operator interface for robotoy project

2.5 Tele-Garden

The Tele-garden project is designed to have different application from the projects discussed earlier. The users are allowed for watering, planting and viewing the garden. The users are presented with a simple interface that displays the garden from the top view, the garden from a global composite view, navigation and information view in the form of a robot schematic. Grids are provided on both of the images to guide the users. By clicking on any of the images, the user can command the robot to move to a new absolute location (left image of the Figure 2.11) or one relative to where they just were (right image of the Figure 2.11). The robot, upon completion of the move, will return a refreshed image of the garden.

To water the garden users must first align the camera image over the section of the garden to water and then press the water button. This will command the robot to release a small squirt of water over the area in view. To plant a seed the user is first requested to find a spot that is relatively empty (there are no restrictions to where one can plant) and then asked to press the plant button. This will cause the

robot to poke a small hole in the ground, proceed to the seed bowl, suck up a seed and deposit it back into the previously dug hole. Nevertheless, the button plant, between water and options buttons, is not available at the time of preparing the writing (shown in the Figure 2.11).

Table 2.4: Comparison table for tele-garden system

Institute	University of Southern California, United States.
Application	Watering, planting and viewing the garden.
Type of Robot	6 DOF robot.
Robot/Task-oriented	Task-oriented i) Text/button base; and, ii) Clickable images.
Feedback	i) Real image; and, ii) Virtual environment.
User guidance	i) Grid on the images.
Web site	telegarden.aec.at/index.html



Figure 2.10: The actual site for tele-garden project

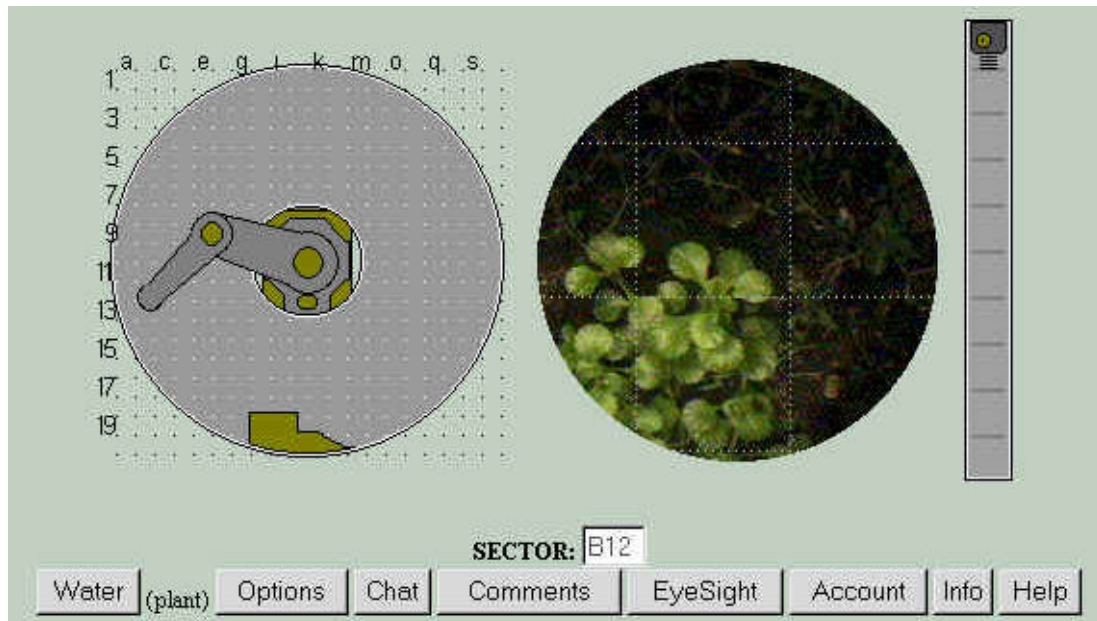


Figure 2.11: Part of the operator interface for tele-garden project

2.6 Painting on the World Wide Web: The PumaPaint Project (Stein, M. R., 2000)

This is a web robot that allows the users to create an original artwork on a World Wide Web. The site allows control of a PUMA 760 robot which is equipped with four paintbrushes together with respective jars of red, green, blue and yellow paint. A white paper is attached to a vertical easel.

The interface provides two types of feedback: one immediate and virtual image while the other time-delayed and real image. The center portion of the interface as shown in the Figure 2.13 is a virtual canvas and the main area of interaction. By clicking, holding and dragging the mouse in this area the user can issues commands to the remote robot to apply paint to the real canvas. In order to increase the fidelity of the virtual canvas, the virtual canvas is coloured as a blob, rather than a shape with sharply defined edges. The blobs contain randomly generated gaps and streaks, and the proportion of area turned to the selected colour progressively decreases as the brush stroke continues.

Table 2.5: Comparison table for PumaPaint project

Institute	Wilkes University, United States.
Application	Painting on white paper attached to a vertical easel.
Type of Robot	PUMA 760, industrial 6-axis robot.
Robot/Task-oriented	Task-oriented i) Virtual canvas.
Feedback	i) Real image; and, ii) Virtual canvas.
User guidance	Not available (not important for this type of application).
Web site	yugo.mme.wilkes.edu/~villanov/

**Figure 2.12: The actual site for PumaPaint project**

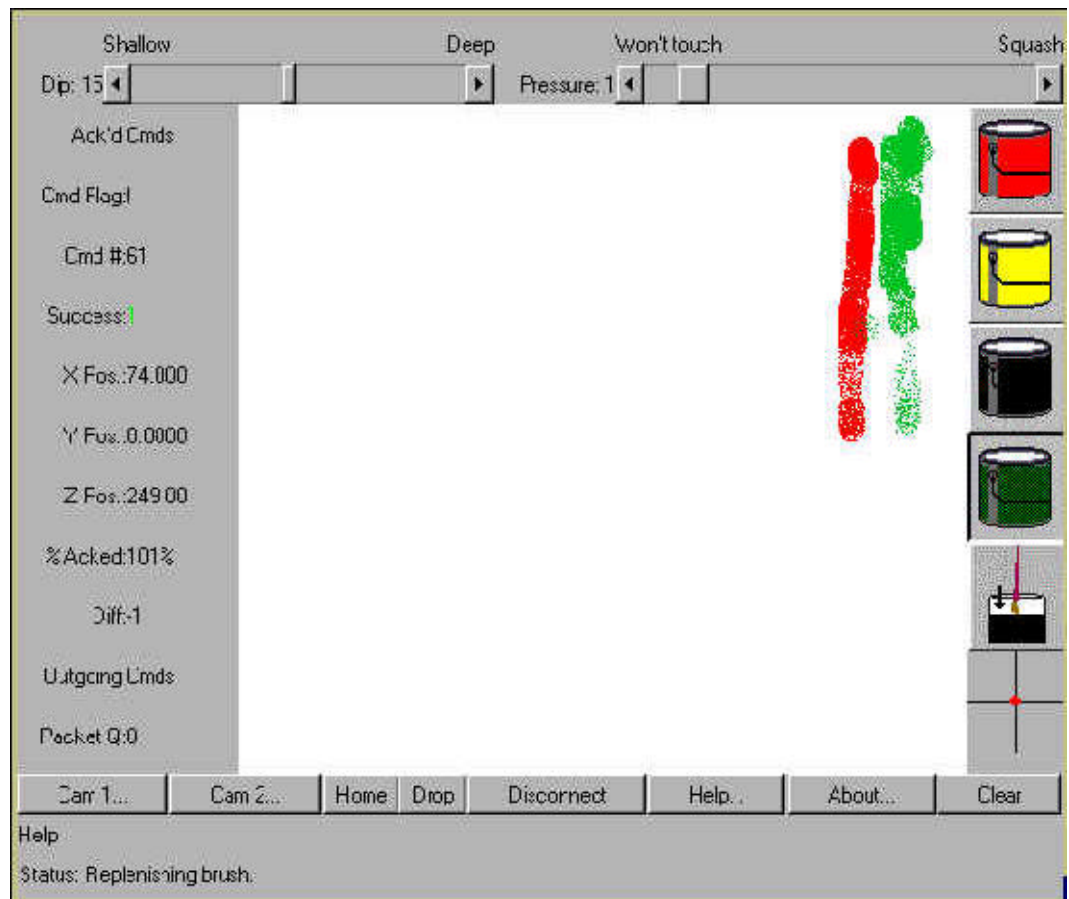


Figure 2.13: GUI with added features to increase the fidelity of the virtual canvas

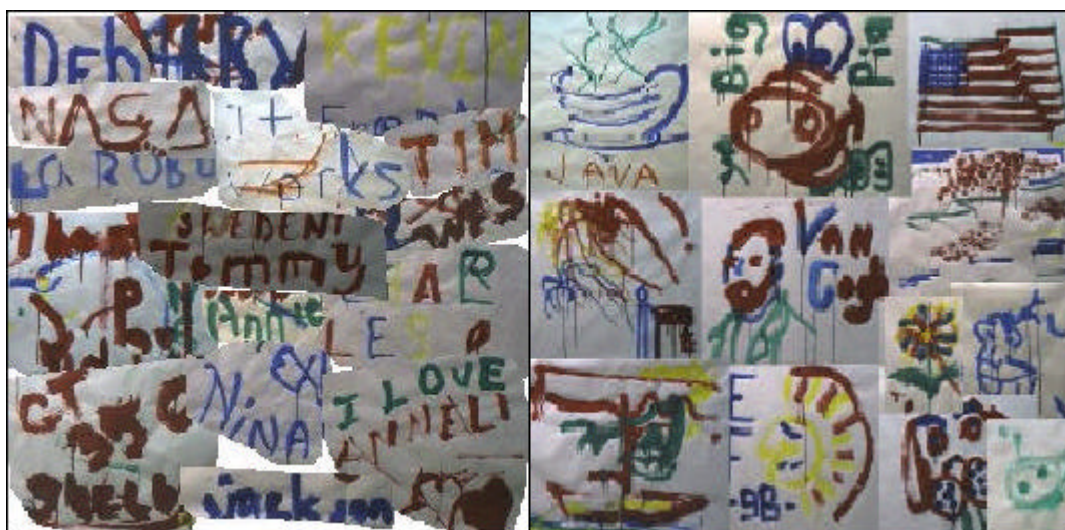


Figure 2.14: Painted text and images

2.7 Natural Communication with Robot (Torrance, M. C., 1994)

This is not an internet-based telerobotic project. The project is highlighted for its effort to apply the natural language for communication with robot. The researcher, Torrance [9], had highlighted the other natural communication approaches as well as their pros and cons. Some of the natural language statements and commands supported are given as below:-

“You are {at|in|on|} *place*”

“You are facing *direction*”

“*Place* is [to the] *direction* of {here|you}”

“Go”

“Stop”

“Go as far as you can”

“Go until you can turn {right|left}”

“Go to *place*”

“What is [to the] *direction* of {here|you}”

The use of the natural language in this project makes the communication between the user and the robotic system become more convenient. The Figure 2.16 is showing the example of the dialogue between the user and the robotic system. This feature is interesting for our implementation in the UTM’s new telerobotic system. The UTM’s new telerobotic system is able to interpret the type-written natural language from the user.

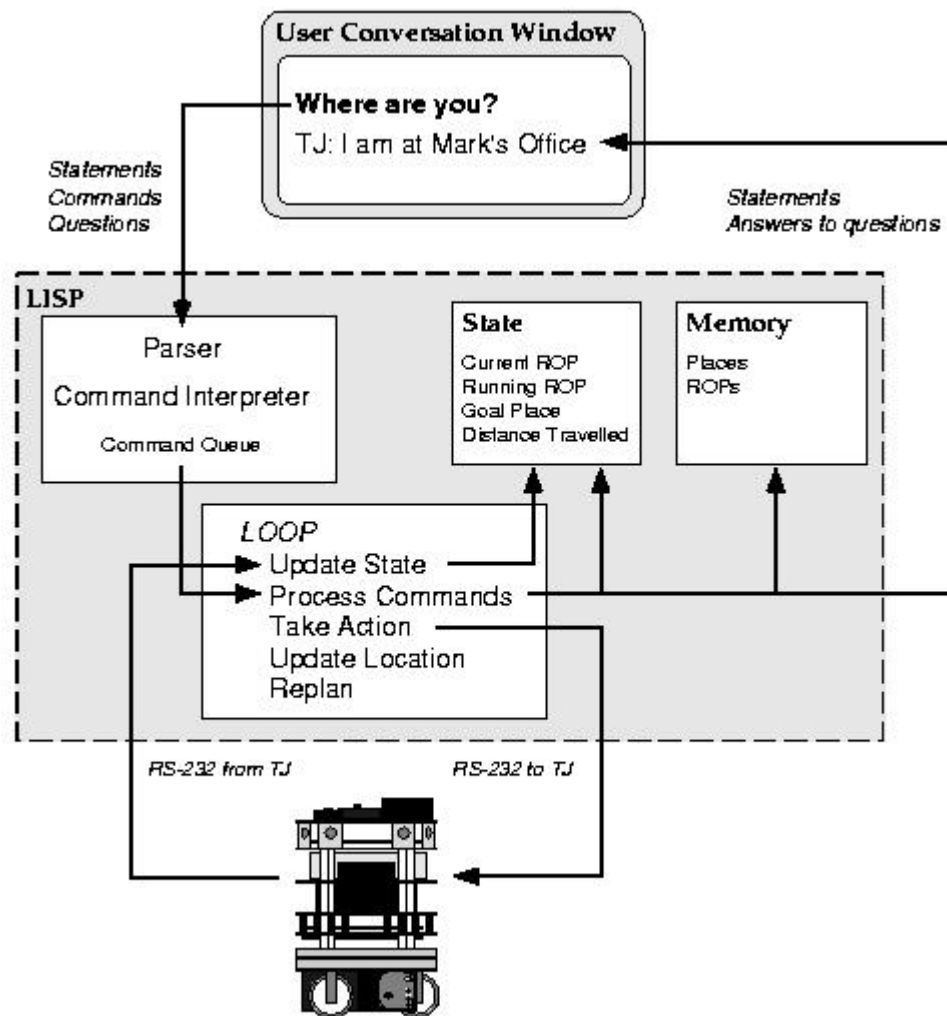


Figure 2.15: Overview of the system

Let's assume the Robot is names Jake and Joe is the "programmer". Starting in the Elevator Lobby...

Joe: We are in the Elevator Lobby.

Jake: Okay.

Joe: We go north and east.

Jake: Is there a door?

Joe: Yes, there is a door with a no smoking sign next to it.

Jake: Okay.

Joe: Going through the doorway we go down the hall to where it splits.

Jake: Okay, are we at the split?

Joe: No, (they walk), now we are.

Jake: Okay.

Joe: We go left and go past a bunch of doors.

Jake: Define bunch.

Joe: More than one.

Joe: Wait, we've now reached a "hole" in the wall.

Jake: What's this "hole" in the wall?

Figure 2.16: Example of dialogue

2.8 Prototype of UTM Web-based Telerobotic System (Fauzi Zakaria, Shamsudin H. M. Amin and Rosbi Mamat, 2000b)

This is the prototype for the Internet-based telerobotic system developed initially at UTM. Figure 2.17 shows the system architecture. The robot used, Rhino (shown in Figure 2.18), is intended for educational purpose and thus the project is facing accuracy and robustness problems of the robot. User can operate the robot through the button, which is not intended for accuracy movement. There is a real image feedback presented to the operator.

Table 2.6: Comparison table for the prototype of UTM web-based telerobotic system

Institute	Universiti Teknologi Malaysia, Malaysia.
Application	Not available.
Type of Robot	Rhino, educational 5 DOF robot.
Robot/Task-oriented	Robot-oriented i) Button base.
Feedback	i) Real image.
User guidance	Not available.
Web site	Not available.

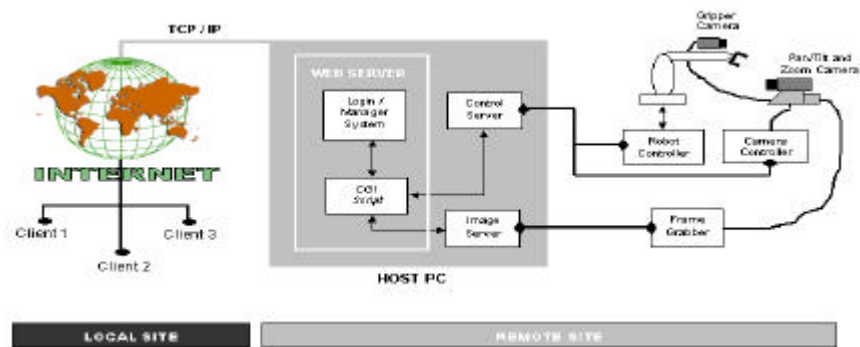


Figure 2.17: System architecture for the prototype of UTM web-based telerobotic system



Figure 2.18: Rhino robot and the actual site

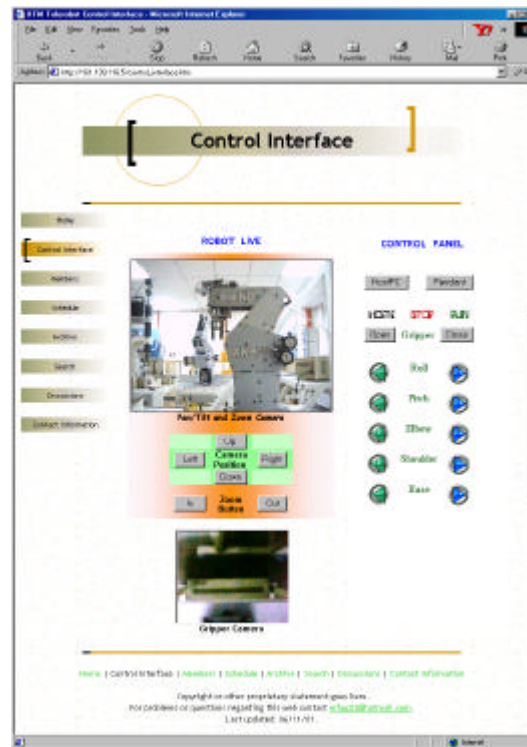


Figure 2.19: UTM Telerobot web-based interface

2.9 Summary

A lot of information is gained from the review of the successful internet-based telerobotic projects developed by the other research groups. This is very helpful for our implementation of the telerobotic system since the problem faced by the other research groups can be avoided in our implementation. The good features of the other project such as the use of natural language can be implemented in our new telerobotic system. The importance factors in the Internet based telerobotics are identified and discussed in the Chapter 3.

CHAPTER 3

IMPORTANT FACTORS IN DEVELOPMENT OF INTERNET-BASED TELEROBOTICS

3.1 Introduction

A success internet-based telerobotic project is not only attracting many users but also get more attention from the users to stay longer for learning and operating the robot. The project must be able to be accessed by the users 24 hours a day and 365 days a year. On the other hand, a good telerobotic system requires less maintenance. There are some important factors which will determine the success of the project. These factors must be considered when designing and developing the telerobotic project.

3.2 Easy to Operate

Basically there are two ways where the internet-based telerobot can be operated. The first method is called robot-oriented approach, where the user can remotely operate the telerobot based on the movement of the robot step by step to perform the work. In this approach, the operation of the system can be controlled by the users through button/text-based interface, model-based interface (2D, 3D or integrated with real image) or master-slave system. However, master-slave system is not suitable for internet-based application since it requires a master robot to exist at

the local site. From the user interfaces mentioned, the user is able to specify and plan the movement of the robot directly.

In button/text-based interface, the operator can use the button or text to specify the values for the movement of the motors at the robot. The Robotoy is an example of the project that uses this type of the interface. Some of the project such as the CSC telerobot extends the button/text-based interface to support the use of the virtual coordinate system as an alternative for the user to specify the coordinate of the end effector. Even so, in the CSC telerobot's user interface most of the robot movement still requires the user to specify and plan step by step.

Model-based telerobotics sometimes is also referred to as teleprogramming. Under this framework, a user interacts with a model of remote site rather than with the remote site directly. The clickable 2 dimensional (2D) wireframe of the robot supported in the earlier interface of the Australia's Telerobot is an example of the model-based interface. A more advanced interface is by using the real robot image with the virtual lines as the guidance instead of using the virtual robot to operate the telerobot. Usher (Friz, H., 1998) is one of the examples that allow users to operate the robot through real image interface.

In the robot-oriented approach, the telerobotic system can be simplified by filtering out the extraneous complications of the system. For example, Taylor, K. and Dalton, B. (1997), noted that for all useful block manipulations only two orientation specifications were required, termed spin, and tilt, rather than roll, pitch and yaw. By limiting the movement, the system has become much easier to be understood and operated without losing any useful functionality.

The second method is called task-oriented approach, where the users need only to specify the task to be done by the robot and then the telerobotic system will plan the path of the movement to complete the task. In this method, the movement of the robot is not controlled directly by the user. The user has no control on how the telerobotic system plans the movement of the robot to complete the task required by the user. The interfaces for task-oriented system can be text-based interface which accepts task-oriented command or model-based (2D, 3D or integrated with real

image). The internet-based mobile robot developed by Roland, S., et al. (1998) is one of the examples that accept task-oriented natural language command. There is another task-oriented system that is non internet-based which was developed by Torrance, M. C. (1994). The PumaPaint project (Stein, M. R., 2000) used 2D model-based interface to implement the task-oriented concept. The user can design the task through the virtual working area.

3.3 Reliability

Reliability is the most difficult criterion for internet-based telerobotics to be made available for 24 hours a day and 365 days a year while requiring minimal maintenance. The system must be able to recover from software and hardware errors. Several internet-based telerobotic projects have been abandoned because of these problems. Some of the suggestions from Taylor, K. and Dalton, B. (1997) to overcome the problems are as below:

- i) Move to more stable and reliable operating system;
- ii) Move to more stable and reliable web-server computer;
- iii) To use hardware and software watchdog; and,
- iv) The workspace is restricted to avoid physical limits such as joint limits.

3.4 Response Time

Response time (Taylor, K. and Dalton, B., 1997) is defined as:

$$tr = tp + \frac{(Ds + Dr)}{Vl} + tc$$

Where tp is request processing time, tc is time taken to initialize communication, Ds and Dr are total data submitted and returned respectively, and Vl is the transmission speed of the link. Since tp and tc contribute less to the value of tr , the main focus is put on Ds , Dr and Vl .

The transmission speed, Vl , is relying most on the medium of transmission. The Integrated Services Digital Network (ISDN) connection provides higher transmission speed than the standard copper Public Switched Telephone Network (PSTN) connection. Meanwhile the fiber optic connection provides higher transmission speed than the Integrated Services Digital Network (ISDN) connection. It is important for the designer to choose the right medium of transmission to be used in the project based on the available resources.

Another way to minimize the response time is to have minimum transmission of data between the server and client. One way of doing this is by minimizing the image size presented to the operator. Further reduction in the data size can be achieved by filtering out unnecessary data. For example, two of the cameras can be calibrated with respect to the robot. This allows automatic cropping of the image to the region of interest centered about the tool center point. The image presented to the operators must be first compressed to the JPEG file format and send only after the completion of the command.

Model-based telerobotics has recently been proposed as a means of overcoming this problem (Lloyd, J.E., Beis, J.S., Pai, D.K. and Lowe, D.G., 1997). Under this framework, a user interacts with a model of remote site rather than with the remote site directly. This allows the client to pre-process the data before sending it to the server. If the data is invalid, the data is verified again with the user until no error. Thus the transmission of data can be reduced. Besides overcoming the time delay, model-based telerobotic system permits other advantages, such as user control of the view point, the ability to test and preview the actions and the introduction of artificial graphical aids for task specification.

Almost all of the projects mentioned in Chapter 2 are using JAVA as the programming language. Due to the nature of the CGI mechanism a whole HTML

page is returned with each request, even if a portion of which does not change between requests. The newer internet technologies of JAVA and frames allows pages to be split up, so that only updated information is refreshed, reducing data transmission further (Taylor, K. and Dalton, B., 1997).

3.5 Human Factors

Most of the internet based telerobotic systems are applying the supervisory control scheme. In supervisory control scheme, human is the central part of the control loop and their behavior becomes an important consideration in the system design. The important information on how to improve the system can be gathered from the analysis of users' behavior. In the Australia's Telerobotic project (Taylor, K. and Dalton, B., 1997), there is a facility for users to register themselves besides all of their activities on controlling the robot is recorded. The incentive to register is the user will be given higher priority for robot access. Some of the important data from the analysis are:

- i) 95 percent registered users are male;
- ii) Indicating a greater interest by youth than older people;
- iii) Less effort to register: registered users operate the robot for only 8% of sessions and 15% of the time;
- iv) Three quarters have given up after waiting for three minutes;
- v) 43% percent do not make any single request to the robot after having gained control; and,
- vi) The users are not staying long enough to learn how to use a complex system.

Besides for new users, they might find out that it is more difficult to control a robot through the Internet rather than controlling the real robot directly. This is because the working objects and the telerobot are both 3 dimensional (3D) while the monitor is able to support 2 dimensional (2D) interface only. The human capabilities such as to estimate the distance, size and locating the objects in 3D environment are

restricted when switched to 2D interface. There must be some kind of guidance to assist the users to extend their capabilities in controlling the robot. In Australia's telerobotic project (Taylor, K. and Dalton, B., 1997) and CSC telerobot project, the user can take advantage of the real coordinate system such as grid on the work space and known objects' size. On the other hand, the virtual coordinate system can be applied on the virtual environment, such as in the PumaPaint project (Stein, M. R., 2000) and the Tele-garden project. It would be more convenient for the users to operate the robot based on virtual coordinate system.

3.6 Interface Design

Interface design is very important since it is where the users interact with the robot and remote site. There are two important criteria to be considered when designing the interface:

i) Informative but simple

A good interface must be informative enough to tell the users on how to operate the robot besides provides others relevant information. All of the information must be made simple and convenient to the users to search and read. For example, the command tilt and spin may be good to be explained by using a diagram rather than text. Roland, S., et al. (1998) had emphasized on interface design in his paper by saying that the design must be “*connect and play*” and any large introduction pages will frighten away most of the users.

ii) Customizable interface

There is no single web page in internet-based telerobotic projects that can be considered as the best suited to all users. The users who use modem internet always face with the bandwidth problems. They have to wait longer for the images to be refreshed. Thus speed is more important than quantity and quality of the images for this group of users. For the convenient of the users, some of the projects include

the facilities to allow the users to customize the interface, such as the resolution of the image, number of the camera views and the approach to control the robot. Both Australia's telerobotic project (Taylor, K. and Dalton, B., 1997) and Tele-garden project support these facilities.

3.7 Summary

The UTM telerobotic system is designed based on the task-oriented approach, where the users need only to specify the task to be done by the robot and then the telerobotic system will plan the path of the movement to complete the task. The task-oriented telerobotic system is more user friendly than the robot-oriented telerobotic system. The overall response time of the task-oriented telerobotic system is shorter than the robot-oriented telerobotic system. The details about both types of the system are discussed in the Chapter 4, 5 and 6.

The new telerobotic system designed is supporting the use of the type-written natural language command. Besides, the user can also plan the task by manipulating the objects in the virtual environment. Every user who manages to login to the UTM telerobotic system is limited to a period of 10 minutes to operate the telerobotic system. The details about the UTM telerobotic system architecture and the design of the user interface are discussed in the Chapter 4.

Meanwhile, the reliability, safety and accuracy design of the system are discussed in the Chapter 7. These include the work cell design, gripper with new fingers design, system self-calibration, working object exception handling, client-server exception handling, log file and error listing.

CHAPTER 4

TERMS DEFINITION AND SPECIFICATION OF UTM TELEROBOTIC SYSTEM

4.1 Terms Definition

Before the system is described, it is necessary to understand some definitions that were used in previous chapters and those are used throughout the chapters and the discussion. The terms definition and explanation are as follow:-

4.1.1 System and Sub-system

A system is a set of connected things (sub-systems) that form a whole or work together. A system has many inputs and outputs. The output is the result of carrying out some process on a set of inputs. A system must have an objective or function. The elements of the system are separated from those things that form part of another system by the boundary (Richards, M., 2001). For example the nervous system is separated from respiratory system and each of the system has their respective functions. The concept of the system is shown in the Figure 4.1.

Meanwhile sub-system is part of a system. Each system is composed of sub-systems, which are themselves made up of other sub-systems. This is because generally every system is part of another system. In other word, a smaller system

which has input, output, boundary and objective but it is a part of a bigger system then the smaller system is called the sub-system of the bigger system. The purpose of the terms system and sub-system is to identify a system until we have a sufficiently clear understanding of the larger system. The Figure 4.2 shows the sub-systems A, B, C, D and E that form a system.

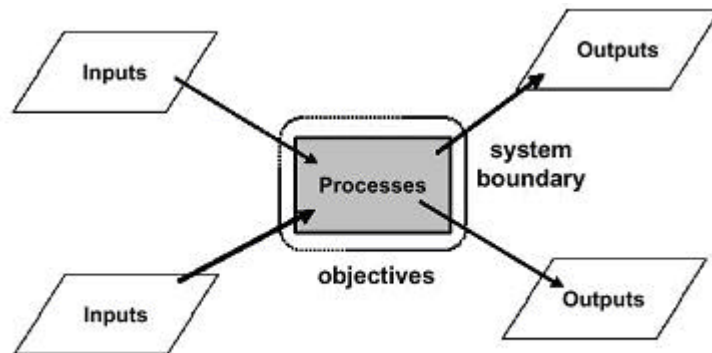


Figure 4.1: System concepts

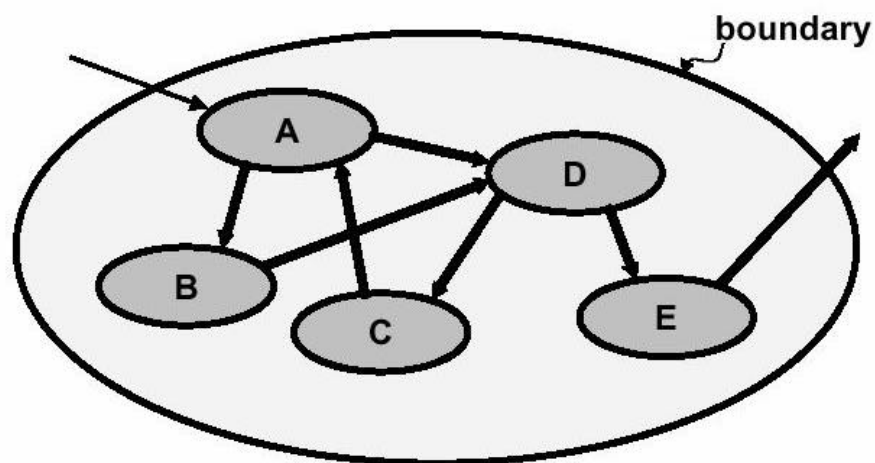


Figure 4.2: Sub-systems of a system

4.1.2 Telerobotic System

The word robot originated from the Czech word *robota*, meaning work. A robot is can be defined as a reprogrammable general-purpose manipulator with external sensors that can perform various assembly tasks (Fu, K. S., et al., 1987). Meanwhile, telerobot is a robot that accepts instruction from a distance, generally from a trained human operator (Fauzi Zakaria, 2000). The word telerobotic system is used to refer to the software, telerobot and equipments used on both of the local and remote sites to work as a system. Human is the user who operates the telerobotic system and thus is excluded from the system. The example of the telerobotic application is the sending of telerobot by NASA to outer space for data collection. The telerobot is controlled from the earth.

4.1.3 Local Site vs. Remote Site

The terms local and remote sites used in Section 4.1.2 are referring to the locations of the telerobot and the operator. The local site is the location where the user operates the telerobot. Meanwhile the remote site is the location where the telerobot situated and works.

On the other hand, the person who operates the UTM telerobotic system from the local site is call as user or operator. Meanwhile the person who setup and manage the system at the remote site is called system administrator.

4.1.4 Client vs. Server

According to the Microsoft Help and Support documentation of the Microsoft Windows XP, client is any computer or program connecting to, or requesting the services of, another computer or program. Client can also refer to the software that enables the computer or program to establish the connection. Meanwhile, server

refers to the computer or program that provides shared resources to network users of a local area network (LAN) or the Internet. As a conclusion, the terms client and server can be used to refer to both the program and the computer. For the ease of understanding and explanation, the terms `telerobotic client program`, `telerobotic server program` and `FTP server` are used to refer to the software. While the terms client and server are referring to the hardware or computers.

4.1.5 Internet

The Internet is a network of networks that connects computers all over the world. The Internet was developed from work done in the 1960s and 1970s by the United States Department of Defense with a project called ARPAnet (Advanced Research Projects Agency net), to connect the computers at some of the colleges and universities where military research took place. By the late 1980s, the Internet had shed its military and research heritage and was available for use by the general public (Young, M. L., et al., 1999).

4.1.6 Robot-oriented System vs. Task-oriented Robotic System

The task-oriented robotic system or so called “task-centric” (Lloyd, J.E., Beis, J.S., Pai, D.K. and Lowe, D.G., 1997) robotic system requires only the operator to specify the task to be done by the system and the system will then plan and carry out a series of actions to complete the task. The task-oriented robotic system is also called a task-level programming system (Craig, J. J., 1986). In contrast, robot-oriented system requires the operator to plan the actions step by step to get the task done.

The robot-oriented system and the task-oriented robotic system can be distinguished by many aspects. The basic command unit for the robot-oriented system is based on the robot movement. For example the commands for arm type

robotic system are shoulder up 30° , elbow down 30° and gripper open. Usually, one basic command unit for the robot-oriented system equals to one robot instruction.

Meanwhile, the basic command unit for the task-oriented system is based on the task designed for the robotic system. For example the commands for the robotic system

which is designed for blocks manipulation can be `PLACE Block3 SO THAT`

`(Block_3_face1 AGAINST Table)`. Usually, one basic command unit for the task-

oriented system equals to a series of robot instructions. The comparison between the

robot-oriented system and the task-oriented robotic system is summarized in the

Table 4.1.

Table 4.1: Comparison between robot-oriented system and task-oriented robotic system

Robot-oriented System	Task-oriented Robotic System
Basic command unit: <ul style="list-style-type: none"> Based on robot movement, e.g.: <ol style="list-style-type: none"> Arm type robotic system: shoulder up 30°, elbow down 30°, gripper open or spray start; Mobile robot: move forward 30 cm, turn left 45°. Usually, one basic command unit equals to one robot instruction. 	Basic command unit: <ul style="list-style-type: none"> Based on the task designed for the robotic system, e.g.: <ol style="list-style-type: none"> Robotic goods sorting system: transfer objects type A to line A and objects type B to line B; Mobile robot: find the target such as heat/light source in unknown environment. Usually, one basic command unit equals to a series of robot instructions.
The system can directly convert the command given to robot instruction since one basic command unit equals to one robot instruction.	The system need to have the ability to “understand” the task required by the user before the task can be converted to a series of robot instructions.
Operator acts as path planner to complete the task. In other word, the operator has full control over how the system completes the task - direct control.	The task controller does the path planning once “understand” the task(s) required to be done. In other word, the operator has no control over how the system completes the task - indirect control.
Autonomy level: low.	Autonomy level: higher (with certain limitations).
Low efficiency in completing the work since every step involved must be manually planned or programmed.	Higher efficiency in completing the work since task controller does the path planning.
Image capturing system (if involved) usually works merely for visual	Image capturing system (if involved) works not only for visual feedback but

feedback.	also as part of the vision system.
Less complicated to be designed and developed.	Complicated to be designed and developed especially the task controller.
Suitable application: usually for repeated/routine work especially in mass production.	Suitable application: usually for the work that is not/less repeated or the work with uncertainties such as goods sorting where the objects may vary in size, shape, orientation and location.
The robot can perform at full capability of the robot	May not be able to do all the task (especially the complicated task) Solution: hybrid system

4.2 UTM Telerobotic System Setup and Application Programs' User Interface

4.2.1 System Setup

The telerobot used in the UTM telerobotic system is the Rhino robot from Rhino Robotics LTD. (Rhino Robot, Inc., 1989). The picture of the Rhino robot is shown in the Figure 4.3. Rhino robot is a 6 degrees of freedom robot. It is a revolute type configuration (RRR) robot arm where the base, shoulder and arm are revolute (R) designed. The robot resembles human arm. The robot dimension is shown in Figure 4.4. The Table 4.2 and Figure 4.5 are showing the motor that is controlling the corresponding robot joint.

The robot is placed in the work cell as shown in the Figure 4.6 and the Figure 4.7. The user can remotely control the robot to manipulate the cube blocks in front of the robot. The dimension of the cube is 18 mm x 18 mm x 18 mm. The cube is taken from the word game, Boggle. The sample of the cubes is given in the Figure 4.8. The cubes are placed on the working area, which is set at 110 mm from the ground.

A camera, Sony X-03 is fixed on the top of the working area. The camera is put at 935 mm exactly on the top of the working area. This is to make sure the whole of the working area can be captured by the camera. The camera is linked to the Matrox Genesis frame grabber for image capturing. The image processing is done

with the Matrox Imaging Library (MIL). The details of the robot selection, the work cell design, the working area and the working object definition are discussed in the Chapter 8.

The telerobot is connected to the serial port 1 of an Intel Pentium III 400 MHz processor's PC with 192 MB RAM. The operating system is Windows 2000 Server with service pack 3. The telerobotic system is controlled by a program named *UTM telerobotic server program*. The user can remotely control the system from a program called *UTM telerobotic client program*. There is another version of *UTM telerobotic client with TCP/IP messaging facilities* which is developed for system maintenance purpose. All the programs are developed using Microsoft Visual C++ 6.0.

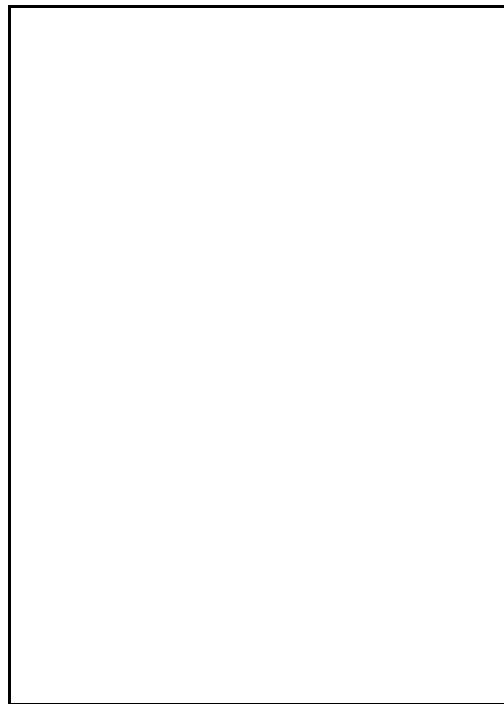


Figure 4.3: Rhino robot

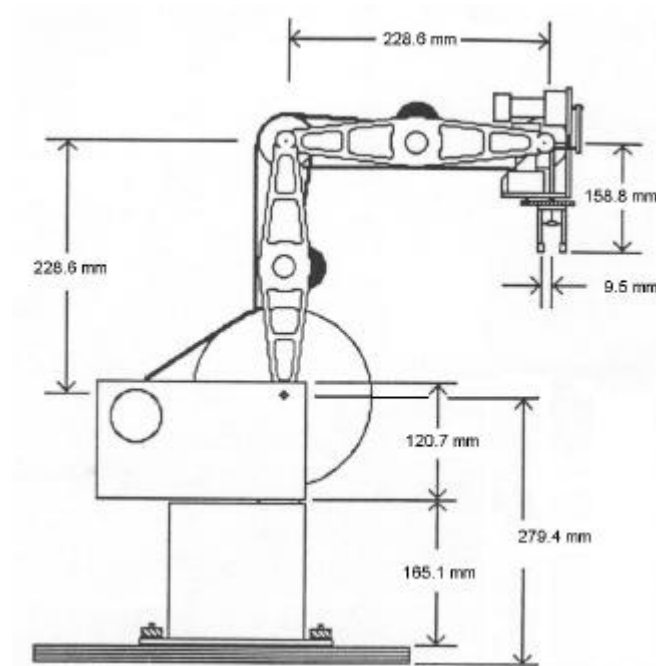


Figure 4.4: Dimension of Rhino robot

Table 4.2: Motor and the corresponding joint

Motor	Joint
A	Gripper open and close
B	Wrist
C	Hand
D	Elbow
E	Shoulder
F	Waist

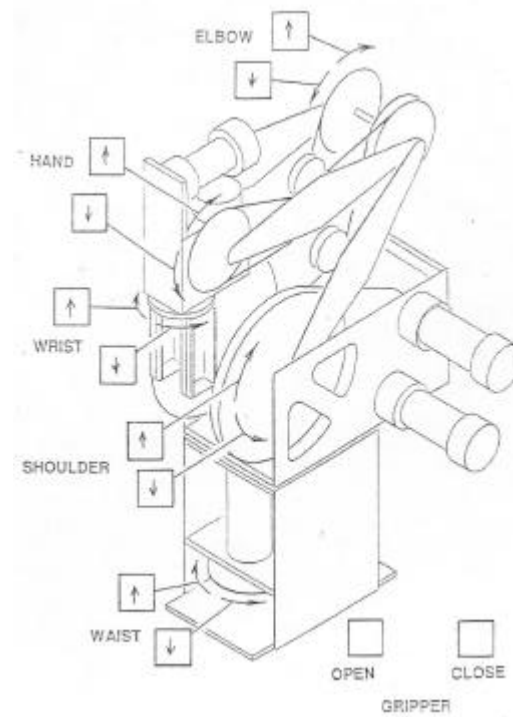


Figure 4.5: Motor and the corresponding joint



Figure 4.6: Front view of the robot work cell



Figure 4.7: Top view of the robot work cell

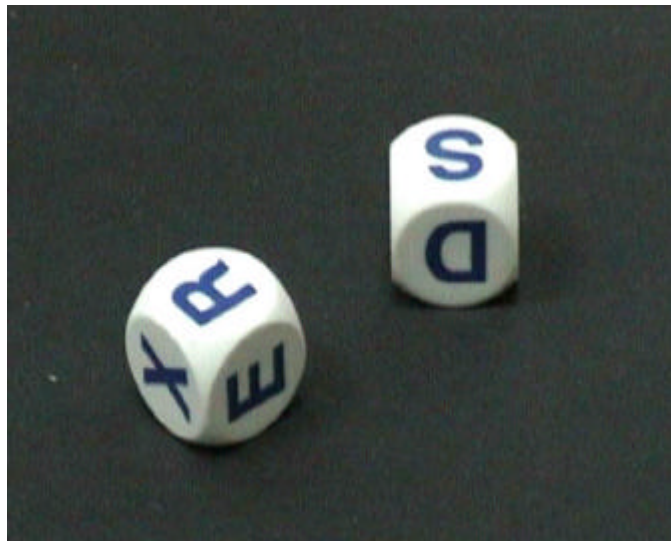


Figure 4.8: Sample of working objects

4.2.2 Application Programs' User Interface

As mentioned in the Section 4.2.1, there are three applications programs developed for the UTM telerobotic system:

- i) *UTM telerobotic server program;*
- ii) *UTM telerobotic client program; and,*
- iii) *UTM telerobotic client program with TCP/IP messaging facilities.*

The user interface of the *UTM telerobotic server program* is shown in the Figure 4.9. There are 2 real images shown on the user interface. The image on the left is a live video of the top view of the working area. It shows the current state of the working area as well as the progress of the task. From the image shown on the left in the Figure 4.9, there are 4 working objects on the working area. The image on the right is showing the progress of the image processing and the object recognition. The image on the right of the Figure 4.9 is displaying the 4 objects recognized from the image captured.

There are four parts of the user interface labeled as 'First 3 Objects Information', 'RS232', 'TCP/IP' and 'Error Code' which are designed for system maintenance and troubleshooting. The part 'First 3 Objects Information' is displaying the information of the first 3 objects recognized. The parts 'RS232' and 'TCP/IP' can be used to send the data to the serial port and *UTM telerobotic client program* manually. The part 'Error Code' displays the error detected from the UTM telerobotic system. The details about the error code are discussed in the Section 7.11.

Meanwhile there are two buttons labeled as 'HOST' and 'Start System' in the 'System' part. When the system is initialized, the button 'HOST' is first clicked to enable the Rhino robot to be controlled from the computer instead of teach pendant. After that the system can be started by a single button click at 'Start System' button. The system will first initialize the robot, vision self calibration and then make online. The system start is designed as a single button click for the convenience of the system administrator.

The Figure 4.10 shows the user interface of the *UTM telerobotic client program* without TCP/IP messaging facilities. There is a message list box with the message "... hi. you are welcome..." for displaying the message the user keyed in and the message conveyed by the program in natural language. There are two buttons labeled as 'Help' and 'Login'. When the 'Help' button is clicked, the importance information is displayed in the message list box. If the user wishes to control the telerobot, he or she required to login the UTM telerobotic system by clicking on the 'Login' button. Once the user is granted the permission, the image on the virtual environment is updated. The user can issue a command either by using the keyboard or through the mouse. The user can key in the natural language command through the keyboard. The user may also right click on the virtual environment to get the command menu and then manipulate the virtual object by using the mouse. On the other hand, the Figure 4.11 shows the user interface of the *UTM telerobotic client program with TCP/IP messaging facilities*. The system administrator can use the program to send the message to the UTM telerobotic program manually.

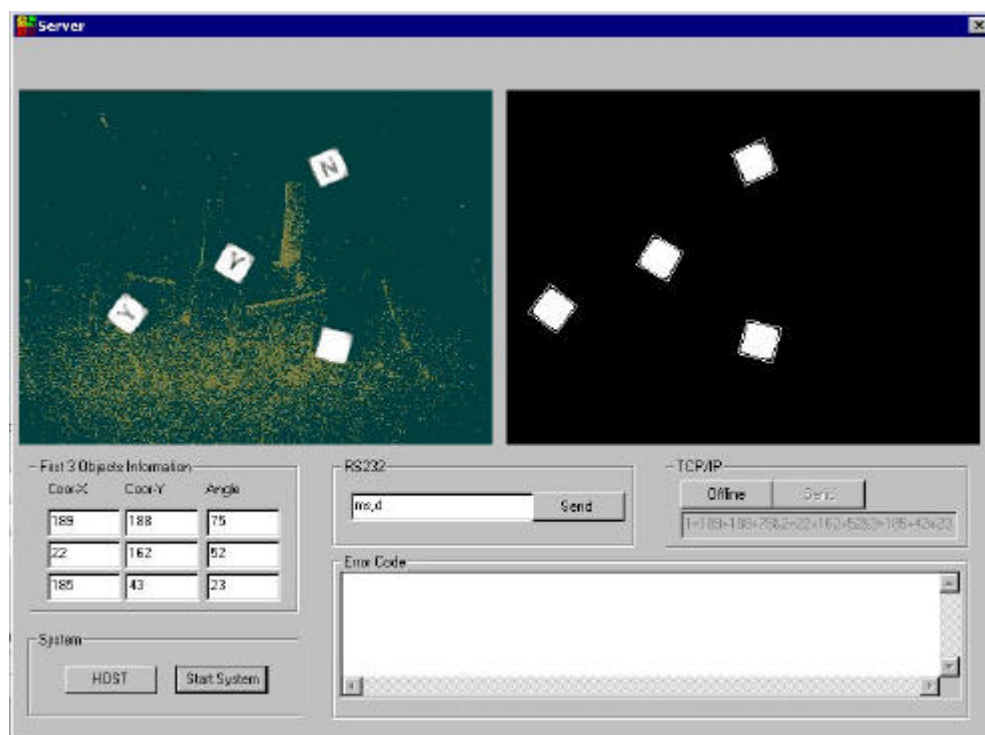


Figure 4.9: User interface of the UTM telerobotic server program

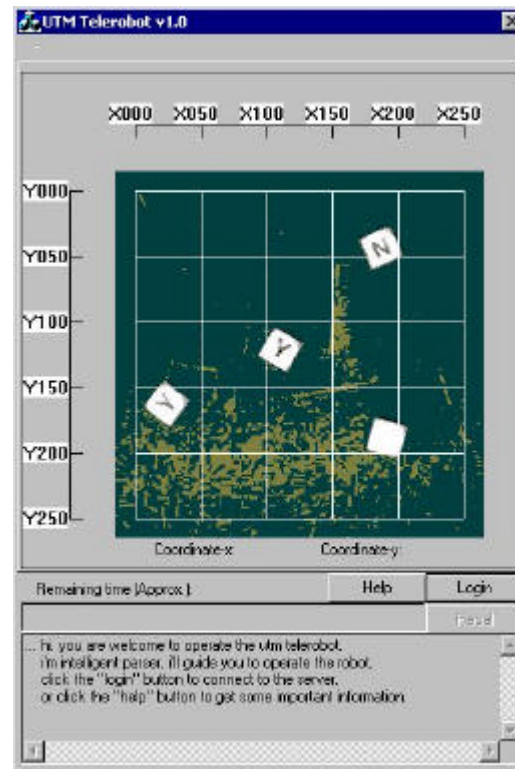


Figure 4.10: User interface of the UTM telerobotic client program

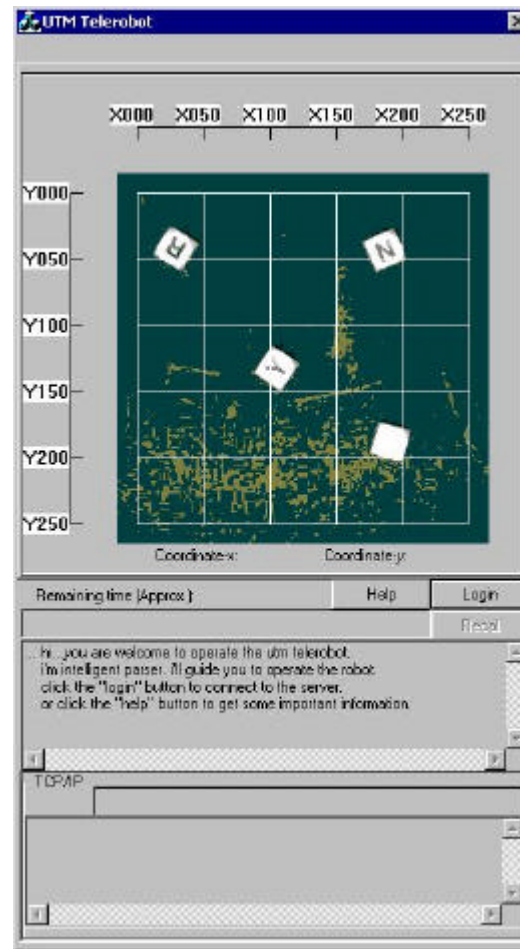


Figure 4.11: User interface of the UTM telerobotic client program with TCP/IP messaging facilities

4.3 System Architecture

The Figure 4.12 shows the relationship between the client and the server of the UTM telerobotic system on a local area network (LAN). The server must be online in order for the telerobotic system to work. Once the client login and accepted by the server, the client is given the control right over the telerobotic system. The user is given 10 minutes to operate the telerobotic system. The architecture of the telerobotic client and server systems is shown in the Figure 4.13 and Figure 4.14. The connection between the client and server is maintained by the client-server connection manager.

