CHAPTER 1

Introduction

1.1 Aims

The aim of this thesis is to develop a remotely operated system to be used for interactive visual monitoring of surgical procedures where the specialist surgeon cannot be physically present. In this thesis the development of a prototype, remotely operated robotic system, consisting of a commercially available industrial robot, a video camera, a teleconferencing package and custom designed software, is described. The system in its final version is intended to be used in the operating theatre or in surgical clinics in country regions where specialist consultation is not possible by other means. Especially in the Australian context where there are large distances between remote regions and city centers, the capability provided by the proposed system for monitoring and advising through a vision and voice link is a development that is expected to contribute to the improvement in the provision of medical and surgical services.

By implementing this prototype system in an extensive experimental program at the University Of Western Sydney and in conjunction with the Nepean Hospital, the potential health benefits to the community of such a system and its practicability can be demonstrated.
1.2 Background

Based on the multidisciplinary relationship between mechatronic engineering and medical and surgical experience and expertise, a research project was begun in the School of Engineering and Industrial Design for developing a system for the visual monitoring of surgical procedures at distant locations.

Existing technology in this area includes a commercially available, voice-activated robot (Aesop) that is used in laparoscopic surgery for holding and controlling the camera during the procedure [17]. This system, being specially designed for this one function, is located within the same volumetric space as the surgeon and is not suitable for this application. Other systems that employ remote visual monitoring involve medical, not surgical applications, or a specific function such as observing eye conditions [17].

What makes this project unique is that this system's final design is intended to not infringe on the working area of the attending surgeon during a procedure and to have an independently controlled camera, operated remotely by a consulting surgeon through a standard data network. This may have very important implications for both the surgeon and the patient as it may eliminate the need to transport the patient over large distances and allows general surgeons to perform more complex surgical procedures with the guidance of remote specialists.
The operation by a surgeon of this complex system, consisting of a robot-like device with a video camera mounted at the end of it, requires a user interface that is both simple and reliable. The development of a remotely operated system that enables the visual monitoring of surgical procedures at distant locations represents an important advantage on current medical technology and provides a new opportunity to benefit health services. In particular, in the Australian context where many country centers lack access to specialist surgeons, the implementation of such a system is considered especially appropriate and is intended to extend the reach of specialist surgeons based in city areas. This project of developing a teleoperated system for providing expert surgical advice through visual observation is a novel application of existing technology in the surgical field as far as can be ascertained from previous published work.

Although teleoperated systems have been the subject of many research publications, the application of these devices to the specific task of surgical consultation and monitoring has not been recorded in the literature to the best knowledge of the author. In the area of surgical education and training, the project system is expected to have an important role as it has the capability to provide real-time visual and audio information of a surgical procedure. With the operation of the camera, being under the control of a remote instructor or adviser, interference with the procedure or the distraction of the attending surgeon is avoided while providing total flexibility in being able to continually change viewing conditions.
The objective of this project is to develop and implement an electrically driven mechanical device with a camera mounted on it, which is controlled via a data network. Based on a development of existing technologies in the areas of telecommunications, real-time mechatronic systems and computer integration. It is intended that the practicability of this integrated system for remote monitoring of surgery from a distant location be demonstrated through an extensive program of development and testing.

As the attending surgeon undertakes a surgical procedure, the camera is moved into a convenient position where the supervisor, controlling the device from a remote location, can observe the procedure and through an audio link advise and consult with the local surgeon.

To achieve this, the necessary techniques and software interface for controlling the system equipped with a video camera in a secure and user friendly manner that is suitable for the use by a surgeon had to be developed. To simulate the system, testing procedures for the system in a laboratory setting, using local-area network (LAN) and modems for communication, and evaluating the reliability and electrical safety of the system forms the main area covered in this thesis. In the longer term, the objective is to adapt the system for application in surgical training and education of medical students.
The current project consists of a number of distinct stages and has been devised to achieve the objectives in developing an integrated mechatronic system that consists of a robot device, a CCD camera for visual monitoring and the controlling software. The design of the system is to incorporate the requirements that it needs to be implemented within the environment of an operation theatre and that a supervising surgeon can easily and reliably control the system.

This study deals mainly with the software development and the serial interface with a robotic device. The robotic device that is going to be used in the final stage of the project will need to be custom designed and constructed such that it does not interfere with the working volume of the operating surgeon. For laboratory testing, an industrial robot was used for verifying the software. The robot used is a welding robot with six-degrees of freedom, the Motoman SV6, and it supports a host computer function for interfacing with a personal computer.

Even though it is considered to be a small industrial robot, this robot device is too limited in terms of working range, for mounting in an operating theatre for the final stage of the project. The current robot is attached to a personal computer at the host site, with a custom made RS-232 cable, made to the specifications of the robot manufacturer. The computer serves as the control center for the robot at the host site. The computer is equipped with a modem and LAN connection, allowing it to receive data from the remote site. The host computer executes the commands it receives from the remote site,
controlling the manipulator movements, and performs other actions related to the control of the manipulator.