At the local site, the telerobotic client system can accept task-oriented command from the operators either through mouse operation, type-written natural language or the combination of the both type of inputs. The command is then processed by the command pre-processor – either by the interpreter or the parser. The purpose of the command pre-processor is to remove the illegal commands such as spelling mistake, syntax error as well as to check the validity of the mouse operation.

Once the system accepts the command from the operator to execute the task, the task is then passed to the task pre-processor. The task pre-processor will check if the task could be performed by the task planner. Apparently not all tasks can be performed by the task controller due to the limitation in the system design. The task is rejected for example when the objects are too close and beyond the ability of the robot. During the command and task pre-processing stage, the information such as the number of objects, location and orientation are required. If the task failed, the user will be informed about the error happened.

On the other hand, if the task is success the client-server connection manager will encode the task information into a URL string. The URL string is then received and parsed by the client-server connection manager at the server system. The task requested is passed to the task controller to decide on the objects that should be moved and rotated. The information is then passed to the path planner to do the path planning as well as to transform the task into action. The progress of the task is feedback to the task planner through the vision sub-system. The path planner is stopped if any error detected. The error is recorded in the log file and error listing. The log file records not only the errors detected but all the system activities since the telerobotic server program is launched until the program is terminated.

The task planner, the robotic sub-system (the robot with its controller) as well as the vision sub-system (combination of the camera and the image controller) can be simplified into a closed-loop block diagram as shown in Figure 4.15. This allows the server to perform self-supervised and this mechanism is called visual servoing (Peter I. Corke, 1996). In other word, the system is able to complete the task given without the supervision from the user.

When the task is completed, the latest top view image of the working area is captured. The image is kept in the directory of the FTP server. The latest environmental information is abstracted from the image captures. The information is kept as the knowledge of the task planner. The information is then passed to the client-server connection manager to be encoded as the URL string. The URL string and the latest top view image of the working area are feedback to the client. The client will update the virtual environment and the real image displayed. After that, the user can plan for the next task.

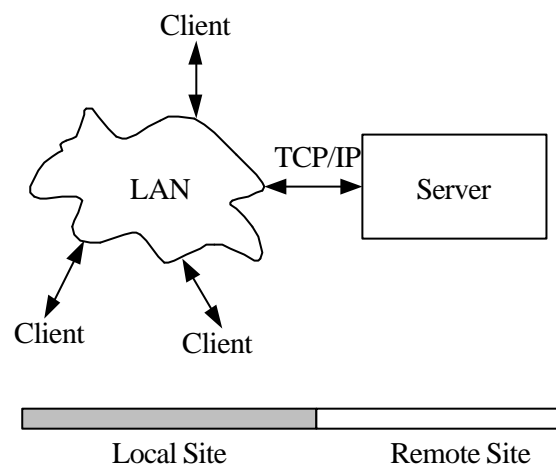


Figure 4.12: Local and remote sites of the UTM telerobotic system

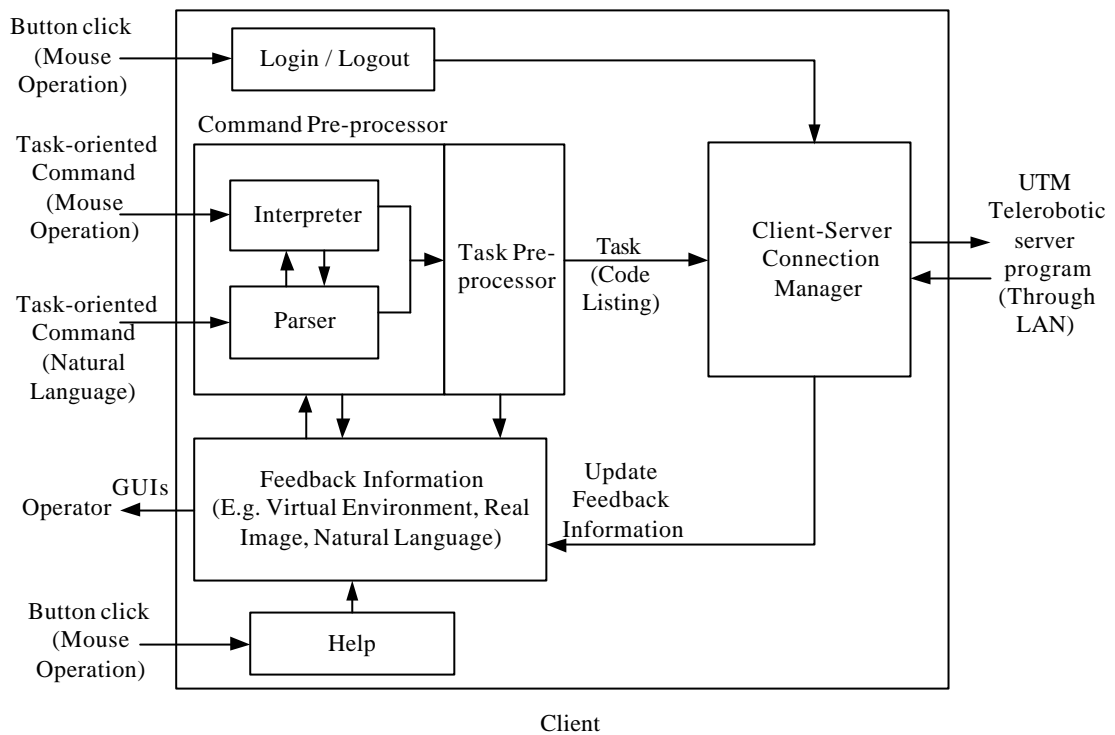


Figure 4.13: UTM telerobotic client program architecture

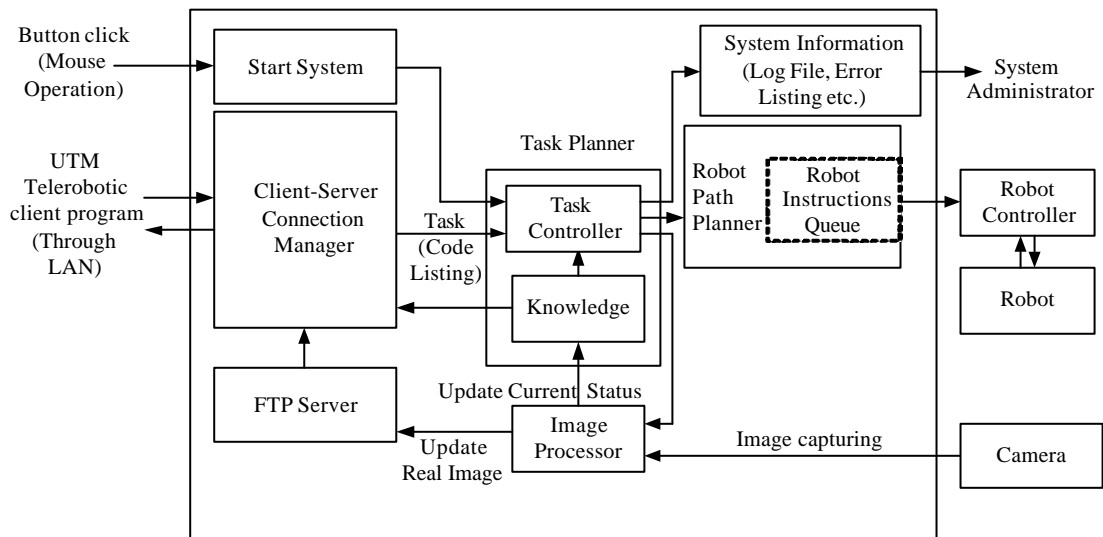


Figure 4.14: UTM telerobotic server program architecture

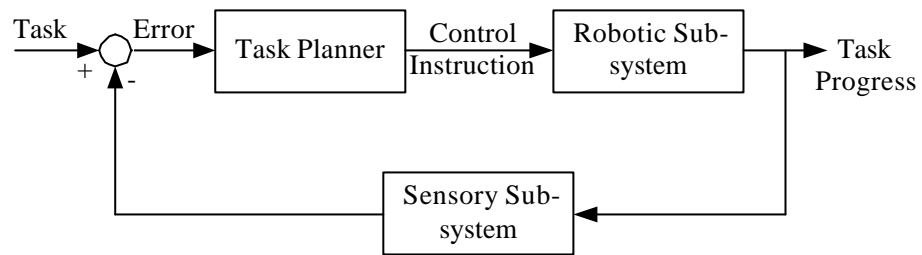


Figure 4.15: Block diagram of the task control sub-system (closed-loop)

4.4 Summary

The important terms are defined and discussed in this chapter. The UTM telerobotic system setup, application programs' user interface and the system architecture are briefly explained. The details of the system architecture are explained in the Chapter 5, 6 and 7.

CHAPTER 5

VIRTUAL WORKING AREA CONSTRUCTION, COMMAND PRE-PROCESSOR AND TASK PRE-PROCESSOR

5.1 Introduction

There are three application programs developed for the telerobotic system. In this report, *application program based telerobotic system* is developed instead of the *web based telerobotic system*. There are several advantages of using the application program instead of the web browser to control the telerobotic system. The application program allows the programmer to have better control over the program functions. The pre-processing function can be easily incorporated in the telerobotic client program, such as the natural language parser, the mouse interpreter, the task pre-processor, the client-server manager, the virtual working area and the virtual working objects. In other word, the application program allows the programmer to have better control over the volume of data transfer between the telerobotic client and server programs.

The discussion in the chapters 5, 6 and 7 are limited to the theory part of the architecture design of the telerobotic client and server programs. The details in the coding are not explained due to the length and complexity of the program. The implementation of the theory in the programming language can be referred to the coding in the CD-ROM attached. The relevant comments are given to the main and important part of the coding.

5.2 Virtual Working Area and Working Objects Construction

In the *telerobotic client program*, the user can manipulate the virtual working objects shown in the virtual working area through the mouse operations and type-written natural language. The virtual working area is built on the window dialogue and is overlapping the working area top view image as shown in the Figure 5.1. The coordinate of the window dialogue starts at coordinate (0, 0) at the top-left corner. The virtual working area is defined from coordinate (100, 90) to coordinate (350, 340) of the window dialogue with certain area left at the left-hand side and the top of the virtual working area. The origin for the virtual working area is defined at the top-left corner for the ease of the programming. The area left is for the scale labeling to guide the user. The dimension of the virtual working area is defined as 250 pixels x 250 pixels and is explained in the Section 7.5. Four grid lines are drawn vertically and horizontally across the virtual area. When the user move the mouse pointer over the virtual working area, the coordinate of the mouse pointer according to the virtual working area is displayed at the bottom of the virtual working area.

The URL string send from the telerobotic server program contains the information of the working objects, such as the orientation and centre of the objects. The dimension of top view of the virtual working object is defined as 25 pixels x 25 pixels square. A mark is labeled at one corner of the square as reference corner. In Visual C++, the functions `MoveTo()` and `LineTo()` are used to draw the straight line. In order to draw a virtual working object, four of the coordinates of the object need to be calculated. For example, to draw a virtual object as shown in the Figure 5.2 with the centre at coordinate (100, 100) and orientation 0° , the coordinate of point at the top right-hand corner is calculated as follows:-

The length of the side, $l_s = 25 \text{ pixels}$

The coordinate-x of the centre, $x_o = 100$

The coordinate-y of the centre, $y_o = 100$

Angle value from the centre to the point at the top right-hand corner, $q = 45^\circ$

The length of the diagonal, $l_d = \sqrt{l_s^2 + l_s^2} = \sqrt{25^2 + 25^2} = 35 \text{ pixels}$ (rounded to nearest integer)

Half of The length of the diagonal, $l_{d/2} = 18 \text{ pixels}$ (rounded to nearest integer)

Offset value for coordinate-x, $o_x = l_{d/2} \times \cos \mathbf{q} = 18 \times \cos 45^\circ = 13$ (rounded to nearest integer)

Coordinate-x of the point, $x_1 = x_o + o_x = 100 + 13 = 113$ (1)

Offset value for coordinate-y, $o_y = l_{d/2} \times \sin \mathbf{q} = 18 \times \sin 45^\circ = 13$ (rounded to nearest integer)

Coordinate-y of the point, $y_1 = y_o - o_y = 100 - 13 = 87$ (2)

Thus, the coordinate for the point at top right-hand corner is (113, 87).

The rest of the corners can be found by using the same working steps with the corresponding value for the \mathbf{q} . For example, the point of the top-left corner can be found by using $\mathbf{q} = 135^\circ$. The coordinate of the point at the top-left corner is (87, 113). The virtual working object is then drawn by linking all points at the corners found. The telerobotic client program is able to draw up to 100 virtual working objects.

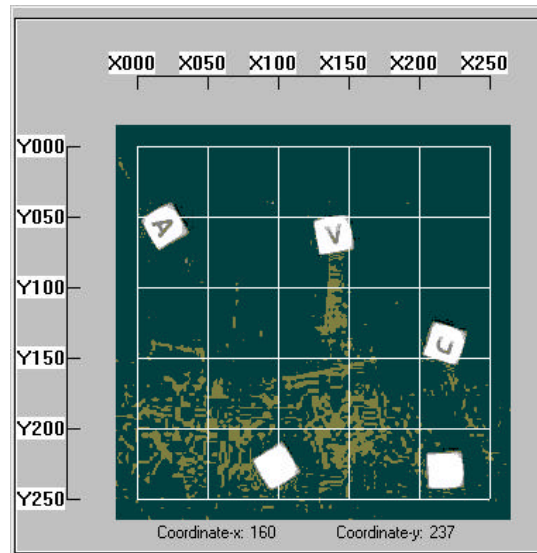


Figure 5.1: Virtual working area with the image of the top view of the working area

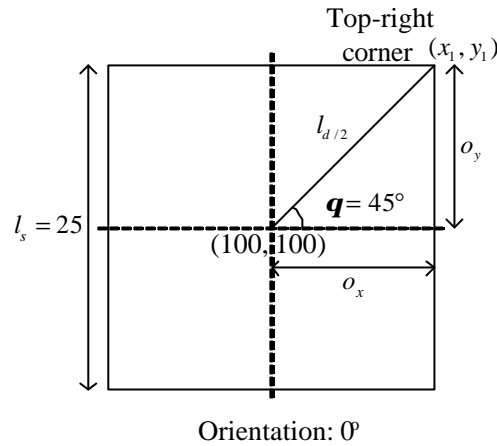


Figure 5.2: Coordinate for the top right-hand corner

5.3 Command Pre-processor

Since the task of the telerobotic system is limited to two-dimension operations, the possible task-oriented commands need to be defined. The basic operations for two-dimension operation are to move the object, to rotate the object and to instruct the system to carry out the task. However some of the advanced operations are supported for the convenience of the user. Some of the advanced commands supported are to move the object vertically (called `offsetX` command), to move the object horizontally (called `offsetY` command), to move an object to the centre between two objects (called `between` command), `undo` and `redo` the commands. Although the virtual working area is defined from (0, 0) to (250, 250), the size of the actual working area allowed for the centre of the virtual working object is limited to (10, 10) to (240, 240). The purpose of the restriction is discussed in the Section 7.5. For the type-written natural language, the coordinate value of the coordinate (10, 10) for example, is accepted as `x10y10` for the convenience of the user.

The task-oriented command is designed to be supported both by the type-written natural language and the mouse operation. The system is designed so that the user can 100% rely on the type-written natural language or 100% rely on the mouse

operations to perform task-oriented command. The system is also offering the flexibility for the user to combine the usage of both the input methods on every task-oriented command. The details of the support for type-written natural language, the mouse operation and the integration of the both input methods are discussed in the Section 5.3.1, 5.3.2 and 5.3.3. The feedback and guidance from the command pre-processor which are conveyed in the natural language are displayed in the message list box as shown in the Figure 4.10.

5.3.1 Command Pre-processor: Task-oriented Natural Language

5.3.1.1 Natural Language Overview

The natural language is the language used by the human in their daily activities such as speaking and writing. A language-comprehensive program must have the knowledge about the structure of the language, including what the words are, how to combine the words into sentences and what the words mean. A language-comprehensive program is always less intelligent than the human. The important aspects of what makes human intelligent are the general world knowledge and reasoning ability of human. There are many different forms of the knowledge (Allen, J., 1987) that might be incorporated into the language-comprehensive program, such as:-

- i) **Phonetics and phonological knowledge:** It concerns how words are realized as sounds. This is an important concern for automatic speech-understanding systems.
- ii) **Morphological knowledge:** It concerns how words are constructed out of more basic meaning units called morphemes. For example, the word “friendly” is constructed from a root form “friend” and the suffix “-ly”.

- iii) Syntactic knowledge: It concerns how words can be put together to form sentences that look correct in the language. This form of knowledge identifies how one word relates to another.
- iv) Semantic knowledge: It concerns what words mean and how these meanings combine in sentences to form sentence meanings.
- v) Pragmatic knowledge: It concerns how sentences are used in different contexts and how context affects the interpretation of the sentence.
- vi) World knowledge: It includes the general knowledge about the structure of the world that language user must have in order to, for example, maintain a conversation.

In the developed *UTM telerobotic client program*, the syntactic knowledge, semantic knowledge and world knowledge are involved. The natural language parser realizes the syntax of the task-oriented natural language commands supported. The meaning of each task-oriented natural language commands allows the natural language parser to perform the corresponding action. The natural language parser is made known of the current virtual objects status in the virtual working area. The details of the application and relationship among the different forms of knowledge are discussed in the Section 5.3.1.3.

5.3.1.2 Natural Language Generation

Natural language generation is the process of producing a set of natural language sentences that realize the goal of the system (Allen, J., 1987). Based on the possible task-oriented operations discussed in the Section 5.3, a set of task-oriented natural language is defined for the developed telerobotic system. The task-oriented natural language defined is in the type-written natural language form. Below are the set of task-oriented natural language developed in this report. The part of the sentence that is bracketed must be replaced by the required data.

5.3.1.2.1 **"move** {coordinate xy of an object} **to** {coordinate xy}"

This command instructs the telerobotic client program to move the object to the coordinates specified. The centre of the object is placed on the coordinates specified. The value for {coordinate xy of an object} is limited from x0y0 to x250y250 while the value for {coordinate xy} is limited from x10y10 to x240y240. For example the command "move x10y100 to x200y200" is telling the telerobotic client program to move the object at the coordinate x10y100 to a new coordinate x200y200.

5.3.1.2.2 **"rotate** {coordinate xy of an object} {degree of rotation}"

This command instructs the telerobotic client program to rotate the object anti-clockwise according to the degree specified. The object is rotated with respect to its centre. The value for {coordinate xy of an object} is limited from x0y0 to x250y250 while the value for {degree of rotation} is limited from 0 to 360 degrees. For example the command "rotate x10y100 45" is telling the telerobotic client program to rotate the object at the coordinate x10y100 with the angle 45 degree anti-clockwise with respect to the centre of the object.

5.3.1.2.3 **"offsetx** {coordinate xy of an object} **to** {coordinate xy}"

This command instructs the telerobotic client program to offset the object horizontally. The value of coordinate-y of the object is maintained. On the other hand, the value of coordinate-x of the object is changed according to the value of the coordinate-x of the mouse pointer. The value for {coordinate xy of an object} is limited from x0y0 to x250y250 while the value for {coordinate xy} is limited from x10y10 to x240y240. For example the command "offsetx x10y100 to x200y200" is telling the telerobotic client program to offset the object from coordinate-x x10 to the new coordinate-x x200.

5.3.1.2.4 **"offsetY {coordinate xy of an object} to {coordinate xy}"**

This command instructs the telerobotic client program to offset the object vertically. The value of coordinate-x of the object is maintained. On the other hand, the value of coordinate-y of the object is changed according to the value of the coordinate-y of the mouse pointer. The value for {coordinate xy of an object} is limited from x0y0 to x250y250 while the value for {coordinate xy} is limited from x10y10 to x240y240. For example the command "offsety x10y100 to x200y200" is telling the telerobotic client program to offset the object from coordinate-y y100 to the new coordinate-y y200.

5.3.1.2.5 **"between {coordinate xy of an object 1} and {coordinate xy of an object 2} put {coordinate xy of an object 3}"**

This command instructs the telerobotic client program to move the object 3 to the centre between the object 1 and the object 2. The centre of the object 3 is placed exactly on the centre between the object 1 and the object 2. The value for {coordinate xy of an object 1}, {coordinate xy of an object 2} and {coordinate xy of an object 3} are limited from x0y0 to x250y250. Let's assume that the coordinates x100y100 and x200y200 are the centre of the object 1 and the object 2. The command "between x100y100 and x200y200 put x10y100" is telling the telerobotic client program to move the object 3 at the coordinate x10y100 to a new coordinate x150y150 where x150y150 is the centre between object 1 and the object 2.

5.3.1.2.6 **"copy coordinateX {coordinate xy of an object 1} apply to {coordinate xy of an object 2}"**

This command instructs the telerobotic client program to offset the object 2 horizontally to the point where the coordinate-x of the object 2 is equal to the

coordinate-x of the object 1. The value for {coordinate xy of an object 1} and {coordinate xy of an object 2} are limited from x0y0 to x250y250. Let's assume that the coordinate x100y100 is the centre of the object 1. The command "copy coordinatex x100y100 apply to x200y200" is telling the telerobotic client program to move the object 2 at the coordinate x200y200 horizontally to a new coordinate-x of x100.

5.3.1.2.7 "copy coordinateY {coordinate xy of an object 1} apply to {coordinate xy of an object 2}"

This command instructs the telerobotic client program to offset the object 2 vertically to the point where the coordinate-y of the object 2 is equal to the coordinate-y of the object 1. The value for {coordinate xy of an object 1} and {coordinate xy of an object 2} are limited from x0y0 to x250y250. Let's assume that the coordinate x100y100 is the centre of the object 1. The command "copy coordinatey x100y100 apply to x200y200" is telling the telerobotic client program to move the object 2 at the coordinate x200y200 vertically to a new coordinate-y of y100.

5.3.1.2.8 "copy orientation {coordinate xy of an object 1} apply to {coordinate xy of an object 2}"

This command instructs the robot to rotate the object 2 until the orientation of the object 2 is equal to the orientation of object 1. The value for {coordinate xy of an object 1} and {coordinate xy of an object 2} are limited from x0y0 to x250y250. Let's assume that the orientation of the object 1 at the coordinate x100y100 is 45°. The command "copy orientation x100y100 apply to x200y200" is telling the telerobotic client program to rotate the object 2 at the coordinate x200y200 until the orientation is 45°.

5.3.1.2.9 "object information {coordinate xy of an object}"

This command inquires the telerobotic client program about the centre and the orientation of the object specified. The value for {coordinate xy of an object} is limited from x0y0 to x250y250. Let's assume that the orientation and the centre of an object are 45° and x100y100. The command "object information x110y100" is resulting the telerobotic client program to feedback the orientation and the centre of the object at the coordinate x110y100, which are 45° and x100y100 respectively.

5.3.1.2.10 "undo"

This command cancels the last task-oriented command issued by the user. The virtual environment and the virtual working objects are restored to the state before the last task-oriented command issued and implemented. The telerobotic client program is able to undo up to 100 commands. Let's assume an object is rotated clockwise for 45° by the user. The command "undo" is telling the telerobotic client program to cancel the last task-oriented command issued by the user, which is to rotate the object anti-clockwise for 45° to restore the telerobotic client program to the previous state.

5.3.1.2.11 "redo"

This command repeats the last task-oriented command which was undoing by the user. The virtual environment and the virtual working objects are restored to the state before the last undo command. The telerobotic client program is able to redo up to 100 commands. Let's take the example discussed in the Section 5.3.1.2.10, the command "redo" is resulting the object "undo" to be rotated clockwise again for 45°.

5.3.1.2.12 "cancel"

This command terminates the last task-oriented command that is currently activated by the user. The virtual environment and the virtual working objects are restored to the state before the current task-oriented command is activated. Let's assume that the user issues a command "move x100y100". This command instructs the telerobotic client program to move the object at the coordinate x100y100. The "move" command is currently activated until the user specified the location where to put the object. The "cancel" command terminates the "move" command which is currently activated.

5.3.1.2.13 "restore"

This command causes the telerobotic client program to restore all the objects to their previous location before any task-oriented command was issued. All records for the "undo" and "redo" commands are reset. The "restore" command has no effect once the "execute" command was issued. The "execute" command is discussed in the next section. Let's assume the user has instructed the telerobotic client program to move and rotate all the objects in the virtual environment. The "restore" command is resulting all the objects to be restored to their previous state.

5.3.1.2.14 "execute"

This command causes the command pre-processor to inform the task pre-processor about the final output as required by the user in the virtual working environment. The details of the process done by the task pre-processor are discussed in the Section 5.5. At the end of the process, the telerobotic server program will instruct the telerobot to achieve the final output as required by the user.

5.3.1.3 Task-oriented Natural Language Processing

The bottom-up method is used to process the task-oriented natural language command issued by the user. The natural language command is first undergone the low level processing, then the next level of processing until the highest level of processing. The levels of the processing are:-

i) Character filtering

The characters used by the system are limited to the alphabets and numbers. The unsupported characters such as the symbols +, =, ? and @ are removed automatically during the character filtering. Thus the user needs not to re-key in again the task-oriented natural language.

ii) Capital to small letter conversion

The "move" command, for example, might be keyed in as "Move", "mOve", "moVe" and "MOVE" by the user. This will cause the difficulty in the supported word filtering which is done in the next level of processing. Thus all the capital letters keyed in by the user are converted to the small letters.

iii) Word filtering

At this level, the words in the natural language command keyed in are checked word by word. The natural language parser checks the word with the library of the supported words. All the possible formats for the coordinate value are supported. If the user key in the coordinate value x1y200 as x01y200 or x001y200, the natural language parser treats it as x1y200. The user is prompted of the unsupported word keyed in if found in the task-oriented natural language command.

iv) Sentence structure conversion

The next level of the processing is the sentence parsing. It is inconvenient to process the sentence in their original format. The sentence structure is converted to the corresponding symbols according to the value given in the Table 5.1. For example, the natural language command "move x10y10 to x100y100" is converted into two strings "v+n+p+n" and "1+7+2+7". The symbols v stands for verb, n stands for nouns and p stands for preposition.

v) Parsing

Parsing is the process of analyzing a sentence to determine its structure according to the grammar (Allen, J., 1987). The process of the parsing is now simplified since the sentence structure was converted to the corresponding symbols and values. The combination of the symbols and values is compared with the supported syntax. The user is prompted of the unsupported syntax if found in the task-oriented natural language command.

vi) Semantic interpretation

The meaning of the sentence is interpreted at this level. The corresponding action according to the command is carried out. For example the meaning of the natural language command "move x10y10 to x100y100" is to move the virtual working object at coordinate x10y10 to the coordinate x100y100. Once the meaning of the sentence is interpreted, the natural language parser checks if the virtual object exists at the coordinate specified (world knowledge). If the virtual working object is found, the new virtual working object is redrawn at the coordinate specified while the previous virtual working object is deleted. On the other hand, if the virtual working object cannot be found, the user is prompted about the error by the intelligent parser.

Table 5.1: Words to symbols and values conversion

Supported Words	Symbols	Values
Move	v	1
Rotate	v	2
OffsetX	v	3
OffsetY	v	4
Copy	v	5
Execute	v	6
Put	v	7
Undo	v	8
Redo	v	9
Cancel	v	10
Restore	v	11
Apply	v	12
CoordinateX	n	1
CoordinateY	n	2
Orientation	n	3
Object	n	4
Information	n	5
Degree value, e.g. 45	n	6
Coordinate value, e.g. x1y1	n	7
Between	p	1
To	p	2
And	c	1

5.3.2 Command Pre-processor: Mouse Operation

Task-oriented natural language is easy to understand. The task-oriented natural language allows the value of the coordinate and the rotation angle to be specified exactly. However the drawback is that the user might feel inconvenient to

key in the whole sentence of the task-oriented natural language. In view of this, the task-oriented command is designed to be supported by mouse operation and type-written natural language. The input method is easy to be learnt and used. However it might be a bit time consuming and inconvenient for specifying the exact value for the coordinate and the rotation angle.

It is assumed that the user has been connected to the server. First of all, the user has to right click on the virtual working area. A popup menu is displayed as shown in the Figure 5.3. The user can choose any task-oriented command from the menu. The intelligent parser will guide the user through the rest of the process. When the command is activated, for example if the user chose the "move" command from the menu, the intelligent parser does provide the opportunity for the user to cancel the activated command. The user can right click on the virtual working area. Another popup menu is displayed. The menu is shown in the Figure 5.4. Note that, the "continue" command is available only for the mouse operation and is not available in type-written natural language. If the user chose the "cancel" command, the activated command is canceled. If the user then right clicks again on the virtual working area, the popup menu showing the task-oriented command is displayed.

The mouse operation is handled by the mouse interpreter. The processing of the mouse operation is much easier than the natural language processing. In Windows programming, every operation of the mouse such as right click is considered as an event. The corresponding function is activated. For the example given, the right click will activate the `OnRButtonDown()` function in the VC++ and the value of the point of the right click can be accessed from the function. Let's take a comparison between the task-oriented natural language command "move {coordinate xy of an object} to {coordinate xy}" as implemented by the mouse operation.

When the user chooses the "move" command from the popup menu, the Boolean value for the variable `bMove` is set to `TRUE`. The intelligent parser will guide the user to left click on the virtual working object to be moved. The left click from the user will activate the `OnLButtonDown()` function in the VC++. The

coordinate of the left click is checked if the virtual working object is chosen. The sides of the virtual working object can be transformed into four linear equations. The four linear equations of the virtual working object with the centre at the coordinate (100, 100) and orientation θ are given in the Figure 5.5. If the coordinate of the left click is fulfilling all the inequalities $x \geq 87$, $x \leq 113$, $y \geq 87$ and $y \leq 113$, the object with the centre at the coordinate (100, 100) and orientation θ is left clicked by the user. The Boolean value for the `bObjClick` is set to TRUE.

When the Boolean values of `bMove` and `bObjClick` are set to TRUE, the function `OnMouseMove()` which is activated by the mouse movement event will cause the virtual working object being move according to the location of the mouse pointer. The intelligent parser will guide the user to left click on the location of the virtual working area where the virtual working object is placed. The second left click on the virtual working area will cause the object to be dropped and the "move" command is terminated.

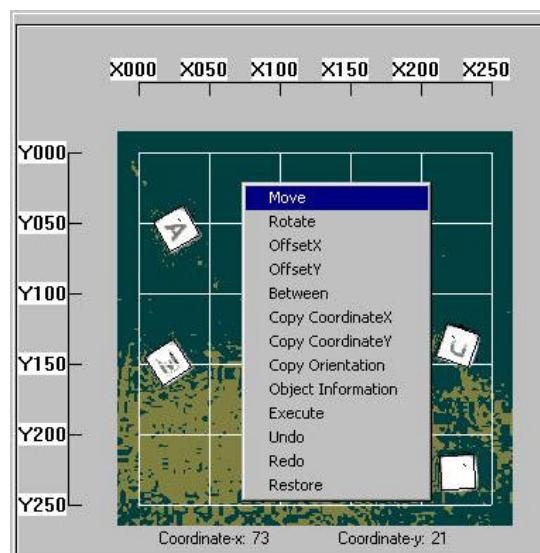


Figure 5.3: Popup menu 1

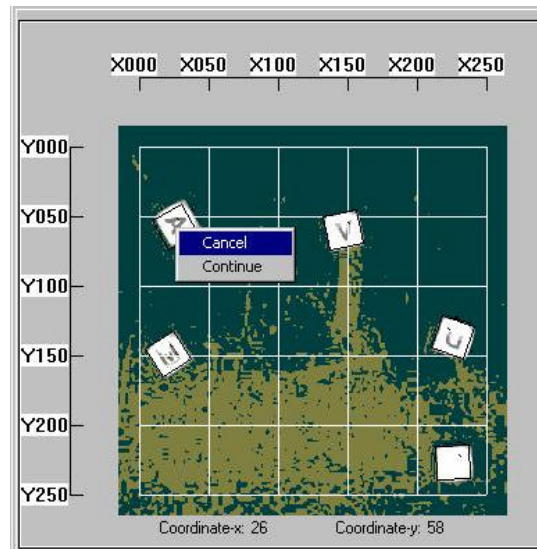


Figure 5.4: Popup menu 2

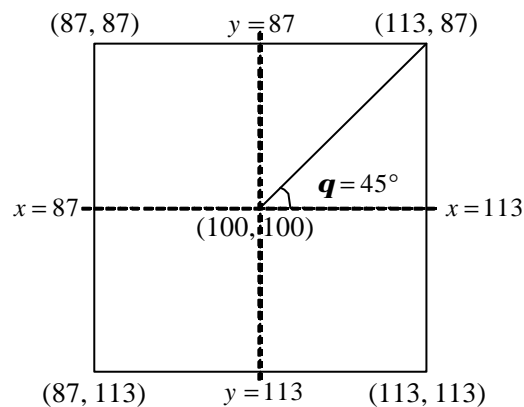


Figure 5.5: Linear equations for the sides

5.3.3 Integration of Natural Language Parser And Mouse Events Interpreter

The telerobotic client program offers the integration of both of the command input methods discussed above. The ease of the task-oriented command input through mouse operation and the ease of the exact value specification through type-written natural language can be achieved at the same time. The user can activate the task-oriented command through the mouse operation and then use the type-written

natural language to specify the exact value. In order to achieve the integration of the both command input methods, the ability of the natural language parser and the mouse interpreter are extended to be able to understand the every sub-command in the task-oriented command supported. For example, the natural language parser must be able to understand the meaning of the sub-commands such as "move", "move x20y20", "move x20y20 to", "x20y20" and "move x20y20 to x200y100".

Let's take the example discussed in the Section 5.3.2 where the user is moving the virtual working object by using the mouse operation. After the "move" command was activated and the virtual working object was clicked, the Boolean values of `bMove` and `bObjClick` are set to TRUE. The intelligent parser will guide the user to left click on the location of the virtual working area where the virtual working object is placed. The intelligent parser will also inform the user about the support of the coordinate value specification through the type-written natural language. If the user decided to specify the coordinate value through type-written natural language, the input is processed by the natural language parser. The action is then carried out by the natural language parser since the last sub-command of the task-oriented command is a type-written natural language.

5.4 Intelligent Parser

In the process of operating the system, the intelligent parser will guide the user to operate the system. The intelligent parser is called "intelligent" because it can guide the user to operate the system as well as to detect the error done by the user. The intelligent parser is able to detect the spelling error, the syntax error and logical error in the task-oriented natural language command. The intelligent parser is conveying in natural language. This makes the system become more user friendly. The symbol placed in front of the message is indicating the type of messages. The symbols used are:-

```

>> the message keyed in by the user or the message send from
    the server
!!! Error message
... Feedback from the intelligent parser (if no error
    detected)

```

In the message listing, the latest message is placed on the top of the message list. For the example as shown in the Figure 5.6, the line numbering is purposely labeled for the explanation. The first message is started at line 4 and ended at line 1. The next message is at line 5 which is indicating the user tries to login to the server. The line 8 is telling the user that he or she is now connected to the server. The following messages are given at line 9, 10, 11 and 13.

```

13    ... which object to be moved?
12    you may click on the object or key in the coordinate value.
11    >> move
10    ... you may choose any command from the menu.
9     >> command menu
8     ... you are now connected to the server.
7     you can start to operate the utm telerobot.
6     right click on the virtual environment to get the command menu.
5     >> login
4     ... hi...you are welcome to operate the utm telerobot.
3     i'm intelligent parser. i'll guide you to operate the robot.
2     click the "login" button to connect to the server.
1     or click the "help" button to get some important information.

```

Figure 5.6: Natural language listing

5.5 Task Pre-Processor

The objective of the task pre-processor is to check if the task can be carried out by the telerobotic system. When the user issuing the "execute" command, the environmental information of the virtual working area is passed to the task pre-processor. There might be some cases where the task issued by the user is not able to be performed by the telerobotic system. For example, the telerobotic system is not

being able to grip or to place the working objects too close to each other due to the physical design of the gripper. In certain cases, the user might try to overlap the virtual working objects in hoping that the telerobotic system will stack the working object one on the other.

In order to avoid the objects are being too close or being overlapped by the user, the distance between the virtual working objects must be checked. The distance allowed between the objects is limited to 65 pixels. Let's assume that there are two virtual working objects located at the coordinate (x_1, y_1) and (x_2, y_2) respectively. The formula for the distance between two virtual working objects is given as below:-

$$\text{Distance, } d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

If there are five virtual working objects labeled as 1, 2, 3, 4 and 5, it is not necessary to check the distance between all the combinations. The combinations needed to be checked are 5-4, 5-3, 5-2, 5-1, 4-3, 4-2, 4-1, 3-2, 3-1 and 2-1. The combinations such as 4-5, 3-5 and 2-5 are the same as the combinations 5-4, 5-3 and 5-2 and thus can be ignored. The number of combinations is given by the formula as below:-

The number of combinations, $q = {}_nC_r$

where n is the number of objects to be arranged in the combination of r objects.

For the example given, there are 5 objects arranged in the combination of 2 objects. Thus the number of combinations, $q = {}_5C_2 = 10$. This can avoid the redundancy in the distance between two objects checking. If the distance between the centres of the object is less than 65 pixels, the user is informed about the location of the combination which is too close.

On the other hand, there is no limitation on the number of the rotation allowed for the virtual working object manipulated through mouse operation. However, in the type-written natural language the virtual working object is allowed

to be rotated from 0° to 359° . Since the virtual working object is a square from the top view, the rotation at the value 90° , 180° and 270° make no difference from the original top view. This is shown in the Figure 5.7 where the original orientation of the cube is 0° . Thus the effective working angle is limited from 0° to 89° . The rotation angle of the virtual working objects are converted by the task pre-processor to the corresponding value.

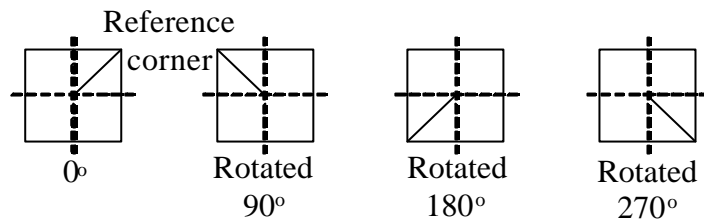


Figure 5.7: Object rotated 90° , 180° and 270°

5.6 Client-server Connection Manager

There are two client-server connection managers. The client-server connection manager at the telerobotic server program is responsible for listening to the request from the client-server connection manager at the telerobotic client program. The connection process is simplified and shown in the Figure 5.8. The client-server connection manager at the telerobotic server program is designed so that one user can login and control the system at a single time.

When the server is online and there is no other user connected to the server, the request from the telerobotic client program is accepted. The user is allocated for 10 minute to operate the system. This is to avoid a single user from occupying the whole system for a long period of time. The telerobotic client program will receive an image of the top view of the working area in JPEG file format and the working objects information contained in the URL string.

The working objects information is encoded in URL string by using the URL encoding scheme specified by MIME. A small modification has been made to simplify the encoding and parsing process. The semicolon sign (;) at the ending of the message is omitted. Every working object is represented by the values of the coordinate-x, coordinate-y and the orientation. The working objects are labeled from the integer number 1 and so on for the identification. If an object is labeled with number 1 and the coordinate-x, the coordinate-y and the orientation are 10, 20 and 30 respectively then in the URL string the values of the object are separated with a plus sign (+) as 10+20+30. The working object's identifier is separated from the values with an equal sign (=) as 1=10+20+30. If there is another working object labeled with number 2, the identifier-values pairs of both of the working objects are separated with an ampersand (&) as 1=10+20+30&2=100+200+60.

After the user complete the task assignment and command the system to execute the task, the information of the virtual working objects is encoded to the URL string by the client-server connection manager at the telerobotic client program before send out. The other URL strings that might be sent by the telerobotic server program to the telerobotic client program are listed in the Table 5.2.

When the time is out, the user is automatically logged out by the server. The client-server connection manager is also able to accept the manual logout from the user before the time is out. If there is another user trying to login to the server while the server is currently having a client connected to it, the user is able to receive only the top view of the working area in JPEG file format before automatically logout by the server. The client-server connection manager is also responsible for the client-server exception handling. The types of the client-server exception handling are discussed in the Section 7.10.

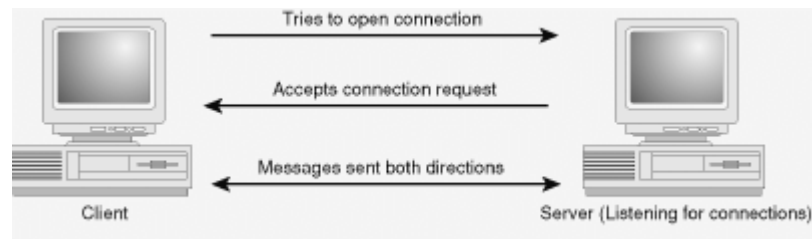


Figure 5.8: Client-server connection process

Table 5.2: URL strings

URL Strings	Explanation
1=10+20+30&2=100+200+60	Information of the working objects or the virtual working objects.
e=1	Error code 1: Working object exception.
e=2	Error code 2: Vision calibration failed because of model find.
e=3	Error code 3: Vision calibration failed because of out of range.
e=4&t=9	Error code 4: Login attempt while server is busy. The remaining time for the current user is about 9 to 10 minute.
e=5	Error code 5: Time out.
e=6	Error code 6: Login attempt while task in progress.
e=7	Error code 7: Log file cannot be opened.

5.8 FTP Server

The objective of the FTP server is to serve the FTP request from the telerobotic client program. The FTP server is set by using the Internet Information Services (IIS) program come with the Windows 2000 Server. The FTP directory is set at the c:\inetpub\ftproot\ . The working area top view image captured is

compressed by using the ActiveMIL command `save()`. The compressed file is using the JPEG format. When the user login to the client-server connection manager, either accepted or rejected, a copy of the image is send to the telerobotic client program.

5.8 Summary

This chapter is mainly focusing on the process of interpreting the task-oriented command from the user. The user can manipulate the virtual working object through the mouse operation and type-written natural language. The details on the virtual working area and working objects construction are covered. The mouse operation on the virtual working area and working objects are discussed in details. Next, the construction and processing of the task-oriented natural language are explained. The intelligent parser is then introduced. The intelligent parser can guide the user to operate the UTM telerobotic system through the mix usage of the mouse operation and the type-written natural language discussed. Finally, the client-server connection manager and FTP server are discussed. Both the client-server connection manager and FTP server are playing an important role in maintaining the connection between the client and server.

CHAPTER 6

HIGH LEVEL COMMAND TO LOW LEVEL COMMAND TRANSLATION

6.1 Introduction

As mentioned in the Section 4.3, the robotic sub-system (the robot with its controller) and the vision sub-system (combination of the camera and the image controller) is equivalent to a closed-loop system. The details of the process in the task conversion to a series of robot command and the robot command execution are discussed in this chapter. The architecture of the UTM telerobotic system is given in the Section 4.3.

6.2 Task Planner

The task planner consists of a task controller and the knowledge defined to support the operation of the task controller. In the knowledge of the task controller, it knows that the working objects can be manipulated within the area defined from $x_{10}y_{10}$ to $x_{240}y_{240}$. As compared with command pre-processor, the task controller knows only the basic operations that are move and rotate commands. The working object can be either moved within the area from $x_{10}y_{10}$ to $x_{240}y_{240}$ or rotated for 0° to 89° . The smallest unit for the working object movement is 1 pixel while the smallest unit for the working object rotation is 1° . The information of the working objects is kept in a 2 dimensional matrix defined as `Coordinate[3][100]`. The

matrix is capable to store information up to 100 working objects. The information stored for each object is the coordinate-x of the centre, the coordinate-y of the centre and the orientation of the working object. There are another two 2 dimensional matrices defined as `CoordinateClient[3][100]` and `PathPlan[3][200]`. The matrix `CoordinateClient[3][100]` is used to keep the information of the virtual working object that is being send by the *telerobotic client program*. While the matrix `PathPlan[3][200]` is used to keep the details of the task planned for the robot path planner.

The task of the telerobotic system is to manipulate the cube blocks placed in front of the telerobot. So the main objective of the task controller is to identify the sub-task from the task send by the *telerobotic client program*. In the task send, it might contain more than one virtual working objects being manipulated by the user. The task controller has to identify each of the objects being manipulated as well as the information of how the virtual working object is being manipulated.

For example, let's assumed there are five working objects in the working area at the remote site. The details of the working objects are given in the Table 6.1. The information is stored in the 2 dimensional matrix `Coordinate[3][100]` and 5 out of 100 records available are used. Let's assumed that the virtual working objects labeled as 3, 4 and 5 are being manipulated by the user. The details of the virtual working objects after manipulation are given in the Table 6.2. The information is stored in the 2 dimensional matrix `CoordinateClient[3][100]` and 5 out of 100 records available are used. The task controller will then make a comparison between the information kept in both of the 2 dimensional matrices. The differences between the information kept in both of the 2 dimensional matrices are shown in the Table 6.3. The pair of the records, for example the first and second records are referring to the same working object. The first record is indicating the original state of the working object while the second record is indicating the final state of the working object requested by the user. The information shown in the Table 6.3 is stored in the 2 dimensional matrix `PathPlan[3][200]` and 6 out of 200 records available are used. The 2 dimensional matrix `PathPlan[3][200]` is used by the robot path planner.

Besides identifying the sub-task from the task send by the telerobotic client program, there are some other objectives defined for the task planner. The task planner is in charged for system initialization when the “system start” button is clicked by the system administrator. The task planner will first initialize the robot, vision self-calibration and then make the system online. Furthermore, the task planner must supervise the progress of the task. An appropriate action is taken if any system error is detected during the system initialization and task progress.

Table 6.1: Details of the working objects at the remote site

Object number	Coordinate-x of the centre	Coordinate-y of the centre	Orientation
1	104	138	10 °
2	10	187	20 °
3	217	189	30 °
4	10	10	40 °
5	230	10	50 °

Table 6.2: Details of the virtual working objects send by the telerobotic client program

Object number	Coordinate-x of the centre	Coordinate-y of the centre	Orientation
1	104	138	10 °
2	10	187	20 °
3	164	232	50 °
4	10	10	10 °
5	120	20	50 °

Table 6.3: Differences between the information kept in the Table 6.1 and Table 6.2

Sub-task number	Coordinate-x of the centre	Coordinate-y of the centre	Orientation
1	217	189	30°
2	164	232	50°
3	10	10	40°
4	10	10	10°
5	230	10	50°
6	120	20	50°

6.3 Robot Path Planner

The Rhino robot is supporting both the joint coordinate system and the xyz coordinate system (Rhino Robots, INC., 1989). In the joint coordinate system, the motors are given distances to move in units of encoder count. In the xyz coordinate system, the motors are given distances to move in units of millimeters or degrees. The xyz coordinate system is easier to be used since it requires the programmer to specify only the final coordinate and orientation of the end effector. The end effector can be controlled directly to the coordinate of the object to be gripped. However, in this report the joint coordinate system is chosen instead of xyz coordinate system. This is because in the xyz coordinate system the details of motors movement cannot be controlled directly and this has caused the objects being collided by the end effector. To avoid this from happening, the joint coordinate is used in UTM telerobotic system.

Once the coordinate is known, the next step is to define the behavior for the robot arm to grip and to place the working object. The Rhino robot is a revolute type configuration (RRR) robot arm. The details of the revolute type configuration were discussed in the Section 4.2.1. Since the process of the object gripping and placing is

better done with the opening of the gripper 90° pointing downward. It will take less space if compared with the other orientation of the gripper. Thus in the arm behavior definition, the motor C can be ignored since the gripper is already 90° pointing downward after initialization.

During the process of the working object gripping and placing, one motor is moved at a single time. This is decided after the testing to move more than one motor at a single time. The robot arm becomes shaky and not stable when more than one motor is moved at a single time. Besides by moving one motor at a single time, it will simplify the process of the robot path prediction. There are many possible combination of the different motors movement to grip the same object. The best combination is chosen after the testing. It is assumed that the robot has been initialized. The motor F is first moved to align the robot arm with the working object to be gripped. The motor F is first moved before the arm is extended so that the inertia can be reduced. Then the motor E is moved and followed by the motor D to extend the robot arm to the coordinate of the working object to be gripped. The motor E is moved before the motor D to avoid the physical limit being reached by the motor D. After that, the motor B is moved according to the orientation of the working object. The motor A is moved to grip the working object. Then the motor D is moved and followed by motor E to move the arm to the soft home defined in the Table 6.4. The working object is now gripped on the gripper waiting for the object to be placed on the working area.

During the process of object placing, the motor F is first moved to align the robot arm with the coordinate of the working object to be placed. Then the motor E is moved. After that, the motor B is moved according to the orientation of the working object as required by the user. Next the motor D is moved. Now the motor A is moved to release the working object at the desired coordinate. The motor B is moved again so that the opening of the gripper is aligned with the path of the robot hand to avoid the collision with the working object being placed. Then the motor D is moved and followed by motor E to move the arm to the soft home defined. The real pictures for the process of working object gripping and placing are shown and discussed in the Chapter 8.

In the end of the working object gripping and placing cycles, the robot arm is moved to the soft home defined. The unit encoder count for the motor D and E at the soft home are -200 and 500 respectively. At the soft home, the robot arm is totally out of the area viewable by the camera. The robot configuration at soft home is shown in the Figure 6.1. An image is then captured and processed. The working object exception is checked in case the working object is failed to be gripped. The details of the working object exception handling are discussed in the Section 7.9.

The next step is to find the angle value for the motors rotation so that the working object can be gripped. First of all, the coordinate systems for the robot and the working area have to be defined. The coordinate systems for the working area and the robot are shown in the Figure 6.2. The coordinate system for the working area is 110 mm higher than the coordinate system for the robot. The direction of the x-axis is defined opposite of the standard direction so that the coordinate system for the real working area is exactly similar to the coordinate system used in the virtual working area as discussed in the Section 5.2. The transformation matrices for both the coordinate systems are given below:-

Transformation matrix refer to robot-based coordinate system (xyz0),

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & -92.5 \\ 0 & 1 & 0 & 228 \\ 0 & 0 & 1 & 110 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Transformation matrix refer to working area coordinate system (xyz1),

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 92.5 \\ 0 & 1 & 0 & -228 \\ 0 & 0 & 1 & -110 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Let say, there is a point $a_{xyz1} = (0, 0, 0)^T$, which is referred to working area coordinate system. The calculation shown below shows the way to find the point a_{xyz0} referred to robot-based coordinate system.

$$\begin{aligned}
 a_{xyz0} &= T_0^1 a_{xyz1} \\
 a_{xyz0} &= \begin{bmatrix} 1 & 0 & 0 & -92.5 \\ 0 & 1 & 0 & 228 \\ 0 & 0 & 1 & 110 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
 a_{xyz0} &= \begin{bmatrix} -92.5 \\ 228 \\ 110 \\ 1 \end{bmatrix}
 \end{aligned}$$

Thus, $a_{xyz1} = (-92.5, 228, 110)^T$ when expressed with respect to the robot-based coordinate system.

During the process of the working object gripping and placing, the motor F, E, D, B and A are involved. The geometric approach is used to solve the inverse kinematics problem of the robot to find the angle value for each of the motor during the process of gripping and placing of a working object. The angle value for the motor A can be ignored since it controls only either to open or close the gripper. The Figure 6.3 and Figure 6.4 are showing the top and side views of the robot. The gripper of the robot is located at the coordinate (P_x, P_y, P_z) refer to robot-based coordinate system. The value for the angles \mathbf{a}_B , \mathbf{a}_D , \mathbf{a}_E , \mathbf{a}_F and the top view distance between the origin and the point (P_x, P_y, P_z) , d , can be found as follows:-

$$d = \sqrt{P_x^2 + P_y^2} \quad (3)$$

$$\mathbf{a}_E = \cos^{-1}[(d/2)/228.6] = \cos^{-1}(d/457.2) \quad (4)$$

$$\mathbf{a}_D = \mathbf{a}_E = \cos^{-1}(d/457.2) \text{ (Isosceles triangle)} \quad (5)$$

$$\mathbf{a}_F = \tan^{-1}(P_x / P_y) \quad (6)$$

$$\mathbf{a}_B = 90^\circ + \mathbf{a}_F = 90^\circ + \tan^{-1}(P_x / P_y) \quad (7)$$

Next the angle values found have to be converted to the unit of encoder count which is required in the robot command. Unfortunately, the conversion is not given in the manual or the Rhino official website. A lot of testing is carried out to find the

approximate unit of encoder count for the corresponding angle value at each of the motor. The Table 6.4 is giving the approximate value for the conversion from angle value to the unit of encoder count and the relationship between the distance in the working area with respect to the distance in the image captured. A reference unit encoder count for each of the motor with respect to the physical position has to be defined. The unit encoder count for each of the motor at the reference point is given in the Table 6.5. At the reference point, the robot shoulder and the elbow as well as the robot elbow and the hand are perpendicular to each other. The robot configuration is shown in the Figure 6.5.

When implemented in the programming, the direction of the motor rotation has to be considered. For example if the robot arm is turned to right-hand side, the motor F unit encoder count is in negative value. The initial value of the motor at the soft home has to be considered. Let's assumed there is a working object placed at the coordinate (x, y) with the orientation \mathbf{q} . The equations given below are showing the corresponding unit encoder count for the motors to reach the working object mentioned.

$$a_{xyz1} = (x, y, 0)^T$$

$$a_{xyz0} = T_0^1 a_{xyz1}, \text{ where } T_0^1 \text{ is from equation (1)}$$

$$a_{xyz0} = \begin{bmatrix} 1 & 0 & 0 & -92.5 \\ 0 & 1 & 0 & 228 \\ 0 & 0 & 1 & 110 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

$$a_{xyz0} = \begin{bmatrix} x - 92.5 \\ y + 228 \\ 110 \\ 1 \end{bmatrix}$$

From the equation (3), the distance (top view) between the origin and the point (x, y) ,

$$d = \sqrt{(x - 92.5)^2 + (y + 228)^2} \quad (8)$$

By combining the equations (4) to (7) and the data from the Table 6.4 and Table 6.5, the unit encoder count for the motors B, D, E and F can be found as follows:-

Motor B (unit encoder count),

$$B = [\mathbf{a}_B - \mathbf{q}] \times (1165/90^\circ) = \left[90^\circ + \tan^{-1} \left(\frac{x-92.5}{y+228} \right) - \mathbf{q} \right] \times (1165/90^\circ) \quad (9)$$

Motor D (unit encoder count),

$$D = -72 - \mathbf{a}_D \times (3200/90^\circ) = -72 - \cos^{-1}(d/457.2) \times (3200/90^\circ) \quad (10)$$

Motor E (unit encoder count),

$$\begin{aligned} E &= 1120 + [90^\circ - \mathbf{a}_E] \times (3200/90^\circ) \\ E &= 1120 + [90^\circ - \cos^{-1}(d/457.2)] \times (3200/90^\circ) \end{aligned} \quad (11)$$

Motor F (unit encoder count),

$$F = \mathbf{a}_F \times (1590/90^\circ) = \tan^{-1} \left(\frac{x-92.5}{y+228} \right) \times (1590/90^\circ) \quad (12)$$

With the equations given above, all the sub-tasks passed from the task planner, which is contained in the 2 dimensional matrix `PathPlan[3][200]` can now be transformed into the corresponding robot commands. The coordinate value given in the equations is based on the millimeter while the unit used in the virtual working area is in pixel. The first two records in the 2 dimensional matrix `PathPlan[3][200]` are used for the discussion. Based on the conversion data from the Table 6.4, the unit for the coordinate values of the first two records to move the object at the coordinate (217, 189) to coordinate (164, 232) is converted to the millimeter as shown below:-

$$217 \text{ pixel} = 217 \times \left(\frac{350}{476} \right) = 160 \text{ mm}$$

$$189 \text{ pixel} = 189 \times \left(\frac{350}{476} \right) = 139 \text{ mm}$$

$$164\text{ pixel} = 164 \times \left(\frac{350}{476} \right) = 121\text{ mm}$$

$$232\text{ pixel} = 232 \times \left(\frac{350}{476} \right) = 171\text{ mm}$$

\therefore The new coordinates are (160, 139) and (121, 171).

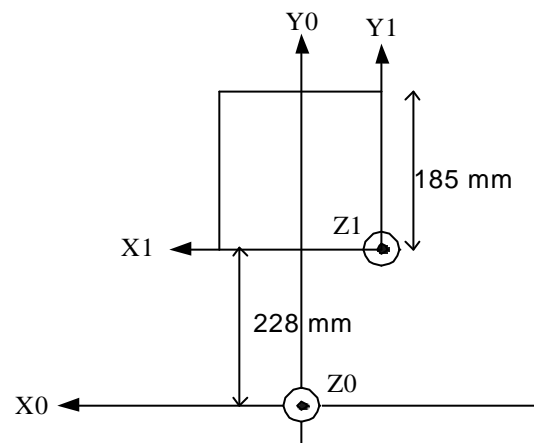
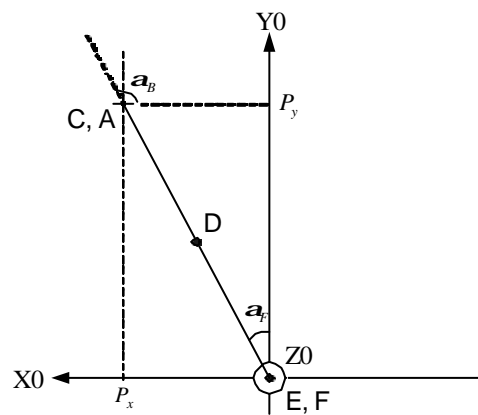
The first two records can now be transformed into the equivalent robot commands as shown in the Figure 6.6. The numbering is purposely labeled for every line for the ease of explanation. A comment is given for every command. The command given at the line number 1 is telling the robot controller the unit encoder count for the motor F is 184. The value 184 is calculated from the equation (12). The command at the line 2 is instructing the robot controller to move the motor F. The command at the line 9 is to turn on the auxiliary port 1 to drive the motor A. The auxiliary port 1 is used instead of the port A due to the sensor at the motor A is damaged and cannot be driven by the port A. The robot command is send through the serial port 1. The serial port 1 is set at the baud rate 9600 bps, 7 data bits, 2 stop bits and 1 odd parity bit. The robot command to move the motor is send after the prior motor command is completely carried out. The system is set at the maximum system velocity for faster task completion.



Figure 6.1: Robot soft home

Table 6.4: Motors' unit encoder count at soft home

Motor	Unit encoder count
A	Gripper opened (Robot command: $x_S, 1, -40$)
B	1165
C	0
D	-200
E	500
F	Depending on the previous state

**Figure 6.2: Working area and robot-based coordinate systems****Figure 6.3: Top view of the robot**

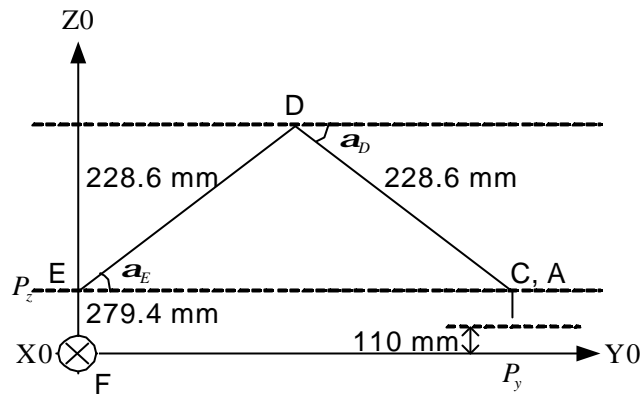


Figure 6.4: Side view of the robot

Table 6.4: Conversion table

Unit 1	Unit 2
90° (motor B)	1165 unit encoder count
90° (motor D, E)	3200 unit encoder count
90° (motor F)	1590 unit encoder count
350 mm	476 pixel

Table 6.5: Motors' unit encoder count at reference point

Motor	Unit encoder count
A	Gripper opened (Robot command: $x_S, 1, -40$)
B	1165
C	0
D	-72
E	1120
F	0

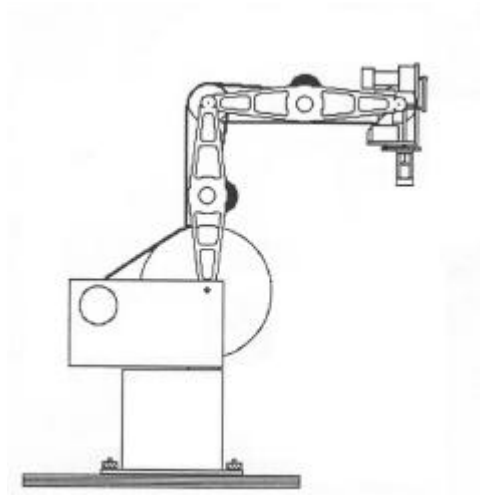


Figure 6.5: Robot physical configuration at reference point

```

28  ms, e          //Start motor E
27  pd, e, 500     //Set motor E destination position (Soft home)
26  ms, d          //Start motor D
25  pd, d, -200    //Set motor D destination position (Soft home)
24  ms, b          //Start motor B
23  pd, b, 1165    //Set motor B destination position (Soft home)
22  xs, l, -40     //Set auxiliary port level (open gripper)
21  ms, d          //Start motor D
20  pd, d, -103    //Set motor D destination position
19  ms, b          //Start motor B
18  pd, b, 571     //Set motor B destination position
17  ms, e          //Start motor E
16  pd, e, 3290    //Set motor E destination position
15  ms, f          //Start motor F
14  pd, f, 72      //Set motor F destination position
13  ms, e          //Start motor E
12  pd, e, 500     //Set motor E destination position (Soft home)
11  ms, d          //Start motor D
10  pd, d, -200    //Set motor D destination position (Soft home)
9   xs, l, 40      //Set auxiliary port level (close gripper)
8   ms, b          //Start motor B
7   pd, b, 912     //Set motor B destination position
6   ms, d          //Start motor D
5   pd, d, -1328   //Set motor D destination position
4   ms, e          //Start motor E
3   pd, e, 3064    //Set motor E destination position
2   ms, f          //Start motor F
1   pd, f, 184     //Set motor F destination position

```

Figure 6.6: Robot commands

6.4 Vision Sub-system

The camera is put at 935 mm exactly on the top of the working area. This is to make sure the whole of the working area can be captured by the camera. There is no extra lighting required since the white color of the working object and dark color of the working area are giving enough contrast in image processing. The vision sub-system is playing an important role in the translation of the high level command to the low level command. The vision sub-system is the “eye” of the telerobotic system that help the telerobotic system to “see” the objects specified by the user in the task. In other word, the objective of the vision sub-system is to abstract the information of the working objects from the image captured. The vision sub-system is also used in the system self-calibration.

6.4.1 Working Object Recognition

The steps involved in the working object recognition are as below:-

i) Image capturing

This is the process to capture an interested image from the continuing image captured by the camera. For example, after the soft home the working area top view image is captured for the image processing. The ActiveMIL function `Grab()` is used to capture the image. A sample of the image captured is shown in the Figure 6.7.

ii) Segmentation

Segmentation is the process that partitions an image into objects of interest. Since the size of the image captured is bigger than the actual size required, only the interested part of the image is segmented. The ActiveMIL function `CopyRegion()` is used to perform the segmentation. A sample of the image segmented is shown in the Figure 6.8.

iii) Pre-processing

Pre-processing deals with the techniques such as noise reduction and enhancement of details. First of all, the image segmented is smoothed by using the ActiveMIL function `Smooth()`. Smoothing is the process for reducing noise that may be present in an image as a result of sampling and transmission. The result is shown in the Figure 6.9. Then the image is binarized by using the ActiveMIL function `Binarize()`. A binarizing operation reduces an image to two grayscale values: 0 (black) and 255 (white). The result is shown in the Figure 6.10. After that, the image is pre-processed by using the ActiveMIL functions `Open()` and `Close()`. The opening operation is to remove small particles in the image while the closing operation is to remove holes from the blobs. The result is shown in the Figure 6.11 and Figure 6.12.

iv) ModelFinder

The ActiveMIL `ModelFinder` control is used to find the working objects in the image captured. A model of the working object is first defined in the ActiveMIL `ModelFinder` as shown in the Figure 6.13. Then the working objects are searched from the image pre-processed by using the ActiveMIL function `Find()`. The result of the processing is kept in the 2 dimensional matrix `Coordinate[3][100]` mentioned in the Section 6.2.

v) Object identifying

The boxes are drawn to identify the working objects found from the previous process. The ActiveMIL function `Draw()` is used to draw the boxes. The result is shown in the Figure 6.14.

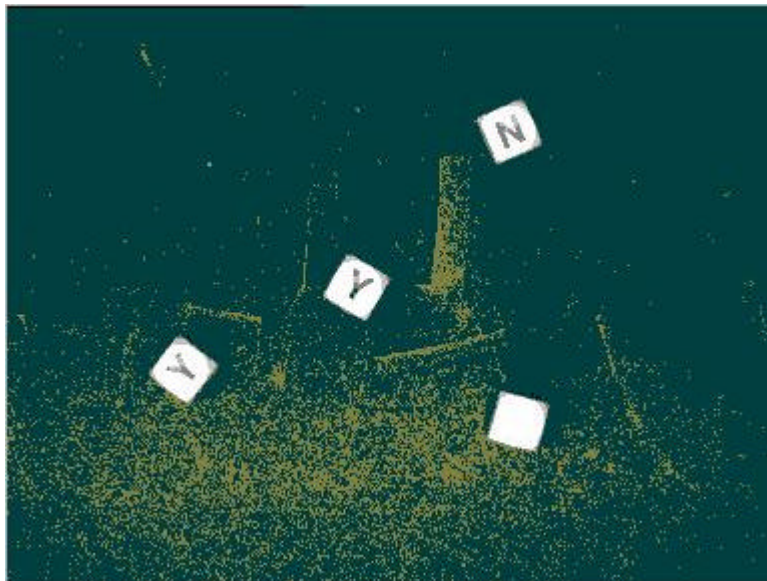


Figure 6.7: Image captured

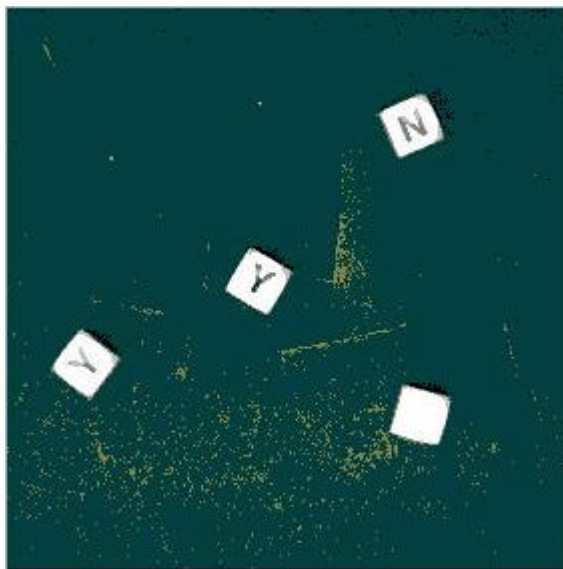


Figure 6.8: Image segmented

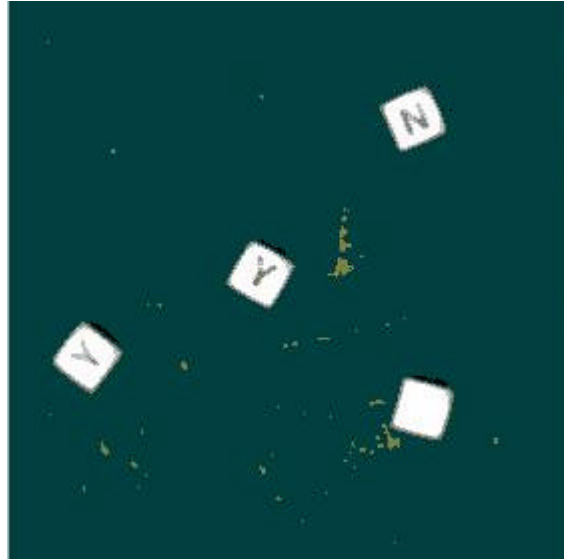


Figure 6.9: Image smoothed

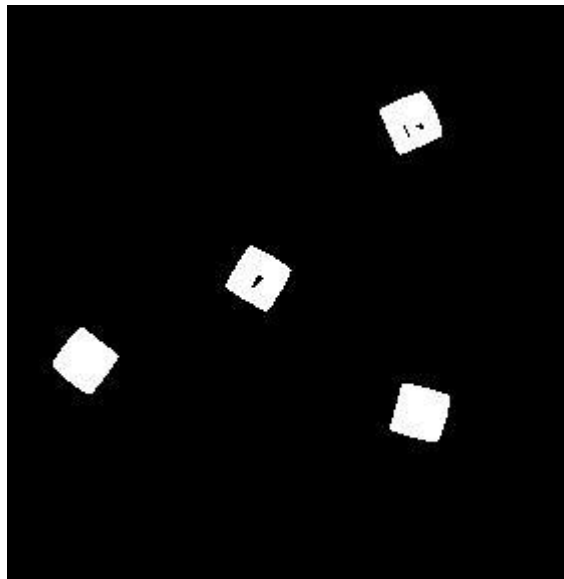


Figure 6.10: Image binarized

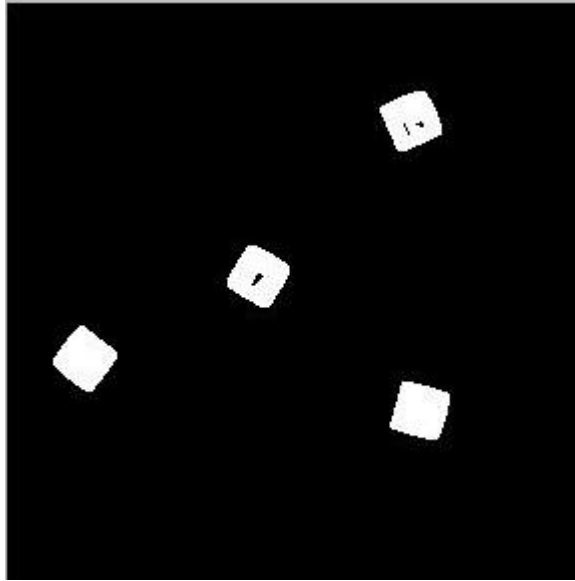


Figure 6.11: Image opened

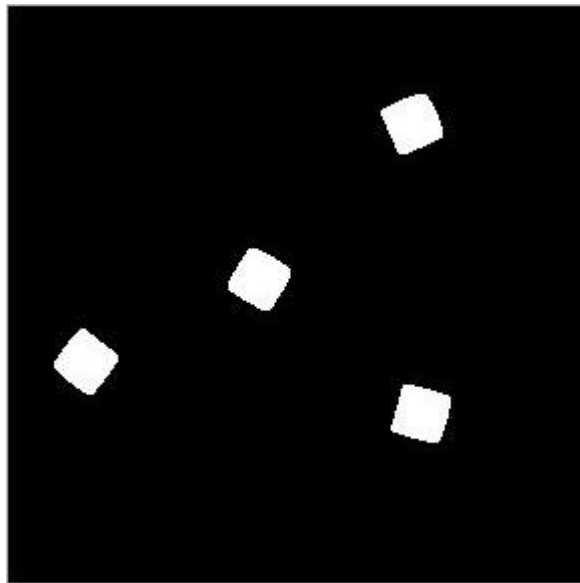


Figure 6.12: Image closed

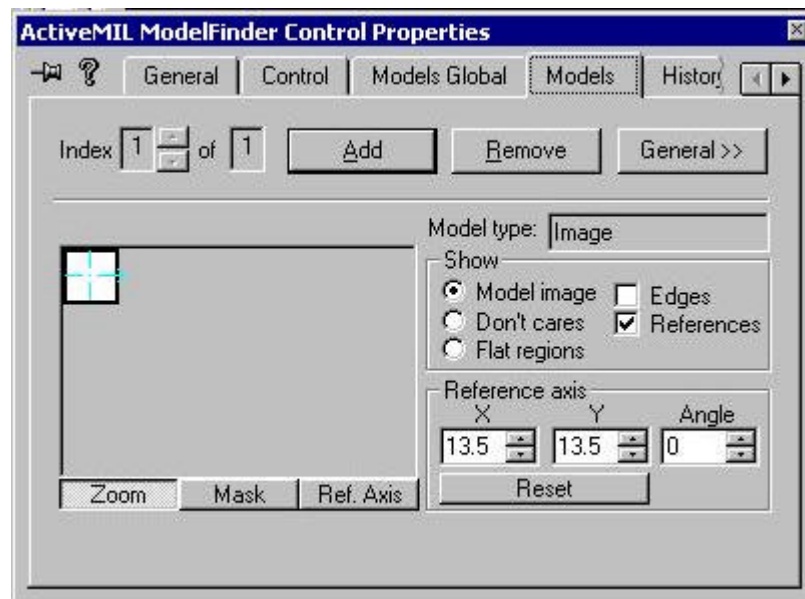


Figure 6.13: Model defined in the ModelFinderer Control

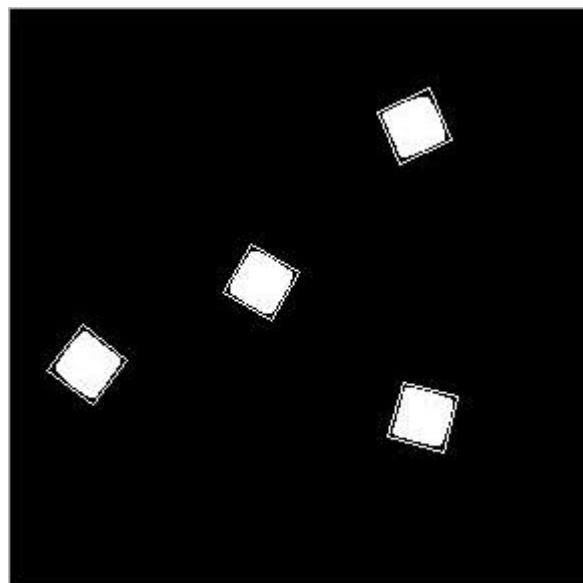


Figure 6.14: Boxes are drawn at the working objects recognized

6.4.2 System Self-calibration

The details of the system self-calibration are discussed in the Section 7.8. The discussion here is limited to the image processing to identify the reference point in the image captured. The image processing involved is exactly as the same as discussed in the Section 6.4.1 except the steps (ii) and (iii) are skipped. The pre-processing is not required to enhance the image captured since the model defined in the `ModelFinder` control is big enough to overcome the noise in the image captured.

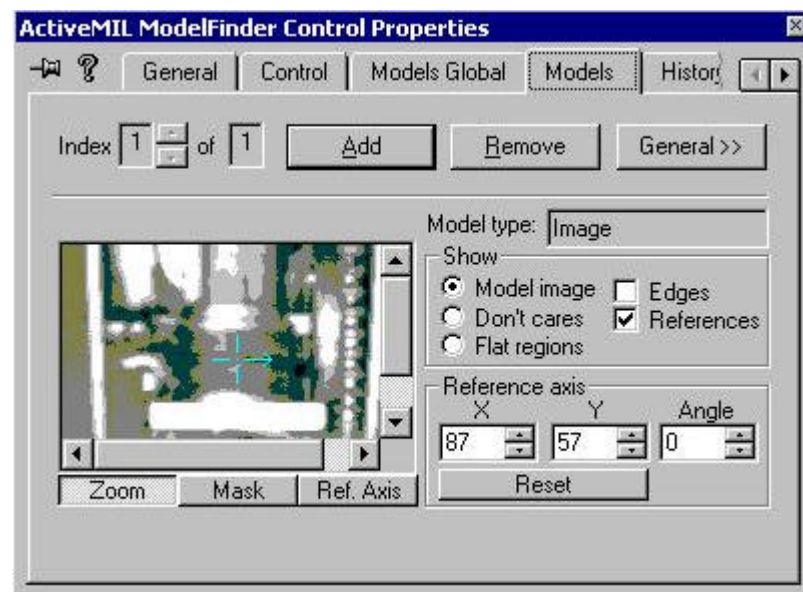


Figure 6.15: Model defined in the `ModelFinder` Control

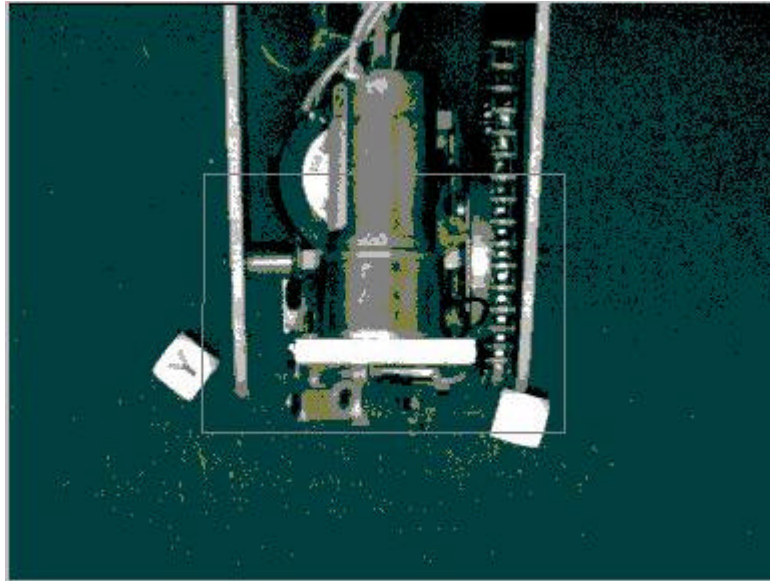


Figure 6.16: A box is drawn at the part of the gripper recognized

6.5 Summary

This Chapter explains the process of translating the task required by the user to the robot command. The task planner will process the task required by the user to the sub-task level. Meanwhile the robot path planner will then convert every sub-task to a series of the robot command. The conversion of the sub-task to a series of the robot command is based on the robot arm behavior defined in the Section 6.3. After that, the robot command is executed by the robot path planner one by one. On the other hand, the role of the vision sub-system in the translating the high level command to low level command are also discussed.

CHAPTER 7

SAFETY, RELIABILITY AND ACCURACY DESIGN OF THE TELEROBOTIC SYSTEM

7.1 Introduction

Safety, reliability and accuracy are some of the importance factors needed to be considered when designing the telerobotic system. Among three of the factors, the safety design is first considered since a bad safety design will make the system costly. The telerobotic system design must take consider the safety of the human, robot and others such as working objects, working area and the equipments. Among three of the factors, human safety must be given the first priority. Since the telerobotic system is remotely control thus the safety of the remote user can be ignored. However the safety of the people who might do the system maintenance, system setup and the visitors must be considered. A work cell with the fence has been setup to prevent the people from accidentally entering the work area of the robot. The dimension of the work cell design is discussed in Section 7.4.

Second priority is given to the safety of the robot. The system is built based on the task-oriented concept and thus the user has no direct control on the robot. The term “no direct control” is referring to the ability of the *UTM telerobotic client program* to issue a command that can instruct the *UTM telerobotic server program* to control the movement of the robot as desired by the user. On the other hand, the task assigned by the user is processed by the *UTM telerobotic server program* to perform the path planning. During the path planning, the safety of the robot and others such

as working objects and the equipments are taken into consideration. Indirectly the safety of the working objects, working area and the equipments are taken care. Furthermore the UTM telerobotic system is self-supervised, any abnormal events can be recovered without causing damage to the robot and others.

The reliability of the telerobotic system will determine the period between the first system start and the next system restart is required. A reliable telerobotic system seldom hangs or required system restarts during the 24 hour per day 365 days per year operation. The UTM telerobotic system is able to function under various conditions especially during abnormal events. The UTM telerobotic system is able to cope with the abnormal events by carrying out some special activities to compensate the abnormal events. For certain critical and complicated abnormal events, the UTM telerobotic system is not able to perform the error recovery activities. However the UTM telerobotic system will stop the system from the following system activities that might cause damage to the system. The client will be informed about the UTM telerobotic system error and a record of the abnormal event will be made by the *UTM telerobotic server program* for further system investigation.

The *UTM telerobotic server program* is running on the Windows 2000 server with the service pack installed. The operating system is chosen to host *the UTM telerobotic server program* for its stability and reliability. On the other hand, reliability criterion is also one of the factors considered when choosing the hardware and developing tools for the vision sub-system. Some of the other facilities designed in the UTM telerobotic system to increase the reliability of the system are discussed in the subtopics below.

As mentioned above, a reliable telerobotic system will have longer period between the first system start and the next system restart is required. So in the UTM telerobotic system developed, the system is halted only for the critical abnormal events which can not be recovered by the system automatically. This situation has to be kept to the minimum. The UTM telerobotic system is halted under the conditions:

- i) System calibration failed caused by the reference point is out of the range or the model of gripper can not be found in the image captured. This type

of system failure will affect the accuracy and functionality of the UTM telerobotic system and thus the system has to be halted. The failure happen only during the system initialization and the person in charged can detect the error spontaneously. The error can be fixed before the UTM telerobotic system is online. In another word, the error will not affect the continuity of the UTM telerobotic system since it can be detected and fixed before system is online.

- ii) Log file failed to be opened. This happen when the log file is write protected where the record can not be made. This type of error can be detected during UTM telerobotic system initialization. Under normal system operation the chance of the error happen is almost zero unless the system is infected by the virus, hacked by someone or the log file is intentionally set to 'read only'. The UTM telerobotic system is halted for safety purpose. In the case the log file is missing either before system is initialized or during the operation the log file will be recreated automatically.
- iii) The working objects are too close and make it impossible for the gripper to grip either of the working objects. The error can be detected during UTM telerobotic system initialization. The chance of the error happen during the operation is kept to the minimum. The minimum distance required is discussed in the Section 7.6. If the error is detected, the system is halted for the safety of robot, working objects and working area.

The next factor to be considered is the accuracy factor. Accuracy refers to the error between the measured and commanded position of the robot. Errors is introduced if the assumed kinematic structure differs from that of the actual manipulator. Such errors may be due to manufacturing tolerances in link length or link deformation due to load (Fu, K. S., *et. al*, 1987).

The term accuracy can be used to refer to the accuracy of the input, system and the accuracy of the output. The accuracy of the output will rely on the accuracy of the input and the system. An inaccurate input and processing will give the

inaccurate output as what is called Garbage In Garbage Out (GIGO). The accuracy of the input from the vision system is acceptable since the system is widely used and recommended in industrial application. In the telerobotic system developed, the accuracy of the output is affected mostly by the accuracy and the repeatability of the robot. The Rhino robot is an educational robot. The accuracy and repeatability problems of the robot have to be considered. Thus the size and shape of the working objects (discussed in Section 7.3) and the gripper with new fingers (discussed in Section 7.7) are designed to achieve the optimum accuracy of the UTM telerobotic system.

7.2 Robot Selection and Task Definition

In the real application, the robot with the appropriate end effector is chosen based on the task assigned for the system. However this is a research project, the existing resources have to be optimized. So the task of the system is defined based on the robot chosen. The robot chosen for the telerobotic system is the Rhino robot from Rhino Robotics LTD.. There are three robot of the same type available. Two of the robot can be used as standby robot. The component of the robot can be interchanged within a short time if the component was damaged. This will reduce the system downtime once the system is launched on the Internet. The robot comes with a gripper. Thus the most suited task for the robot is object picked and placed.

7.3 Working Object Definition

The task of the robot was defined. The next step is to define the dimension of the working object that is best suited the robot gripper and reliable. The original finger of the robot is made by a rectangle metal. The surface of the touch area of the finger is flat and both of the touch area of the fingers is parallel. Thus the possible shape of the working objects is either cuboid or cube. The cube is the best choice since the length of the four of the sides are equal.

The next question is the dimension of the working object. From the design of the gripper, the maximum dimension of the object that can be gripped safely is about 28 mm. The dimension of the gripper opening is shown in the Figure 7.1.

Theoretically the dimension of the cube must be equal or less than 28 mm. A bigger cube will give a better result than a small cube during object recognition which is discussed in the Section 6.4. However a smaller cube has less chance to be hit by the gripper during gripping attempt. The collision happened due to the accuracy and the repeatability problem of the Rhino robot. From the try and error testing, the optimum dimension for the cube is about 20 mm.

The consistency of the object dimension is another importance factor to be considered. During the object recognition as discussed in the Section 6.4, the object from the image captured is compared with the model defined. The consistency in the object size will improve the result of the object recognition. It is cost effective if the cube can be found directly from the market. Luckily the cube from the word game, Boggle fulfills all the conditions mentioned. The dimension of the cube is 18 mm x 18 mm x 18 mm and the size is quite consistent. The sample of the cube with the gripper is shown in the Figure 7.2.

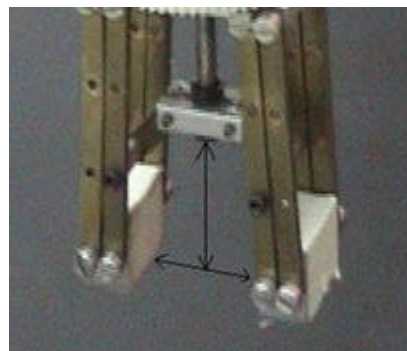


Figure 7.1: Dimension of gripper opening

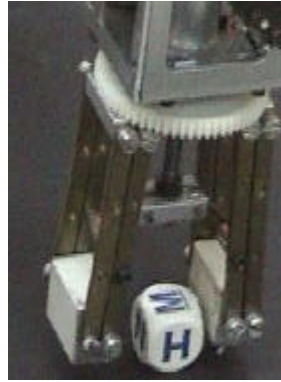


Figure 7.2: Gripper and cube

7.4 Work Cell Design

When designing the robot work cell, the safety of the human, robot, working objects, working area and equipment have to be considered. The work cell has to be able to cover all the work volume of the robot. The robot work envelope is shown in Figure 7.3. The maximum high of the robot is 895.4 mm as calculated from the dimension given in the Figure 4.4 (Chapter 4). The work cell is also designed with consideration of the ease of the system setup and maintenance. For example, the robot and camera fixture are integrated with the work cell. The work cell is easy for entering and does provide enough space for the person entered.

The details of the work cell dimension are given in the Appendix A while the Figure 7.4 is showing the real work cell. The centre of the robot work volume is fixed at the centre of the work cell. The work cell does provide the robot fixture. On the other hand, the camera is fixed on the top of the working area. The location of the camera is set to be out of the robot work volume for the safety consideration. The work cell is designed with the camera fixture. Meanwhile the working area of the robot is fixed in front of the robot. The dimension of the working area is discussed in the Section 7.5.

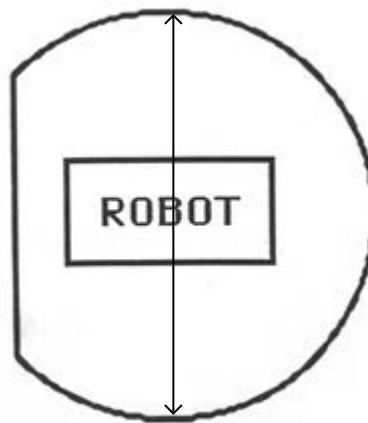


Figure 7.3: Robot work envelope



Figure 7.4: Robot work cell

7.5 Working Area Definition

As the working object definition, the shape of the working area has to be first determined. The image captured by the camera is in rectangle shape. Thus for ease of image processing, real and virtual area presentation, the shape of the working area is set to be square.

The working area is placed in front of the robot and parallel with the ground surface. As mentioned in Section 6.3, the gripper is always pointing downward and perpendicular with the ground surface. Certain gap has to be reserved between the gripper and the working area during the object gripping and placing process. From the experiment, the working area is optimum if the level of the working area is set at 110 mm from the ground. The explanation diagram is given in Figure 7.5.

The next step is to define the dimension of the working area. From the experiment, the minimum acceptable radius for the robot is about 180 mm while the maximum radius is about 460 mm. The experiment is based on the assumption that the working area is at 110 mm from the ground. Based on the experiment result, the dimension of the working area is defined as 250 pixels x 250 pixels, or 185 mm x 185 mm. The details of the calculation are given in the Figure 7.6 and the virtual working area is given in the Figure 7.7.

Although the working area is set to 250 pixels x 250 pixels, the real image presented in the *UTM telerobotic client program* is 280 pixels x 280 pixels for the convenience of the user. The client actual workable area in the *UTM telerobotic client program* is set to 230 pixels x 230 pixels. The client actual workable area in the *UTM telerobotic client program* refers to the area where the centre of the object is valid for moving. The size of the image used in image processing is 280 pixels x 280 pixels. The robot workable area in the remote site is greater than 280 pixels x 280 pixels for reliability consideration. The robot workable area refers to the achievable area for the telerobot gripper to grip and place the working object. The area where the vision can recognize must be larger than the client actual workable area where the user can work on while the robot workable area must bigger than the area where the vision can recognize. The same concept applied for the client viewable in the *UTM telerobotic client program* which must be larger than the client workable area. The areas are shown in the Figure 7.8.

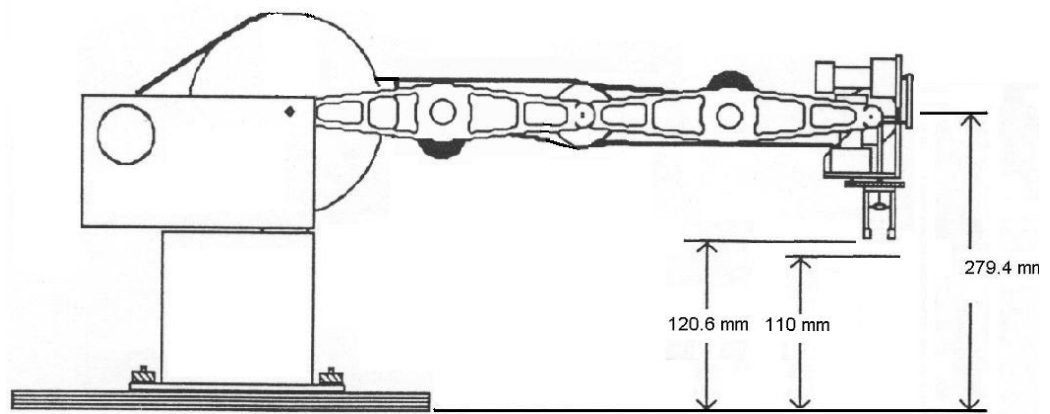


Figure 7.5: Optimum height of the working area

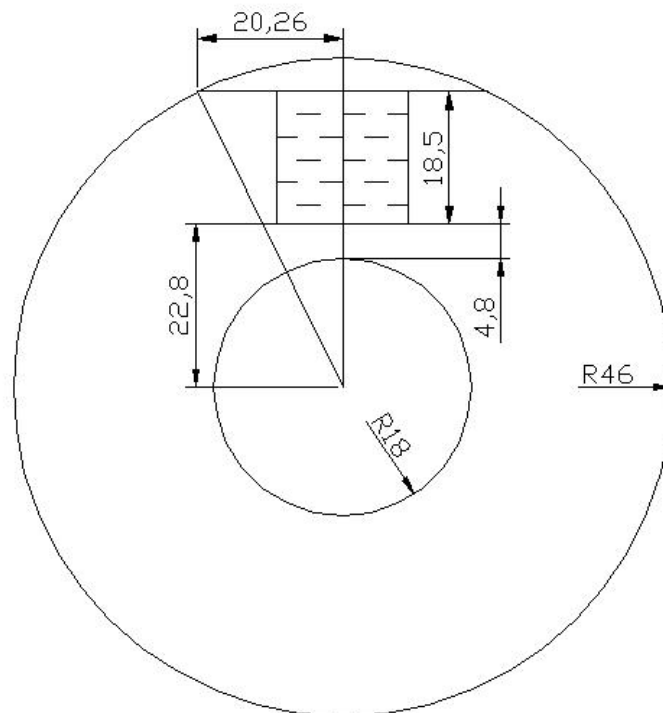


Figure 7.6: Working area dimension calculation

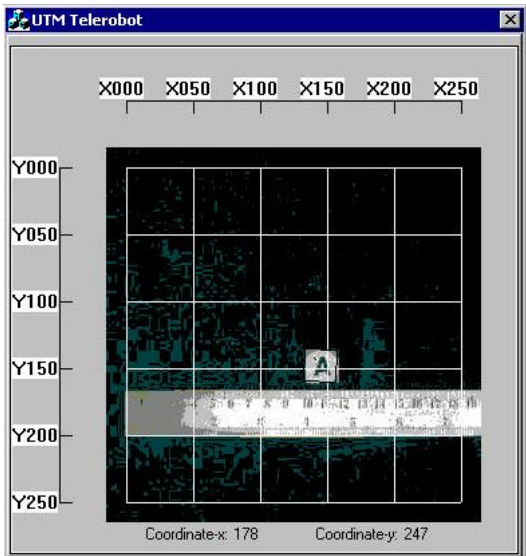


Figure 7.7: Virtual working area

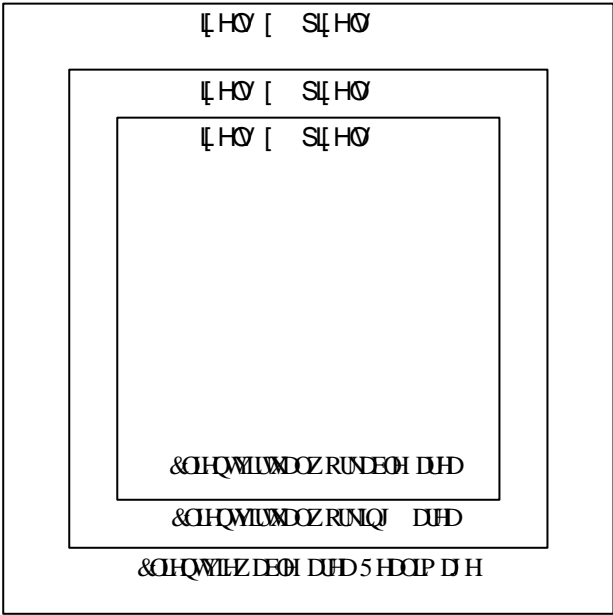


Figure 7.8: Workable, viewable areas comparison

7.6 Distance Between Objects Definition

Due to the robot architecture and the gripper design, certain area from the object to be gripped must be cleared from obstacle. As discussed in Section 6.3, the gripper is opened and then the opening is aligned with the path before move inward and downward to the location of the object to be gripped. The gripper is rotated with respect to the orientation of the object, gripping the working object then moved outward and upward. The path for the object placing is slightly different but the size of the area must be cleared is the same as in the object gripping.