A color CDD camera, with auto-focus, is mounted at the end-effector of the manipulator. The camera is part of a commercially available videoconferencing package, which has been acquired for the project. The flexibility afforded by the six-degrees of freedom of the robot, is necessary to allow the device to be positioned anywhere within the working volume, with any orientation, and to provide unrestricted control of the viewing directions.

For the software development, visual C++ software was used to ensure that the serial interface with the modems and robot functions properly. The visual C++ compiler supports ms-dos based programs and windows based programming. Two software programs were developed using visual C++, one in ms-dos, this being for handling the data coming from the remote site and the robot controller, and a Windows based program at the remote site to permit the operator to control the robot from the remote location. The windows based program for the control of the robot is easier for the operator to handle, and is more user-friendly than an equivalent ms-dos-based program. The ms-dos program does not affect the operator, as it is a layer below the user-interface program.
Communication between the mechanical system and the specialist surgeon, the remote operator, is by means of a video conferencing package. The package includes all the tools needed for normal videoconferencing. It also has the capability to implement either a Local Area Network (LAN) or the telephone network, ISDN, as the means of data transmission. In the future stage of the project, the video conferencing will employ an ISDN line, 128 kbit/second, for transmission over longer distances. This speed is sufficient for the intended application.

In the current project the software that handles the command execution for the manipulator communicates through a high-speed modem, while the video conferencing uses a Local Area Network (LAN). For the final stage of the project, ISDN will be used for communicating between the host site and the remote site.

1.3 Thesis Structure

In Chapter 2 the literature in the area of telemedicine, telesurgery and robotic surgery is reviewed. Here other projects, existing technology and future aspects of telemedicine are discussed. A laboratory version of the longer-term prototype system is described in Chapter 3, and the set-up is described in more detail. The hardware and software setup and the techniques that are implemented in this system are also presented.

Chapter 4 describes the implementation of the prototype system. It describes in detail the software development, the testing phases starting with simple programs and the hardware
aspects such as the camera and the manipulator. The communication aspects of the project are also described here, dealing with the RS-232 connection both to the robot and the personal computer. In this chapter the hardware handshaking and flow control used for the serial communication with the robot is also covered. The implementation of the modems to affect the remote connection is also included in this chapter.

Chapter 5 deals mostly with the videoconferencing system and different forms of video image transmissions and compressions, such as ISDN, ATM and h.261.

In chapter 6, robot path control is presented, which is designed to ensure that the end effector always points towards the target being observed and the video image does not get rotated due to movements of the end-effector. This path control is part of the future development where the control of the manipulator is achieved by means of a joystick.

The conclusion, including recommendations for further research is presented in chapter 7.
CHAPTER 2

Literature Review.

2.1 Introduction

There have been many attempts at producing an all-encompassing definition of telemedicine. The word ‘telemedicine’ is often regarded as too limiting, and definitions such as telecare and telehealth are also used to refer to the availability of nursing and community support and other public health services by the use of telecommunications. Applications that come under the broad headings of telemedicine, telecare and telehealth include the transmission of visual media, such as pathology slides (telepathology) [16, 24, and 25].

The emerging discipline of ‘telesurgery’ encompasses both the mentoring and training of surgeons in the operating theatre via video-conferencing systems and the performance of surgical procedures using robotic devices guided by real surgeons at a remote site using computer data of diagnostic studies and scans [12, 17 and 18]. E-health applications, which include PC-based video-conferencing, the Internet and e-mail, are becoming increasingly practical as a means of communication between patients and health professionals, as well as between health professionals themselves for the purpose of consultation, education and second opinions [16].

The application of the new technologies of computers, robotics and communications to the efficient provision of medical and surgical services in remote locations has been an area of recent research interest in both the medical and engineering disciplines [12,15, and 16].
This new area of study has been referred to in the literature by the general description of telemedicine and telesurgery. These terms describe the ability to transmit medical data and visual information through the data communication network, examples of which are the telephone system and the computer network, generally called the Internet. The field of telemedicine and telesurgery has encompassed research into remotely operated manipulators and robots as well as the use of cameras and the consideration of the problems in transmitting visual data or images through the available data networks [20, 21 and 22].

2.2 Robots in Telemedicine

One aspect of telemedicine that needs to be considered in order to deal with the overall issues of delivering medical services remotely is the method of obtaining visual images from a remote location. In a surgical procedure for example, the site of the surgery on the patient on the operating table may need to be viewed from different positions and orientations to be able to observe in detail what is taking place. For this to be achieved, a robot or a manipulator with a camera attached to its end-effector would need to be controlled remotely. There are many different types of Telemanipulators and the status of this area of research is reviewed here first.

2.3 General Features of a Teleoperated System

A teleoperated system consists of a master-slave system, where the master is the control unit (the computer) and the slave is the manipulator, and a videoconferencing unit for providing visual feedback for the surgical procedures. There are several different systems that are developed already [4, 12, 15, 17, and 18]. The system may be equipped
with a video camera, as in this project where a CCD camera is used, for video images and a personal computer to handle the controls and the video processing.