When the gripper is opened as shown in Figure 7.9, the maximum outer length of the finger is 56 mm. The area of a circle with radius of 34.7 mm and centered at the centre of the object to be gripped must be cleared from other obstacle. This is to give enough space for the gripper to be rotated around the object. The details are shown in the Figure 7.10. Thus the minimum distance between the working objects required is 43.7 mm. The explanation is shown in the Figure 7.11. During the system operation, the actual distance allowed is set to be 48 mm or 65 pixels. The purpose of this is to compensate the accuracy and repeatability problems of the Rhino robot.



Figure 7.9: The maximum outer length of the robot finger

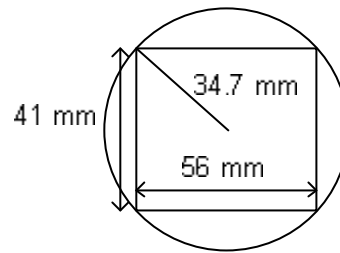


Figure 7.10: The area cleared for the gripper rotation

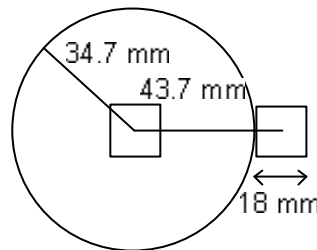


Figure 7.11: The minimum distance between the working objects required

7.7 Gripper with New Fingers Design

Rhino Robotics LTD. is supplying a wide range of the end effectors for the robot sold, such as triple fingers, narrow fingers, long fingers and vacuum fingers as shown in Figure 7.12. The best option for the application is vacuum fingers.

Although some modification has to be done on the surface of the working object, it would allow the gripper to do some complicated operation such as to place the working objects side by side. Due to some technical problem of getting a better end effector, the plan had to be given up. The focus is put on the existing gripper. The fingers of the gripper are redesigned as shown in Figure 7.13. The design is done based on the study of the application of the telerobotic system. The advantages of the new fingers are:-

- i) The inner distance between fingers is maximized. This reduces the chance that the fingers might hit the top of the object. The safety of the robot, working objects and the working area are improved.
- ii) Object gripped is centralized automatically by the new fingers design. Thus the accuracy of the output is improved.
- iii) The friction and sticky problem with the surface of the fingers that touch with the object are minimized. The object can be released spontaneously when the gripper is opened. This can increase the accuracy of the system output.



Figure 7.12: Types of gripper available for Rhino robot

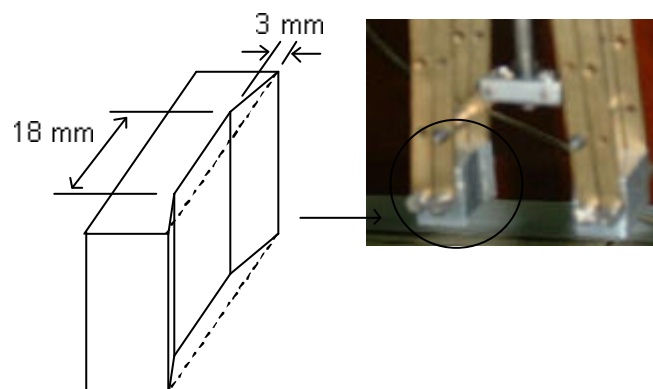


Figure 7.13: New fingers design

7.8 System Self-calibration

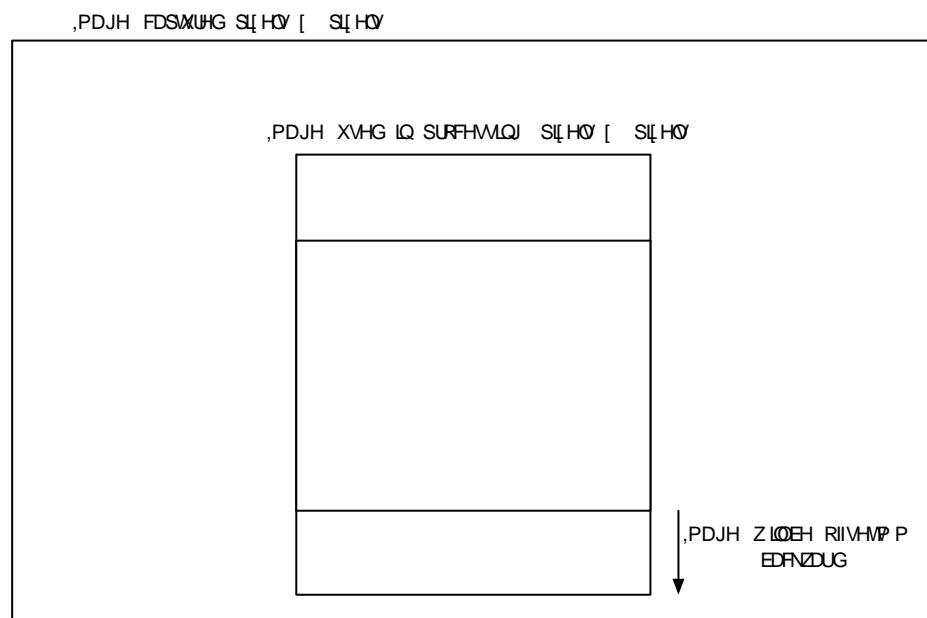
In the Section 7.4, the design of the work cell has been discussed. The dimension for the work cell is optimized for the current system setup. However when the work cell was build, there might be some tolerance occurred. This will affect the inverse kinematic equations derived in the Section 6.3. The term “self-calibration” referred here means the coordinate systems checking and calibration. The calibration is done in term of the software compensation. The hardware calibration, especially the Rhino robot can be done only by the manufacturer.

In the Section 6.4.2, the `ActiveMIL ModelFinder Control` is used to search the model defined during the system self-calibration. In order to define the model, first of all a working object is placed at the centre of the virtual working area. The gripper is moved to the top of the working object. The unit encoder count of the motors are recorded in the Table 7.1. An image is then captured and cropped until the part interested. A reference axis of the model is defined as shown in the Figure 6.15. The coordinate of the axis is overlapping the centre of the virtual working area.

When the system is initialized, the motors on the robot arm are moved according to the unit encoder count recorded in the Table 7.1. An image is captured. After the `ActiveMIL ModelFind` process, the coordinate of the gripper is identified. If the position of the robot is changed, it will affect the coordinate of the gripper found. Due to the design of the robot base holder at the work cell, the robot can be shifted backward only. If the robot is moved 10 mm backward so will be the gripper. The distance of the gripper offset is recorded. In order to compensate the error, the client workable area is offset with the same distance. Although the position of the camera and the physical working area are fixed, the position of the client workable area in the image captured can be offset in term of the image processing. This is shown in the Figure 7.14. Since the image size captured is 768 pixels x 576 pixels and the size of the image required for processing is 280 pixels x 280 pixels, the coordinate of the gripper found is valid provided it is still within the rectangle form by coordinates (140, 140) and (628, 436).

Table 7.1: Motors position for the gripper model defined

Motor	Unit Encoder Count
A	Gripper opened (xs, 1, -40)
B	1165
C	0
D	-1399
E	2710
F	0

**Figure 7.14: Image offset to calibration the system**

7.9 Working Object Exception Handling

During the process of object gripping and placing, the object exception might be happening. The object exception is the event where the object is not properly gripped or placed according to the procedure defined. For example, the object might be failed to be gripped because of the coordinate systems calibrated during the

system was initialized is no longer valid after the system is being used for a long period of time.

At the end of the system initialization, the number of the working objects is recorded. As mentioned in the Section 6.3, in the working object gripping and placing cycles, the robot arm is moved to the soft home defined to allow for the image capturing. The number of the working objects after gripping and placing cycles are recorded and compared with the number of the working objects after system initialization. For example, if the number of the working objects after placing is not equal to the number of the working objects after system initialization, the system is halted. The logic of the working object exception handling is given in the Table 7.2.

Table 7.2: Working object exception handling

Condition	Possible events and cause	Error recovery activities
Number of objects after placing ? Number of objects after initialization	? The object dropped ? The object placed out of working area	The system is halted.
Number of objects after gripping + 1 ? Number of objects after initialization	? Failed to grip the object	The system is halted.

7.10 Client-server Exception Handling

The data transfer between the telerobotic server and client programs are kept to minimum. The data transferred is limited to the important data such as the image of the top view of the working area and the working objects information encoded in the URL string. Thus, the connection between the client and the server is not checked

from time to time. Therefore the user may not be able to realize if some exception has happened to the connection. The actual connection between the server and client might lose but are not made realize by the other side, such as when the illegal operation is detected in the telerobotic client program and caused the operating system to terminate the program before the logout message is send.

When the user login to the telerobotic system, he or she is allocated for 10 minute to operate the system. Both of the telerobotic server and client programs will start the timer respectively. If the telerobotic program is terminated due to the illegal operation mentioned, the server will automatically logout the user when the time is out so that the other user can operate the system. On the other hand, if the telerobotic server program is terminated due to the illegal operation, the telerobotic client program is able to terminate the connection by itself.

The data send through the network is delayed. When the user is trying to login to the server, the request will take a certain time to reach the server and the delay will also happen to the message feedback from the server. Thus when the user login to the telerobotic system, 10 second is allocated for the telerobotic client program to wait for the reply from the server before the request is terminated.

As mentioned in the Section 5.6, the telerobotic server program is designed to handle a client at a single time. When there is a client connected to the telerobotic server, any other user might try to login to the telerobotic server. In order to avoid the current user being interrupted by the other user, the telerobotic server will reject the rest of the users. By the way, a working area top view image and the remaining time for the current user is send to the user who is trying to login.

From the experiment, the system will take about 30 to 40 second to move a working object. When the task is still in progress but the remaining time of the current user is out, the current user is automatically logout by the server. The server will not accept any user until the current task is completed.

Meanwhile, the telerobotic system is halted when there is working object exception or system calibration error detected. The user who is currently connected

to the UTM telerobotic system is automatically logout and informed about the working object exception detected. Any other user who is trying to login after the exception was detected will receive the same error message. For the system calibration error, the client-server connection manager will not make the system online after the system initialization.

7.11 Log File and Error Listing

Testing is a process of checking by means of actual execution whether a system behaves as expected. It is an effort of finding error in the system. A good testing is able to remove almost all the errors from the system. However it is impossible to make the system 100% error free. The purpose of the log file in the UTM telerobotic system is to record the activities of the *UTM telerobotic server program*. The information recorded in the log file must be sufficient enough for system trace back during system investigation. The log file is stored in the telerobotic server. The Figure 4.9 in the Chapter 4 might help to understand the explanation below. The types of the events recorded in the log file are as below:

- i) Application launched: When the *telerobotic server program* is launched.
- ii) Application terminated: When the *telerobotic server program* is terminated.
- iii) System start: When the telerobotic server system is being started. The system can be started by a single click on the “system start” button of the *telerobotic server program*. The system will first initialize the robot, system self-calibration and then make the system online. The system start button can also be used as the system restart function when the error happen.
- iv) System online: When the telerobotic server system is online and ready for the client remote control. The system is automatically online during the system start. After the system being started, the user can manually make

the system online by clicking on the online button (provided that the system is offline) of the *telerobotic server program*. When the system is online the label of the online button is changed to offline so that the system administrator can offline the system by clicking on the offline button.

- v) *System offline*: When the telerobotic server system is manually made offline by clicking on the offline button (provided that the system is online) of the *telerobotic server program*. When the system is offline the label of the offline button is changed to online so that the system administrator can online the system by clicking on the online button.
- vi) *Robot initialization*: During the start of the robot initialization. The robot is automatically initialized during the *system start*.
- vii) *System calibration*: During the start of the vision self calibration. The vision sub-system is automatically calibrated during the *system start*.
- viii) *Login*: When there is a client successfully login to the telerobotic system.
- ix) *Logout*: When the client manually logout from the telerobotic system before time out.
- x) *RefPoint> ...*: When the reference point is identified during the vision self calibration. For example, *RefPoint> x180y142* means the reference point is at the coordinate (180, 142) of the full vision area 756 pixels x 482 pixels.
- xi) *Env> ...*: When the environment information is being abstracted from the image captured. For example, *Env> 1=123+20+302&2=54+60+55&3=189+83+168* means there are 3 objects where the first object is at coordinate (123, 20) of the working area and the orientation is 302 degree.

- xii) TCPIn> ...: When the message is received from the client. For example,
TCPIn> 1=123+20+302&2=54+60+55&3=189+83+168.
- xiii) TCPOut> ...: When the message is send to the client. For example, TCPOut>
1=123+20+302&2=54+60+55&3=189+83+168.
- xiv) SerialOut> ...: When the message is send to the robot through the serial port.
For example, SerialOut> pd,f,-5.
- xv) Err> ...: When an error is being detected. There are various types of error that
can be detected and recorded in the log file, such as:-

```
Err> Code 1: Objects x57y130 and x18y131 are too close
Err> Code 1: Object number error after placing
Err> Code 1: Object number error after gripping
Err> Code 1: Login attempt while objects error
Err> Code 2: System calibration failed because of model find
Err> Code 2: Login attempt while system calibration failed
Err> Code 3: System calibration failed because of out of range
Err> Code 3: Login attempt while system calibration failed
Err> Code 4: Login attempt while server is busy
Err> Code 5: Time out
Err> Code 6: Login attempt while task in progress
Err> Code 8: Logout (System offline)
```

There is a special type of error that cannot be recorded in the log file. This type of error happened due to the log file error where the file exists but cannot be opened for event recoding. Normally this happened when the log file is write protected. Since the error cannot be recorded and it is still needed to be informed to the system administrator, the error listing on the *UTM telerobotic server program* is designed to solve this problem. All the errors mentioned above are also listed in the error listing for the convenience of the system administrator. The log file error is listed as below:-

```
Err> Code 7: Log file cannot be opened
```

The example of the events recorded in the log file and the error listing are shown in the Figure 7.15 and 7.16 respectively. Besides the types of event, the date and the time of the events happened are also being recorded. The log file is a text file and the name of the file is based on the date of the system for the convenience of the system administrator. The name of the log file is based on the format YYYY-MM-DD.txt, for example 2002-12-20.txt. The size of the log file will not be a problem since the size of the hard disk can be found in the market is quite huge. Besides log file and error listing facilities, the information recorded in the Event Viewer of the Windows 2000 Server can also be used as a source to trace the events of the telerobotic server program.

2002/12/20	14:16:34	Application launched
2002/12/20	14:16:48	System start
2002/12/20	14:16:48	Robot initialization
2002/12/20	14:17:8	SerialOut> xs,1,-40
2002/12/20	14:17:11	System calibration
2002/12/20	14:17:14	SerialOut> pd,e,2710
2002/12/20	14:17:14	SerialOut> ms,e
2002/12/20	14:17:17	SerialOut> pd,b,1165
2002/12/20	14:17:17	SerialOut> ms,b
2002/12/20	14:17:20	SerialOut> pd,d,-1399
2002/12/20	14:17:20	SerialOut> ms,d
2002/12/20	14:17:23	RefPoint> 181,142
2002/12/20	14:17:26	Robot initialization
2002/12/20	14:17:50	System online
2002/12/20	14:17:51	Env> 1=119+24+27&2=53+60+56&3=189+91+82
2002/12/20	14:17:59	Login
2002/12/20	14:18:0	Env> 1=119+24+27&2=53+60+56&3=189+91+81
2002/12/20	14:18:0	TCPOut> 1=119+24+27&2=53+60+56&3=189+91+82

Figure 7.15: Example of the events recorded in the log file

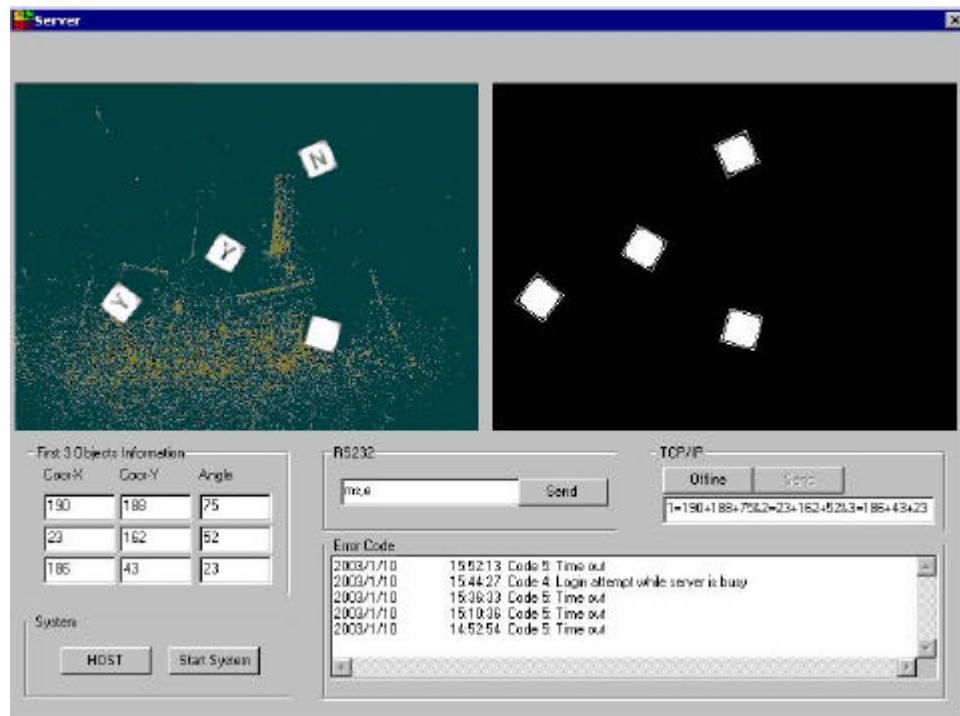


Figure 7.16: Events recorded in the error listing

7.12 Summary

In this chapter, some of the measures are taken to cope with the safety, reliability and accuracy issues of the UTM telerobotic system. The robot, task and working object selection as well as the work cell and working area design have been thoroughly carried out. The new fingers design and system self-calibration can increase the accuracy of the UTM telerobotic system. Meanwhile, the working object exception handling, client-server exception handling, log file and error listing will enhance the reliability of the UTM telerobotic system.

CHAPTER 8

SYSTEM TESTING, RESULT ANALYSIS AND SYSTEM ARCHITECTURE COMPARISON

8.1 Introduction

After the UTM telerobotic system being developed, the functionality of the UTM telerobotic system has been tested. In order to make the explanation easy to understand, some of the pictures are captured and attached. The testing and analysis discussed below represent a small number of the total testing. Only the importance testing and analysis are discussed. For example, the discussion about the tasks is limited to the cases:-

- i) one object moved;
- ii) one object moved and rotated; and,
- iii) two objects moved and rotated.

This is because the tasks mentioned above cover enough movements in order to test the ability of robot arm behavior discussed in the Section 6.3 to handle different tasks. The tests carried out below include the functionality test, accuracy analysis and platform testing. Finally, the UTM telerobotic system is compared with the other similar telerobotic system. A CD-ROM contains the video clips for the importance testing is attached with the report.

8.2 Command Pre-processor Testing

The command pre-processor is designed to support the use of the type-written natural language, the mouse operation and the integration of both of the input methods. The system has been tested to support all the commands with only the mouse operation. The test includes the logic testing. For example when the working object is required for the move command, the mouse is clicked on the area without any working object.

The system has also been tested to support all the commands with only the use of the natural language. The natural language has been tested in full sentence and partially. For example the move command has been tested as "move", "move x100y100", "move x100y100 to" and "move x100y100 to x200y200". The logic of the command such as to move a working object to a point that is out of the virtual working area has been tested.

The integration of both of the input methods has been tested. For example the move command, the user can first entered the command in natural language "move x100y100" and followed by the mouse operation to move the object to the coordinate desired. The user can also activate the move command by using the mouse operation and then followed by the use of the natural language "x100y100" and "x200y200". All the possible combinations of both the input methods have been tested.

8.3 Task Analysis: Single Object Moved

The progress of the task to move a single working object is observed. The Figure 8.1 is showing a virtual working object is being moved from the coordinate x100y140 to x210y40. As discussed in the Section 6.3, the robot path planning for the cycle to grip a working object is as below:-

- i) The motor F is first moved to align the robot arm with the working object to be gripped. This is shown in the Figure 8.2 (left).
- ii) Then the motor E is moved and followed by the motor D to extend the robot arm to the coordinate of the working object to be gripped. These are shown in the Figure 8.2 (right) and Figure 8.3 (left).
- iii) After that, the motor B is moved according to the orientation of the working object. This is shown in the Figure 8.3 (right).
- iv) The motor A is moved to grip the working object. This is shown in the Figure 8.4 (left).
- v) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.4 (right) and Figure 8.5 (left).

The working object is now gripped on the gripper waiting for the object to be placed on the working area. The robot path planning for the cycle to place a working object is as below:-

- i) During the process of object placing, the motor F is first moved to align the robot arm with the coordinate of the working object to be placed. This is shown in the Figure 8.5 (right).
- ii) Then the motor E is moved. This is shown in the Figure 8.6 (left).
- iii) After that, the motor B is moved according to the orientation of the working object as required by the user. This is shown in the Figure 8.6 (right).
- iv) Next the motor D is moved. This is shown in the Figure 8.7 (left).
- v) Now the motor A is moved to release the working object at the desired coordinate. This is shown in the Figure 8.7 (right).
- vi) The motor B is moved again so that the opening of the gripper is aligned with the path of the robot hand to avoid the collision with the working object being placed. This is shown in the Figure 8.8 (left).
- vii) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.8 (right) and Figure 8.9 (left).

The Figure 8.10 is showing the updated image of the top view of the working area after the task is done. This testing shows the ability of the UTM telerobotic system to move any working object in the task.

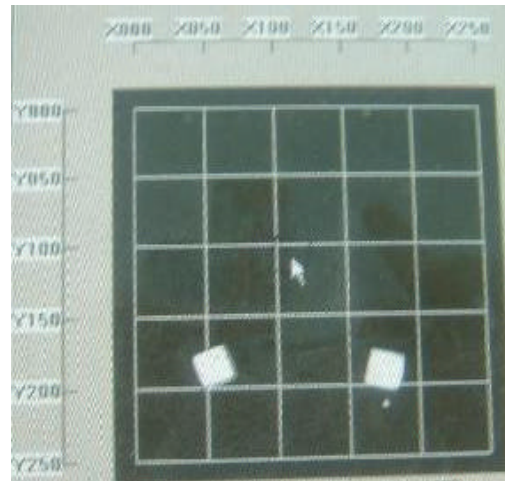


Figure 8.1: Task planned



Figure 8.2: Motor F is moved (left) followed motor E (right)



Figure 8.3: Motor D is moved (left) followed by motor B (right)



Figure 8.4: Motor A is moved (left) followed by motor D (right)



Figure 8.5: Motor E is moved (left) followed by motor F (right)



Figure 8.6: Motor E is moved (left) followed by motor B (right)



Figure 8.7: Motor D is moved (left) followed by motor A (right)



Figure 8.8: Motor B is moved (left) followed by motor D (right)



Figure 8.9: Motor E is moved

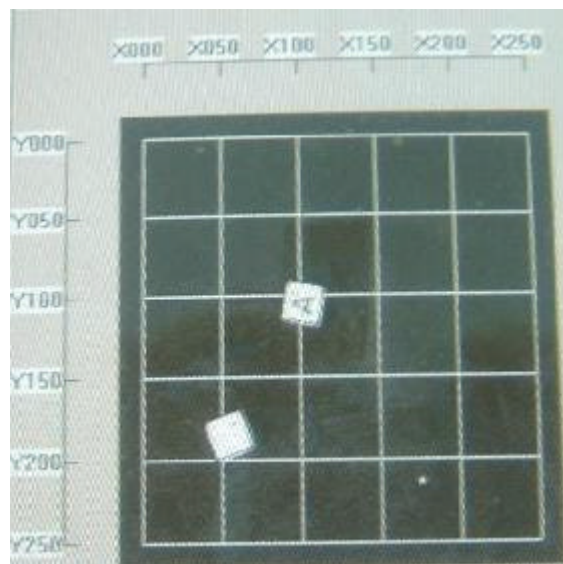


Figure 8.10: Task completed

8.4 Task Analysis: Single Object Moved and Rotated

The Figure 8.11 is showing a virtual working object is being rotated and moved from the coordinate x50y130 to x70y70. As discussed in the Section 6.3, the robot path planning for the cycle to grip a working object is as below:-

- i) The motor F is first moved to align the robot arm with the working object to be gripped. This is shown in the Figure 8.12 (left).
- ii) Then the motor E is moved and followed by the motor D to extend the robot arm to the coordinate of the working object to be gripped. These are shown in the Figure 8.12 (right) and Figure 8.13 (left).
- iii) After that, the motor B is moved according to the orientation of the working object. This is shown in the Figure 8.13 (right).
- iv) The motor A is moved to grip the working object. This is shown in the Figure 8.14 (left).
- v) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.14 (right) and Figure 8.15 (left).

The working object is now gripped on the gripper waiting for the object to be placed on the working area. The robot path planning for the cycle to place a working object is as below:-

- i) During the process of object placing, the motor F is first moved to align the robot arm with the coordinate of the working object to be placed. This is shown in the Figure 8.15 (right).
- ii) Then the motor E is moved. This is shown in the Figure 8.16.
- iii) After that, the motor B is moved according to the orientation of the working object as required by the user. This is shown in the Figure 8.17.
- iv) Next the motor D is moved. This is shown in the Figure 8.18 (left).
- v) Now the motor A is moved to release the working object at the desired coordinate. This is shown in the Figure 8.18 (right).
- vi) The motor B is moved again so that the opening of the gripper is aligned with the path of the robot hand to avoid the collision with the working object being placed. This is shown in the Figure 8.19 (left).
- vii) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.19 (right) and Figure 8.20.

The Figure 8.21 is showing the updated image of the top view of the working area after the task is done. The robot path planning is exactly the same as discussed in the Section 6.3. The working object is rotated to the orientation specified by the user at the step (iii) during the working object placing cycle. Thus the move and rotate operations for the same working object can be done in the same cycle to reduce the working time. This testing shows the ability of the UTM telerobotic system to move and rotate any working object in the same cycle in the task.

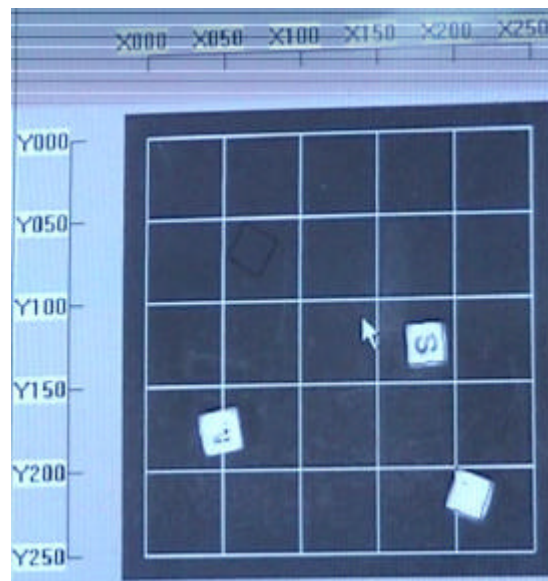


Figure 8.11: Task planned

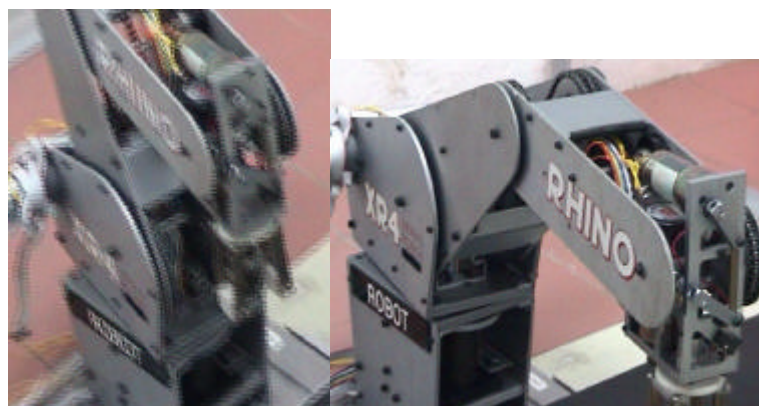


Figure 8.12: Motor F is moved (left) followed by motor E (right)



Figure 8.13: Motor D is moved (left) followed by motor B (right)



Figure 8.14: Motor A is moved (left) followed by motor D (right)

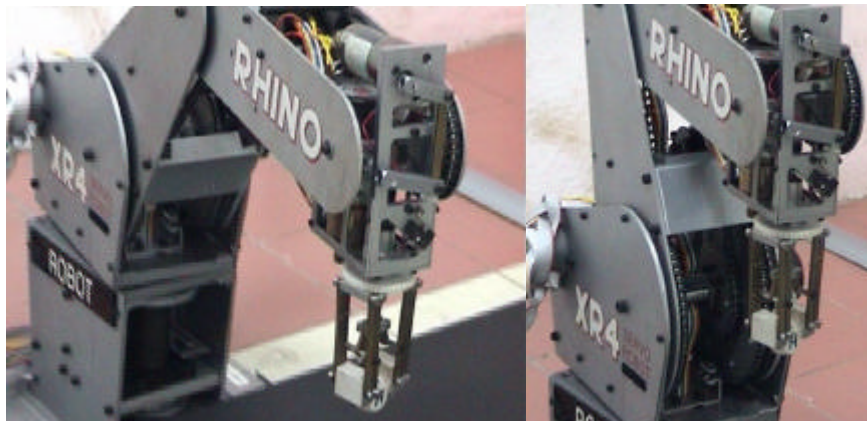


Figure 8.15: Motor E is moved (left) followed by motor F (right)



Figure 8.16: Motor E is moved

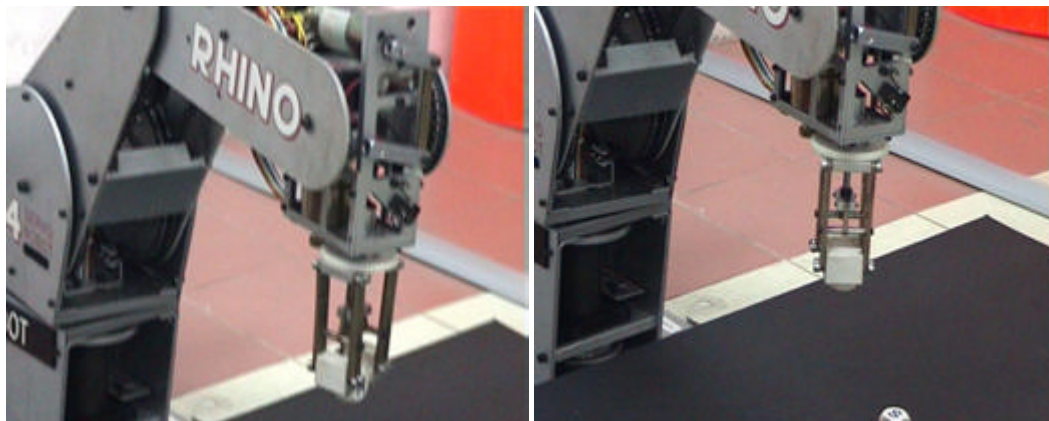


Figure 8.17: Motor B is moved

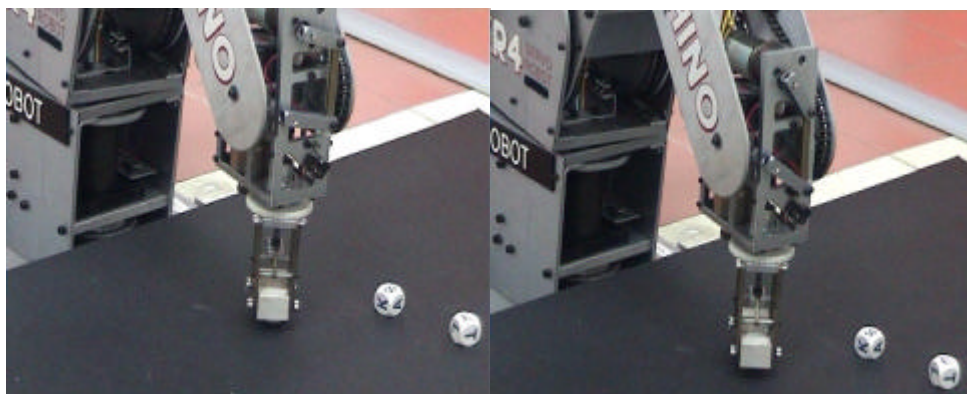


Figure 8.18: Motor D is moved (left) followed by motor A (right)

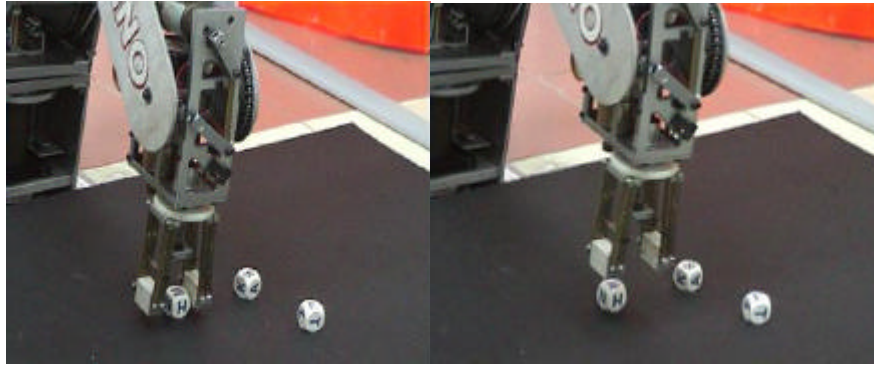


Figure 8.19: Motor B is moved (left) followed by motor D (right)

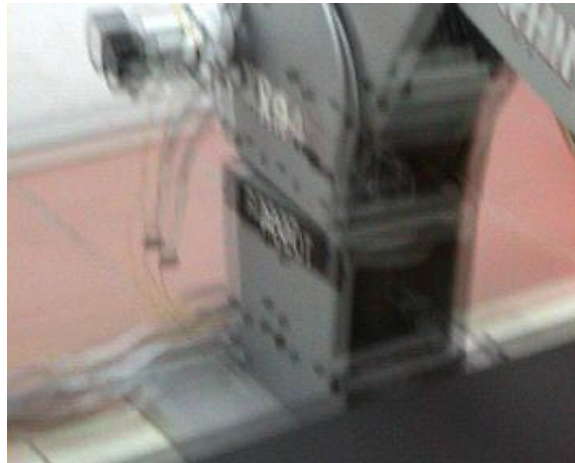


Figure 8.20: Motor E is moved

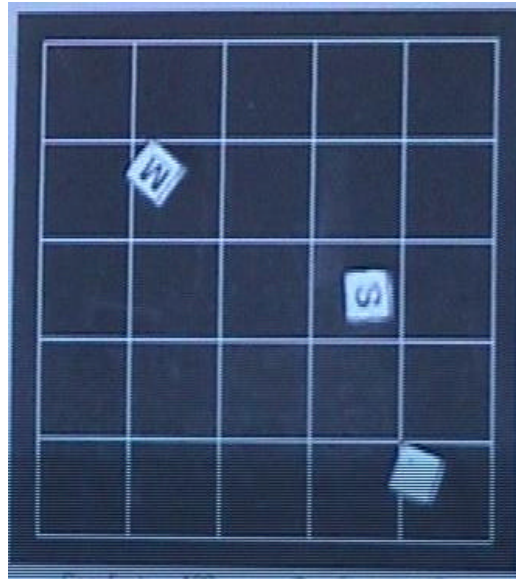


Figure 8.21: Task completed

8.5 Task Analysis: Two Objects Moved and Rotated

The Figure 8.22 is showing the virtual working objects are being moved from the coordinate $x210y40$ to $x180y130$ and from the coordinate $x20y80$ to $x40y210$. As discussed in the Section 6.3, the robot path planning for the cycle to grip a working object is as below:-

- i) The motor F is first moved to align the robot arm with the working object to be gripped. This is shown in the Figure 8.23 (left).
- ii) Then the motor E is moved and followed by the motor D to extend the robot arm to the coordinate of the working object to be gripped. These are shown in the Figure 8.23 (right) and Figure 8.24 (left).
- iii) After that, the motor B is moved according to the orientation of the working object. This is shown in the Figure 8.24 (right).
- iv) The motor A is moved to grip the working object. This is shown in the Figure 8.25 (left).

- v) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.25 (right) and Figure 8.26 (left).

The working object is now gripped on the gripper waiting for the object to be placed on the working area. The robot path planning for the cycle to place a working object is as below:-

- i) During the process of object placing, the motor F is first moved to align the robot arm with the coordinate of the working object to be placed. This is shown in the Figure 8.26 (right).
- ii) Then the motor E is moved. This is shown in the Figure 8.27 (left).
- iii) After that, the motor B is moved according to the orientation of the working object as required by the user. This is shown in the Figure 8.27 (right).
- iv) Next the motor D is moved. This is shown in the Figure 8.28 (left).
- v) Now the motor A is moved to release the working object at the desired coordinate. This is shown in the Figure 8.28 (right).
- vi) The motor B is moved again so that the opening of the gripper is aligned with the path of the robot hand to avoid the collision with the working object being placed. This is shown in the Figure 8.29 (left).
- vii) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.29 (right) and Figure 8.30 (left).

The cycle for the working object gripping and placing is repeated for the next working object. The robot path planning for the cycle to grip a working object is as below:-

- i) The motor F is first moved to align the robot arm with the working object to be gripped. This is shown in the Figure 8.30 (right).
- ii) Then the motor E is moved and followed by the motor D to extend the robot arm to the coordinate of the working object to be gripped. These are shown in the Figure 8.31 (left) and Figure 8.31 (right).

- iii) After that, the motor B is moved according to the orientation of the working object. This is shown in the Figure 8.32 (left).
- iv) The motor A is moved to grip the working object. This is shown in the Figure 8.32 (right).
- v) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.33 (left) and Figure 8.33 (right).

The working object is now gripped on the gripper waiting for the object to be placed on the working area. The robot path planning for the cycle to place a working object is as below:-

- i) During the process of object placing, the motor F is first moved to align the robot arm with the coordinate of the working object to be placed. This is shown in the Figure 8.34 (left).
- ii) Then the motor E is moved. This is shown in the Figure 8.34 (right).
- iii) After that, the motor B is moved according to the orientation of the working object as required by the user. This is shown in the Figure 8.35 (left).
- iv) Next the motor D is moved. This is shown in the Figure 8.35 (right).
- v) Now the motor A is moved to release the working object at the desired coordinate. This is shown in the Figure 8.36 (left).
- vi) The motor B is moved again so that the opening of the gripper is aligned with the path of the robot hand to avoid the collision with the working object being placed. This is shown in the Figure 8.36 (right).
- vii) Then the motor D is moved and followed by motor E to move the arm to the soft home defined. These are shown in the Figure 8.37 (left) and Figure 8.37 (right).

The Figure 8.38 is showing the updated image of the top view of the working area after the task is done. This testing shows the ability of the UTM telerobotic system to handle more than one working objects in the task. Regardless the number of the working objects being manipulated in the task, the same path planning discussed in the Section 6.3 is followed until all the sub-tasks are done.

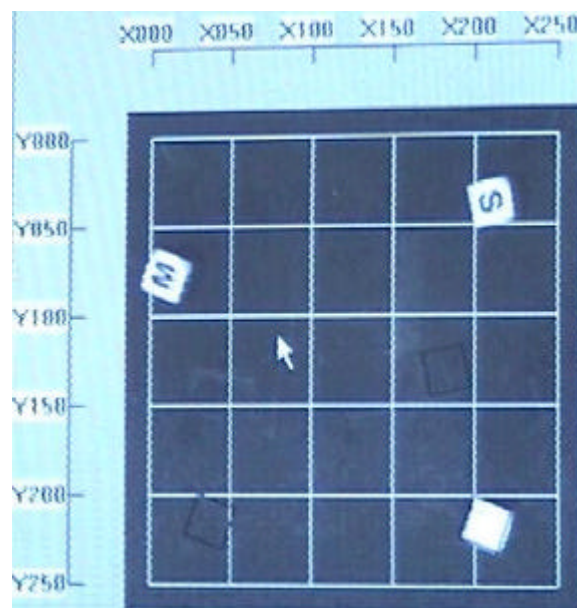


Figure 8.22: Task planned



Figure 8.23: Motor F is moved (left) followed by motor E (right)

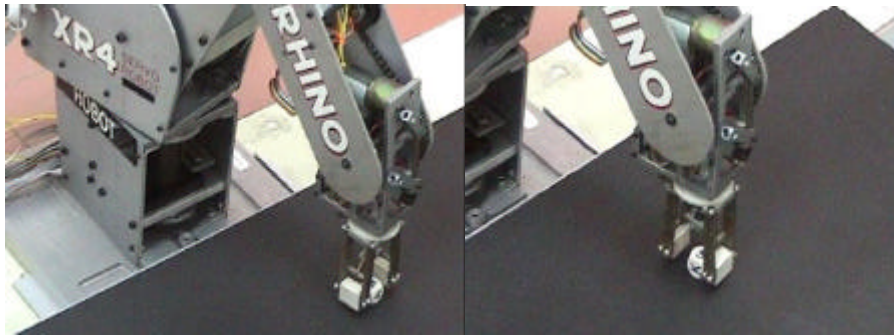


Figure 8.24: Motor D is moved (left) followed by motor B (right)

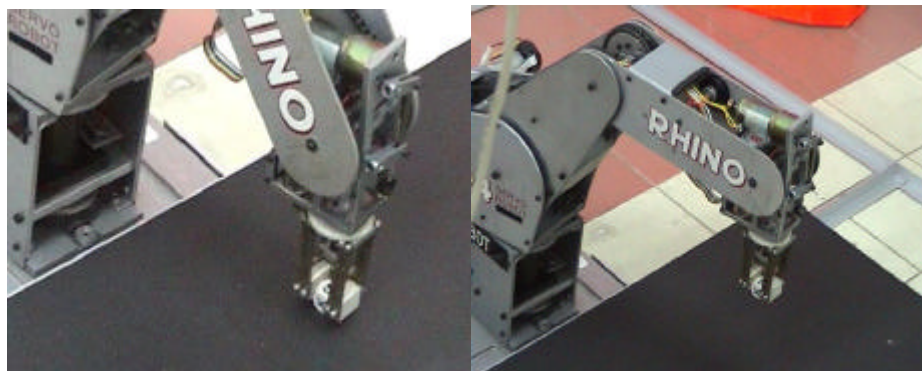


Figure 8.25: Motor A is moved (left) followed by motor D (right)

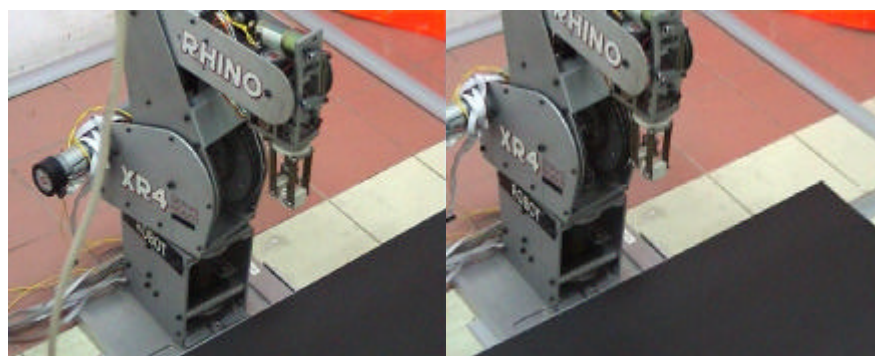


Figure 8.26: Motor E is moved (left) followed by motor F (right)

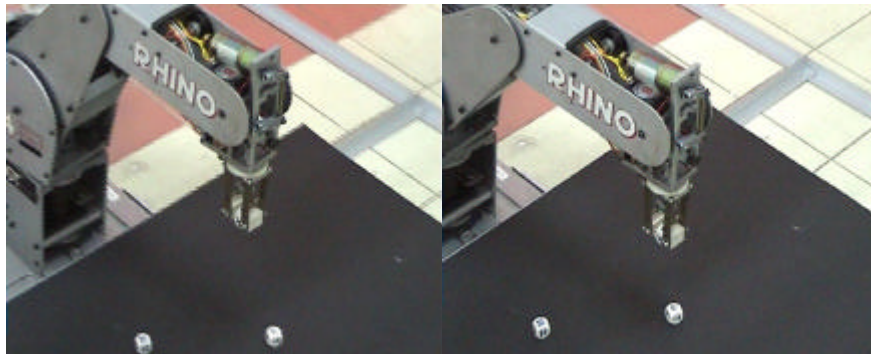


Figure 8.27: Motor E is moved (left) followed by motor B (right)



Figure 8.28: Motor D is moved (left) followed by motor A (right)



Figure 8.29: Motor B is moved (left) followed by motor D (right)

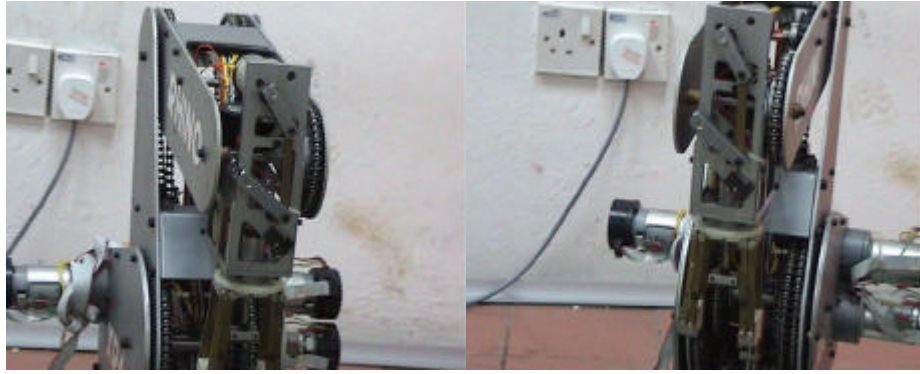


Figure 8.30: Motor E is moved (left) followed by motor F (right)

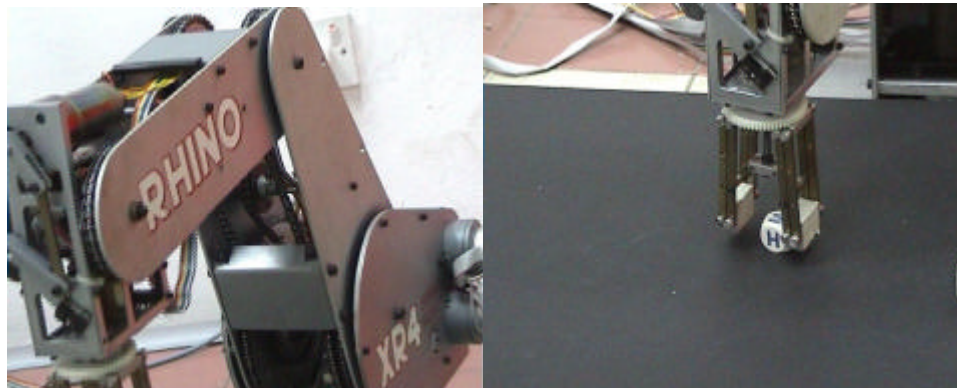


Figure 8.31: Motor E is moved (left) followed by motor D (right)

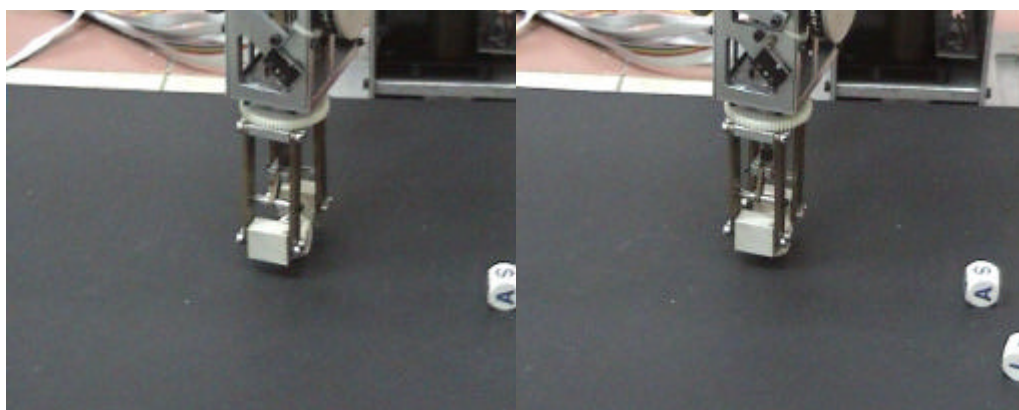


Figure 8.32: Motor B is moved (left) followed by motor A (right)

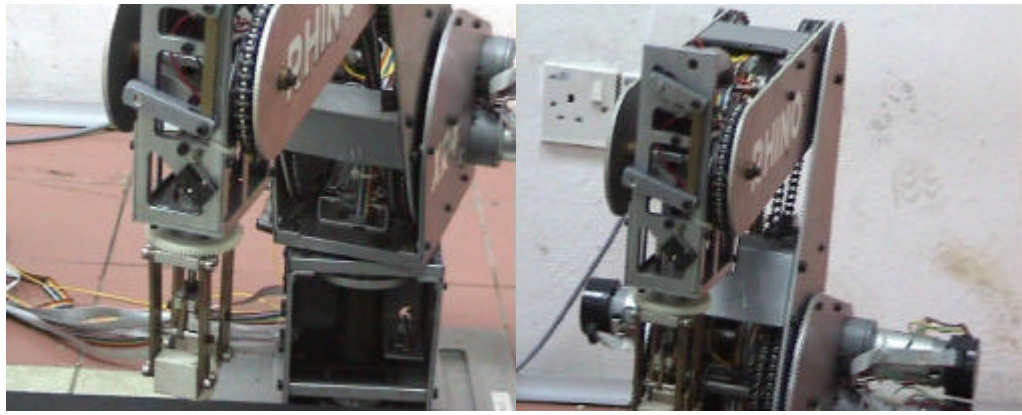


Figure 8.33: Motor D is moved (left) followed by motor E (right)

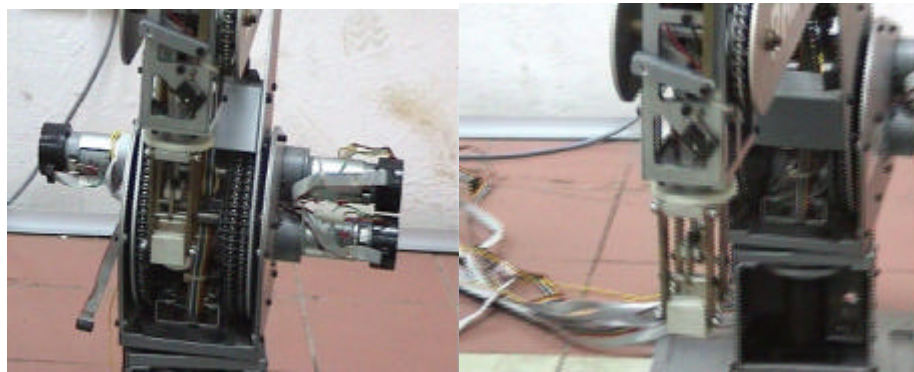


Figure 8.34: Motor F is moved (left) followed by motor E (right)

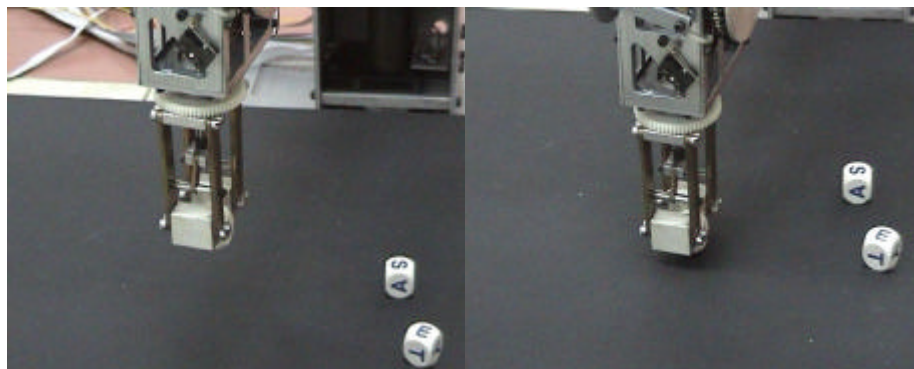


Figure 8.35: Motor B is moved (left) followed by motor D (right)

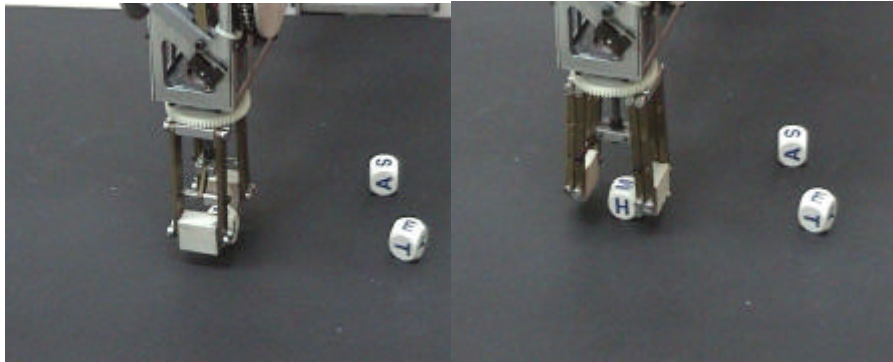


Figure 8.36: Motor A is moved (left) followed by motor B (right)

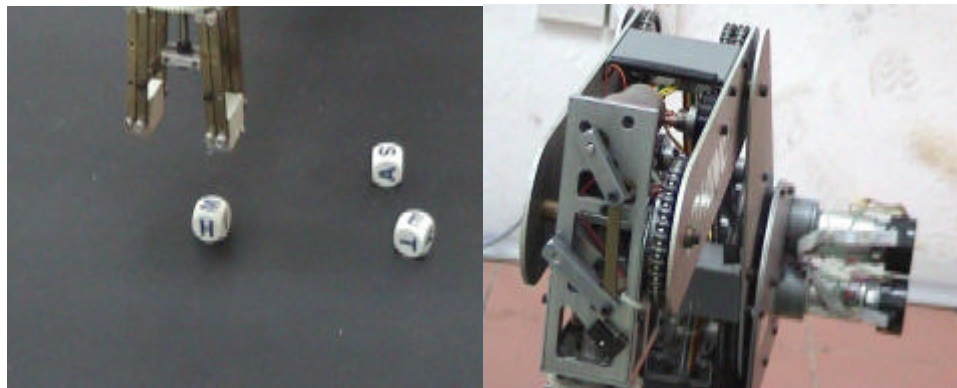


Figure 8.37: Motor D is moved (left) followed by motor E (right)

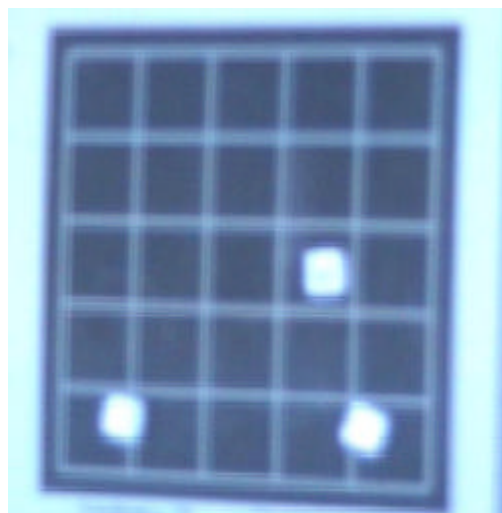


Figure 8.38: Task completed

8.6 Output Accuracy Analysis

The telerobotic system is designed to manipulate with the cube blocks. The accuracy of the output is analyzed. The accuracy of the output referred is the error between the measured and commanded position of the cube block. It is hard to measure the position and the orientation of the cube block. The reading of the position and the orientation of the cube block is based on the value obtained from the vision sub-system. The command “object information” is used to abstract the information of the working object. Five sets of the reading are taken for the system using the standard gripper and the gripper with new fingers design. The Table 8.1 and Table 8.2 are showing the readings taken. By comparing the reading from the tables, it shows the accuracy of the system with new fingers design has been improved.

Table 8.1: Output analysis for standard gripper

Commanded position			Actual position			Difference (Absolute value)		
Coordinate-x	Coordinate-y	Orientation	Coordinate-x	Coordinate-y	Orientation	Coordinate-x	Coordinate-y	Orientation
121	126	0	116	125	1	5	1	1
226	25	37	225	23	37	1	2	0
24	26	1	16	20	4	8	6	3
24	224	12	17	219	11	7	5	1
222	226	32	216	228	33	6	2	1
Average						5.4	3.2	1.2

Table 8.2: Output analysis for new fingers design

Commanded position			Actual position			Difference (Absolute value)		
Coordinate-x	Coordinate-y	Orientation	Coordinate-x	Coordinate-y	Orientation	Coordinate-x	Coordinate-y	Orientation
130	143	0	134	140	1	4	3	1
25	200	15	23	205	15	3	5	0
30	60	10	25	61	11	5	1	1
30	190	23	31	188	21	1	2	2
0	250	40	3	252	39	3	2	1
Average						3.2	2.6	1

8.7 Exception Handling

The working object exception handling has been tested. The number of the working object is purposely increased and reduced while the task is in progress to test the ability of the system to detect and react according to the object exception detected. Meanwhile the ability of the system to handle the client-server connection exception is also tested. For example the other user has tried to login the server while there is a user connected to the server.

8.8 Platform Testing

The telerobotic client and server programs are developed by using Microsoft Visual C++ 6.0. The advantage of using the Microsoft Visual C++ is the software developed is well supported by the Microsoft operating system. The telerobotic

server program is installed on the Windows 2000 Server. However the operating system for the telerobotic client program is depending on the operating system used by the user, thus the compatibility of the telerobotic client program with the most commonly used operating systems is tested. The telerobotic client program has been installed and tested on the Windows operating systems such as Windows 98, Windows 98 Second Edition, Windows Millennium Edition, Windows 2000 and Windows XP.

8.9 Local Area Network (LAN) Testing

The telerobotic system has been tested on the local area network. The system is controlled from the different computers at different building. Only the computers that are connected to the Gateway 10.5.0.1 can be used to control the telerobotic system. This is due to the local area network setting and configuration. However this will not stop the system being accessed and controlled from the public once the fixed IP address is obtained from the Internet Service Provider (ISP).

8.10 System Architecture Comparison

System architecture developed is compared with the systems developed by other institutes and the prototype of UTM web-based telerobotic system. There are three of the systems that are designed to manipulate with the blocks, namely Australia's telerobotic system, CSC telerobot system and robotoy system. The advantages of the UTM telerobotic system architecture developed as compared with the other systems with the same application are as below:-

- i) Task-oriented where the system developed requires only the user to focus on the task, i.e. the blocks manipulation instead of the robot movement. For three of the telerobotic system mentioned above, the user is required to control the system based on the robot joint value.

- ii) Intelligent parser is used to guide the user to operate the system. The concept of the intelligent parser is not used in any other of the telerobotic system.
- iii) The natural language command is supported in UTM telerobotic system.
- iv) Virtual grid instead of real grid is provided. This allows the system to perform system self-calibration. The Australia's telerobotic system and CSC telerobot system are using real grid on the working area in the remote site.
- v) The complexity of the system is hidden thus no robotic knowledge is required to operate the UTM telerobotic system. For example, the knowledge about the part of the robot such as elbow and shoulder is not required as compared with three of the projects mentioned.
- vi) The safety of the robot and the working objects are protected. For example, it is almost impossible to control the robot to purposely hit on the working object. For the other telerobotic system mentioned, there is the possibility for the robot gripper to hit on the working object.
- vii) The system is developed using Microsoft Visual C++ instead of Java. This is because at the time of the telerobotic system development, Microsoft is planning not to support Java language in the Microsoft Windows XP and the future operating system.

There are some limitations on the system developed such as:-

- i) The block manipulation is limited to 2 dimensional operations. The working object stacking is not allowed. This is due to the single camera used on the top of the working area in the UTM telerobotic system compared with the other telerobotic system.
- ii) The distance required between the working objects is too far as compared with the other projects mentioned. The distance is required in UTM telerobotic system to avoid the possibility that the gripper will hit on the working object during robot path planning.
- iii) The requirement to download the application program whereas the other projects are using web-based telerobotic system.

- iv) The telerobotic client software is not supported by the Linux and Mac operating system. The UTM telerobotic application program is designed specifically for the Windows operating system.

8.11 Summary

The UTM telerobotic system is first tested on its functionality such as command pre-processor testing and the variety task handling. The accuracy of the UTM telerobotic system is studied. The ability of the UTM telerobotic system on exception handling is also analyzed. After that, the *UTM telerobotic client program* is tested on different Windows operating system and on the local area network (LAN). Finally, the UTM telerobotic system architecture is compared with the other similar telerobotic systems.

CHAPTER 9

CONCLUSION

9.1 Introduction

During the development and implementation of the system, various disciplines of the knowledge are involved. This has made the project quite challenging. The major knowledge is from the disciplines as below:-

i) **Robotics**

In order to be able to control the robot movement efficiently, the inverse kinematics of the robot has been derived. The lack of the information about the encoder rotation unit and the degree of the robot physical rotation has made the situation more complicated. Many experiments, try and error had been carried out to find the relationship between the two of the units. Meanwhile, the construction of the virtual environment, working object and mouse events processing such as object moving and rotation involve a lot of calculation and mathematics equations derivation.

ii) Natural language

A study on the natural language processing has been carried out to develop a suitable method of natural language processing to be used. In the project developed, the natural language is integrated with the mouse operation and thus the natural language processing is more complicated.

iii) Vision system

Although the vision system is developed based on ActiveMIL from Matrox Imaging Library (MIL), it is impossible to rely only on the high level ActiveMIL command to complete the object recognition. The knowledge of low level image processing is required. The low level ActiveMIL commands are used to filter out the noise at background of the objects to increase the accuracy of the output. Furthermore the hardware was not properly installed by the supplier and a lot of effort had been carried out to solve the hardware problems.

iv) Information technology

The application programs in the UTM telerobotic system are developed based on C++ language by using Object-oriented Programming (OOP) method. The ActiveMIL from Matrox Imaging Library (MIL) can be supported only by either C++ or Visual Basic. Thus the skills in the programming are very importance in determining the successful of the project.

Furthermore, the protocol used in the data transfer between the telerobotic server program and client program is a result of the study from the encoding scheme call URL (specified by the MIME). The study of the FTP server of the Windows 2000 Server has enabled the telerobotic client program to download the image file from the server.

In the telerobotic programs produced, most of the coding is self-developed by referring to the Microsoft Foundation Classes (MFC) documentation and the relevant books. The functions of image processing are called from Matrox Imaging Library (MIL). Due to the fact that the Matrox Imaging Library (MIL) requires a run-time license, the JPEG file displaying function can not be incorporated in the telerobotic client program. Thus the JPEG file displaying function is called from a downloaded library for the ease of programming.

9.2 Objectives Achievement

As mentioned in the Section 1.3, the objectives of the project is to study the latest finding in the internet-based telerobotics, develop and implement a new system architecture design for use in the Internet-based telerobotic application. The system designed must take the consideration of the problems faced in the Internet-based telerobotic application, such as time delay, safety and reliability factors.

This project is successfully developed and implemented by considering the importance factors that will determine the success of the telerobotic system. The importance factors are learnt from the experience of the other projects developed. The system developed, as compared with the other systems developed, has achieved a lot of improvement from the perspective of:-

i) User friendly of GUI

The client program is providing the options for the user whether to use the mouse, type-written natural language or the combination of mouse and type-written natural language to issue a task-oriented command. The buttons at the telerobotic client program are kept to as minimum as possible. On the other hand, the telerobotic server system is designed so that the system can be initialized by single click on the “start system” button.

ii) User friendly of system architecture

The architecture of the system is designed so that the user needs only to specify the final output required and then the system will carry out the task. Every user is allowed to control the system for 10 minute only to avoid the other user to keep waiting. The remaining time of the current user will be informed so that the other user can estimate the time to login again.

iii) Reliability

The application of the system is designed for 24 hour per day 365 days per year operation, many of the designs are aiming for system reliability, such as the log file, error listing, object size definition and the operating system selection. However, it is almost impossible to achieve the target for 24 hour per day 365 days per year operation with the existing hardware. There are still some factors that affect the continuity of the system such as the use of PC as server and the type of the robot chosen. By the way, it is obvious that the system has been optimized to make it as reliable as possible.

iv) Safety

The system is designed based on the task-oriented concept. The robot can be safely protected from the damage caused by the client and the abnormal events. Furthermore the robot is isolated from the people in the work cell and thus the robot and the people are safely protected.

v) Time delay

The data transfer between the telerobotic client and server programs are kept to as minimum as possible. The data are limited to the image of the top view of the working area in JPEG file format and the URL strings as discussed in the Section 5.6. The content of the data transferred can be controlled easily when the system is build by using the application program.

In the web-based telerobotic project, some other data will be added to the original data automatically by the web server and this will increase the size of the data transferred. For example it need only 1 second to transferred 5 kb of original data at transfer rate of 5 kbps. If the size of the original data is increased by the web server from 5 kb to 10 kb then the time required will be increased to 2 second. The response time of the system will become longer. In UTM telerobotic system, the application programs are developed so that the volume of data transferred between the client and server can be controlled and kept to minimum.

vi) Accuracy

The accuracy of the output is enhanced by the gripper with new fingers design. The tolerance achieved by the gripper with new fingers design as compared with the use of standard gripper are compared and discussed in the Section 8.6. Even though, the application of the UTM telerobotic system is still limited to education and entertainment application.

The system designed and developed had been tested on the local area network (LAN). The communication protocol is based on TCP/IP as the protocol used on the Internet. Thus the system can be directly connected to the Internet once a fixed IP address and direct Internet connection are obtained from the Internet service provider (ISP).

9.3 Contribution

As a conclusion, this report has successfully developed and implemented a task-oriented telerobotic system for use in Internet-based telerobotic application. In some of the aspects as discussed in the Section 8.8, the UTM telerobotic system manages to surpass the telerobotic system developed by the other research groups. The system has been tested from various aspects. Although the application of the telerobotic system developed is limited to education and entertainment application,

the knowledge and the experience gained from the project is valuable and useful for the future project.

The UTM telerobotic system developed and implemented in this report involves a wide discipline of knowledge. Thus the contribution of the project can be viewed from various aspects as below:-

i) Command pre-processor

The command pre-processor is well designed to support mouse events, type-written natural language and the integration of the both types of command inputs. The command pre-processor can be easily expanded to support spoken natural language since both of the spoken and type-written natural language can use the similar command list and the difference is limited only with the form of input. The command pre-processor can also be easily expanded to support gesture recognition on the remote site. The object specification for move operation as well as the location to be placed can be specified by using the finger pointing.

ii) Task-oriented concept for telerobotic application

This project has successfully implemented a task-oriented robotic system for Internet-based telerobotic application. Various aspect of the design such as reliability, safety and accuracy consideration has been widely covered in the Chapter 8. The knowledge and experience can be used for the future project.

iii) Robot arm behavior

The robot, Rhino has been optimized for the object pick and place. The behavior of the robot arm is specifically optimized for the cube gripping with the cube dimension about 2 cm x 2 cm x 2 cm. In the Chapter 8, the robot arm behavior has been tested for complicated task such as moves and rotates an object in a single task. The robot arm behavior is also tested for

manipulating two objects in a single task. A new behavior for different application can be derived from the data given in the Section 6.3.

iv) Tested communication protocol for tele-application

The message transfer between telerobotic server and client programs is based on the URL encoding scheme specified by the MIME which is used on Internet form application. A small modification has been made to simplify the encoding and parsing process. The semicolon sign (;) at the ending of the message is omitted. The data transferred by using the protocol is easy to understand and to be parsed. The data is identified by the name and value in pair, for example the message used in the system $e ? 7$ means error with the type 7. The protocol enables the sending of many pairs of data by separating each pair of data by an ampersand (&). For example the message $1=100+100+0&2=100+200+45$ means there are two objects with the attributes given after equal sign (=). The protocol has been tested for telerobotic application and it can also be used in wireless applications.

v) New fingers design for the Gripper

The new fingers design in this report brings some improvement to the UTM telerobotic system. For example, the safety of the robot, working objects and the working area are improved. The details of the improvement have been discussed in the Section 7.7.

9.4 Recommendations and Future Work

Due to the limitation and the constraint, many of the new proposals and designs that were discovered and identified during the development and implementation of the project are not be able to be implemented in the current project. The proposals for the future work are as below:-

- i) Change the robot from the revolute type configuration (RRR) arm to SCARA type configuration which is more suitable for object pick and place from the top.
- ii) Change the end effector from gripper type to vacuum finger type. Thus the size of the area required to be cleared from the obstacle around the object to be picked and placed can be reduced.
- iii) Support more cameras to allow 3 dimensional operations such as object stacking.
- iv) The camera that supports zoom operation can be fixed on the robot end effector. This allows the objects to be view from various better perspectives. The accuracy for object gripping can be improved.
- v) Better users interactive where more than one user can be connected to the telerobotic server program for chatting.
- vi) Better system application such as allowing the user to manipulate with different shape and size of objects.
- vii) Voice recognition facility for spoken natural language.
- viii) Gesture recognition facility for object and coordinate specification on remote site.
- ix) User registration and feed back for data collection and analysis.
- x) Security precaution steps for the Internet connection such as by using the firewall, password login and the data encryption.

From the knowledge and experience gained, it is possible to extend from the current project to a totally new project to build a self-supervised task-oriented system such as Aibo from Sony Company and Asimo from Honda Company. New features

can be added to the command pre-processor to support voice and gesture recognition. The natural language conveyed by the system can be realized as sound. A new design of the gripper is required for more flexible and advanced arm behavior for different object gripping and placing. The knowledge of the system can be expended to make the system more intelligent such as the system can choose the best object picking behavior among behavior 1, 2 and 3 that are supported for the object of type A under various situation. The new system must be designed to work in a dynamic environment.

REFERENCE

- Allen, J. (1987). "Natural Language Understanding." Benjamin/Cummings Publishing Company, Canada.
- Allen, P. K. (1987). "Robotic Object Recognition Using Vision and Touch." Klumer Academic Publishers, USA.
- Asada, H. and Slotine, J. J. E. (1986). "Robot Analysis and Control." Wiley-Interscience Publication, USA.
- Brunner, B., Arbter, K. and Hirzinger, G. (1994). "Task-directed Programming of Sensor-based Robots." in Graefe, V.. "Intelligent Robots and Systems." Netherlands: Elsevier. 387 - 400.
- Corke, P. I. (1996). "Visual Control of Robots: High-Performance Visual Servoing." Research Studies Press, England.
- Craig, J. J. (1986). "Introduction to Robotics Mechanics and Control." Addison-Werley, Canada.
- Deitel, H. M. and Deitel, P. J. (1998). "C++ How to Program." Prentice-Hall Book Company.
- Farzin, B. R., Goldberg, K. and Jacobs, A. (1989). "A Minimalist Telerobotic Installation on the Internet." In 1st Workshop on Web Robots, Inter. Conf. on Robots and Intelligent Systems, Victoria, Canada, 7 - 13.

Fauzi Zakaria (2000). "Design and Development of a Prototype Internet-based Telerobotics." Universiti Teknologi Malaysia: Report of Research Proposal.

Fauzi Zakaria, Shamsudin H. M. Amin and Rosbi Mamat (2000). "Design and Development of Control System for Internet-based Telerobotics." Proc. of TENCON 2000, Kuala Lumpur, Malaysia, Vol. II, 338 - 342.

Friz, H. (1998). "Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views." Technical University of Clausthal: Diploma Thesis.

Fu, K. S., et al. (1987). "Robotics: Control, Sensing, Vision, and Intelligence." McGraw-Hill, Singapore.

Goldberg K. (Ed.) (1999). "The Robot in the Garden: Telerobotics and Telepresence in the Age of the Internet". The MIT Press, Massachusetts Institute of Technology, Cambridge.

Goldberg K., et al. (2000). "The Mercury Project: A Feasibility Study for Internet Robots." *IEEE Robotics and Automation Magazine*, 7(1):35-40.

Gomi, T., Ide, K. and Maheral, P. (1994). "Vision-based Navigation for an Office Messenger Robot." in Graefe, V.. "Intelligent Robots and Systems." Netherlands: Elsevier. 619 - 635.

Hager, G. D., Grunwald, G. and Toyama, K. (1994). "Feature-based Visual Servoing and Its Application to Telerobotics." in Graefe, V.. "Intelligent Robots and Systems." Netherlands: Elsevier. 415 - 430.

Heath, L. (1985). "Fundamentals of Robotics." Reston Publishing, USA.

Ishikawa, S., et al. (2000). "Man-machine Collaborative Work Based on Visual Communication." Proc. of the TENCON 2000, Kuala Lumpur, Malaysia, Vol. II, 321 - 325.

Kang, S. B. and Ikeuchi, K. (1994). "Robot Task Programming by Human Demonstration: Mapping Human Grasps to Manipulator Grasps." in Graefe, V.. "Intelligent Robots and Systems." Netherlands: Elsevier. 119 - 136.

Klafter, R. D., Chmielewski, T. A. and Negin, M. (1989). "Robotic Engineering." Prentice-Hall Book Company.

Lim, C. S., Rosbi Mamat and Zamani Md. Zain (2001). "Design of Task-Oriented System for Internet-based Telerobotic System." Proc. of the MSTC 2001, Malacca, Malaysia, B136.

Lim, C. S., Rosbi Mamat and Zamani Md. Zain (2002). "Natural Language in Task-oriented Telerobotic Application." Proc. of the WEC 2002, Kuching, Sarawak, Malaysia, 407-410.

Lloyd, J.E., Beis, J.S., Pai, D.K. and Lowe, D.G. (1997). "Model-based Telerobotics with Vision." Proc. of the ICRA '97, Albuquerque, New Mexico, 1297 - 1304.

Matrox Electronic Systems Ltd. (2001). "Matrox ActiveMIL User Guide." Canada.

Norhayati A. M, Shamsudin H. M. Amin and Rosbi Mamat (2001a). "Internet Based Leg Motion Control of a Mobile Robot." Proc. of the 2nd International Conference on Advances in Strategic Technologies (ICAST 2000), Selangor, Malaysia, Vol. 1, 283 - 292.

Norhayati A. M, Shamsudin H. M. Amin and Rosbi Mamat (2001b). "Development of the Internet Interface for Leg Motion Control of a Mobile Robot." Proc. of the 1st International Conference on Mechatronics (ICOM'01), Kuala Lumpur, Malaysia, Vol. 2, 665 – 675.

Rembold, U. (1986). "Programming of Industrial Robots, Today and in the Future." in Rembold, U. and Hörmann, K.. "Languages for Sensor-based Control in Robotics." Germany: Springer-Verlag. 3 – 23.

Rembold, U. and Hörmann, K. (Eds.) (1987). "Languages for Sensor-based Control in Robotics." Germany: Springer-Verlag. 601 – 611.

Rhino Robots, INC. (1989). "Mark IV Axis Controller Owner's Manual." USA.

Rhino Robots, INC. (1995). "Introduction to Robotics Student Workbook I." USA.

Richards, M. (2001). "Systems Development." NCC Education Limited, UK.

Roland, S., et al. (1998). "Guiding Mobile Robots through the Web". Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '98), 1-6.

Shamsudin H. M. Amin, Rosbi Mamat, Md. Fauzi Zakaria, Rudas, I. J., Horvath, L. and Tar, J. (2001). "User Interface Design for Internet-based Telerobotics." Proc. of SCORED 2001, Kuala Lumpur, Malaysia, CD Vol.

Shamsudin H. M. Amin, Rosbi Mamat, Mohamad Fauzi Zakaria, Norhayati A. M. and Lim, C. S. (2001). "Internet-based Telerobotics: UTM's Experience and Future Direction." Proc. of the ICAR 2001, Budapest, Hungary, 313 - 319.

Stein, M. R. (2000). "Interactive Internet Artistry." *IEEE Robotics and Automation Magazine*, 7(2):28-32.

Taylor, K. and Dalton, B. (1997). "Issues in Internet telerobotics." Inter. Conf. on Field and Service Robotics (FSR 97), Canberra, Australia, 151-157.

Taylor, K. and Dalton, B. (2000). "Internet Robot: A New Robotics Niche." *IEEE Robotics and Automation Magazine*, 7(1):27-34.

Torrance, M. C. (1994). "Natural Language with Robots." Massachusetts Institute of Technology MIT: MSc. Thesis.

Trent, R. (2001). "IIS 5.0: A Beginner's Guide." McGraw-Hill, USA.

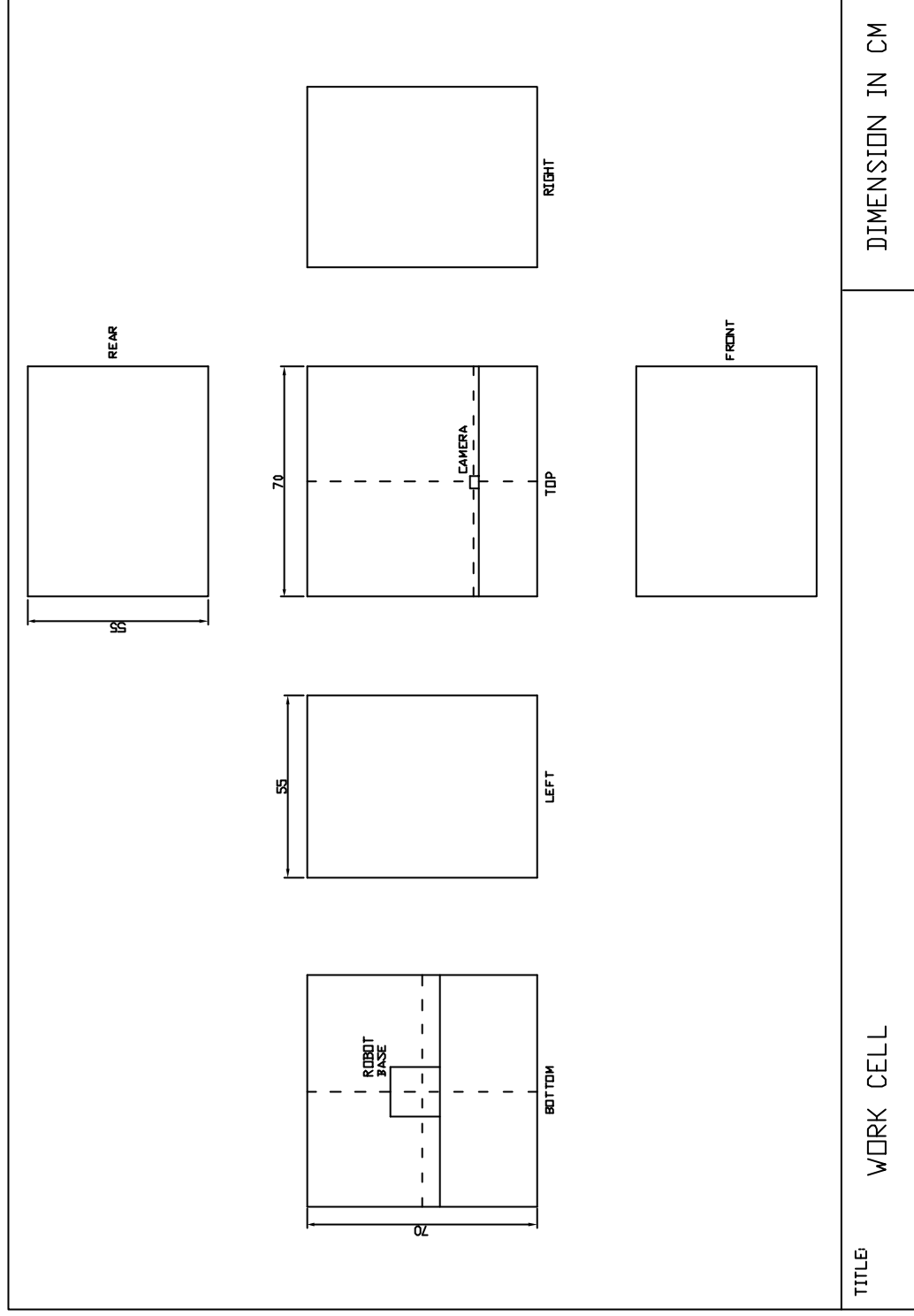
Wolovich, W. A. (1986). "Robotics: Basic Analysis and Design." CBS College Publishing, New York.

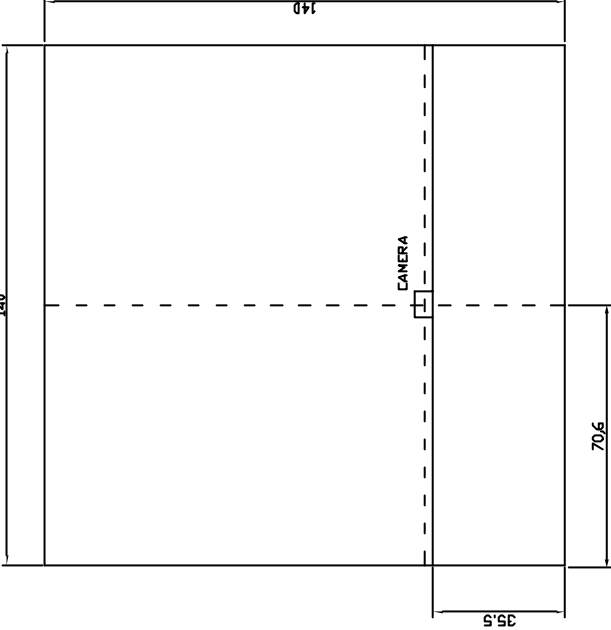
Young, M. L., et al. (1999). "Internet: The Complete Reference, Millennium Edition." McGraw-Hill, USA.

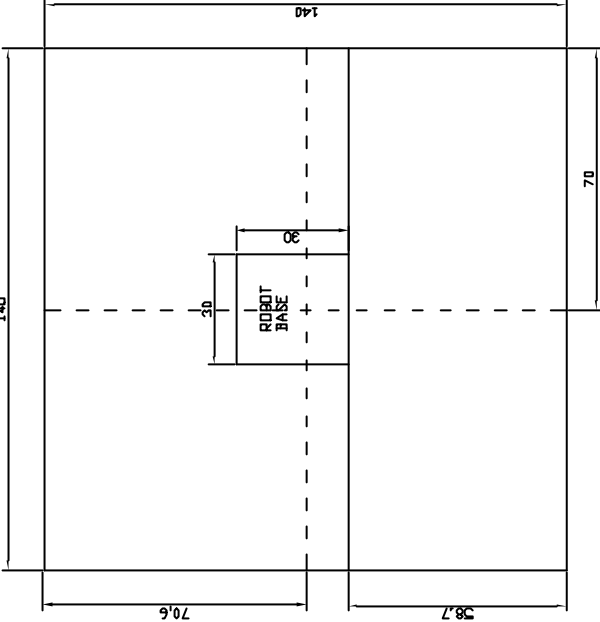
APPENDIXES

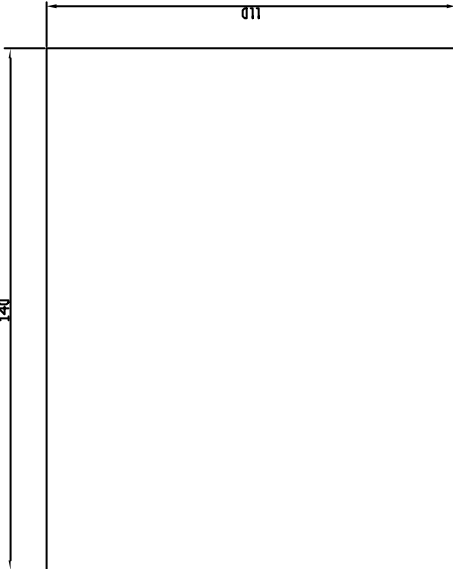
APPENDIX A

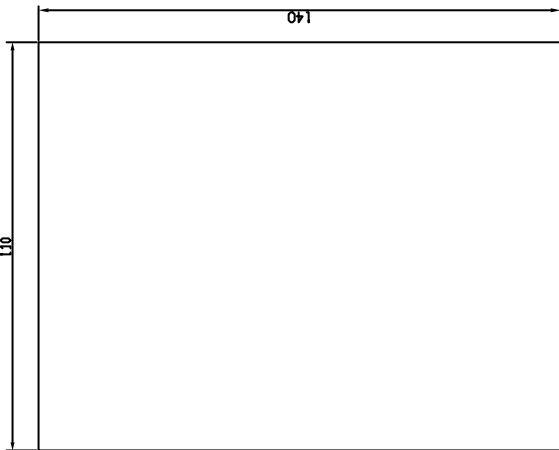
WORK CELL DIMENSION



 <p>Technical drawing of the top of a work cell. The drawing shows a rectangular layout with dimensions: 140 (width), 70.6 (height), 35.5 (width), and 140 (width). A dashed line indicates a camera position labeled CAMERA.</p>		DIMENSION IN CM
TITLE	TOP OF WORK CELL	

		TITLE	
BOTTOM OF WORK CELL		DIMENSION IN CM	

 <p>A technical drawing of a rectangular work cell. The drawing shows a rectangle with a vertical dimension line on the left side labeled '140' and a horizontal dimension line on the top side labeled '110'.</p>		TITLE	FRONT AND REAR OF WORK CELL	DIMENSION IN CM

 <p>A technical drawing of a rectangle. The vertical dimension on the left is labeled 110, and the horizontal dimension on the top is labeled 140. The drawing is centered within a large rectangular frame.</p>		TITLE	LEFT AND RIGHT OF WORK CELL	DIMENSION IN CM

APPENDIX B**PAPER PUBLISHED**

INTERNET BASED LEG MOTION CONTROL OF A MOBILE ROBOT

Norhayati A.Majid, Shamsudin H.M.Amin and Rosbi Mamat
Centre for Artificial Intelligent and Robotics (CAIRO)
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor, Malaysia

Keywords: Internet; Internet-based telerobotics; leg motion control

ABSTRACT

Nowadays, advancement in technology has enabled us to control machines or robot that are connected through the Internet. The application of Internet-based telerobotics technology can also be implemented for hazardous operation such as in nuclear reactors, or for difficult to reach places in underwater operation, teleoperated surgery, manufacturing and space application. This paper presents the basic implementation of a user interface for controlling the leg motion of a multi-legged mobile robot via internet including the graphic user interface for the server-side and client-side, implemented through TCP/IP network protocol. The design includes the controller of the leg and will be expanded to the supporting system like vision. In this work, the movements including forward, reverse, up, down and brake of a leg of a mobile robot can be controlled via Internet through the graphic interface. The robot motion can be seen through the digital camera that was focussed on the robot. A robot called ROBOC UTM II serves as a web robot. This work is part of the Malaysia-Hungary joint research project on development of Internet-based telerobotics. Up to this moment, the client and the server is standalone. Description of system architecture and steps of implementation for this project are fully described in this paper.

INTRODUCTION

The early Internet telerobotics system was introduced by Goldberg et al. (1995a) in the Mercury project installed in 1994; the Australia's Telerobot on the web by Taylor and Trevelyan (1995), which came on-line in nearly at the same time and Telegarden project, popularised by Goldberg et al. (1995b), which replaced the

Mercury robot. After that Internet telerobotics has moved on to mobile robot applications. This inspiration has led to the design the Internet mobile robot. Therefore, this project concerns with creating an Internet environment for motion control of a leg of a multi-legged mobile robot. The system consists of one leg, which is actuated by three DC motors, driven by logic signals from a computer. A digital video camera is used to view the movement of the leg. All the hardware systems are combined together and connected to the computer (set as server). Instead of controlling the leg directly from the computer, it is extended to the Internet users. The intention is to provide a basic internet-based mobile robot environment that can be implemented on real mobile robot system. Therefore several issues need to be considered such as the capabilities of the controller and design interface on controlling the motion of the leg of mobile robot. This work is part of the Malaysia-Hungary joint research project on development of Internet-based telerobotics proposed by Shamsudin et al. (1998). Up to this moment, the client and the server is standalone, which means that the client will have to download and execute the running program from the server and start connecting to the server by calling an Internet protocol (IP) address.

SYSTEM ARCHITECTURE

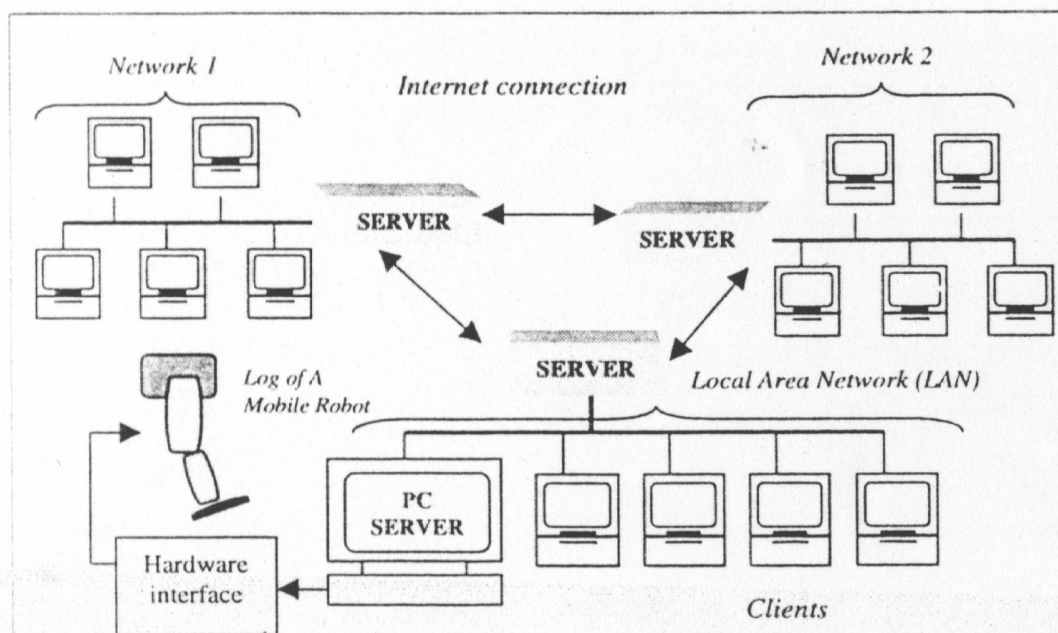


FIGURE 1 Hardware Architecture of the Software Development Environment for Internet Based Leg Motion Control of Mobile Robot

The Internet architecture of the leg of mobile robot system is shown in Figure 1. The robot is connected to the computer, which is set as a local server. Then the computer is connected to a server which has many computers connected to it. Other computers can act as clients and can download the interface and control the motion of the leg of mobile robot through the local area network (LAN) area. When the system goes to the public network, other users from any computer can also control the robot.

Internet-based control systems must rely on the available communication protocols to exchange real time data between the controller and the process. Today, most of the protocols provide a transparent and reliable support for data exchange among computers using the transfer control protocol (TCP). The protocol first used by Roberts and Sharkey (1995) which was subsequently used by Fiorini and Oboe (1997). TCP is guaranteed, connection based control protocol, which allows information to be sent between two processes, which may run on different machines. TCP/IP is a backbone of the Internet and referring to group of protocol related to the TCP and IP protocol. It is used to build a server that can work over long distances. This protocol can provide full duplex stream service, with automatic error handling, retransmission, packet re-ordering and guarantee of self-delivery and it can be used to enable the communication between the user and the robot.

CLIENT-SERVER MODEL APPLICATION

Client-server model describes the relationship between two computer programs or more. According to Carlos and Caroline (1994), the World Wide Web consists of a network of computers, which can act, in two roles: as servers, providing information; or as clients, requesting for information. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common and have become one of the central ideas of network computing. In the usual client/ server model, one server is activated and awaits client request. Typically, multiple client programs share the service of a common server program. Both client programs and server programs are often part of a larger program or application. Relative to the Internet, the Web browser is a client program that requests services (the sending of data or web pages) from the Web server in other computer somewhere in the Internet. Similarly the user computer with TCP/IP installed is allowed to make client request from other computers on the Internet and in this project a similar approach is implemented.

DESIGNING THE GRAPHIC USER INTERFACE (GUI)

A graphic user interface (GUI) is a set of graphic pages for the user interface that can be designed by using GUI programming language. Paulos and Canny (1996) have implemented several Internet robots with elegant user interface. Paulos and Canny (1995) has faced the similar limitation problem that was early faced by Goldberg (1994). The design interface for the system was challenging due to the limitations of the Hyper Text Markup Language/Hyper Text Transfer Protocol (HTML/HTTP) environment, as well as network traffic considerations. The restrictions imposed by the HTML made it difficult to design an intuitive user interface to a robotics system. Therefore, in this project the graphic interface is designed using C++ Builder which is an object-oriented, visual programming environment for general-purpose and client/server application. In C++ Builder, the socket programming can be written connecting to the Internet through the TCP/IP protocol. The client and the server is a stand alone where the client has to download and execute the program from the server and start connecting to the server by calling the server IP address. Meanwhile, the server must be in active condition and listen to any client who wants to communicate and control the robot. This design is suitable for any interface design for Internet-based telerobotics environment. The client-server interface is shown in Figure 2 below.

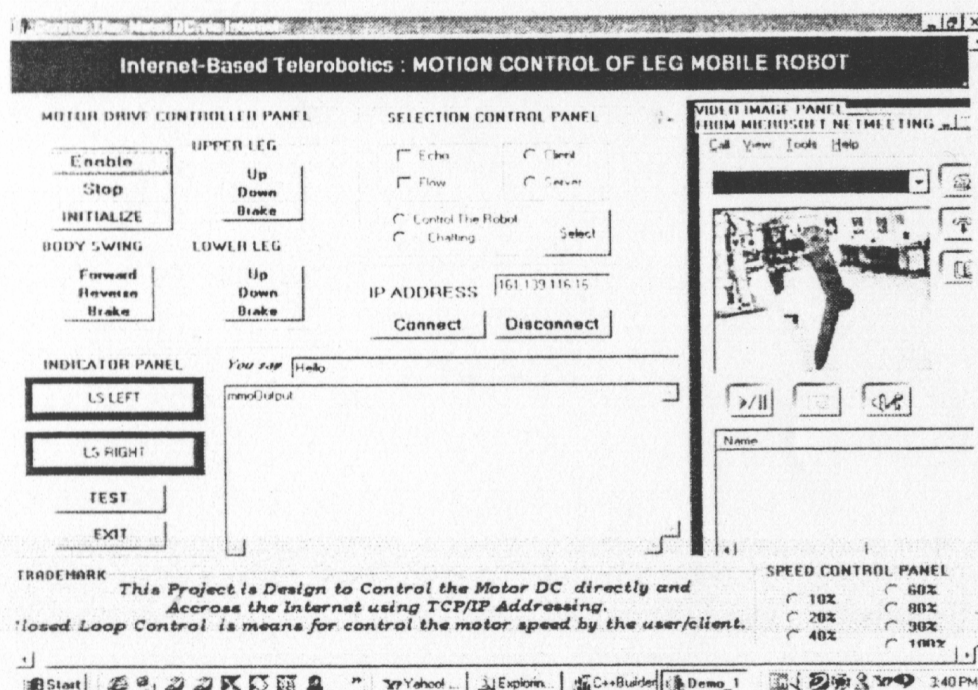


FIGURE 2 The Graphic User Interface for Client and Server

The listening connection introduced by Calvet et al. (1998) has been chosen as a type of socket connection. The server does not locate the clients but associate a queue with the listening connections; the queue records client connection requests as they come in. When the server socket accepts a client connection request, it forms a new socket to connect to the new client, so that the listening connection can remain open to accept other client request. The software architecture includes with the simple socket component that enables the client to connect to a server and perform simple socket communications. It also enables the client to send a stream of data to the server, and allowing for easy transmission of table data. Therefore, many users from anywhere in the networks that are connected to the server can send the command to the robot. The server socket component can be used to manage the client connections. When the program reads or writes data to the socket, the program present input/output request to that socket and then continue executing. A non-blocking socket is used when the socket needs to synchronise reading and writing with server sockets. The client type has to be set as non-blocking to enable the client socket to respond to asynchronous reading and writing events.