Figure 2.1 shows an example of what a typical setup that may consist of:

- Two personal computers
- Video card (mounted in an empty slot in the PC)
- Monitors for high resolution.
- RS-232 ports for serial interface.
- Audio and video decoder.
- Manipulator/robot, which supports serial interface.

There are many published papers on the issue of telemedicine. Many of the Telemanipulators described in these papers, are remotely controlled robotic arms that, for surgery, are continuously monitored through human vision. They therefore overcome the problems of pre-programmed robots. With the manipulator described in this thesis, the vision control is the key issue [12, 13, 14, 21, 22 and 23].
There are different kinds of telemanipulators, but crucial to this and other projects are the use of visual feedback. A significant area within the general field of Telemanipulators used in surgery is laparoscopic surgery [6, 12, 15, 16, 20, 21 and 22]. Laparoscopic surgery first made an impact upon the surgical scene in the late 1980’s [2]. With the help of a laparoscope, patients are able to undergo major surgery with little or no postoperative morbidity. Robotics may help to reduce the stresses of this type of surgery on surgeons and thereby make these operations safer, more efficient and faster [2]. The aim in implementing any robotic surgery, however, is that the surgeon should always have full control. There is no suggestion that a fully automated machine will be able to operate alone on humans at this stage [2].
Industrial Robots

Robots that are used for industrial purposes and those used in medical and surgical applications have to meet different requirements. Robots are widely used in factory automation, where they perform repetitive tasks with high accuracy and speed. These requirements are normally achieved by using robots with powerful actuators, and the working volumes of these robots are usually not shared by humans while in operation. The introduction of a robot in the surgical theatre imposes additional specific requirements that are not yet satisfied by conventional industrial robots. Problems that occur using industrial robots, the robot used in this thesis being one instance, can be identified as follows: Industrial robots have high torque, for handling different operation tasks using various tools as the end-effector. In the operating theatre environment such manipulators present a danger to the surgeon(s) with their high torque and force capabilities. The question of safe operation arises when the user shares the working volume with these robots and it represents a dangerous situation for the operator and has the potential to cause damage to other equipment as well.

Other design features of industrial robots that should be taken into account when considering their suitability for medical applications are the dimensions and kinematics that must be selected to be compatible with the limited working space available. With the presence of medical staff and other equipment around the operative table, there is a need for high dexterity, which may achieved by increasing the working range of each robot axis, and with low speed and low acceleration rates. In order to maintain arm sterilization, the robot should have smooth surfaces where bacteria are less likely to exist. For the same reason, all wiring should be routed internally within the robotic structure itself. If the manipulator is designed to work in hazardous environments such
near radioactive processes or x-ray machines in a hospital, wiring should be protected against radiation [26]. This proves to be a very difficult task without some sort of shielding. One solution to this particular problem may be achieved by using pneumatic or hydraulic drive systems, but this gives rise to other complications such as leakage and the need for much heavier equipment. The manipulator should also have high precision, repeatability and resolution for positional control as well as fail safe brakes. The overriding criteria that needs to be satisfied is the protection of the well being of the patient and the medical staff sharing the working environment with the robot.

Most industrial robots operate on a pre-programmed cycle, which allows the robot to work at high speed, since the path is pre-determined. The difference with Telemanipulators is that they use visual feedback, where the operator does not usually know where to move the manipulator before actually seeing the desired spot for its location. The ability to maneuver a manipulator within a specified workspace to every position with any orientation requires a robot that has 6 degrees of freedom.

2.4 Applications of Telemanipulators in Surgery.

A Telemanipulator is a device, which allows an operator to perform a task at a distance. These devices work as a master/ slave system, where the master is the control system, in this thesis a Windows based control program, and the slave is the manipulator.

Laparoscopic surgery has been one of the first areas to use manipulators in surgery. Such robots are controlled by an assistant or by other means of control such as voice control, foot pedal or head sensor [2]. The manipulator holds the laparoscope, where the control of the manipulator movement can be implemented in different ways. It can be
controlled through an infrared link or by using an RS-232 cable or, as in this project, by a combination of using a modem and a direct connection (RS-232) to the manipulator.

Since in practice manipulators in surgery are almost exclusively used in assisting laparoscopic surgery, this surgical procedure will be discussed in a little more detail. As mentioned earlier for the new keyhole method of surgery, a manipulator can be employed for holding and controlling the laparoscope during the surgical procedure [16, 12, 15 and 21]. The manipulator allows for a smooth non-shaky movement to be achieved giving a better image for the surgeon to view during the procedure.