VISION IMAGE FEEDBACK

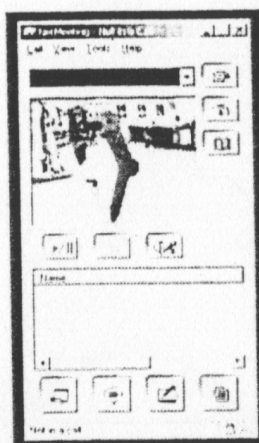


FIGURE 3 The View of the Leg of Mobile Robot Movements

The remote telerobotic experiments have used high bandwidth communication link, typically with real-time video. To make the control environment look real, the client can control the robot through the graphic interface and the movement of the robot can be seen through the digital camera that was focussed on the robot. In the interface designed, clients cannot see the real movement of the leg of mobile robot. Here, the system is supported by the view of the digital video

camera. Of course there is a delay that occurs on the vision because a big information or data of the leg motion need to be sent through the TCP/IP to the client. The vision can be seen using Microsoft NetMeeting. Figure 3 shows picture of the leg of mobile robot that is connected through the Internet.

HARDWARE DEVELOPMENT

In the development stage of the hardware part, the robot is designed using the leg of the mobile robot, which is actuated by three brushless DC servomotors. The actuator system is actuated by the interface circuits using power MOSFETs. The typical motor functions include open loop speed, forward or reverse direction, run enable and dynamic braking. The driver MC33035 is used because it is designed to operate efficiently on controlling the brushless DC servomotor. The PC parallel port can be very useful I/O channel for connecting the circuits to PC. PC Parallel port has 25 pins D-shaped female connector in the back of the computer. In open loop controller, only 8 output pins (data lines) will be used to control the motion of the mobile robot leg. In this case every single command that was sent by the client will be executed on the server by sending the certain output to the interfacing circuits. The command will invoke the robot to move to the certain position without knowing the actual position. All the hardware is assembled together to build the simple leg of mobile robot system. The mobile robot system architecture is shown as Figure 4 below.

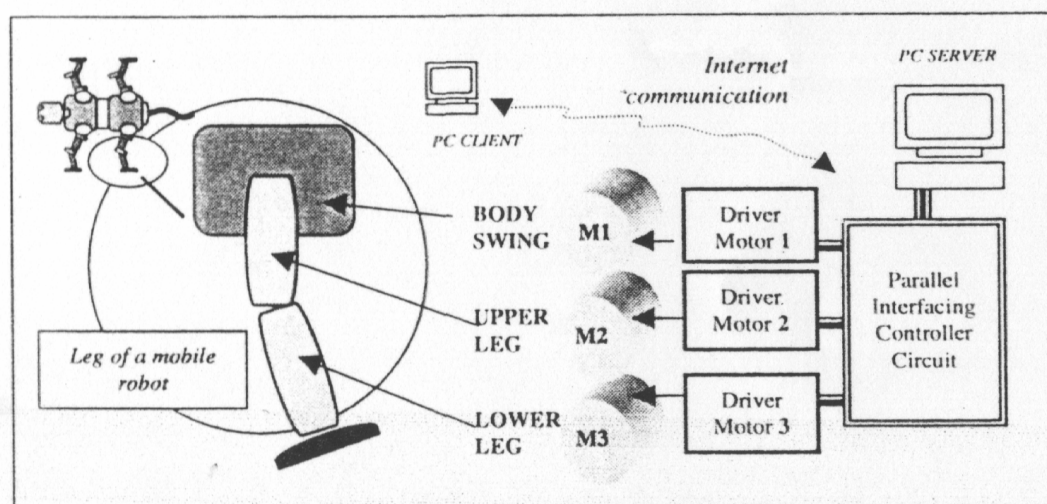


FIGURE 4 The Hardware Architecture for Parallel Interfacing with the Computer Server

MOTION CONTROL CAPABILITIES

The leg has capabilities to move with certain degrees of motion. The GUI including the control motion panel for controlling the bodies swing, upper leg and lower leg movements as shown in Figure 5 below. The interface also includes the control speed panel and indicator panel as shown in Figure 6(a) and 6(b) below.

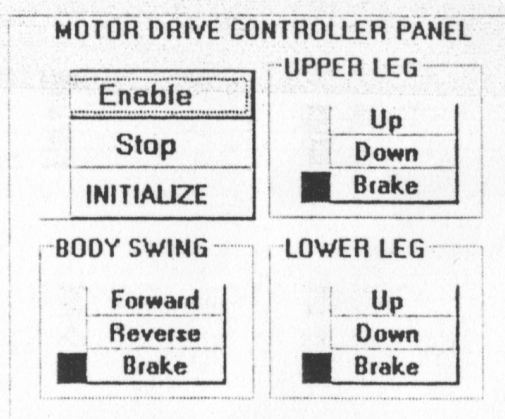


FIGURE 5 The Motor Drive Control Panel Interface

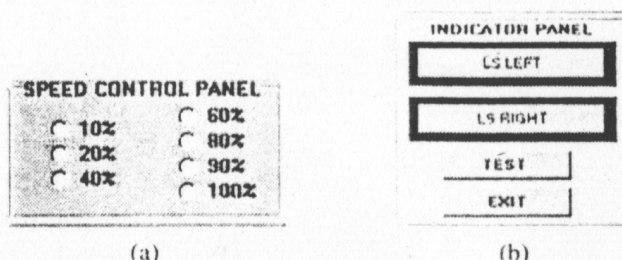


FIGURE 6 (a) Speed controls Panel Interface; (b) Indicator Panel Interface

INTERFACING WITH THE PARALLEL PORT

A PC printer port is an appropriate platform for implementing projects dealing with the control of real world peripherals. The printer provides eight TTL outputs, five inputs and four bi-directional leads and it provides a very simple means to use the PC interrupts structure. Each printer ports consists of three port addresses; data, status and control. These addresses are in sequential order.

Typically the data port is at address 0x0378, the corresponding status port is at 0x0379 and the control port is at 0x037a. In this project, certain PC output or input values need to be specified for controlling the leg motion of the mobile robot. To make the robot perfectly move the interface is programmed to send an output and receive an input data from the parallel port. The pins assignment and the data signal for motion control of leg mobile robot are listed in Table 1 below.

TABLE 1 Pins Assignment for Motion Control of Mobile Robot

Functions		Command	Input data (8 byte)							
			D7	D6	D5	D4	D3	D2	D1	D0
Stop / Enable		0 x 01	0	0	0	0	0	0	0	1
A. Body	Forward	0 x 06	0	0	0	0	0	1	1	0
	Swing	Reverse	0 x 04	0	0	0	0	1	0	0
		Brake	0 x 02	0	0	0	0	0	1	0
B. Upper	Up	0 x 18	0	0	0	1	1	0	0	0
	Leg	Down	0 x 10	0	0	0	1	0	0	0
		Brake	0 x 08	0	0	0	0	1	0	0
C. Lower	Up	0 x 60	0	1	1	0	0	0	0	0
	Leg	Down	0 x 40	0	1	0	0	0	0	0
		Brake	0 x 20	0	0	1	0	0	0	0
			Motor 3		Motor 2		Motor 1			

RESULTS AND DISCUSSIONS

This Internet based leg motion control is a one-client or multi-clients to one-server basis communication. The telerobotics session can execute between any pair of client and server provided the IP server is known prior to starting the session. Several tests have been made on the session by controlling the leg of mobile robot from various places namely within the local region. Besides controlling the robot, both server and client can communicate through the 'chat' communication service in the package. The test that has been done in three different situations including direct control, short distance and long distant by single and multi clients.

The first test is done using direct control from the computer server. The use of parallel interface may be advantageous for teleoperation system. The data still can be sent or received almost in real time. This is very important to perform the task efficiently. The control motion capabilities have been successfully tested.

Then the single and multi-client in a LAN environment tested to controlled the leg of mobile robot. The LAN is a network of interconnected workstations sharing the resources of a single processor or server within a relatively small geographic area. Typically, a suite of application programs can be kept on the

LAN server. The users who need the application frequently can download it once and then run it from their local hard disk. The test also has been successfully done and the robot can perform as the client request. The first request from the first client is done first. If there is another request from other client the server will execute the request by following the sequence.

When the test is done in the wide area, there is delay in response and conflict may occur. The users from the Internet area are set as a client and then activate the client program. When sending the request to control the motion of the robot, the result is not the same as controlling in the LAN environment. And the vision is not real time because of the delay.

CONCLUSIONS

At this moment, the design of the basic interfacing of Internet-based telerobotics for leg of mobile robot has been successfully carried out. The project can be extended to advance application on a wider environment. The development of the interface for leg motion control of multi-legged mobile robot involves the construction of the hardware, software and network protocol part. The software interface part has been developed using C++ Builder and all the available components provided by the software. The hardware is the interface and controller for the robot leg. Both systems are combined together and produced the environment of leg motion control via Internet through the TCP/IP.

NOTATIONS

TCP	Transfer Control Protocol	GUI	Graphic User Interface
IP	Internet Protocol	HTML	Hyper Text Markup Language
DC	Direct current	HTTP	Hyper Text Transfer Protocol
LAN	Local Area Network		

ACKNOWLEDGEMENTS

The authors would like to thank the Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

REFERENCES

- Calvet, C. et al. 1998. *Borland C++ Builder 3 Unleashed*, United States: Sams Publishing.
- Carlos, A.V. and Caroline, C.H. 1994. Providing Data on the Web: From Examples to Programs, *In Second International World Wide Web Conference*, Chicago, <http://fiaker.ncsa.uiuc.edu:8080/www94-2/paper.html>.
- Fiorini, P. and Oboe, R. 1997. Internet-Based Telerobotics: Problems and Approach, *In Proc. 8th Int. Conf. on Advanced Robotics (Icar'97)*, Monterey, Vol. 1:pp 765-770.
- Goldberg, K. et al. 1994. Beyond the Web: Excavating the Real World Via Mosaic, *In Second International World Wide Web Conference*, Chicago, <http://www.icor.berkeley.edu/~goldberg/>.
- Goldberg, K., Mascha, M., Gentner, S., Rothenberg, N., Scutter, C., and Wiegley, J. 1995a. Desktop Tele-operation via the World Wide Web, *In Proc. IEEE Int. Conf. on Robotics and Automation*, Vol.1: pp.654-659.
- Goldberg, K., Santarromana, J., Bekey, G., Gentner, S., Morris, R., Wiegley, J., and Berger, E. 1995b. The Telegarden, *In Proceedings ACM SIGGRAPH*, <http://cwis.usc.edu/dept/garden>.
- Paulos, E. and Canny, J. 1995. A World Wide Web telerobotics remote environment browser, *In 4th Int. World Wide Web Conf.*, Boston, Massachusetts, <http://www.cs.berkeley.edu/~paulos/papers/www4/index.html>.
- Paulos, E. and Canny, J. 1996. Delivering real reality to the world wide web via telerobotics, *In Proceedings IEEE Int.Conference on Robotics and Automation (ICRA)*, Minneapolis, Minnesota, pp. 1694-1699.
- Roberts, D.J. and Sharkey, P.M. 1995. A Real-time, Predictive for Distributed Virtual Reality. *In Proceedings 1st Conf. Simulation and Interaction in Virtual Environment*, Vol. 1, pp. 72-81.
- Shamsudin H.M Amin et al. 1998. *Project Brief: Development of Internet-Based Telerobotics*, Malaysia: University Technology of Malaysia.
- Taylor, K. and Trevelyan, J. 1995. Australia's telerobot on the web, *In Proc. 26th Int. Symposium Industrial Robots*, Singapore, <http://telerobot.mech.uwa.edu.au/>.

DEVELOPMENT OF THE INTERNET INTERFACE FOR LEG MOTION CONTROL OF A MOBILE ROBOT

Norhayati A.Majid

Shamsudin H.M.Amin

Rosbi Mamat

Centre for Artificial Intelligent and Robotics (CAIRO)

Faculty of Electrical Engineering

Universiti Teknologi Malaysia

81310 UTM Skudai, Johor, Malaysia.

yatie_531@lycos.com

sham-amin@ieee.org

rosbi@suria.fke.utm.my

ABSTRACT

Recent development in internet-based robotics has enabled many types of robots becoming accessible to the public including fixed and mobile types. This field of research has captured the interest and motivated the researchers and web developers to design and develop new telerobotics web applications that have not been practical in the past. Several studies have been made on searching the concepts of World Wide Web telerobotics that is still in its infancy. The telerobotics web developers need to know how the robot users behave and how can the interface design be made to be attractive, useful and intelligent to make users remain interested in controlling and playing with the robot. The research include the studies on the level of commitment that robot operators show. This telerobotics web interface may result in easier and practical ways to design the web interface. The applications of internet-based robotics include telemanufacturing, training, services and entertainment. There are many approaches in designing the WWW telerobotics interface including implementation of COBRA, RRI, Java Applets, C++, CGI and etc. Some developers may consider on how this technology approaches may interest Internet users. In this project, the Internet interface is designed using C++ Builder and has been tested on the leg of a mobile robot system. The leg of a mobile robot is a part of ROBOC UTM version II that is still under development stage. The leg of the mobile robot can be controlled using socket connections via Transfer Control Protocol / Internet Protocol (TCP/IP) network protocol and the robot movement can be seen through a digital video camera that focuses on the robot. This paper emphasizes more on the design and architecture of the interface for UTM's internet-based telerobotics. This will also include a summary on client-server model and windows socket programming that have typically been used in internet application. Up to this moment, the previous design of the interface has been improved to make it intelligent and easy-to-use. Steps of design and implementation for this project are described in this paper.

Keywords: Internet-based telerobotics; Leg motion control; Client-Server architecture; Windows socket programming.

1. INTRODUCTION

Telerobotics web is a concept that involves control of a remote robot or device from a within a client-server application or a web browser over the internet. It requires a supervisory control scheme to avoid instability and makes a robot or device available to vast numbers of users, thus opening up a new range of applications [1]. Many researchers have tried to manipulate the concepts of teleoperation to telerobotics. So that the concept of teleoperation can be extended to the advances application such as in outer space and underwater exploration. In most teleoperation system, a single user input controls a single robot. However in the telerobotics environment, the interfaces need to be designed to manage working with many users. The early internet telerobotics system was introduced by Goldberg et. al [2] in their Mercury project installed in 1994; the Australia's Telerobot on the web developed by Taylor and Trevelyan [3], and Telegarden project popularised by Goldberg et. al [4], which replaced the Mercury robot. After that internet telerobotics has been implemented on mobile robot such as Khepera [5], Minerva the second-Generation Museum tour-guide robot introduced by Sebastian [6], Rhino robot and Xavier-the autonomous indoor mobile robot [7].

2. DESIGN CONSIDERATION

The goal of this project is to fulfil the telerobotics user requirements. Therefore, there are a few factors that the designer must consider while working with the internet telerobotics interface. The most important thing is to make the interface simple but attractive and still can provide enough information about the robot motion. Helping user to easily understand how to control the robots is one of the basic tasks that need to be concern. Most of the users do not have an ideas how to start when first time access the robot control interface. Some users face difficulty to access the controller and they have to queue for a long time without knowing what are the current status of the robot. The interface can be more useful is the users can be help to know the status of the robot up to date even when other users control the robot. The most challenging part is how to design the interface that can manage to handle errors originating from the users by executing an appropriate command that may be under the user command. Perhaps in future the telerobotics can behave autonomously.

2.1 Client-Server Model Architecture

Client-server model describes the relationship between two computer programs or more. This architecture is one of the possible architecture for the software to implement the roles of windowing system. Alan Dix et. al [8] showed this architecture could provide the most portability, since the management function is written as a separate application in its own right and operating systems. The World Wide Web consists of a network of computers, which can act, in two roles: as servers, providing information; or as clients, requesting for information as explored by Carlos and Caroline [9]. In a network, the client-server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. In the usual client-server model, one server is activated and awaits client request. Multiple clients can share the service of a server program. Relative to the internet, the Web browser is a client program that requests services (the sending of data or web pages)

from the Web server in other computer somewhere in the internet. Similarly the user computer with TCP/IP installed is allowed to make client request from other computers on the internet and in this paper a similar approach is implemented.

2.2 Windows Socket Programming

Windows sockets provide connections based on the TCP/IP protocol. Sockets will give some of the power of DCOM without loading down with concerns regarding connectivity, security and Windows NT domains. The disadvantage of sockets is that they lack the backing of a full object-oriented scheme such as in COM. Technically, sockets are easy-to-use and flexible and is the ideal solution when building distributed applications. This technology is built on TCP/IP and it should appeal to people who want to work across very large distances using the Internet. There are two components that can be used for windows socket programming, which is known as Client Socket and Server Socket. TClientSocket is used to connect to a Server socket. The TClientSocket is the primary interface on the client side. There are a few steps need to be completed before the designer can use this component [10]:

1. Write an Event handler for the OnRead Event (such as **Connect** button).
2. Set the Address property to whatever the server address is (i.e., 161.139.116.16).
3. Set the Port to whatever pre-determined number the Server will be listening on (i.e., 1234).
4. Set ClientType to ctNonBlocking
5. Call the Open method to connect to the remote Server. The server should already be running or the Open call will fail.

2.2.1 Writing Data to the Server

The description on how the client can write data to a server over the connection is shown as Fig. 1:

```
MyClientSocket->Socket->SendText ("this is a test");
    ↑           ↑
TClientSocket TClientWinSocket
```

Figure 1: Writing data to a server socket

2.2.2 Reading Data from the Server

To read data from the server, the programmer must include code such shown in Fig. 2.

```
void __fastcall TfrmMain::sckClientRead(TObject *Sender,
TCustomWinSocket *Socket)
{
    String Data = Socket->ReceiveText();
    MmoOutput->Lines->Add(Data);
}
```

Figure 2: Code for reading data from a server.

2.2.3 Writing Data to a Client

To send data to a client, the array is sent using TServerSocket as below.

```
MyServer->Socket->Connections[3]->SendText("this is a test");
```

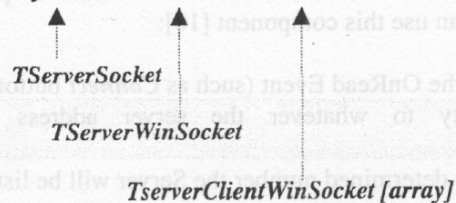


Figure 3: Writing data to a specific client socket

2.2.4 Reading Data from a Client

To read data from a client, the programmer must include this code.

```
void __fastcall TfrmMain::sckServerClientRead(TObject *Sender,
TCustomWinSocket *Socket)
{
    String Data = Socket->ReadText();
    MmoOutput->Lines->Add(Data);
}
```

Figure 4: Reading data from a client.

2.3 TCP/IP network protocol

Today, most of the network protocols provide a transparent and reliable support for data exchange among computers using the transfer control protocol (TCP). The protocol was first used by Roberts et.al [11], which was subsequently used by Fiorini and Oboe [12]. TCP is guaranteed, connection based control protocol, which allows information to be sent between two processes, which may run on different machines. TCP/IP is a backbone of the Internet and referring to group of protocol related to the TCP and IP protocol. It is used to build a server that can work over long distances. This protocol can provide full duplex stream service, with automatic error handling,

retransmission, packet re-ordering and guarantee of self-delivery and it can be used to enable the communication between the user and the robot [13].

2.4 Hardware Interface

The hardware set-up consists of a leg of a mobile robot, which has three parts that move in three different kinds of movements. The first part is joined to the body, second joint is connected between the body and the upper leg and the third joint is connected between the upper leg and the lower leg. The controller devices are on-board motor drivers, which were connected to the computer through the interfacing board. The host computer, which acts as a server, communicates with the robot via parallel port interfacing. Every single command that send to the server interface will invoke the server to perform a certain task to the robot. Overall overview of the hardware configuration set-up can be shown as in Fig. 5 below.

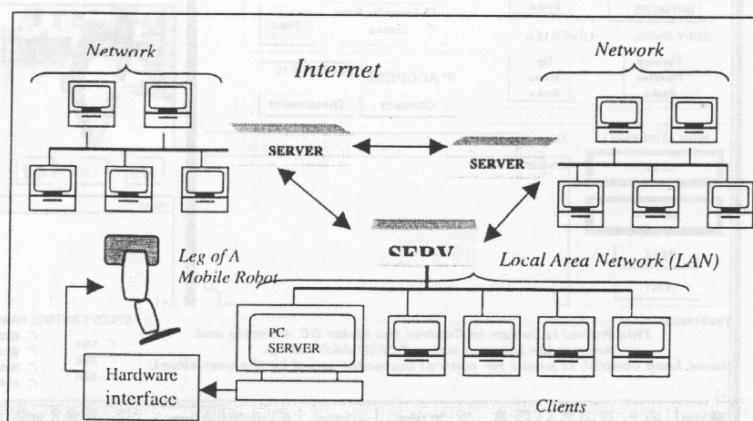


Figure 5: Hardware Architecture of the Software Development Environment For Internet based Leg Motion Control of Mobile Robot.

3. GRAPHICAL USER INTERFACE (GUI)

First effort in this work is to design the basic interface that has the internet telerobotics concepts. This basic interface has been designed using Borland C++ Builder because it is an object-oriented programming language, provides a VCL (Visual Component Library) which is needed to generate the graphical user interface for any type of application. The architecture of the basic design shows in Fig. 5 above. The leg of the mobile robot system is connected to the computer and is set as a local server. The computer is also connected to a server which has many computers connected to it. Other computers can act as clients and can download the interface and control the motion of the leg of mobile robot through the local area network. When the system goes to the public network, other users from any computer can control the robot.

3.1 Preliminary Design

One of the most important components of any telerobotics system is the user interface. The display in the user interface should be designed sufficiently so that the user receives enough information about the remote environment. The preliminary basic design of the GUI is shown in Fig. 6. This GUI consists of few panels including motor drive controller panel, speed control panel, selection of either to be a client or a server, indicator panel and video image panel.

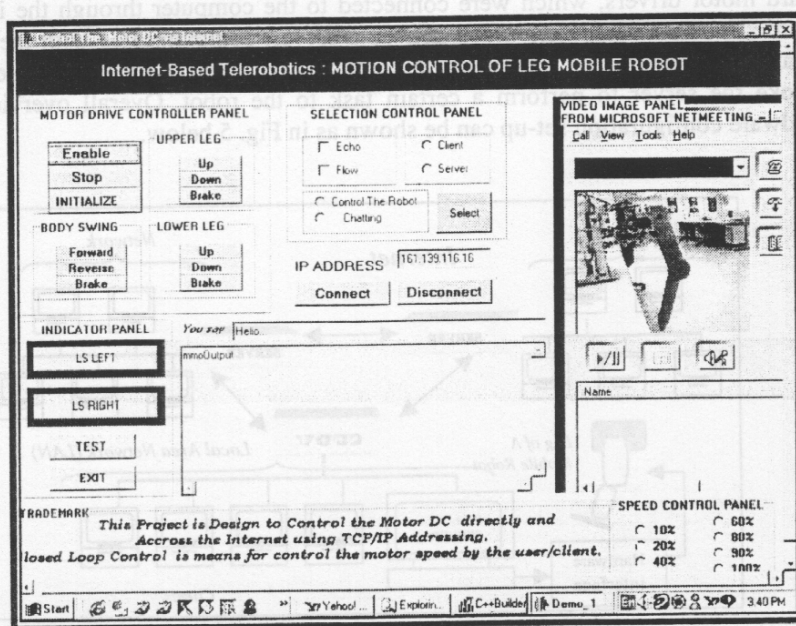


Figure 6: Preliminary basic design of GUI

3.2 Internet Interface

The control interface for telerobotics can be designed using high-level language with visual and internet components and other available methods. Normally the data will be sent through the network using a better and faster method and the user can assure that the robot executes the command. The interface of motion control can be classified as intelligent if its can behave intelligently even with the novice users. They can provide all the information needed and make the user feel the robot is in front of him. Therefore, the preliminary design of GUI has been improved to achieve the project goal. The interface architecture is designed for continuous co-operations between the user and the interface.

3.3 Design Structure Improvement

Some modification has been made on the preliminary design. The architecture of the design has been re-structured so that the preliminary design can be improved. The re-structured architecture is shown in Fig. 7 below.

3.4 Preliminary Results

After some modification and improvement has been made to the previous design, the basis interface can perform much better. The new features including log in panel and channel selection panel are shown in Fig. 8(a) and Fig. 8(b). After the client receives permission from the server to log in, the client will be connected to the server side either in the control robot channel or chat communication channel. The server can know the client identification beside the client IP address.

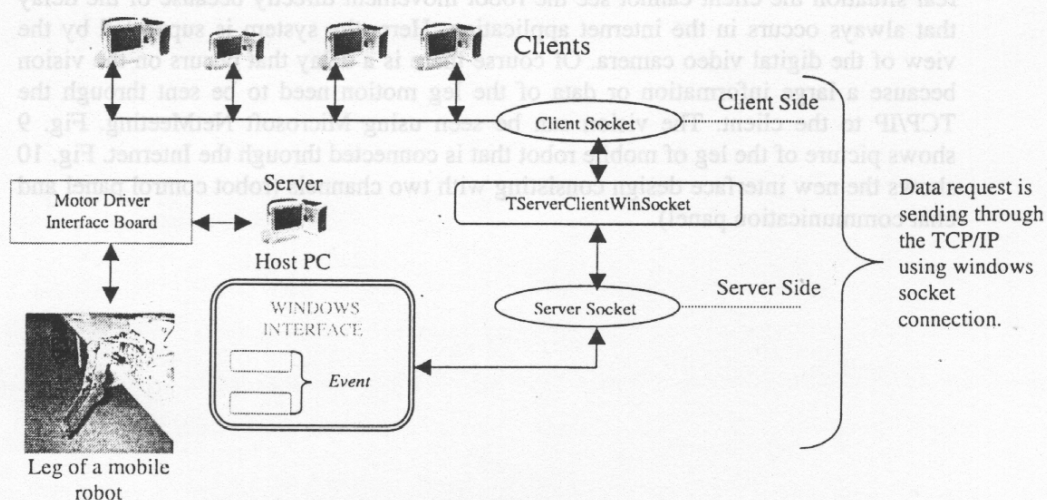


Figure 7: System Architecture

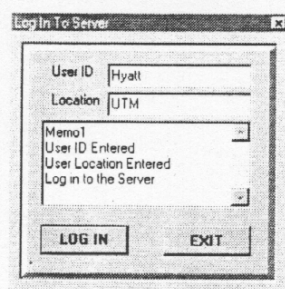


Figure 8 (a): The Log In Panel

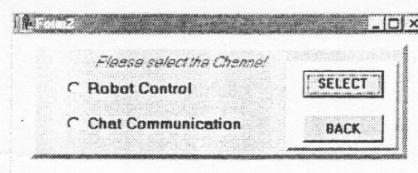


Figure 8 (b): The Channel Selection Panel

After log in session, the client needs to select either to control the robot or to communicate with other clients through the chat communication channel. Here, the clients are continuously notified about the current robot status and who is currently controlling the robot by looking at the Memo panel.

The telerobotics remote interface is designed using high bandwidth communication link, typically with real-time video. To make the control environment look real, the client can control the robot through the graphical interface and the movement of the robot can be seen through the digital camera that was focussed on the robot. In the real situation the client cannot see the robot movement directly because of the delay that always occurs in the internet application. Here, the system is supported by the view of the digital video camera. Of course there is a delay that occurs on the vision because a large information or data of the leg motion need to be sent through the TCP/IP to the client. The vision can be seen using Microsoft NetMeeting. Fig. 9 shows picture of the leg of mobile robot that is connected through the Internet. Fig. 10 shows the new interface design consisting with two channels (robot control panel and chat communication panel).

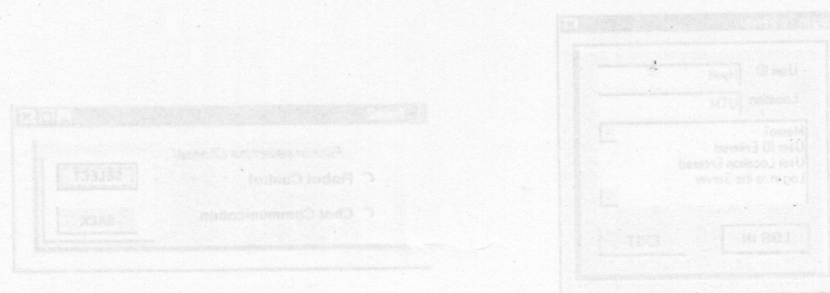
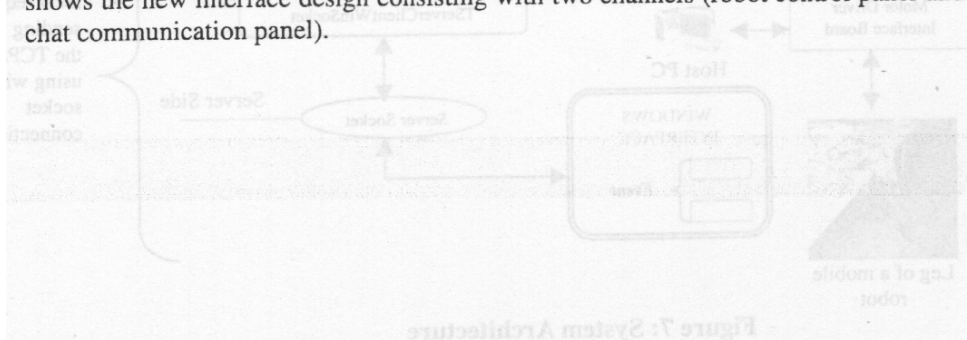


Figure 8 (b): The Channel Selection Panel

Figure 8 (a): The Log in Panel

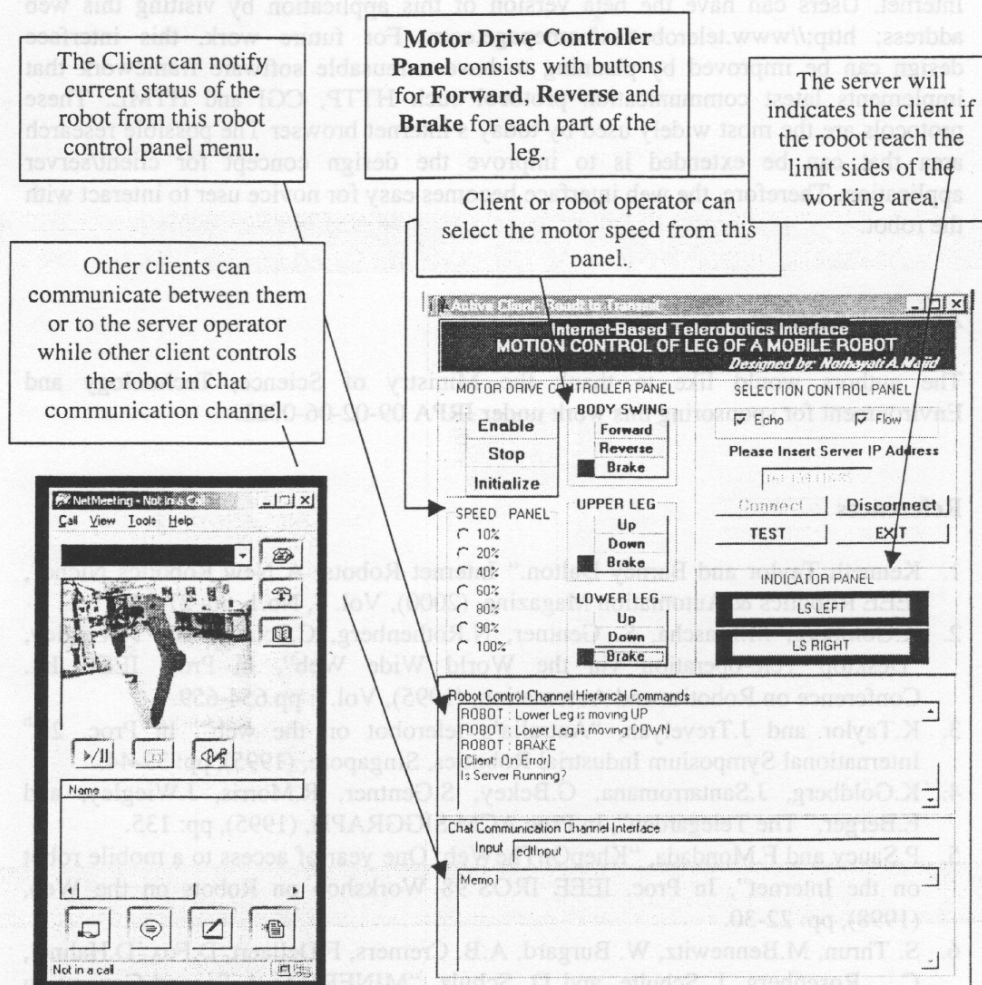


Figure 6: View from the Digital Web Camera

Figure 7: Graphical User Interface

4. CONCLUSION

At this moment, the interface design of internet-based telerobotics for leg of mobile robot has been successfully carried out. The development of the interface for leg motion control of multi-legged mobile robot involves the construction of the hardware, software and network protocol part. The software interface part has been developed using C++ Builder and all the available components provided by the software. The hardware interface is the controller for the robot leg. Both systems are combined together and produced the environment of leg motion control via Internet through the TCP/IP. The client and server will behave like a standalone application, which means the user must download and execute the application program from the server. The user will be allowed to control the robot after obtaining permission from the server. The interface has been tested locally but is not yet freely available on the

Internet. Users can have the beta version of this application by visiting this web address; <http://www.telerobotics.homepage.com>. For future work, this interface design can be improved by planning to have a reusable software framework that implements latest communication protocol such HTTP, CGI and HTML. These protocols are the most widely used by today's internet browser. The possible research area that can be extended is to improve the design concept for client/server application. Therefore, the web interface becomes easy for novice user to interact with the robot.

Acknowledgement

The authors would like to thank the Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

References

1. Kenneth Taylor and Barney Dalton, "Internet Robots: A New Robotics Niche", IEEE Robotics & Automation Magazine, (2000), Vol. 7, No.1, pp: 27-34.
2. K.Goldberg, M.Mascha, S. Gentner, N.Rothenberg, C.Scutter, and J.Wiegley, "Desktop Tele-operation via the World Wide Web", In Proc. IEEE Int. Conference on Robotics and Automation, (1995), Vol.1: pp.654-659.
3. K.Taylor and J.Trevelyan, "Australia's telerobot on the web", In Proc. 26th International Symposium Industrial Robotics, Singapore, (1995), pp: 39-44.
4. K.Goldberg, J.Santarromana, G.Bekey, S.Gentner, R.Morris, J.Wiegley, and E.Berger, "The Telegarden", In Proc.ACM SIGGRAPH, (1995), pp: 135.
5. P.Saucy and F.Mondada, "KhepOnTheWeb: One year of access to a mobile robot on the Internet", In Proc. IEEE IROS'98 Workshop on Robots on the Web, (1998), pp: 22-30.
6. S. Thrun, M.Bennewitz, W. Burgard, A.B. Cremers, F Dellaert, D.Fox, D.Hahnel, C. Rosenberg, J. Schulte, and D. Schulz, "MINERVA: A Second-Generation Museum Tour-guide Robot", In Proc. Int. Conf. Robotics and Automation (ICRA'99), (1999), pp: 1997-2085.
7. Reid Simmons, Joaquin L.Fernandez, Richard Goodwin, Sven Koenig, Joseph O'Sullivan, "Article on Lesson Learned from Xavier", IEEE Robotics & Automation Magazine, (2000), Vol.7, No.2, pp: 33-39.
8. Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction: 2nd Edition, Europe, Prentice Hall, 1998.
9. Carlos A.V, Caroline C.H., "Providing Data on the Web: From Examples to Programs", In Second International World Wide Web Conference, Chicago, (1994), <http://fiaker.ncsa.uiuc.edu:8080/www94-2/paper.html>.
10. John Crawford, "Socket Chat: Developer's Corner Journal", (1995); <http://www.dcjournal.com>.
11. D J Roberts and P M Sharkey, "A Real-time, Predictive for Distributed Virtual Reality", In Proc. 1st Conf. Simulation and Interaction in Virtual Environment, (1995), Vol.1: pp 72-81.

12. P.Fiorini, and R.Oboe, "Internet-Based Telerobotics: Problems and Approach", In Proc. 8th International Conference on Advanced Robotics (Icar'97), Monterey, CA, (1997), Vol. 1: pp 765-770.
13. C.Calvet et.al, Borland C++ Builder 3: Unleashed, United States: Sams Publishing, 1998.

USER INTERFACE DESIGN FOR INTERNET-BASED TELEROBOTICS SYSTEM

Shamsudin H. M. Amin

(sham@suria.fke.utm.my)

Rosbi Mamat (rosbi@suria.fke.utm.my)

Mohamad Fauzi Zakaria

(fauzi@nadi.fke.utm.my)

Center for Artificial Intelligence and Robotics
(CAIRO)

Faculty of Electrical Engineering

Universiti Teknologi Malaysia

81310 UTM Skudai

Johor Darul Ta'zim, Malaysia

Fax: +607 5566272

Imre J. Rudas

(Rudas@Zeus.banki.hu)

Laszlo Horvath

(Lhorvath@Zeus.banki.hu)

Josze Tar

(Jktar@Zeus.banki.hu)

Budapest Polytechnic

H-1081 Budapest Nepszinhaz u.8

Hungary

Tel. 36 1 333-4513

Fax: 36 1 333-9183

Abstract ? One of the most important components of any telerobotics system is the user interface, as it determines the extent to which the user can sense the remote environment and consequently control the robot. The display in the user interface should be designed so that the user receives sufficient information about the remote environment. The controller part in the user interface has to be designed such that the user can effectively control the robot. This paper presents ways to design the user interface for controlling the robot via internet by defining the user operator, task and environment requirement. Besides, the problem in the internet also need to be considered before the user interface system is developed. Lastly, this paper also describe the implementation of development of user interface in Universiti Teknologi Malaysia (UTM) Telerobot.

Keywords : Internet-based telerobotics, user interface, user operator.

I. INTRODUCTION

Internet-based Telerobotics is a system that accepts instructions from a distance at anywhere in the world, generally from a trained user operator. The user operator thus performs live actions in a distant environment and through sensors can gauge the consequences [1]. The basic telerobotic system to be launched in the Internet normally has a robot system, a camera and a personal computer [2]. System architecture of this telerobotic system is shown below:

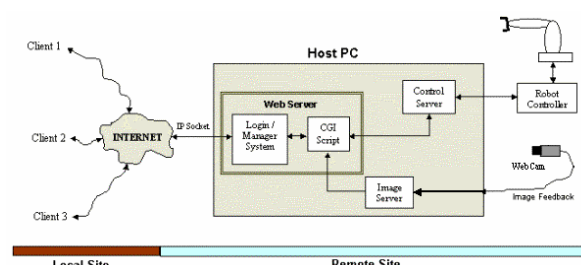


Fig. 1: System Architecture Design

II. DESIGN APPROACH

User interface design is a very difficult business. It combines two awkward disciplines [3]: psychology and computer science. These disciplines have very different cultural backgrounds: psychology is concerned with people; computer science with computer machinery. Psychologists are supposedly sympathetic and understanding; computer scientists are supposedly mathematical and precise. Psychologists have enough trouble understanding people even when they are not using computers; computer scientists have enough trouble getting programs to work even when they are not being used by people. Good user interface design requires these two perspectives to be united.

Defining Requirement

Before designing any telerobotic system, it is crucial to define several functional requirements. Those are:

- ? Who are the *users* of the specific application and what are their experiences, aptitudes, motivations, and needs?
- ? What is the *task* for the system and what is required to do it?
- ? What is the *environment* in which the application will be used, and what is the context in which the task will be done?

Defining the user, task, and environment is essential in specifying appropriate technology for user operator interaction in general and in creating usable systems [4].

User experience is divided to novice or expert. In teleoperation, the concern for novices is generally ease of first time use, clarity of what the user can and can't do, and recoverability from error. For the expert user, more focus tends to be on the "power" the system provides: high functionality, speed which users can accomplish routine tasks, and flexibility of the systems to accommodate the needs of expert users (e.g., to let users customise the way an interaction is accomplished). In addition, expert users might spend a great deal of time with a system or use it for very demanding tasks.

In order to develop the teleoperation system, we should identify the goals of the application, the tasks that will be required to achieve those goals, and how the tasks will be accomplished in the teleoperation. Telerobotics devices are typically developed for situations or environments that are too dangerous, uncomfortable, limiting, repetitive or costly for humans to perform [5]. Some applications or tasks are listed below in different of environments:

- ✍ Underwater: inspection, maintenance, construction, mining, exploration, search and recovery, science, surveying.
- ✍ Space: assembly, maintenance, exploration, manufacturing, science.
- ✍ Resource industry: forestry, farming, mining, power line maintenance.
- ✍ Process control plants: nuclear, chemical etc., involving operation, maintenance, decommissioning, and emergency.
- ✍ Military: operation in the air, undersea and on land.
- ✍ Medical: patient transport, disability aids, surgery, monitoring, remote treatment.
- ✍ Construction: earth moving, building construction, building and structure inspection, cleaning and maintenance.
- ✍ Civil security: protection and security, fire fighting, police work, bomb disposal.
- ✍ Education and entertainment.

Design Consideration

Based on the experiences in accessing telerobotics websites and literature review, there are two problems that affect the internet-based telerobotics performance that should be solved. These problems are related to time delay response and operator's skill or behaviour. The time delay response occurs when transmitting real time visual feedback and control command to the client-server. To solve these problems, the size of visual image has to be reduced by choosing the compressed JPEG (Joint Photographic Experts Group) file [6][7] and the part of the software that controls the hardware should be created as a plug-in component package [8].

Whereas, to solve the user operator's problem who is unfamiliar (inexperienced) with the system, there is a need to create a security (safety) system in user interface. This is especially to limit the robot workspace so that the damage to the manipulator or other objects in the task space may be avoided. Another problem that will occur is when many users try to access the site at the same time. In that case a database system may be used to arrange the user list that only one should be allowed to control the system on the particular time.

III. WEB AND APPLICATION INTERFACE

There are two ways the user interface can be launched in the internet by using a web or application interface. The web is designed as a hyper-text distributed information storage system for technical documentation [9]. Data is stored at many servers, and can be accessed by many clients, seemingly simultaneously. The client programs used by people are usually referred to as Web browsers because they allow a user to explore inter-related data on different topics. Whereas, application interface is using client-server model. The client-server model provides a convenient way to interconnect programs that are distributed efficiently across different locations.

Web-based interface

A web-based interface is usually a platform independent hypertext mark-up language (HTML) form that is coordinated with a server side common gateway interface (CGI) program [10]. The CGI program, in turn, controls the robot. Web browser forms allow the designer to distribute the interface in a platform independent manner with little or no programming on the interface side. The HTML language contains several different window system components that mimic some standard user interface components. The interface is designed in a manner easily understood by users familiar with such environments.

The bulk of the processing behind an HTML interface is handled by the CGI program on the server side. These programs can involve sophisticated access control subsystems and routines, which will decode the interface input (motion and other commands) and generally pass them on to the actual control programs for the robot.

This method is currently a popular choice for existing Internet-based robotics because interfaces can be created easily and because there are multiple platforms to which it can be distributed. However, it suffers from the "set-submit" cycle, it has potentially wide security loopholes, and it raises the challenge of controlling concurrent users accessing multiple copies of the CGI server programs. The example of web-based interface is shown in Fig. 2.

Application Interface

With an application interface (for example please refer to Fig. 3), the user interacts directly to an executing program. The program is written and compiled to a specific hardware platform and utilises the communications capabilities of the platform to connect to a server program. The interface program can then be

released to users having the same hardware and operating system platform. This approach, therefore, only benefits those users with the same platform.

Applications generally require a greater effort to design and code the interface when compared to HTML forms, but they benefit from the ability to be more complex,

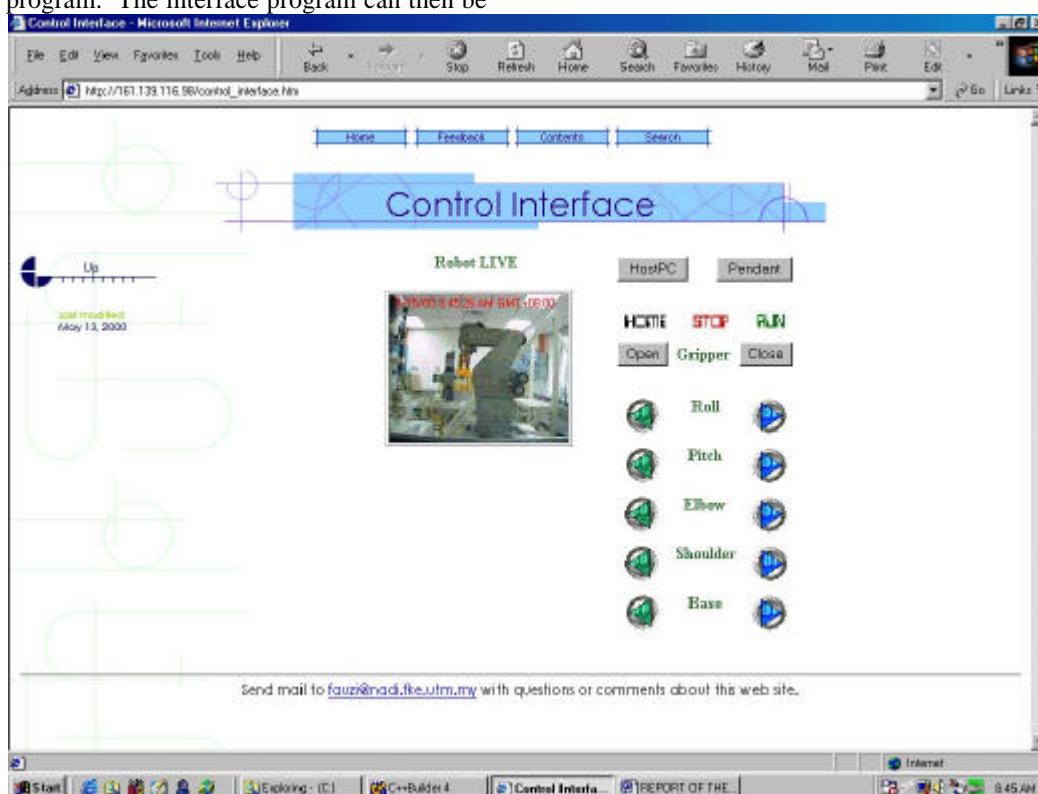


Fig. 2: UTM Telerobot web-based interface

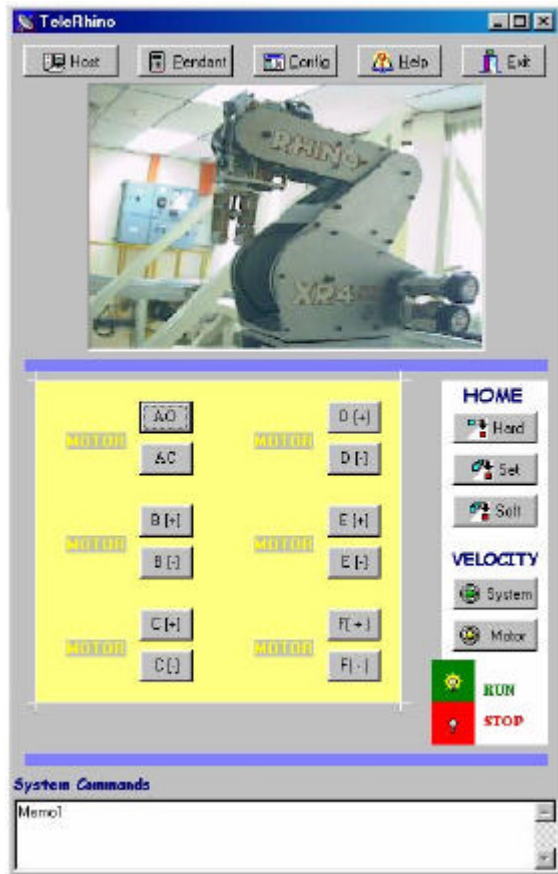


Fig. 3: UTM Telerobot application-based interface

supporting interactive tasks in a decidedly improved fashion. Applications have the added benefit of distributing the processing. The client-side application deals with the interface and interaction with the user, and the server-side controls the robot. Many of the range checking and manipulation limits can be implemented in the interface application, relinquishing the server program from these duties.