Laparoscopic surgery requires the use of a laparoscope, which consists of a small camera nested into a tube for viewing the site of the internal surgery. The laparoscope is a rigid metal tube with the camera lens being similar to a microscope, magnifying the image. The system is known in minimally invasive operations that include in addition to laparoscopic surgery, thoracoscopy (chest cavity), arthroscopy (joints) and pelviscopy (pelvis)[12]. Laparoscopes, which use the Hopkins rod lens system, have high intensity sources for illumination and small CCD video cameras attached to the eyepieces. An image is produced on one or more video monitors mounted at the side of the operating table in front of the surgeon and assistants. The laparoscope is inserted into the abdomen through a cannula, or port, previously placed through a 10mm skin incision using a trocar. The problem arises, when the laparoscope is inserted. There are two ways of holding the laparoscope; either by using a clamp attached to the side of the operating table, which keeps it steady but makes movement difficult, or hand held by an assistant, which makes movement easier but demands close cooperation between assistant and surgeon and introduces some shakiness in the projected image. The
assistant is usually a junior doctor or nurse who moves the telescope on instructions from the surgeon. This is an expensive use of junior doctors or nurses and can also give rise to stress and mistakes if the assistant cannot easily interpret the surgeon’s commands. One of the obvious problems is that a left-to-right movement of the camera produces a right-to-left movement of the video image [2]. A telemanipulator may give back control to the surgeon during the operation and with voice activation can free both hands for handling instruments.

Telemanipulators are remotely controlled robotic arms, the movements of which for surgery are continuously monitored by using vision feedback. In this way, the problems of pre-programmed robots working automatically in soft tissues without boney landmarks for calibration are overcome.

There are various methods for controlling such manipulators. Voice activation is already implemented in some systems [17], but there are some problems with these with respect to interference in the working volume of the surgeon and in voice recognition reliability. Some other methods of robot control include roller ball control, single foot pedal or a head sensor [2 and 20]. An example of a head sensor controlled system was reported in [2] where the camera, mounted on the robot reproduced the head movements of the surgeon. This allowed the surgeon to be in complete control over robot movements. Telemanipulators and teleoperated systems can also be useful in environments that are harmful to humans, as in nuclear power plants and in space applications [26].

A telemeter system, described in [5] consists of two television cameras for obtaining stereoscopic vision within a specific working volume and is capable of providing three
dimensional co-ordinates of any point inside it. A point can be identified automatically by the system by marking it with a laser beam or by means of diode light emitters directly on the target object. Some basic surgical operations such as a simple incision or a biopsy may be possible to carry out with this system. It has been claimed by the authors [5] that such a system can provide a simplified method of robotic telepositioning in a surgical application.

**Robotic Control Systems**

The movements of robots used in various projects may be controlled differently using head sensors or foot pedals, but the control system consists usually of three control loops.

The control system can be divided into three control loops: local loop, remote loop and general loop. The operator, a personal computer and the interactive media form the local loop. The robot and a remote controller form the remote loop. The general loop connects the local and remote loops together so they can interact with each other.
An example of a general loop is a neurosurgical tool, described in [3]. This telemanipulator uses vision feedback for positioning the end-effector. The manipulator is connected to a microscope supported by a mobile platform. There are also other projects that involve telemanipulators. In Finland, a laboratory version of a surgical robot is under development [4]. It has been shown that there are limitations in this design, such as being too heavy and large for application in surgery.
2.5 Visual Monitoring in Telemedicine

Telemedicine involves the use of live, two-way audiovisual communication between two or more sites, using state-of-the-art video-conferencing technology. The vision data can be transmitted via the Internet or directly between to or more computers using modems or networks.

As part of a remotely operated monitoring system, videoconferencing packages are useful in easily transmitting images over a data network or using the phone system. Hospitals and medical staff already use videoconferencing to exchange information such as x-ray data and ultrasound images between different locations [7, 14, and 27].

According to [7], one of the first live interactive surgical teleconsultation broadcasted through the Internet was in 1996, where previously visual tele-consultation was only carried out via modems [7]. The transmission was performed from Pontiac, Michigan, to Laguna Hills, California, and Buenos Aires, Argentina, and the participants interacted in audio and video formats. Video images were transmitted at a rate of 1-2 frames/s and displayed in a 320 x 240 window at the remote sites. The loss of audio packets averaged 17 % with a delay of 0.5- 2 s. The broadcasting computer station was also able to receive video images and sound from the distant computer, allowing complete interaction between both parties over the duration of each transmission session. Average broadcast time was 1.5-h +/- 30 min, and the cost of each transmission was equal to that of a regular phone call. The conclusion for the project was that videoconferencing via the internet is a viable method for transmitting information in real-time allowing surgeons worldwide to work together during surgical procedures [7]. Although, this project is similar to the one described in this thesis, there are some different approaches.
The project in this thesis uses modem connection, but ultimately ISDN is intended to be implemented (LAN in the laboratory), and connected with the host site directly without going via the Internet.

The video images in this project are transmitted from the host to remote site and displayed in a 352 x 288 pixel window at the remote site. Using an ISDN line and various compression techniques incorporated in the video conferencing software, frame rates of 26-30 frames per second are achievable. In addition to the video conferencing unit, both projects support audio connection between the two or more parties. It is expected to have the same costs as the project described in [7], corresponding to a regular phone call and transmission delay in both the audio and video packages are expected to be considerably reduced.

The use of telemedicine involves telepresence surgery, as mentioned above and in other papers described in this thesis. In addition to using video conferencing, the wide area within telemedicine may use three-dimensional (3-D) visualization of anatomy, such as 3D models of the heart and brain for the purpose of medical education [55]. Other equipment that may contribute to education as well is VR (virtual reality), surgical simulators, and virtual prototyping of surgical equipment and operating rooms. According to [55], approximately 90% of the knowledge a physician requires can be obtained through electronic means, such as diagnostic sensors and imaging modalities, directly viewing the patient with a video camera for medical consultation, or using electronic medical records. Thus, when a surgical practitioner is separated from the patient by a physical distance information relating to the patient that is already in an electronic form can be transmitted electrically over a small or large distance [55].
Teleconferencing Applications

There are some requirements for the staff and patients involved with the equipment that a videoconferencing system used for a medical or surgical application must fulfill. Audio and video quality must enable easy interaction between the physically separated participants. Patients and staff must be able to clearly see and hear each other and the videoconferencing unit with the camera must be easily moved to accommodate changing conditions in the operating theatre.

Since the CCD camera mounted on the telemanipulator in this thesis provides visual feedback, the image transferring has to be fast and reliable, because during a surgical procedure delays and interruptions may cause harm to the patient. Several projects [13 and 28] show that the plain old telephone system (POTS), which has a normal speed of 56 kbps is not capable of transmitting vision data fast enough for live video-transmission and the quality of the image becomes degraded with significant time delay in the data transmission [9].

As stated earlier, an ISDN connection with a high-speed modem is capable of transferring data between two sites with a baud rate of 128 kbps [54] and with the video image compression techniques incorporated into the video conferencing software, gives the application the necessary data transmission speed. An ISDN solution is probably the cheapest and most commercially viable alternative today. One way to illustrate these requirements is to consider that, to be able to transmit a picture equivalent to a television image, one would need to use the equivalent capability of over 1000 telephone lines [33], this means that for the average television image a transmission rate of 56000 kbps is necessary.
For videoconferencing units used for consulting with patients in hospitals where they are already in use for other applications, it is desirable to be able to move the system around, so it can be used for patients in different locations in the hospital. This can be solved by having the phone lines connected to a network, where every room has access to the videoconferencing system [8].

New innovations in medicine, such as remote consultations, may give rise to resistance by patients. The idea of consultations via a video-monitor and not face-to-face with a doctor may be threatening to a patient needing personal reassurance and advice.

The University Hospital of Tromso and the Norwegian Telecom Research project “Telemedicine in north Norway” has used video conferences as a tool in making remote diagnoses in a number of applications including pathology, diagnosis from microscope images, dermatology, endoscopy in otorhinolaryngology, echocardiology and radiology. Experience to date suggests that the quality of telecommunication in diagnostic services is sufficiently developed that access to these facilities provides measurable benefits to the patient care and management [10].

Videoconferencing has been used in several areas in medicine and are still increasing. Surveys have been done to compare the merits of telemedicine with conventional methods. In one such study, two hundred patients were compared for the accuracy of teleconsultation for minor injuries with face-to-face consultations [11]. Color change, swelling, decreased movement, tenderness, instability, radiological examination, severity of illness, treatment and diagnosis were recorded for both telemedicine and face-to-face consultations. The severity of the illness or injury was overestimated in one
case and under-prescribed in three cases. The final diagnosis was correct in all but two cases in which mistakes were made in the teleradiology. The overall conclusion was that acceptable accuracy could be achieved by using teleconsultations [11].

2.6 Data-transmission

The project described in this thesis and other applications within the area of telemedicine, use the same means of data transmission to communicate between the remote and host site.

Video images are transmitted from one location to another by employing different data compression and transmission techniques. Although the protocol for sending the data is the same for every computer, the rate of the data-transmission relies on the particular data network, various data compression techniques and the bandwidth of the network.

In telemedicine, where the diagnostician or surgeon completely relies on the video image for interaction and for continuously maintaining visual contact, the reliability and method of transmitting such images plays an important role. In this context it is also very important to keep the transmission delay to a minimum in order that the process is as close to a real-time process as possible. The maintenance of visual contact is a safety consideration as its loss may produce catastrophic consequences in a surgical procedure, for example.
The essential minimum requirements for data transmission systems and for the hardware networks that implement the transmission is summarized as follows [13]:

- Network reliability
- Acceptable end-to-end delay
- The ability to transfer data sources with different data rates.
- Low data error rates (DER).

Plain old telephones lines (POT) have a bandwidth of about 56 kbps. This has been shown to be perfectly adequate for voice telephony, but proves to have limited applicability if services, such as the transmission of real time video images and other types of data requiring similar rates, and bandwidth, with various data compression methods, are to be implemented, [54].