IV. IMPLEMENTATION

The UTM Telerobot system basically consists of three main hardware systems that must be integrated. These systems are robot system, camera system and host computer system. Robot system includes robot controller and its arm. The robot is a fixed base Rhino XR-4 robot that has five degree of freedom (5 DOF) and a gripper. The camera system is a webcam (Logitech Quickcam Pro) type that is used to capture the entire robot environment. These systems must be integrated together before being launched to Internet by programming in host computer. System architecture of this telerobotic system is shown in Fig. 1. Actually this system is built to perform simple tasks such as to move a small plate of steel. The target of application is in education and training sectors. Many research and educational institutions cannot afford to purchase industrial robots, mainly because they are very

expensive. By introducing Internet-based Telerobotics system, it is a chance to expose to any users especially students in Malaysia on robotics area.

UTM Telerobot User Interface

User interface system in UTM telerobot basically consists of three basic services as shown in Fig. 4. These three services are login service, robot guidance (control) service and visual feedback service. The login service, provides communication with the other services, and allows the system manager to get information about established connection. This part is important to enable the system manager to arrange the priority user to control the robot by following the database system. The second service, robot guidance (control) service allows the user to send high-level commands to the server, where a Common Gateway Interface (CGI) script decodes and builds the corresponding order for the robot. The CGI is a standard way for the Web server (HTTPD etc) to run and talk to other programs on the remote computer. The last one is video feedback service is a part to allow feedback from different video cameras. The users can view the status of robot image.

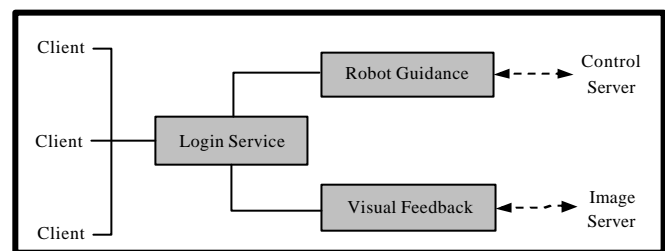


Fig. 4: Schematics of the User Interface

Robot Guidance

Actually, this telerobot system is using the point-to-point controller type. So, the program should be designed following the controller specifications. In this system, C++ programming language was used to program the robot control system. In order for Host PC to talk with the robot controller, the communication link and protocol must be established. This must be done each time a program run. An open serial communication program is shown below [11][12].

```
DCB dcbCommPort;
hComm = CreateFile("COM1",
    GENERIC_READ | GENERIC_WRITE,
    0, // exclusive access
    NULL, // no security
    OPEN_EXISTING,
    0, // no overlapped I/O
    NULL); //null template
```

```
SetupComm(hComm, 128, 128);
```

```
GetCommTimeouts(hComm, &ctmoOld);
ctmoNew.ReadTotalTimeoutConstant = 100;
```

```

ctmoNew.ReadTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutConstant = 0;
SetCommTimeouts(hComm, &ctmoNew);

dcbCommPort.DCBlength = sizeof(DCB);
GetCommState(hComm, &dcbCommPort);
BuildCommDCB("9600,N,8,1", &dcbCommPort);
SetCommState(hComm, &dcbCommPort);

```

To send data or commands to the serial port the *WriteFile* call is used. For example, the following call sends "GO" (open gripper) command to the controller:

```
WriteFile(hComm, "GO\r", 3, &lpNumber, NULL);
```

If a command sent to the controller is a responsive command, that is, one that results in data being sent back to the host, the data is retrieved using the *ReadFile* call.

Camera Image Programming

Live image from camera (webcam) is a robot movement feedback. Therefore the programming of camera is very important to capture a live image. Normally, the web cam camera can capture the image up to 30 frames per second (fps) based on image size, resolution and computer system. This image feedback was developed using the AVICap window class that is programmed in C++. AVICap provides applications with a simple, to view a live incoming video signal by using the overlay or preview methods.

The Robot Control Step

To control the robot, a user needs to follow the control flow shown in Fig. 5. Firstly, the user needs to go to the UTM Telerobot website at <http://161.139.116.98>. He or she must understand the condition and rules given by the Webmaster and then must register before being allowed to control the robot. Second step, the user must enter the password into the Password Form and then the password will be processed. After that, the user operator will get the result either he will be able to control and view the system or just view only the system handled by system manager (software). To control the system, only one user is accepted and the others just view the status and image until the first user quit or reach maximum limit time (10 minutes). After that, the second user will substitute as first user. If there is only one user accessing the telerobot web, the system manager will give permission to that user to control a robot, as he or she likes until other users come in.

Result

UTM Telerobot Graphics User Interface (GUI) was developed using Hyper Text Mark-up Language (HTML) and C++ Builder. The C++ builder 4 Web Broker Technology allows developer to build CGI Web server applications without having to worry about too

many low-level details. The Robot Control Panel was launched to web page (internet) after it is programmed in CGI Web Broker. HTML was used to integrate CGI – Robot Control and Camera Live Image page to one web page by using FRAME and IFRAME technology. The GUI for this basic telerobot is shown in Fig. 2 and is working successfully.

V. CONCLUSION

The aims of internet-based telerobotics are to control the robot at a distance, at any time and to allow for a good feedback response. So, the user interface design is an important part to implement in an internet-based telerobotics system. This paper has described the methodology in design of user interface system and development of basic telerobotics system. The user interface system has been developed by using C++ programming language to program the robot guidance (control) and visual feedback. Both of these systems are combined together and produced in World Wide Web (WWW) by using HTML technology.

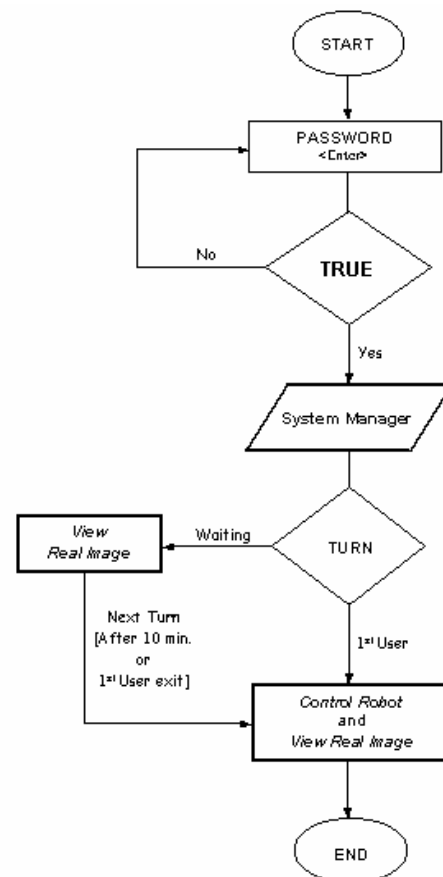


Fig. 5: Steps to Control the Telerobotics System

VI. ACKNOWLEDGEMENTS

The authors would like to thank the Malaysian Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

VII. REFERENCES

- [1] K. Goldberg (ed.). "The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet". The MIT Press, Massachusetts Institute of Technology, Cambridge. 1999.
- [2] Mohamad Fauzi Zakaria, Shamsudin H.M. Amin, Rosbi Mamat. "Design and Development of Control System for Internet-based Telerobotics". Proc. of TENCON 2000 Vol. II, p. 338-342, 2000.
- [3] H. Thimbleby. "User Interface Design". ACM Press, New York. 1990.
- [4] R. Stuart. "The Design of Virtual Environments". McGraw-Hill, New York. 1996.
- [5] "Virtual reality: Scientific and Technological Challenges". Computer Science and Telecommunications Board Press. 1994.
- [6] K. Taylor and B. Dalton. "Issues in Internet Telerobotics". Proceeding of FSR'97 International Conference on Field and Service Robotics, The Australian National University, Canberra, Australia. 8-10 December 1997.
- [7] R. Siegwart, C. Wannaz, P. Garcia, R. Blank. "Guiding Mobile Robots through the Web". Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '98), p. 1-6, 1998.
- [8] K.P. Leu, M.H. Ang and Y.S. Wong. "A Telemanufacturing Workcell Over the Internet". Proc. SPIE Vol. 3524, Telemanipulator and Telepresence Technologies V, Paper No. 32, 1998.
- [9] K. Taylor and J. Trevelyan. "A Telerobot On The World Wide Web". National Conference of the Australian Robot Association, Melbourne, Australia. 1995.
- [10] P. DePasquale, J. Lewis, M. Stein. "A Java Interface for Asserting Interface Telerobotic Control". Proc. SPIE Vol. 3206, Telemanipulator and Telepresence Technologies IV, Paper No. 19, 1997.
- [11] J. Miano, T. Cabanski and H. Howe. "Borland C++ Builder How-To". Waite Group Press, United State of America. 1997.
- [12] K. Reisdorph et al, "Borland C++Builder 4 Unleashed". Sams Publishing, United State of America. 1999.

Design and Development of Control System for Internet-Based Telerobotics

Mohamad Fauzi Zakaria
fauzi@nadi.fke.utm.my

Shamsudin H.M. Amin
sham@suria.fke.utm.my

Rosbi Mamat
rosbi@suria.fke.utm.my

Center for Artificial Intelligence and Robotics (CAIRO)
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai
Johor Darul Ta'zim, Malaysia
Fax: +607 5566272

Abstract: This paper presents ways to design the basic development of control system for a prototype internet-based telerobotics using a fixed type robot by considering the philosophy of design. There are three issues should be considered before telerobot control system is developed. They are operation safety and error issues, response time issues and continuous control issues. After that, the telerobotics system is developed depend on the design approach, system architecture and control scheme. In this project, robot control system has developed by using C++ programming language, whereas Java programming language has been used to program the camera image feedback. Both of this system are combined together by using HTML technology. Finally, this telerobot system has been successful done and allowed the user to manipulate a robotics arm, through the web browser interface, to perform simple tasks such as moving small plate of steel.

Keywords

Internet, Internet-based Telerobotics, Control System, Telerobot System.

I. INTRODUCTION

The Internet has become the most important network for communication and the biggest data storage. It connects a million of computers all over the world giving access to communication, data, pictures, videos and even real time images of distant environments.

There are several factors that make the Internet an attractive medium for teleoperation applications [1]. Firstly, *the Internet has an extensive geographical reach*. An estimated 147 million people and 9.5 million machines are now plugged into the Internet, with the figures doubling or tripling every year. Teleoperated devices can be controlled and operated from any part of this global network of computers.

Secondly, *the Internet is network and platform independent*. This enables computers of different hardware and operating system platforms to be connected and communicate with each other over different kinds of

network and physical links. This widens teleoperations development on any hardware and software platform to be shared and accessed by a significantly larger audience of computers.

Thirdly, *the Internet is standards-based and open*. For instance, standards such as HTTP and CGI simplify the development of online applications while HTML provides a means of creating consistent and open interfaces. Teleoperation application can be developed with reduced time and effort and be accessed easily from anywhere on the Internet through standard interfaces such as the Web.

Finally, *a wave of technological development, from high bandwidth networks to new software technologies, is revolutionizing the Internet*. These developments are alleviating constraints and enhancing the capabilities of Internet-based teleoperations (telerobotics).

As a result, telerobotics system, which uses Internet as a platform, can easily to allow operators to relocate at a little cost and at anywhere. The telerobotics systems are applicable to different tasks in education (training), telemanufacturing, entertainment and hazardous environments.

II. DESIGN APPROACH

This paper focused on the basic development of the Internet-based telerobotics prototype system using a fixed type robot. The basic telerobot system to be launched in the internet, normally has a have a robot system, a camera and a personal computer (PC). Therefore, to integrate and to design the control system programming; the following issues should be considered.

A. Operation Safety and Error Issues

One of the biggest concerns during teleoperation is that the manipulator should not collide with other objects in task space, to avoid damage to the manipulator or other objects in the task space [2]. Also, damage might occur when the manipulator end-effector moves with very high velocity near its singularity points. While in contact with other objects, the manipulator should not exert excessive forces on them, though this problem is diminished when

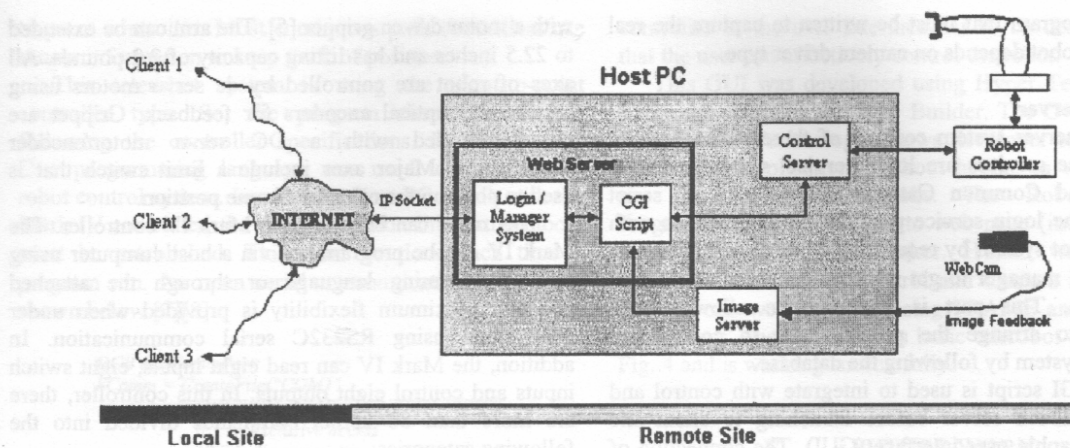


Fig. 1: System Architecture Design

the operator receives feedback of forces being applied by the end-effector, or when the manipulator is compliant.

Teleoperation errors involve undesirable manipulations, such as mis-positioning of the end-effector with respect to other objects, a hit and miss trial approach by the operator to achieve appropriate contact between the manipulator and object, and slipping of an object which has been gripped by the manipulator. Some of these teleoperation errors may be attributed to human error, such as mistakes and slips. Mistakes occur due to incorrect interpretation of the remote environment or the task, or when an incorrect manipulation sequence is selected even with correct interpretation of the environment. Slips are accidental movements of the manipulator caused by the operator, while other errors could be attributable to the fact that the user does not get sufficient feedback from the remote site.

B. Response Time Issues

While an operator is waiting for a response from the telerobot they are unable to plan/submit the next request. To minimise this waiting period the response time should be as small as possible. Response time t_r is defined as [3]:

$$t_r = t_p + \frac{(D_s + D_r)}{v_l} + t_c$$

Where t_p is request processing time, t_c is time taken to initialise communication (approximately 1 second), D_s and D_r are total data submitted and returned, and v_l is the transmission speed of the link. As a server of web content there is no control over the speed of the link, therefore minimum response time can only be achieved by fast processing of a request and minimum transmission of data. For a typical robot request 8 seconds is spent processing (a large portion of this being dead time while the robot moves) while data transmission can take over 15 seconds. Hence in this application data reduction is likely to be more beneficial than reducing response time. One way of doing this is by minimising image sizes presented to the operator. Further reduction in data size can be achieved by minimising data transmissions to robot controller in remote site. The part of the software that

controls the hardware should be created as a plug-in component package. Plug-in contains binary codes that implement a defined set of functions.

C. Continuous Control Issues

The user should have continuous control over the robot and receive continuous feedback about the remote environment. The update rate of the transmitted video images should be as fluent as possible to provide a good feeling for reality. In the other hand, the control system should be designed to handle the problem when having many users accessing at the same time. It is important to allow only one user on continuous control at a particular time.

III. SYSTEM OVERVIEW

The telerobot system basically consists of three main hardware systems that must be integrated. These systems are robot system, camera system and host computer system. Robot system includes robot controller and its arm. The robot is a fixed base Rhino XR-4 robot that has five degree of freedom (5 DOF) and a gripper. The camera system is a web cam (Logitech Quickcam Pro) type that is used to capture the entire robot environment. These systems must be integrated together before being launched to Internet by programming in host computer (Pentium III). System architecture of this telerobot system is shown in Fig. 1 and its explanation is given below.

A. Control Server

The Control Server is a place that handles instructions and feedbacks to/from robot controller. The instructions command should be sent to controller via serial communication port if we want to move the arm and know the position of robot.

B. Image Server

Visual feedback server means a place that controls the camera images feedback before launching Internet server.

So, the program that must be written to capture the real image of robot depends on camera driver type.

C. Web Server

The web server system consists of three basic services. These three services are login service, system manager service and Common Gateway Interface (CGI) script service. The login service provides communication with the telerobot system by requesting a password and allows the system manager to get information about established connection. This part is important to allow system manager to arrange the priority user to control the telerobot system by following the database.

The CGI script is used to integrate with control and visual feedback server before launching to client site through graphic user interface (GUI). The connection of this system into Internet is using Transmission Control Protocol and Internet Protocol (TCP/IP). TCP/IP is a software-based communications protocol and it handles errors in transmission, manages the routing, the delivery of data, and control the actual transmission by the use of predetermined status signals [4].

IV. TELEROBOTICS CONTROL SYSTEM

A. Control Scheme

Control system for this telerobot is classified and shown in Fig. 2 as closed-loop system. The closed loop systems are more accurate since they can detect any error in the output and adjust for it. The user at Client PC is able to submit individual or a sequence of moves to the robot by submitting the instructions command to Server PC. The Server PC will send or modify commands if they exceed the robot's workspace to robot controller. Users can view the latest position and live image as a robot feedback action. If no action is shown in live image action, the user must send again the command. The live image from camera is also as a feedback to determine that the robot moves or not.

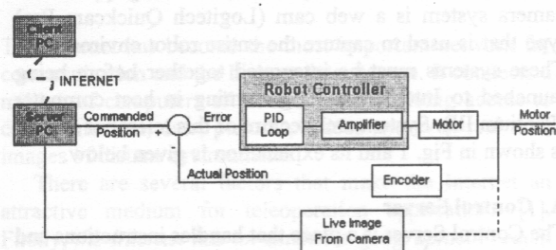


Fig. 2: Closed-loop Control System

B. Robot and Controller Description

This telerobot is using XR-4 series Robot Arm produced by Rhino Robots Incorporation, United State of America. The XR-4 arm is a five axis revolute coordinate robot arm

with a motor driven gripper [5]. The arm can be extended to 22.5 inches and has lifting capacity of 2.2 pounds. All axes of robot are controlled by dc servo motors using incremental optical encoders for feedback. Gripper are also controlled with a DC servo motor/encoder combination. Major axes include a limit switch that is used to position the robot to a home position.

This robot can be driven by Mark IV controller. The Mark IV can be programmed via a host computer using any programming language or through the attached pendant. Maximum flexibility is provided when under host mode using RS232C serial communication. In addition, the Mark IV can read eight inputs, eight switch inputs and control eight outputs. In this controller, there are more than 80 kernel commands divided into the following categories:

- System functions
- Motor functions
- Gain functions
- Teach Pendant functions
- Configuration functions
- Input/Output functions

C. Control System Programming

The robot needs programs for its controller to execute. Similarly to telerobot system, it needs programs to control and produce the control panel interface to the web (internet). There are three parts to be programmed namely robot control program, camera image program and graphics user interface (GUI) which would be developed by following the design approach.

Robot Control Programming

Robot control programming depends on the controller capabilities and styles. At present, there are three styles of robot controllers: the limit-sequence controller, the point-to-point controller and the continuous-path controller. Each has unique programming requirements [6]. The limited-sequence controller is used on the limited-sequence (or bang-bang) robot. The simplest bang-bang robots are pneumatic-powered and have a mechanical controller. Their application programming is done by mechanical set-up, such as using a motor to turn a timing drum. More modern this robot operating systems use electronics circuit and read only memory (ROM). Whether electronic or mechanical, the operating system translates applications program commands into pneumatic value activations.

Whereas, the point-to-point controller uses some type of electronics memory to record of number of positions for the manipulator. Each position represents a value for each axis and each feedback sensor for the robot. It must also support control panels and teaching pendants. The last type of controller is continuous-path. This controller must be able to record and play back the robot's position many times each second. This requires an electronic memory many times larger than that possessed by a point-

to-point controller. Most continuous path controllers use sensor information to keep track of position.

This telerobot system is using the point-to-point controller type. So, the program should be design following the controller specifications. In this system, C++ programming language was used to program the robot control system. In order for Host PC to talk with the Mark IV controller, the communication link and protocol must be established. This must be done each time a program run. An open serial communication program is shown below [7][8].

```
DCB dcbCommPort;
hComm = CreateFile("COM1",
    GENERIC_READ | GENERIC_WRITE,
    0, // exclusive access
    NULL, // no security
    OPEN_EXISTING,
    0, // no overlapped I/O
    NULL); //null template
```

```
SetupComm(hComm, 128, 128);
```

```
GetCommTimeouts(hComm, &ctmoOld);
ctmoNew.ReadTotalTimeoutConstant = 100;
ctmoNew.ReadTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutConstant = 0;
SetCommTimeouts(hComm, &ctmoNew);
```

```
dcbCommPort.DCBlength = sizeof(DCB);
GetCommState(hComm, &dcbCommPort);
BuildCommDCB("9600,N,8,1", &dcbCommPort);
SetCommState(hComm, &dcbCommPort);
```

To send data/commands to the serial port the *WriteFile* call is used. For example, the following call sends "GO" (open gripper) command to the Mark IV:

```
WriteFile(hComm, "GO\r", 3, &lpNumber, NULL);
```

If a command sent to the Mark IV is a responsive command, that is, one that results in data being sent back to the host, the data is retrieved using the *ReadFile* call.

Camera Image Programming

Live image from camera (web cam) is a robot movement feedback. We can know that robot have moved or not. Therefore the programming of camera is very important to capture live image. Normally, the web cam camera can capture the image up to 30 frame per second (fps) based on image size, resolution and computer system. This image feedback was developed using Java programming language. Java is an ideal development tool for web cam applications because it has an applet for easy changing of the image [9].

Graphics User Interface (GUI)

One of the most important components of any telerobotics system is the user interface, as it determines the extent to which the user can sense the remote environment and consequently control the manipulator. The display in the user interface should be designed so that the user receives sufficient information about the remote environment. The

controller in the user interface has to be designed such that the user can effectively control the robot.

This GUI was developed using Hyper Text Mark-up Language (HTML) and C++ Builder. The C++ builder 4 Web Broker Technology allows developer to built CGI Web server applications without having to worry about too many low-level details. The Robot Control Programming was launched to web page (internet) after program in CGI Web Broker. HTML was used to integrate CGI – Robot Control and Camera Live Image page to one web page by using FRAME and IFRAME technology. The GUI for this basic telerobot is shown in Fig. 4 and is working successfully.

V. FUTURE WORK

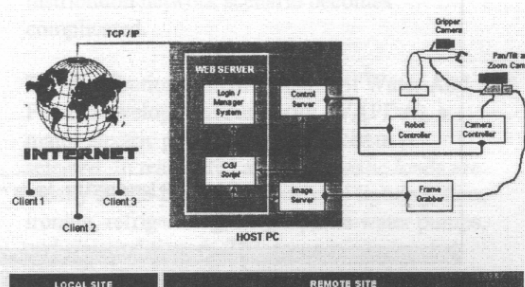


Fig. 3: New System Architecture Design

There are several parts in which the internet-based telerobot will be improved and developed:

1. The old system will be change to the new system architecture (Fig. 3) by adding the pan/tilt/ zoom camera used to view entire robot environment and a web cam will be attached on gripper. So that, the control system for camera movement will be developed.
2. The robot control package will be improved for better performance of time delay response.
3. The program as continuous-path controller will be built to record and play back the robot's position movement.
4. New GUI elements should be created to improve the control over the robot arm by considering the robot and camera control panel, zoom panel and record of movement panel.

VI. CONCLUSION

This paper has described a basic design approach for telerobotics control system, general system architecture and development of basic telerobotics system especially its control system. The major task in this project is the programming part in accordance with the hardware's specifications and the performance of programming language. Robot control system has been developed by using C++ programming language, whereas Java programming language has been used to program the

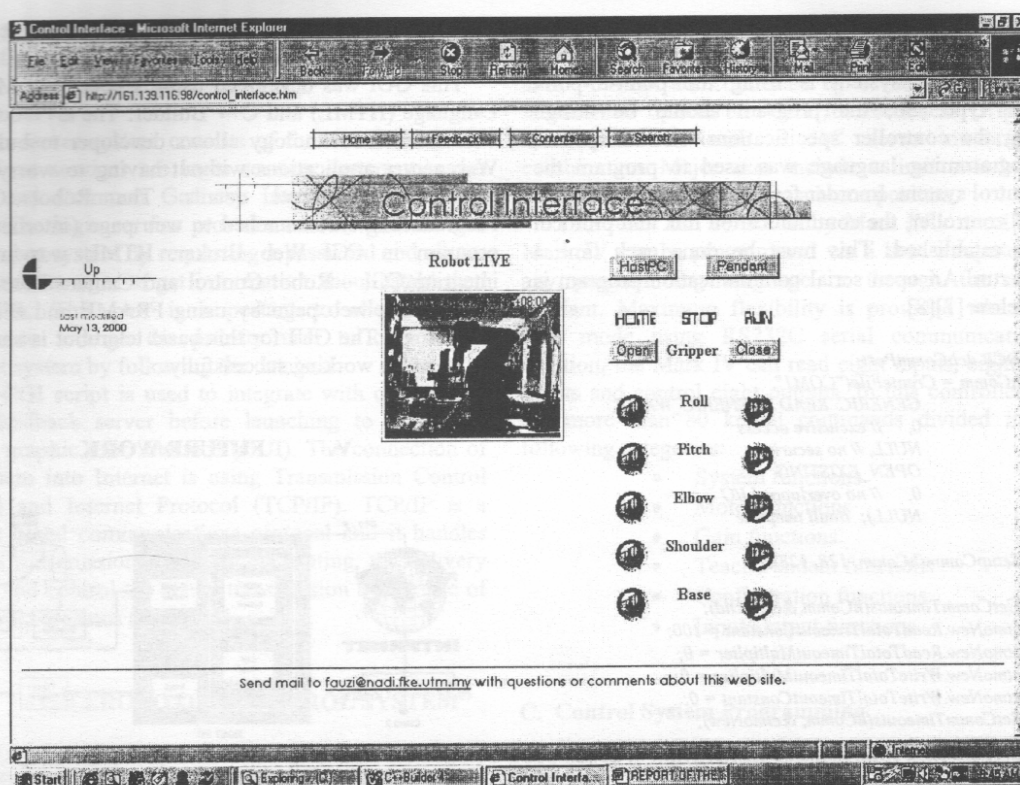


Fig. 4: Graphics User Interface

camera image feedback. Both of these systems are combined together and produced in World Wide Web (WWW) by using HTML technology. The future work development also has been described to improve the telerobot system.

VII. ACKNOWLEDGEMENTS

The authors would like to thank the Malaysian Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

VIII. REFERENCES

- [1] K.P. Leu, M.H. Ang and Y.S. Wong. "A Telemanufacturing Workcell Over the Internet". Proc. SPIE Vol. 35247, Telemanipulator and Telepresence Technologies V, p. 230-23, 1998.
- [2] Anu Rastogi, Design of an Interface for Teleoperation in Unstructured Environments using Augmented Reality Displays, Canada : University of Toronto. 1996.
- [3] K. Taylor and B. Dalton. "Issues in Internet telerobotics". In International Conference on Field and Service Robotics (FSR 97), p. 151-157, Canberra, Australia, 8-10 December 1997.
- [4] S. Kuppaswami, Marie Stanislas Ashok, Paul Rodrigues and K. Palanivel. "Design Patterns On TCP/IP". Proceeding of International Conference on Robotics, Vision and Parallel Processing for Automation (ROVPIA'99), p. 436-445, 1999.
- [5] Mark IV 8 Axis Controller Owner's Manual Book Version 4.01, United State of America: Rhino Robots Inc. August, 1990.
- [6] J. L. Fuller, Robotics: Introduction, Programming and Projects, New York: Macmillan Publishing Company. 1991.
- [7] J. Miano, T. Cabanski and H. Howe, Borland C++ Builder How-To, United State of America: Waite Group Press. 1997.
- [8] K. Reisdorph et al, Borland C++ Builder 4 Unleashed, United State of America: Sams Publishing. 1999.
- [9] J.Dwight and M.Erwin, Special Edition Using CGI, United State of America: Que Corporation. 1996.

Internet-based Telerobotics: UTM's Experience and Future Direction

Shamsudin H. M. Amin ¹
Rosbi Mamat
Mohamad Fauzi Zakaria
Norhayati A. Majid
Lim Cheng Siong

Center for Artificial Intelligence and Robotics (CAIRO)
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai
Johor Darul Takzim, Malaysia
Tel: +607 557 6160
Fax: +607 556 6272

¹ sham@suria.fke.utm.my

Imre J. Rudas ²
Laszlo Horvath
Joszef Tar

Budapest Polytechnic
H-1081 Budapest Nepszinhaz u. 8
Hungary
Tel. 36 1 333-4513
Fax: 36 1 333-9183

² Rudas@Zeus.banki.hu

Abstract

This paper reports on the development of the Universiti Teknologi Malaysia's (UTM) Internet telerobotics systems. The design can be categorised into three phases. In the first phase, the leg of a mobile are tested and controlled through local internet connection. The task is called robot-oriented control system. The second phase of the project is the implementation of the web-based robot using Rhino robot and the last phase is future works on task-oriented of Rhino robot through the Internet.

Keywords : Internet-based telerobotics, robot-oriented, task-oriented.

1. Introduction

The goal of our project in the telerobotics area is to discover and develop the system by combining network technology with capabilities of robots. Using Internet technology for telerobotic application offers the advantage of low-cost deployment. There is no longer a requirement for expensive purpose built equipment at each operator's location. Almost every computer connected to the Internet can be used to control a teleoperable device. The downside is the limitation of varying bandwidth and unpredictable time delays [1]. These Internet features should be defined and considered before designing an efficient telerobotic system. Besides that, several functional requirements should also be defined before designing any telerobotic system:

- ? Who are the *users* of the specific application and what are their experiences, aptitudes, motivations, and needs?

- ? What is the *task* for the system and what is required to do it?
- ? What is the *environment* in which the application will be used, and what is the context in which the task will be done?

Defining the user, task, and environment is essential in specifying appropriate technology for user operator interaction in general and in creating usable systems [2].

In this paper we focus mainly on the overview of our past and recent projects and present some preliminary results.

2. UTM Telerobotics Project

The user interface of telerobot has two ways to be launched in the internet either by using an application or a web interface. Interface using an application usually used a client-server model. The client-server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Whereas, web is designed as a hyper-text distributed information storage system for technical documentation [3]. Data is stored in many servers, and can be accessed by many clients, seemingly simultaneously. The client programs used are usually referred to as Web browsers because they allow a user to explore inter-related data on different topics.

2.1 Non-web-based Telerobotics System

Our early telerobotics system used an application interface which is based on client-server model [4]. We call this non-web-based telerobotics system. This basic interface is designed using Borland C++ Builder, an

object-oriented programming environment which provides a Visual Component Library which is needed to generate the graphical user interface for this interface. Data is transfer using windows sockets element.

System overview

The architecture of the basic system is shown in Fig. 1. The hardware set-up consists of a leg of a mobile robot, which has three joints that move in three degrees of freedom. The leg is controlled by a PC which also acts as a local server. The motor drivers and interfacing electronics are connected to the PC through the parallel port. Every single command that is send to the server interface will invoke the server to perform a certain task to the robot. That is why we call this task as robot-oriented.

The computer is also connected to a main server which has many computers connected to it. Other computers can act as clients and can download the interface and control the motion of the leg of the mobile robot through the local area network. When the system goes to the public network, other users from anywhere can control the robot. Windows sockets provide connections based on the TCP/IP protocol.

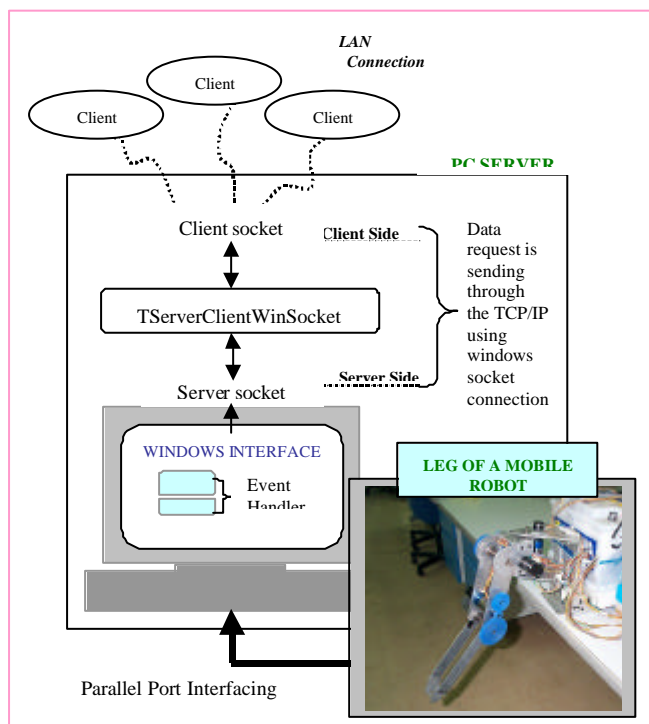


Fig1. : System Architecture for non-web-based

Graphical user interface

One of the most important components of any telerobotics system is the user interface. The display in the user interface should be designed sufficiently so that the user receives enough information about the remote environment. The preliminary basic design of the graphical user interface (GUI) is shown in Fig. 2. This GUI consists of a few panels including motor drive controller panel, speed control panel, selection of either to be a client or a server and indicator panel. Image feedback is shown in Fig. 3.

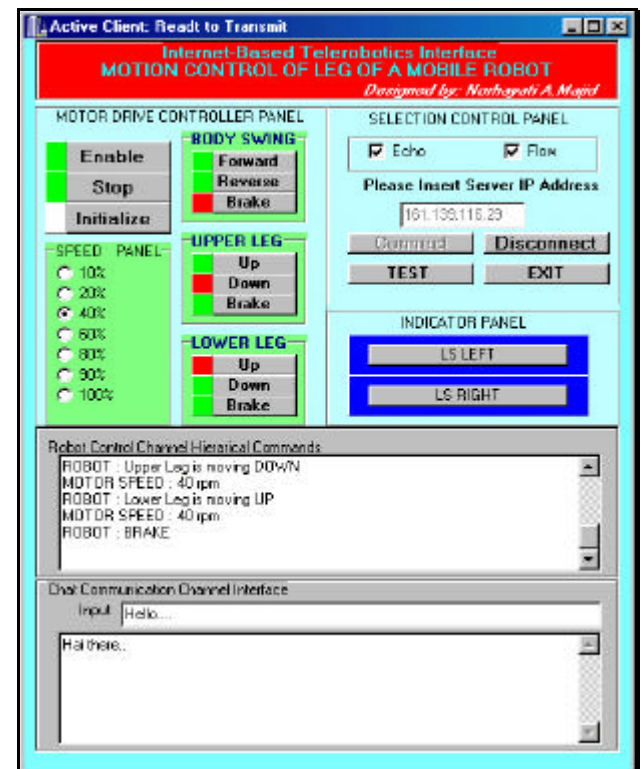


Fig 2: Graphical User Interface for Non-web-based Telerobotics System



Fig. 3: Image Feedback

System achievement

The client can control the robot through the graphical interface and the movement of the robot can be seen through the digital camera that is focused on the robot, which is provided by Microsoft NetMeeting software. In the real situation the client cannot see the robot movement directly because of the delay that always occurs in the Internet application. The system is a robot-oriented system where the client and server will behave like a standalone application. The user must download and execute the application program from the server. The user will be allowed to control the robot after obtaining permission from the server. The interface has been tested locally but is not yet freely available on the Internet yet.

2.2 Web-based Telerobotics System

Our current telerobotics project are based on web interface. A web-based interface is usually a platform independent hypertext mark-up language (HTML) form that is coordinated with a server side common gateway interface (CGI) program [5]. Web browser forms allow the designer to distribute the interface in a platform independent manner with little or no programming application on the interface side. The HTML language contains several different window system components that mimic some standard user interface components. This method is chosen since interfaces can be accessed easily in the web browser and there are multiple platforms in which it can be distributed .

2.2.1 Robots Oriented Interface

Robot-oriented telerobotic is a system that requires the operator to control the robot step by step in implementing a task. In UTM, robot-oriented telerobotics system was built to perform simple tasks such as to move a small plate of steel, which is used in education and entertainment (edutainment) purposes.

System overview

The UTM robot-oriented telerobot system basically consists of three main hardware components that must be integrated. These comsystems are robot system, camera system and host computer system [6]. The robot system includes robot controller and its arm. The robot is a fixed base Rhino XR-4 robot that has five degree of freedom and a gripper. The camera system has two cameras, they are a Sony EVI-D31 pan/tilt/zoom camera type that is used to capture the entire robot environment and a webcam that is attached on the gripper. These systems must be integrated together before being

launched to the Internet by programming in the host computer server. System architecture of this telerobot is shown in Fig. 4 and its explanation is given below:

? *Control server*

The Control Server handles instructions and feedbacks to/from robot controller. The instructions command should be sent to controller via serial communication port if we want to move the arm and know the position of robot.

? *Image server*

Visual feedback server that controls the camera images feedback before launching Internet server. The actual image of robot depends on type of camera used.

? *Web server*

The web server system provides three basic services. These three services are login service, system manager service and Common Gateway Interface (CGI) script service. The login service provides communication with the telerobot system by requesting a password and allows the system manager to get information on established connection. This part is important to allow system manager to schedule the user to control the telerobot system by following the database. The CGI script is used to integrate the control and visual feedback server before launching to the client site through GUI.

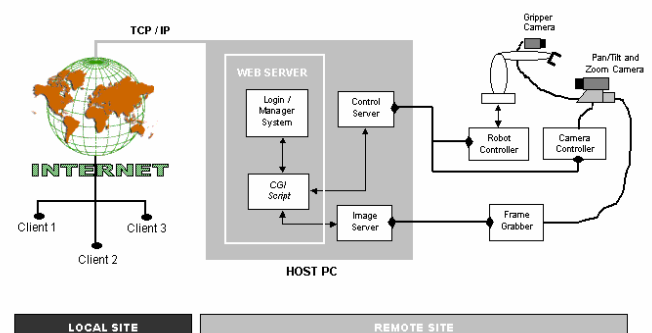


Fig. 4: Web-based Telerobotic System Architecture

Graphical user interface

Robot-oriented GUI shown in Fig. 5 is developed using HTML and C++ Builder. The Robot Control Panel is launched to the web after it is programmed in CGI. HTML was used to integrate CGI – Robot Control and Cameras Live Image page to one web page by using FRAME and IFRAME technologies [7].

System achievement

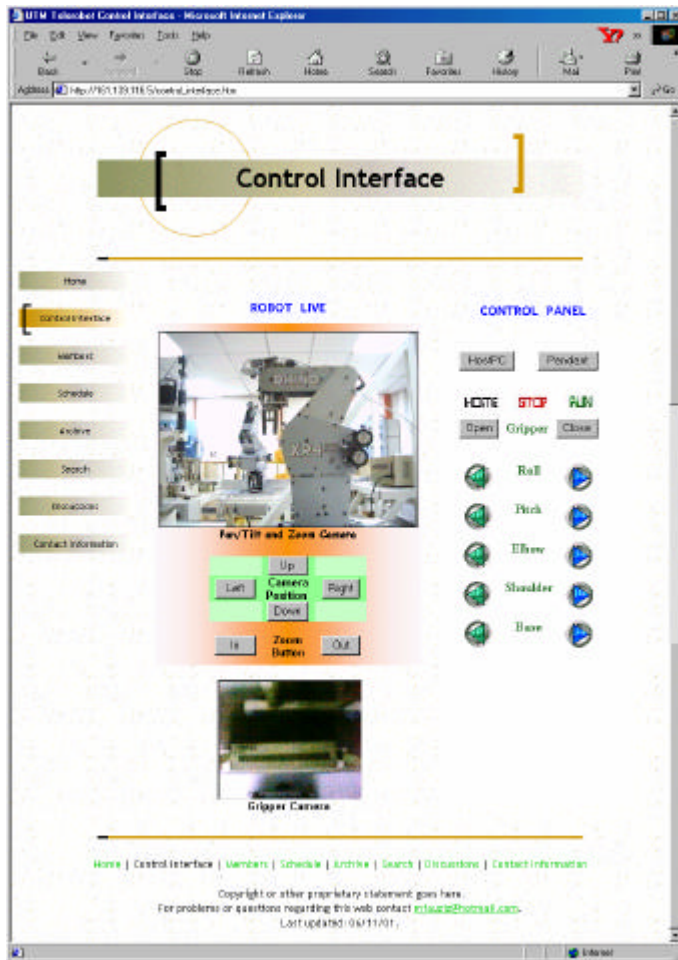


Fig. 5: UTM Telerobot web-based interface

The GUI for this robot-oriented telerobotic system shown in Fig. 5 has been successfully tested on Internet explorer and Netscape navigator web browser. There are some problems especially in internet response time and difficulties to achieve the task target are to be overcome.

2.2.2 Task Oriented Interface

Task-oriented robotic system or so called “task-centric” robotics system requires only the operator to specify the tasks to be done by the system and the system will then plan and carries out a series of action to complete the tasks. In contrast, robot-oriented system will require the operator to plan the actions step by step to get the tasks done. Compared to a robot-oriented system, task-oriented robotic system has higher degree of autonomy. Table 1 shows the comparison between both of the systems.

Advantages of the task-oriented system

The task-oriented internet-based telerobotic system provides better solution to the problems mentioned in previous discussion.

- i) *Easy to operate*
Basically task-oriented robotic system is easier to be operated than robot-oriented system since one task in task-oriented robotic system may equal to a set of commands in robot-oriented system. For example the task to move a cube from one location to another which may require the operator to specify a set of commands to move the various motors in robot-oriented system.
- ii) *Response time*
Certain processes such as command and task pre-processing will be carried out on the client site thus reduce the waiting time for the response from the server. Furthermore the system may carry out the steps in completing the task without delay between the steps compared with robot-oriented system where each step followed must be specified upon completion of the latest command.

Table 1: Comparison between robot-oriented system and task-oriented robotic system

Robot-oriented System	Task-oriented Robotic System
<p>Basic command unit:</p> <p>? Based on robot movement, e.g.:</p> <ol style="list-style-type: none"> a) Arm type robotic system: shoulder up 30°, elbow down 30°, gripper open or spray start; b) Mobile robot: move forward 30 cm, turn left 45°. <p>? Usually, 1 basic command unit equals to 1 robot instruction.</p>	<p>Basic command unit:</p> <p>? Based on the task designed for the robotic system, e.g.:</p> <ol style="list-style-type: none"> a) Welding/spray painting system: spot, straight, arc or follows certain marks/pattern; b) Robotic goods sorting system: transfer objects type A to line A and objects type B to line B; c) Mobile robot: find the target such as heat/light source in unknown environment. <p>? Usually, 1 basic command unit equals to a series of robot instruction.</p>

The system can directly convert the command given to robot instruction since 1 basic command unit equals to 1 robot instruction.	The systems need to have the ability to “understand” the task given (requires task specified method) before the task can be converted to a series of robot instruction.
Human will act as path planner to complete the task such as welding and spray painting.	The task controller will do the path planning once “understand” the task(s) required to be done.
Autonomy level: low.	Autonomy level: higher (with certain limitations).
Low efficiency in completing the work since every steps involved must be manually planned/programmed.	Higher efficiency in completing the work since task controller will do the path planning.
Image capturing system (if involved) usually works merely for visual feedback.	Image capturing system (if involved) works not only for visual feedback but also as part of the vision system.
Less complicated to be designed and developed.	Complicated to be designed and developed especially the task controller.
Suitable application: usually for repeated/routine work especially in mass production.	Suitable application: usually for the work that is not/less repeated or the work with uncertainties such as goods sorting where the objects may vary in size, shape, orientation and location.

System architecture

The system is built based on the task-oriented robotic system concept. The task of the system is to manipulate the cubes in front of the robot. The operator only needs to tell the system what to be done (task) rather than how to do it. The operator can tell the system to move some of the blocks to certain locations as well as the pattern of arrangement. Then the system will plan the path on which cube is to be best moved first than the other as well as how the gripper will move the cube.

Fig. 6 shows the system architecture without providing the web service. The preliminary GUIs design is shown in Fig.9. The system can accept task-oriented command from the operators either through mouse operation or natural language. The command will then be processed by the command preprocessor – either interpreter or parser. The purpose of the command preprocessor is to remove the illegal commands such as spelling mistake, syntax error as well as to limit the mouse operation. Information such as the number of objects as well as the location and orientation of respective object are required by the command preprocessor.

Once the system accepts the command from the operator to complete the task, the task will then be passed to the task preprocessor. The task preprocessor

will do the simulation if the task could be performed by the task controller. This is very important since the system is designed based on task-oriented approach. Apparently not all tasks can be performed by the task controller due to the limitations in the design and the task-oriented robotic system itself. The complicated task may need to be divided into sub-tasks with the assistance from the operator.

The task will then be passed to the task controller to do the path planning as well as the transformation to action. The combination of the task controller, the robotic system (the robot controller and the robot) as well as the sensory system (sensors and the sensory sub-system) will form a closed-loop task control sub-system as shown in Fig. 7. In other words, the system will be able to carry out the task autonomously.

At the end of the project, the system must be able to provide the web service. Web-based and non-web-based system will be developed for comparison. The non-web-based system architecture is shown in Fig. 8. The task control sub-system mentioned will be kept on the server. An application program will be developed to provide the command and task pre-processing. The application program will run on the client site. With the pre-processing carried out on the client site, this absolutely will reduce the data transferred and waiting time for the response from the server.