A telesurgery application may require transfer of commands to a manipulator, video and speech signals, stored and real time medical images. To be able to send such data of, including real time video, 56 kbit/s is inadequate. For this purpose, other methods, or a combination of other methods and compression techniques have to be chosen. To achieve real-time, non-jerky video at least 24 frames/second is needed. If each image frame is a VGA image of 640x480 pixels and each pixel requires 3 bytes, one for each basic color group (red, green and blue), one image consists of: 640x480 pixels x 3 bytes for each frame. Then 24 frames/second demands that 640x480 x 3 x 24 bytes/second are transmitted. This works out to be 22,118,400 bytes/second i.e. approximately 20 Mb/second transmission. For an ISDN connection consisting of two 64 Kbps
multiplexed into one 128 Kbps channel, one can see that to achieve live video transmission various compression techniques has to be used.

A number of other networks are also available for video transmission that satisfies the 24 frame/sec requirement including ADSL (Asymmetric Digital Subscriber Line) and cable TV phone services as well as the previously mentioned ISDN (Integrated Services Digital Network). These networks are each capable of real time video transmission provided some video compression techniques are incorporated. The last method, since the development of the Internet, has become more common.

**Integrated Services Digital Network**

In the conjunction with the introduction of the Internet, the development of the ISDN transmission network using telephone connection has also progressed becoming widely available for commercial as well as domestic use.

ISDN stands for ‘integrated service digital network’. Telecomm networks are almost entirely digital except for the final connection from the local exchange to the home where this part is still analogue when using POT systems. Conversion from analogue to digital takes place in the local exchange. Having digital connection end to end is known as ISDN. Probably the best option available for fast data transmission today is ISDN as this is digital from one end of the system to the other and everyone with a POT line can get ISDN installed. ISDN is widely available and can transfer data to/ from sources with different bandwidth requirements, such as videoconferencing and a variety of file transfer protocols.
A number of research projects, dependent on video communication in their implementation and that use ISDN, have been described in the literature [7,13,14,29]. These projects include video image transmission using various transmission methods. A bit rate of about 128 kbps in addition to data compression techniques is according to [13] the minimum required for transmitting real time video images.

There are a variety of ISDN services available, each with different bandwidth. The most common two ISDN lines are Basic rate, for private households, and Primary rate for office use. The Basic rate is made up of two B channels and one D channel, bringing the total bandwidth to 144 Kbps (64 Kbps + 64 Kbps + 16 Kbps). Primary rate is provided to users with large capacity requirements, such as offices with digital private branch exchanges or local area networks (LANs). This is also called a T1 transmission rate and may provide a bandwidth as large as 1.544 Mbps [54].

**Asynchronous Transfer Mode (ATM)**

In addition to ISDN, which may be the most common one, there are other means of data transmission protocols. One of these methods is the ATM [13].

In today's environment, different protocols and topologies are used in local and wide area networks (LANs, WANs). These networks have generally been designed to carry one specific type of traffic, i.e. voice or images.

With this technology the same protocol can be used throughout the network. Unlike with other protocols, ATM was initially designed to be an all around protocol, that is, a single protocol to cover all types of voice, video and data communications. Therefore.
ATM may provide a solution for the high-bandwidth high-speed transmission application emerging today as well as new technology within the areas of teleconferencing, telemedicine, real-time collaboration and high-speed data transfer [13].

**Transmission Delays**

To be able to use videoconferencing in the applications described here, the video images must be real-time or at least the time delay in transmitting the images must be kept to a minimum. Using ISDN or ATM with some data compression techniques, one should be able to achieve a time delay that falls within the acceptable level for obtaining smooth video images.

In short, ISDN is a synchronous time division multiplexed structure, meaning that the transmission across the physical medium end to end digital, and capable of transmitting 128 kbps of data using two B-channels. ATM uses an asynchronous transfer mode, meaning that the transmission across the physical medium takes place within sections of cell, with each cell has the capacity of 53 bytes of data. ATM can deliver bandwidth between 45 Mbps to 1.2 Gbps [54]. The more bandwidth, the faster the data transmission that can be achieved, with or without data compression techniques.

- **ISDN**
  
  It takes 125μs for an ISDN terminal to compress/ decompress an ISDN frame.

- **ATM**
  
  It takes 2.7μs for an ATM terminal to compress/ decompresses an ATM cell [13].
As can be seen, the ATM compresses a cell ready to send in 2.7 μs, almost 47 times faster than an ISDN terminal compresses a frame ready for transmission.

In addition, one ATM cell may hold more data than one ISDN frame. According to [7], the first time video images were sent live through the Internet in 1996, the average time delay was in the range 0.5-2.0 seconds. The delay in transmission can be kept to a minimum by employing the appropriate transmission system for the specific task required. A project that used still-images for a telemedicine application on the Internet resulted in 2-5 seconds delay for 640 x 480 pixels to 10-65 seconds delay for 1000 x 1000 pixels for the transmission and the display of the images [56]. A case study report on a videoconferencing between Alice Springs and Adelaide showed it took 3-4 minutes to send a set of ultrasound images using an ISDN line (128 kbit/s) [32].