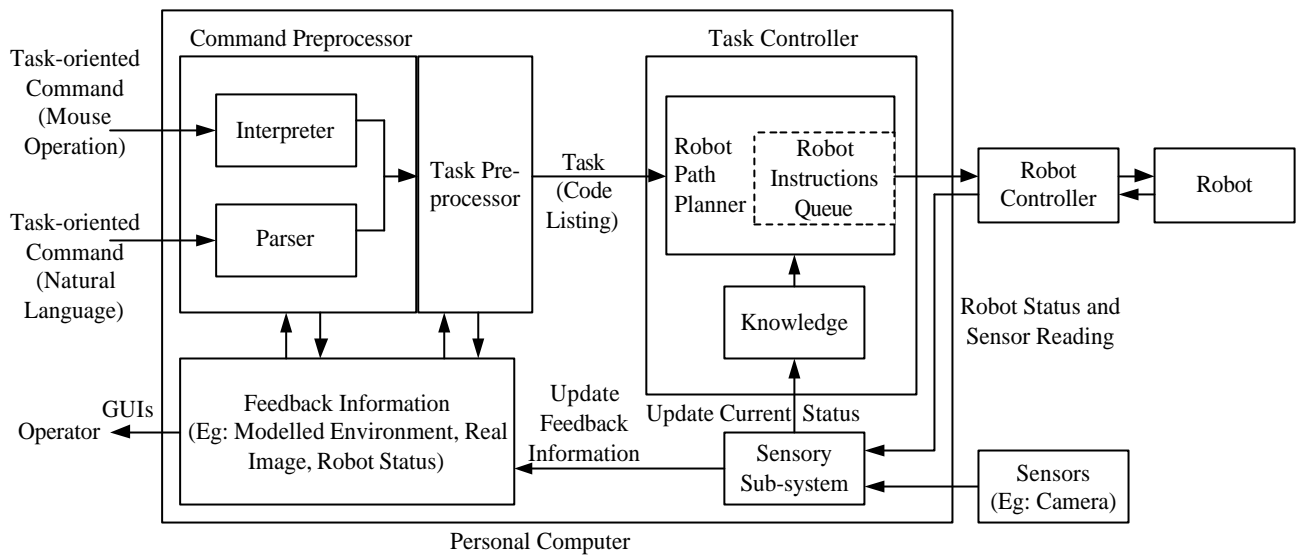


Fig. 6: Task-oriented robotic system architecture (without web service)

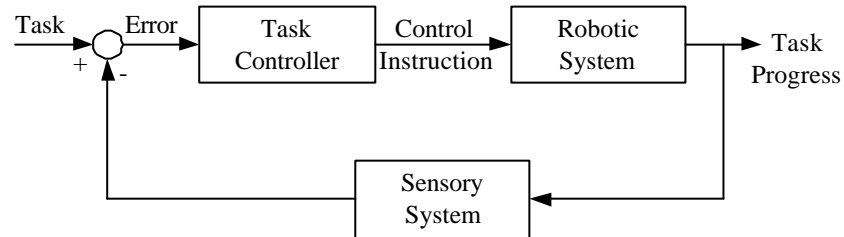


Fig. 7: Block diagram of the task control sub-system (closed-loop)

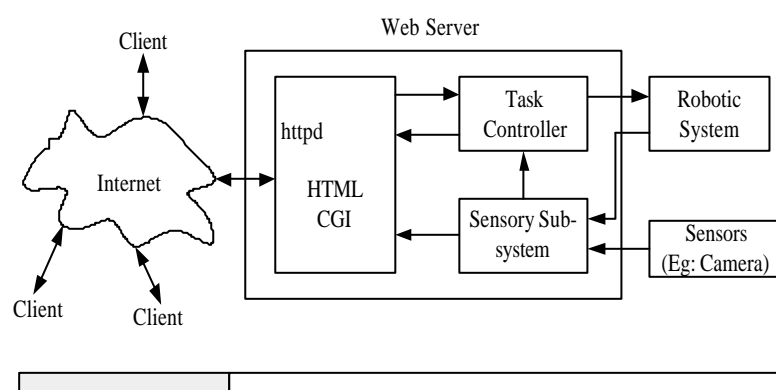


Fig. 8: Internet-based telerobotics system architecture

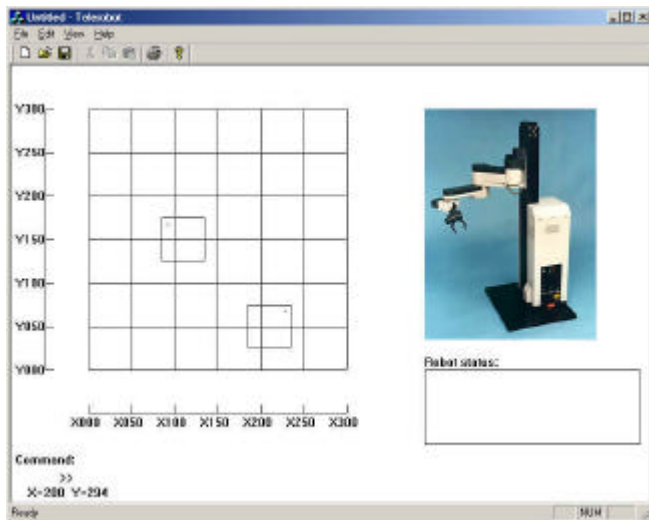


Fig. 9: Preliminary GUIs design (without web service)

3. Future Work

There have been many internet-based telerobotic projects developed since the first robot launched on Internet in 1994. Some of the projects are designed for critical applications such as telesurgery and telemanufacturing. Nevertheless, these applications are too risky and not practical for the current technologies available for the Internet. Unless in the future there are some break through technologies introduced and are low cost and publicly available or the system must be developed based on better quality connection but higher cost such as leased line and fibre optic. In a nutshell, our future direction of internet-based telerobotic projects will tend toward edutainment which is in line with the nature of today's Internet – publicly available, low cost as well as vulnerable and suffered from time delay.

4. Conclusion

We have successfully developed the internet-based telerobotic system for the leg of a mobile robot as well as the fixed type robot. The systems are designed based on robot-oriented and task-oriented concept. The project for fixed robot is expected to be available on the Internet in July. Currently we are developing the internet-based telerobotic system for the mobile robot and the fixed robot based on task-oriented concept. A comparison will be later made between robot-oriented and task-oriented systems.

Acknowledgements

The authors would like to thank the Malaysian Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

References

- [1] H. Friz. "Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views", Diploma Thesis, Technical University of Clausthal, Germany, Sept. 1998.
- [2] H. Thimbleby. "User Interface Design". ACM Press, New York. 1990.
- [3] Norhayati A.M, Shamsudin H.M.Amin, Rosbi Mamat. "Internet Based Leg Motion Control of A Mobile Robot", Proc. of the 2nd International Conference on Advances in Strategic Technologies (ICAST 2000), 15th –17th August 2000, Selangor, Malaysia, Vol.1, pg: 283-292, 2001.
- [4] Norhayati A.M, Shamsudin H.M.Amin, Rosbi Mamat. "Development of the Internet Interface for Leg Motion Control of A Mobile Robot", Proc. of the 1st International Conference on Mechatronics (ICOM'01), 12th – 13th February 2001, Kuala Lumpur, Malaysia, Vol.2, pg: 665-675, 2001
- [5] Shamsudin H.M. Amin, Rosbi Mamat, Mohamad Fauzi Zakaria, Imre J. Rudas, Laszlo Horvath, Jozsef Tar. "User Interface Design for Internet-based Telerobotics". Proc. of SCORed 2001, CD Vol., 2001.
- [6] Mohamad Fauzi Zakaria, Shamsudin H.M. Amin, Rosbi Mamat. "Design and Development of Control System for Internet-based Telerobotics". Proc. of TENCON 2000 Vol. II, pg. 338-342, 2000.
- [7] Rick Darnell. "HTML 4 Unleashed – Second Edition", Macmillan Computer Publishing, USA, 1998.

DESIGN OF TASK-ORIENTED SYSTEM FOR INTERNET-BASED TELEROBOTIC SYSTEM

Rosbi Mamat
 Zamani Md. Zain
 Lim Cheng Siong
 Center for Artificial Intelligence and Robotics
 (CAIRO)
 Faculty of Electrical Engineering
 Universiti Teknologi Malaysia
 81310 UTM Skudai
 Johor Darul Ta'zim, Malaysia
 Phone: +607 5505004
 Fax: +607 5566272
 chengsiong@hotmail.com

Imre J. Rudas
 Laszlo Horvath
 Jozsef Tar
 Budapest Polytechnic
 H-1081 Budapest Nepszinhaz u.8
 Hungary

Abstract

Task-oriented robotic system or so called "task-centric" [1] robotic system requires only the operator to specify the task to be done by the system and the system will then plan and carry out a series of actions to complete the task. In contrast, robot-oriented system requires the operator to plan the actions step by step to get the task done. Compared to a robot-oriented system, task-oriented robotic system has higher degree of autonomy. In this paper our decisions and approaches used in designing our task-oriented telerobotic system will be discussed and presented.

Keywords: Task-oriented robotic system, task-centric robotic system, robot-oriented system, robot-centric system, internet-based telerobotic system.

1 Robot-oriented System vs. Task-oriented Robotic System

The robot-oriented system and the task-oriented robotic system can be distinguished by

many aspects. The basic command unit for the robot-oriented system is based on the robot movement. For example the commands for arm type robotic system are shoulder up 30°, elbow down 30°, gripper open or spray start. Usually, 1 basic command unit for the robot-oriented system equals to 1 robot instruction. Meanwhile, the basic command unit for the task-oriented system is based on the task designed for the robotic system. For example the commands for the robotic goods sorting system are transferring objects type A to line A and objects type B to line B. Usually, 1 basic command unit for the task-oriented system equals to a series of robot instructions. The comparison between the robot-oriented system and the task-oriented robotic system is summarized in Table 1. It is important to understand that no matter which type the robotic system design is, in practical application both of the systems will have their own specific task such as welding, spray painting and goods sorting. The main different is on how the systems carry out the task.

Table 1: Comparison between robot-oriented system and task-oriented robotic system

Robot-oriented System	Task-oriented Robotic System
Basic command unit: ? Based on robot movement, e.g.: a) Arm type robotic system: shoulder up 30°, elbow down 30°, gripper open or	Basic command unit: ? Based on the task designed for the robotic system, e.g.: a) Robotic goods sorting system: transfer

spray start; b) Mobile robot: move forward 30 cm, turn left 45°. ? Usually, 1 basic command unit equals to 1 robot instruction.	objects type A to line A and objects type B to line B; b) Mobile robot: find the target such as heat/light source in unknown environment. ? Usually, 1 basic command unit equals to a series of robot instructions.
The system can directly convert the command given to robot instruction since 1 basic command unit equals to 1 robot instruction.	The system need to have the ability to “understand” the task given (requires task specified method) before the task can be converted to a series of robot instructions.
Operator will act as path planner to complete the task. In other word, the operator has full control over how the system will complete the task - direct control.	The task controller will do the path planning once “understand” the task(s) required to be done. In other word, the operator has no control over how the system will complete the task - indirect control.
Autonomy level: low.	Autonomy level: higher (with certain limitations).
Low efficiency in completing the work since every step involved must be manually planned/programmed.	Higher efficiency in completing the work since task controller will do the path planning.
Image capturing system (if involved) usually works merely for visual feedback.	Image capturing system (if involved) works not only for visual feedback but also as part of the vision system.
Less complicated to be designed and developed.	Complicated to be designed and developed especially the task controller.
Suitable application: usually for repeated/routine work especially in mass production.	Suitable application: usually for the work that is not/less repeated or the work with uncertainties such as goods sorting where the objects may vary in size, shape, orientation and location.

2 Designing the Task-oriented Robotic System

The first stage in designing a task-oriented robot system is to define the task required to be performed by the system as well as the methods to specify the task. For a wooden plank cutting machine, the task required is to produce a certain shape and size of plank based on the soft copy of the drawing or lines made on the surface of the plank. On the other hand, the task for a welding robotic system is to perform welding based on the joint of two or more metal plates. The tasks mentioned must be understood by the robotic system so that a series of actions can be planned and carried out. A proper task specified methods must be defined. For example the wooden plank cutting machine mentioned must be able to

interpret the soft copy of the drawing (task) provided to the system.

Since the task definition and specified methods have been determined, the feasibility and cost of designing and developing the robotic system must be considered before proceed to the next stage. A robotic system with 2D operation is easier to be designed and developed compared to a robotic system with 3D operation. This is also true for a fixed robot arm system compared to a mobile robot equipped with robot arm, and similarly for robotic system working in structured environment compared to robotic systems working in unstructured environment.

The second stage involved the design of the sensory sub-system. The main purpose of the

sensory sub-system is to ensure the capability of the robotic system to identify the working target as well as miscellaneous purposes such as environmental information feedback and uncertainties occurrence detection. A vision system can be equipped to a welding robotic system to identify the length, shape, position and orientation of the joints of two or more metal plates. From the sensory feedback, the operator can decide on required task. The same kind of vision system can be incorporated to the wooden plank cutting machine so that it can “see” the “shape” to be cut out. As a conclusion, sensory feedback is very important in supporting the task-oriented robotic system to complete the task.

In stage three, the knowledge about the robot, the working area, the working target as well as the sensory sub-system must be modelled. This knowledge is required by the task-oriented controller before it can plan and carry out a series of actions which will be discussed in stage four. For the robot, its inverse kinematics equations must be derived. A proper working area must be developed and set up based on the dimension of the robot, the working objects, the task and the sensory sub-system requirement such as lightning for vision system. The relationship among the robot, the working area, the working target and the sensory sub-system can be formed through transformation matrices.

In stage four, the task which tells the task-oriented robotic system what to be done must be transformed into action, step by step how to get the task done. This is called as path planning and it is one of the functions of task-oriented controller. A well designed path planner will be managed to handle complicated tasks and requires shorter working time to get the task done without scarifying the output quality.

Finally, it involved the design of the user interface for the task-oriented robotic system. Although the system is easier to be operated compared to robot-oriented system, effort is still needed in designing the user interface since it is the medium where human and machine interact to each other.

3 The Design of the UTM’s Task-oriented Internet-based Telerobotic System: Fixed Arm Type Robot

A robot can be broadly defined as a system where a mechanism is controlled by a computer. In telerobotics, the mechanism is remote. It sends back data and is generally controlled by a human at the other end [2]. The first robot has appeared on the internet in 1994. The project, named Mercury project [3], was the first system that allowed WWW users to remotely view and alter the real world via telerobotics. Since the launch of the robots on the internet, an enormous effort has been undertaken by hundreds of researchers to push this technology.

Below describes the stages involved in the designing the UTM’s task-oriented internet-based telerobotic system.

3.1 First Stage: Defining the Task and the Task Specified Methods

The telerobotic system is built to work in structured environment where the system learns about its working environment as well as the working object. The working object is limited to the wooden cube of the size 50 mm x 50 mm. The task defined for the telerobotic system is to manipulate the wooden blocks in front of the robot. The task is limited to 2 dimension operation or in other word the users are not allowed to stack the blocks. The users can tell the system about the task by manipulating the blocks in the virtual environment. This can be done either through mouse operation or by using natural language. The details about the task and the methods used to tell the telerobotic system will be discussed in Section 4.

3.2 Second Stage: Sensory Sub-system

The telerobotic system is equipped with a vision system. The vision system allows the telerobotic system to “see” the working objects and the progress of the task. Besides, the vision system can detect the occurrence of the uncertainties so that the telerobotic system can take the recovery action. The vision system

consists of a colour CCD camera, Sony XC-003, and a frame grabber, Matrox Genesis-LC (PCI bus version). The camera is put at the top of the working area. The captured image is processed by using Matrox Imaging Library (MIL).

3.3 Third Stage: Knowledge Development

The knowledge about the robot, the working area as well as the working object have been modelled and made known to the telerobotic system through the inverse kinematic equations and the transformation matrices derived. The RT100 robot has been chosen to be the candidate for the project because of its high accuracy and repeatability. Figure 1 shows the relationship between the working area and the position of the RT100 robot.

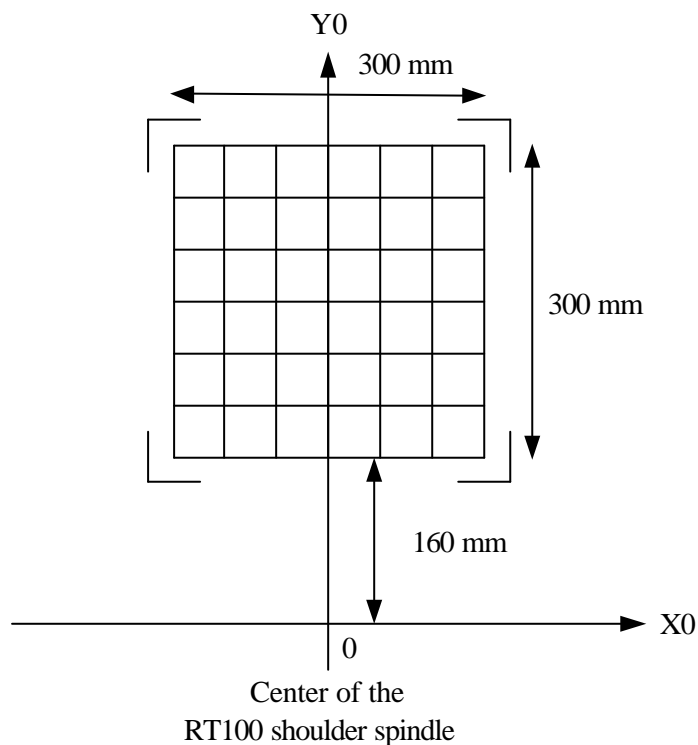


Figure 1: Working area and the position of RT100 robot

3.4 Fourth Stage: Path Planning and Transformation into Action

Since there are too many possible paths for the robot to move even just a block of cube from a point to another, certain criteria has been set. The path preferred is the path that requires minimum working time without hitting the other objects as

well as easy to be transformed into a series of actions. Below are the rules laid down for the path planning and the transformation into action:

- i) Lift the cube at optimum height that allows the cube to be moved across the other cubes. It will take longer time if the cube was lifted too high. On the other hand the cube might hit the other cubes if the height was not enough. In order to simplify the path planning, this rule is obeyed even there is no obstacle along the path;
- ii) All the motors rotate simultaneously to move and open/close the gripper;
- iii) Rotate and move operations for the same cube will be performed simultaneously;
- iv) The time needed by the gripper to travel from point to point is not depending on the distance between the points but the time taken by the motors to complete the degrees of rotation calculated through inverse kinematic equations;
- v) The time needed by the gripper to travel from point to point equals to the maximum time required by the motors to complete the rotation since all the motors rotate simultaneously;
- vi) The number of possible sequences to move the N cubes equals to N factorial ($N!$); and,
- vii) The time required by the gripper to travel from the set point to the first cube will be considered.

3.5 Fifth Stage: User Interface Design

Figure 2 shows the user interface of the client application. First of all, the user has to click the connect button to get connected to the server. If the robot was free from other user, then the user is allowed to manipulate the blocks in the virtual environment either through mouse operation or by using natural language. These input methods have been chosen for their ease of use. A pop-up menu will appear if the user clicked on the square in the virtual environment. There are four options listed in the menu: move, rotate, cancel and execute.

The user can choose the operation he or she wished to. The image will be updated upon the completion of the task.

On the other hand, the user will be restricted to receive image feedback and the chat

room if the robot was controlled by the other user. The purpose of these is to attract the users to remain connected to the server until their turn. Besides, the users can exchange their opinions about the project.

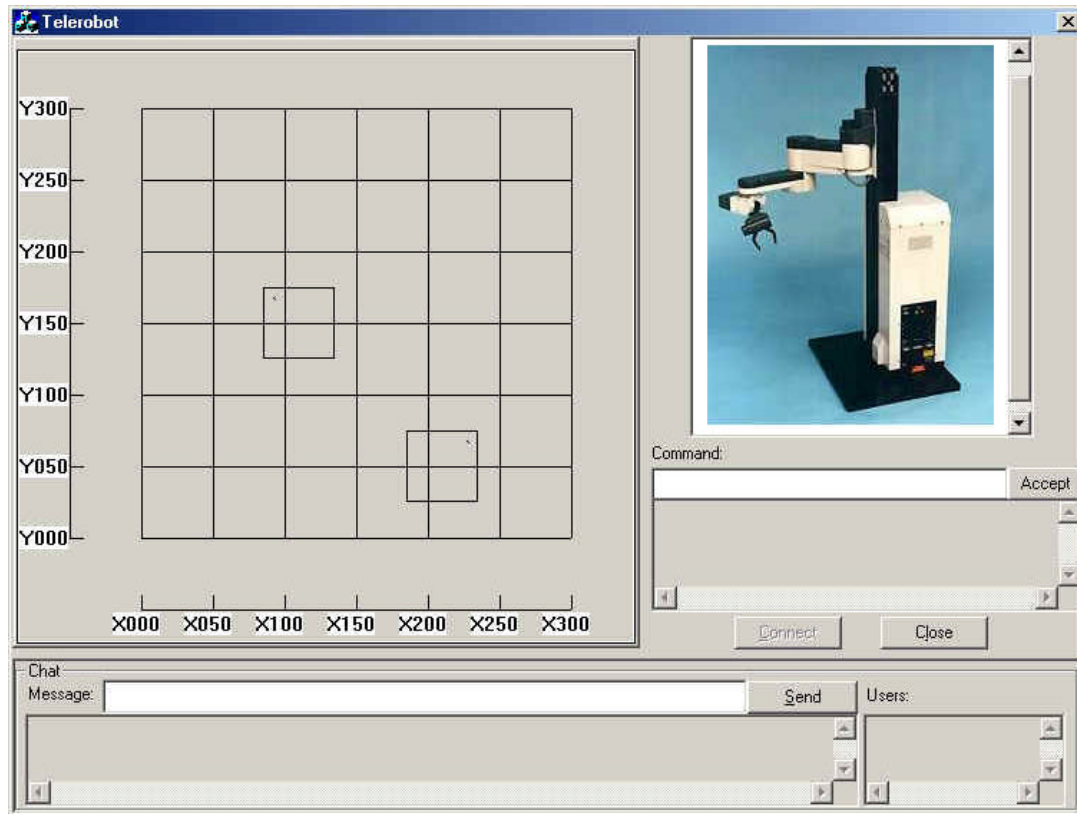


Figure 2: Preliminary GUIs design for client application

4 Task-oriented Robotic System Architecture

Figure 3 shows the architecture of the task-oriented robotic system. The system can accept task-oriented command from the operators either through mouse operation or natural language. The command will be then processed by the command pre-processor – either by the interpreter or the parser. The purpose of the command pre-processor is to remove the illegal commands such as spelling mistake, syntax error as well as to check the validity of the mouse operation. Once the command gets passed from the command pre-processor, the command will be then passed to the task pre-processor. The task pre-processor will do the simulation if the task could be performed by the task controller. Apparently not all tasks can be performed by the task controller due to the

limitation in the system design. The simulation is hidden from the user and if the process succeeded the virtual environment will be updated. During the command and task pre-processing stage, information such as the number of objects, location and orientation are made available to the command pre-processor.

Once the system accepts the command from the operator to execute the task, the task will be then passed to the task controller to do the path planning as well as to transform into action. The task controller, the robotic system (the robot with its controller) as well as the sensory system (combination of the sensors and the sensory sub-system) can be simplified into a closed-loop block diagram as shown in Figure 4. This is what called as visual servoing [8]. In other word, the system

will be able to complete the task given without the supervision from the operator.

At the end of the project, the telerobotic system will be incorporated with the Internet service. The system architecture is shown in Figure 5. An application program will be developed to provide the command and task pre-

processing (shown in Figure 2). On the other hand, the task control sub-system (shown in Figure 4) will be kept remain on the server. The application program will run on the client site. With the pre-processing be carried out on the client site, this absolutely will reduce the data transferred and waiting time for the response from the server.

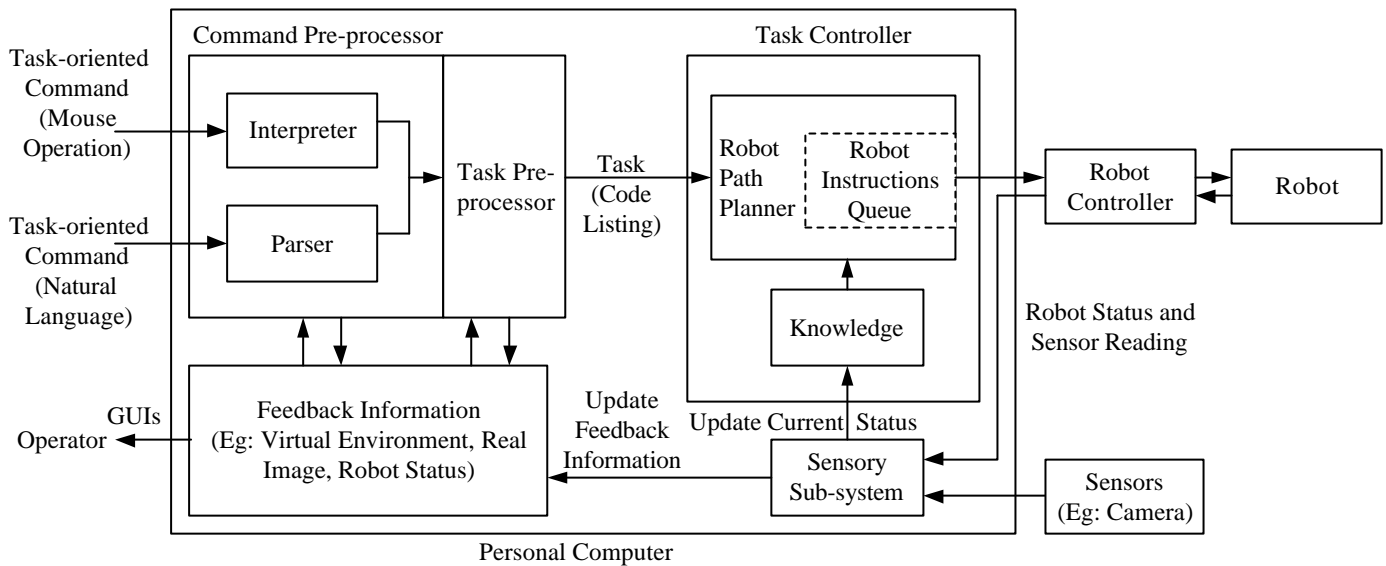


Figure 3: Task-oriented robotic system architecture (without Internet service)

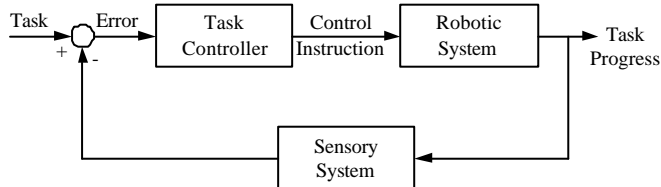


Figure 4: Block diagram of the task control sub-system (closed-loop)

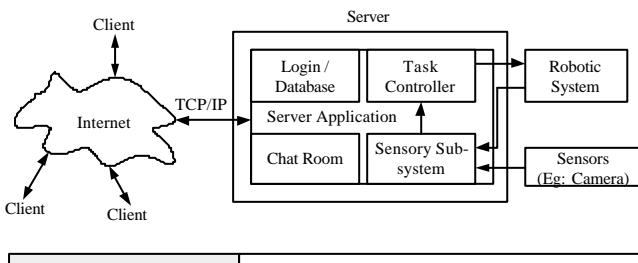


Figure 5: Internet-based telerobotic system architecture

5 Advantages of the System Architecture

The task-oriented system architecture provides better solution to the certain problems faced in internet-based telerobotic application which had been discussed by Taylor and Dalton [4]. The advantages of the system architecture are as below:

5.1 Task-oriented Robotic System: Easy to Operate

Basically the task-oriented robotic system is easier to be operated than robot-oriented system since one task in the task-oriented robotic system equals to a set of commands in the robot-oriented system. For example a simple task to move a cube from one location to another in the task-oriented robotic system requires the operator to specify a set of commands to move the various motors in the robot-oriented system. Furthermore, the complexity of the robotic system is hidden from the users where the robotics knowledge is not required any more. The users just have to

concentrate on the task designed for the robotic system without learning what the elbow, shoulder, gripper, tilt and spin mean.

5.2 Interface Design: Easy to Use

The option of mouse operation as one of the task specified methods makes the application program easy to use. The users who learnt how to use the computer definitely understand the operation of the mouse. The mouse right click will pop up a menu and the users can then left click to select the command wished. On the other hand, even though natural language is a bit more difficult to be learnt compared with mouse operation, it does provide higher accuracy operation. Natural language is chosen to be one of the task specified methods because it is more human-oriented and thus easier to be learnt compared with the command used in robot-oriented system which is tend toward robot-oriented such as rotate shoulder 30°.

5.3 Response Time

A client application program is developed for the users to download. The size of the program is maintained as small as possible. Certain processes such as command and task pre-processing will be carried out on the client site thus reduce the waiting time for the response from the server. Furthermore the robotic system does the path planning and supervises the progress of the task at the remote site thus the delay between the steps can be minimized. In contrast, the robot-oriented system requires the user to specify each step followed upon completion of the latest step.

5.4 Human Factor

In order to attract more users to operate the robot, the users who fail to gain control over the telerobotic system will be restricted to receive image feedback and the chat room. These make the users feel they are not alone and still involved in the project even though they are still waiting for their turn. From the statistics done by Taylor and Dalton [4], three quarters of the users have given up after waiting for three minutes. Furthermore, the use of mouse operation as one of the task

specified methods will be able to attract the users who are not keen to learn how to operate a complex system.

5.5 Safety and Reliability

Since the system is designed based on task-oriented concept, the system architecture is hidden from the users and thus the users are not controlling the robot directly. The problems that the users might cause damages to the robot, working area and working objects have overcome. Furthermore, any uncertainties that happen on the remote site can be detected by the vision system and the appropriate recovery action will be taken.

6 Conclusion

This paper has described our decisions and approaches in designing the task-oriented robotic system for use in Internet-based application. The capabilities of the system architecture in solving the certain problems faced in Internet-based telerobotic application have been highlighted. The objective of the project is intended for edutainment purpose after taking consideration of the nature of today Internet – publicly available, low cost as well as vulnerable and suffered from time delay.

Acknowledgements

The authors would like to thank the Malaysia Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

References

- [1] J.E. Lloyd, J.S. Beis, D.K. Pai, and D.G. Lowe. **Model-based Telerobotics with Vision**. Proceedings of the ICRA '97, page 1297-1304, Albuquerque, New Mexico, April 1997.
- [2] Bobak R. Farzin, Ken Goldberg, and Adam Jacobs. **A Minimalist Telerobotic Installation on the Internet**. In 1st Workshop on Web Robots, International Conference on Robots and Intelligent Systems, September 1998.

- [3] K. Goldberg et al.. **The Mercury Project: A Feasibility Study for Internet Robots**. IEEE Robotics and Automation Magazine, page 35-40, March 2000.
- [4] K. Taylor and B. Dalton. **Issues in Internet telerobotics**. In International Conference on Field and Service Robotics (FSR 97), page 151-157, Canberra, Australia, 8-10 December 1997.
- [5] K. Taylor and B. Dalton. **Internet Robot: A New Robotics Niche**. IEEE Robotics and Automation Magazine, page 27-34, March 2000.
- [6] Matthew R. Stein. **Interactive Internet Artistry**. IEEE Robotics and Automation Magazine, page 28-32, June 2000.
- [7] Fauzi Zakaria. **Design and Development of Control System for Internet-based Telerobotics**. Proceeding of the 2000 TENCON, Vol. II, page 338-342, 2000.
- [8] Peter I. Corke (1996). **Visual Control of Robots: High-Performance Visual Servoing**. Research Studies Press: England.
- [9] K.S. Fu et al. (1987). **Robotics: Control, Sensing, Vision, and Intelligence**. McGraw-Hill: Singapore.
- [10] H. Friz. **Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views**. Diploma Thesis, Technical University of Clausthal, German, 1998.
- [11] J. Allen (1987). **Natural Language Understanding**. Benjamin/Cummings Publishing Company: Canada.
- [12] Mark C. Torrance. **Natural Language with Robots**. Thesis in Master of Science, Massachusetts Institute of Technology MIT, USA, 1994.
- [13] **RT100 User Manual**. Universal Machine Intelligence: England. 1990.

Natural Language in Task-Oriented Telebototic Application

Lim Cheng Siong, Rosbi Mamat, Zamani Md. Zain

Universiti Teknologi Malaysia, Malaysia

ABSTRACT: Task-oriented robotic system or so called “task-centric” (J.E. Lloyd *et.al.*, 1997) robotic system requires only the operator to specify the task to be done by the system and the system will then plan and carry out a series of actions to complete the task. In contrast, robot-oriented system requires the operator to plan the actions step by step to get the task done. Compared to a robot-oriented system, task-oriented robotic system has higher degree of autonomy. In this paper, the design and application of the natural language in the task-oriented telerobotic system will be discussed and presented.

INTRODUCTION

A robot can be broadly defined as a system where a mechanism is controlled by a computer. In telerobotics, the mechanism is remote. It sends back data and is generally controlled by a human at the other end (Bobak R. Farzin *et.al.*, 1998). The first robot has appeared on the internet in 1994. The project, named Mercury project (K. Goldberg *et.al.*, 2000), was the first system that allowed WWW users to remotely view and alter the real world via telerobotics. Since the launch of the robots on the internet, an enormous effort has been undertaken by hundreds of researchers to push this technology.

Our internet-based telerobotic system is developed based on task-oriented concept for the convenient of the operators. The system is more user friendly than robot-oriented system since the operators focus more on the task completion rather than robot movement planning. As a result, natural language has been chosen to be one of the methods for user to operate the system. Natural language is human-oriented thus it is easier to be learnt and used. The natural language designed for the system is a typewritten English like language.

THE UTM'S TASK-ORIENTED INTERNET-BASED TELEROBOTIC SYSTEM: FIXED ARM TYPE ROBOT

The telerobotic system is built to work in structured environment where the system learns about its working environment as well as the working object. The working object is limited to the wooden cube of the size 50 mm x 50 mm. The task defined for the telerobotic system is to manipulate the wooden blocks in front of the robot. The task is limited to 2 dimension operation or in other word the users are not allowed to stack the blocks. Figure 1 shows the relationship between the working area and the position of the RT100 robot, which is chosen for the project because of its high accuracy and repeatability.

The telerobotic system is equipped with a vision system. The vision system allows the telerobotic system to “see” the working objects and the progress of the task. Besides, the vision system can detect the occurrence of the uncertainties so that the telerobotic system can take the recovery action. The vision system consists of a colour CCD camera, Sony XC-003, and a frame grabber, Matrox Genesis-LC (PCI bus version). The camera is put at the top of the working area.

The captured image is processed by using Matrox Imaging Library (MIL).

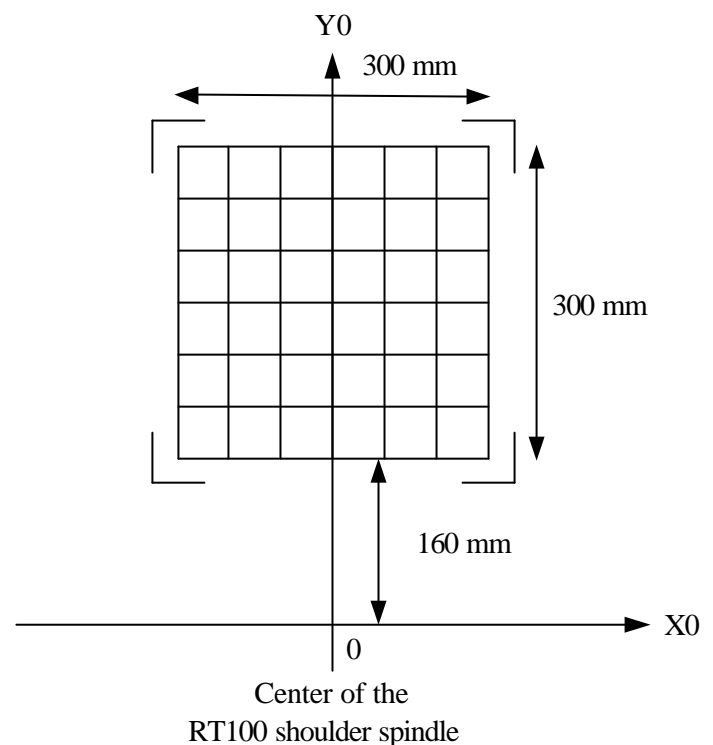


Figure 1: Working area and the position of RT100 robot

TASK-ORIENTED ROBOTIC SYSTEM ARCHITECTURE

Figure 2 shows the architecture of the task-oriented robotic system. The system can accept task-oriented command from the operators either through mouse operation or natural language. The command will be then processed by the command pre-processor – either by the interpreter or the parser. The purpose of the command pre-processor is to remove the illegal commands such as spelling mistake, syntax error as well as to check the validity of the mouse operation. Once the command gets passed from the command pre-processor, the command will be then passed to the task pre-processor. The task pre-processor will do the simulation if the task could be performed by the task controller. Apparently not all tasks can be performed by the task controller due to the limitation in the system design. The simulation is hidden from the user and if the process

succeeded the virtual environment will be updated. During the command and task pre-processing stage, information such as the number of objects, location and orientation are made available to the command pre-processor.

Once the system accepts the command from the operator to execute the task, the task will be then passed to the task controller to do the path planning as well as to transform the task into action. The task controller, the robotic system (the robot with its controller) as well as the sensory system (combination of the sensors and the sensory sub-system) can be simplified into a closed-loop block diagram as shown in Figure 4. This is what called as visual servoing (Peter I. Corke, 1996). In other word, the system will be able to

complete the task given without the supervision from the operator.

At the end of the project, the telerobotic system will be incorporated with the Internet service. The system architecture is shown in Figure 5. The task control sub-system (shown in Figure 4) will be kept remain on the server. A client application program will be developed to provide the command and task pre-processing (shown in Figure 3). The application program will run on the client site. With the pre-processing be carried out on the client site, this absolutely will reduce the data transferred and waiting time for the response from the server.

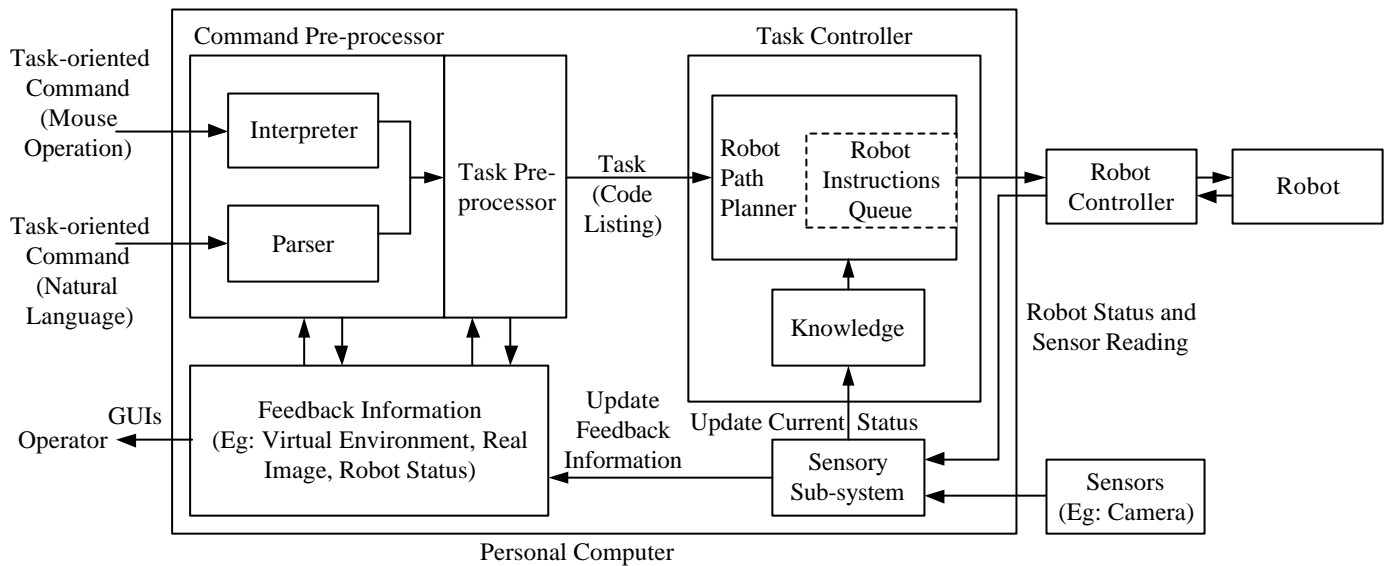


Figure 2: Task-oriented robotic system architecture (without Internet service)

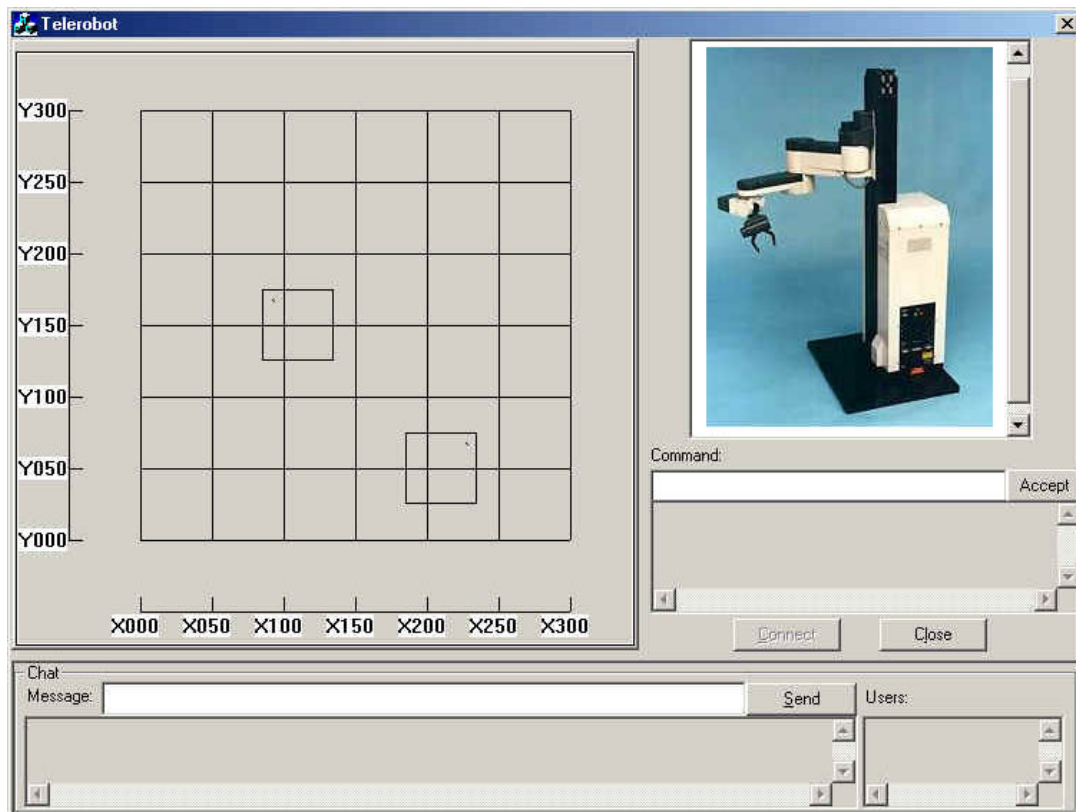


Figure 3: Preliminary GUIs design for client application

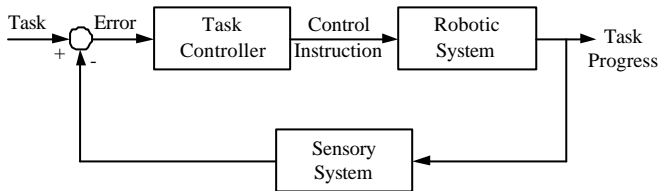


Figure 4: Block diagram of the task control sub-system (closed-loop)

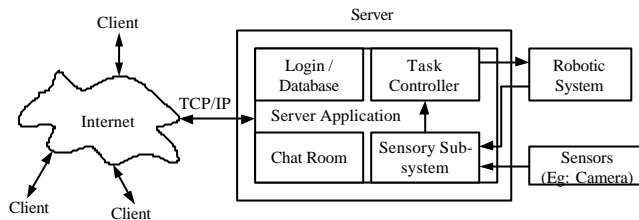


Figure 5: Internet-based telerobotic system architecture

APPLICATION OF NATURAL LANGUAGE

Since the task of the telerobotic system is limited to 2 dimension operation, a limited variety of English sentence constructions is needed to support all the possible operations. Thus, a very constrained grammar will suffice. Below are the set of the language supported by the system:-

“{coordinate xy of an object} is {name given to the object}”

This informs the telerobotic system that the object with the coordinate mentioned is given a name. The command can also be used to rename the name of the object. The name given to the object must be in single word. Besides, the coordinate mentioned must be any point that falls within the area covered by the object.

“{coordinate xy of an object}”

This command is used to inquire the telerobotic system about the name given to the object (if any).

“{object’s name}”

This command is used to inquire the telerobotic system about the coordinate of the object specified (if any).

“Place {object’s name|coordinate xy of an object} to {coordinate xy}”

This will instruct the telerobotic system to move the object to the coordinate specified.

“Place {object’s name|coordinate xy of an object} to {coordinate x|coordinate y}”

This will instruct the telerobotic system to move the object to a new coordinate where the value of coordinate x or y will be changed.

“Place {object’s name|coordinate xy of an object} to {left|right|front|back} {distance in milimeter}”

This command will instruct the robot to offset the object to left/right/front/back of the current coordinate with the distance specified.

“Rotate {object’s name|coordinate xy of an object} {degree of rotation}”

This command will instruct the robot to rotate the object according to the degree specified.

“{command 1} then {command 2}”

This command allows the operator to issue two commands in one statement.

“execute”

This command instructs the telerobotic system to carry out all the commands required by the operator.

“undo”

This command will cause the telerobotic system to restore all the objects to their previous position before the “execute” command.

The application of the natural language will be more effective if the telerobotic system would be able to communicate with the operator through natural language. Intelligent parser that will be able to guide the operator by communicating in natural language is still under construction. For example the parser will be able to point out the error in the command by replying “unknown {error} in {command}”.

CONCLUSION

This paper has described the application of natural language in designing the task-oriented robotic system for use in Internet-based application. Even though natural language is more difficult to be learnt compared with mouse operation, it does provide higher accuracy for objects manipulation. Furthermore, natural language is human-oriented and thus is easier to be learnt compared with the command used in robot-oriented system which is tend toward robot-oriented.

ACKNOWLEDGEMENTS

The authors would like to thank the Malaysia Ministry of Science, Technology and Environment for sponsoring this work under IRPA 09-02-06-0022.

REFERENCES

Lim Cheng Siong, Amin, Rosbi Mamat, Zamani Md. Zain, Design of Task-Oriented System for Internet-based Telerobotic System. *Proc. of the MSTC 2001*, Malacca, Malaysia, pp. B136 (2001).

Shamsudin H.M. Amin, Rosbi Mamat, Mohamad Fauzi Zakaria, Norhayati A.M, Lim Cheng Siong, Internet-based Telerobotics: UTM's Experience and Future Direction. *Proc. of the ICAR 2001*, Budapest, Hungary, pp. 313-319 (2001).

J.E. Lloyd, J.S. Beis, D.K. Pai, and D.G. Lowe, Model-based Telerobotics with Vision. *Proc. of the ICRA '97*, Albuquerque, New Mexico, pp. 1297-1304 (1997).

K. Taylor and B. Dalton, Issues in Internet telerobotics. *Inter. Conf. on Field and Service Robotics (FSR 97)*, Canberra, Australia, pp. 151-157 (1997).

Fauzi Zakaria, Design and Development of Control System for Internet-based Telerobotics. *Proc. of the 2000 TENCON*, Kuala Lumpur, Malaysia, Vol. II, pp. 338-342 (2000).

Bobak R. Farzin, Ken Goldberg, and Adam Jacobs, A Minimalist Telerobotic Installation on the Internet. In *1st Workshop on Web Robots, Inter. Conf. on Robots and Intelligent Systems*, Victoria, Canada, pp.7-13 (1998).

Peter I. Corke, *Visual Control of Robots: High-Performance Visual Servoing*. Research Studies Press: England (1996).

K.S. Fu et al., *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill: Singapore (1987).

J. Allen, *Natural Language Understanding*. Benjamin/Cummings Publishing Company: Canada (1987).

Mark C. Torrance, *Natural Language with Robots*. Thesis in Master of Science, Massachusetts Institute of Technology MIT, USA (1994).

H. Friz, *Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views*. Diploma Thesis, Technical University of Clausthal, German (1998).

RT100 User Manual. Universal Machine Intelligence: England (1990).

K. Goldberg et al., The Mercury Project: A Feasibility Study for Internet Robots. *IEEE Robotics and Automation Magazine*, 7(1):35-40 (March 2000).

K. Taylor and B. Dalton, Internet Robot: A New Robotics Niche. *IEEE Robotics and Automation Magazine*, 7(1):27-34 (March 2000).

Matthew R. Stein, Interactive Internet Artistry. *IEEE Robotics and Automation Magazine*, 7(2):28-32 (June 2000).