The videoconferencing unit in this project uses a LAN connection that is available in the laboratory; this connection provides enough bandwidth so that the user is able to see real-time video images from the local site. The videoconferencing package purchased for the project may also be used to send real time images using an ISDN connection that is intended to be used for implementing the system in an actual surgical environment.

ATM technology has been tested for video image transmission [30]. The project aims for developing a system to assist doctors operating in minimum invasive surgery inside complex and narrow brain blood vessels. To realize this system, the need for high quality moving images was needed. The teleoperation were performed between Nagoya and Tokyo about 350 km away from each other. A prototype for multimedia tele-surgery especially for the intravascular neurosurgery was made. The doctors could
exchange information such as real time color images, sound, text data, numerical data, force information and so on.

The operator in Tokyo controlled the operation. Tokyo and Nagoya were linked together by high-speed optical fiber network. The application used ATM and shared 156 Mbps in the network. The real time video images (24 bit, 320 x 240 pixels) had a time delay on about 11.79 ms. Computer networks (LANs/ WANs) are frequently tied up for seconds due to the transfer of large amounts of data. Such delays are unacceptable in telesurgery.

Using the right data protection techniques, it should be possible to raise the integrity of the data to a point where it is well above the minimum necessary for telesurgery applications. One alternative for ISDN may be the ATM (Asynchronous Transfer Mode) network. It has broader bandwidth than ISDN, but ISDN has the advantage of being more widely available than ATM at the present time.

In the future, ATM and fibre-optic networks having sufficient bandwidth, may overcome the current problems associated with delays in the transmission of video-images [30]. Also fibre-optic networks are inherently more reliable, are able to handle multiplexing of a wider range of data rates and have fewer transmission errors than other networks currently available [13].

2.7 Training and Simulation, Virtual Reality

In addition to applications in surgery and medicine, teleconferencing systems may have a role in the training and education of medical and surgical students and staff.
Using the same setup as for a telesurgery procedure, consisting of a local and remote site equipped with teleconferencing as well as with robotics, perhaps, in a surgical environment, medical students could observe an operation remotely. Similarly, the transmission of visual information could be combined with a simulation model to create a training tool in the education of medical professionals.

Computer simulation of surgery is one area that is emerging as a developing research interest. One system [6] used for laparoscopic simulation and training, employs telemanipulator equipment and has reported problems such as time delay in the transmission of commands and in the feedback process, accentuating the sense of being remote from the actual site of the operation. As in most telesurgery systems, this system includes a master and a slave station. The two stations communicate by means of an ISDN line. The master station consists of a computer where images from the remote site are displayed, including a graphic reconstruction of the operation scenario.

One advantage of using VR-based simulators stems from the inherent ability to repeat procedures as many times as needed [32].

Telemedicine may offer significant advantages in bringing consulting support to distant colleagues. In [31] Gutierrez describes a system for training in laparoscopic surgery that includes teleconferencing. Independent experiments were performed with this system where in one phase a mentor was present during the operation while in the other phase the mentor used voice and vision communications to supervise the surgical procedures. The results have supported the hypothesis that the remote supervision and consultation process does not affect performance in such surgical procedures.
CHAPTER 3

Laboratory Prototype

3.1 Introduction

The previous chapter reviewed published work within the broad area of telemedicine. This chapter describes how the required equipment for this project is set up in the laboratory for testing and simulating a real surgical scenario in an operating theatre. The robot system used in this project is available in the laboratory, supports serial (RS-232) interface with a personal computer and a form of off-line programming. The robot is a small industrial robot normally used for industrial purposes such as welding operations. Two standard Pentium II computers were also available for the project where these handle the interface between the remote site and the host site. In addition to the computers, two standard 56k modems and a local area network (LAN) handles the data transmission between the two sites such as the commands to the robot controller and the videoconferencing instructions.

Overall Setup

Figure 3.1 shows the laboratory setup. The two personal computers are placed at separate locations; one attached to the robot for handling the remote control and the other one for handling the command execution to the robot from the local site.
Both computers are equipped with video conferencing software and VGA circuit boards for the videoconferencing while the host site also includes a CDD camera. In addition, both computers have external high-speed modems and LAN connections for the handling of the communication between remote and host sites.

Figure 3.1 The overall setup.
At the host site, a computer is physically attached to the robot with a custom made RS-232 cable that meets the specifications for the robot controller. A software program, written specifically for handling the interface with the remote site and the command execution for the robot controller is installed on the host computer. For the communication with the remote site, a modem is installed in one of the serial ports. In this way the operator at the remote site can send commands to the manipulator at the host site, such as hold commands, move commands and other status commands.

At the remote site the computer also has a modem and software installed for command transmission. The operator uses a windows based program to execute the commands to the manipulator located at the host site.

The videoconferencing package uses the LAN network to link the two sites together by means of transmitting the video images in both directions. A CDD camera is mounted on the end-effector of the manipulator, for monitoring the movements and to provide visual feedback of the target area at the host site.

3.2 Prototype Sub-Systems

Robot Manipulator

The robot used in the project as a laboratory prototype is an industrial robot, originally designed for welding operations. It has six degrees-of-freedom and supports data transmission functions such as serial port programming and standard RS-232 protocol, making it relatively simple to connect it to a host computer using the RS232 port.
data transmission functions, also called host function, used in this project for sending various commands to the robot controller, can also be used for loading and saving jobs and reading the status of the robot.

Since the manipulator is designed for welding operations, its high torque and power as well as its physical construction makes it inappropriate for use in a real operating theatre. However, since it supports the host function, the robot system is suitable for simulating and testing the communication software and the hardware used in this project.

Figure 3.2 shows an overall picture of the manipulator, which has six degrees of freedom, making it able to move with little restriction within its working volume. It supports several types of move functions, making it capable of moving in different modes, such as linear motion and joint motion both in pulse mode and by specifying xyz coordinates. In linear motion, the end-effector moves in a straight line and in joint motion, each joint moves independently of each other. The move pattern that was chosen as the basis for remote operation at this stage was the joint mode, MOVJ, moving with an incremental value that modifies the current position. This makes it easier to control the end-effector to the manipulator, since it only moves one joint at a time, holding other joints stationary. Since the path control is not yet implemented in the software, the operator has to manually re-align the camera so that it does not loose visual contact with the target object. With the path control implemented, the manipulator will move using MOVL, linear motion, since it then automatically re-aligns the end-effector towards the target object.
In addition to the different move patterns, the operator can choose between four coordinate systems: Joint, Cartesian, Cylindrical and User defined [47]. Since the manipulator was set to move in joint motion, the joint coordinate system is used.

\[\text{Figure 3.2 An overview of the Motoman six degrees of freedoms manipulator used in this project [47].}\]

\textbf{The Computers}

For handling the communication between the remote and host sites, two personal computers with external modems and LAN connections were used. The computers, with the RS-232 interface used with the robot and the remote transmitting, are two standard Pentiums II IBM personal computers, equipped with VGA circuit boards for handling the video images from the videoconferencing unit. In addition to the VGA circuit boards, two standard external high-speed 56k fax modems are installed for the transmission of
commands to the robot controller between the host and remote sites. The videoconferencing unit uses the Local Area Network for transmitting video images between the remote and host site, since ISDN is not available in the laboratory.

One of the computers is directly connected to the robot controller with an RS-232 cable, with the appropriate pin connections. Since the handshaking signals to the robot controller differs somewhat from the standard RS-232 protocol, a special RS-232 cable had to be wired. This is attached to the serial port number 2 (COM2) while the modem that handles the data transmission between the remote and host sites, uses serial port number 1 (COM 1).

The computers are capable of much faster processing when it comes to data transmission rates (baud rates) than the robot controller, so the only restriction to the data transmission is the robot controller since the other equipment is able to meet the maximum baudrate requirements.

**Modem Interface**

In addition to the RS-232 protocol and the Local Area Network (LAN) for data transmission, the system uses modems for allowing data-transmission communication between the remote and host locations. The RS 232 protocol has its limitations when it comes to data transmission. If the cable exceeds a length of over 15 feet, the signal weakens due to resistance within the cable, so errors and misreading of the data may occur [32]. The modems are two Web-Excel 56kbps fax modems that supports AT commands. These commands are developed for controlling the modem, such as dialing a
number and picking up at the other end [48]. The commands are sent through the serial port as ASCII code to the modem, and is then interpreted and executed by the modem internally [48]. Both modems are connected to the telephone network at the laboratory. The modems are external and mounted on top of the computers. External modems are more convenient in the troubleshooting process for the interface including the modems and the software. The operator is able to monitor the changes in the modem status by the means of the indicator lights on the modem, showing the dialing process and the connection to the host site. Chapter 4 has more information about implementing the modem in the control software.

The modems are capable of transmitting data with a baud rate of 56kbps, but there are restrictions to the baud rate for communication between the robot controller and the personal computer, which is limited to 9600 bps. For avoiding inconsistent transmission rates between the modems and the robot controller, the maximum baud rate is set to 9600 bps for this project.

**CCD cameras**

For capturing the video images that are sent from the host to the remote site for visual feedback, two CDD cameras of different sizes are used. The smaller CDD camera is mounted on the end effector of the manipulator. Its small size makes it suitable for the task, and easy to attach to the end effector of the manipulator. The camera at the remote site has only a peripheral role in the set-up since it is the camera at the host site that sends the video images of the surgical procedures.
CHAPTER 4

Implementation of the Laboratory Prototype

4.1 Introduction

While the previous chapter dealt with the laboratory set up for the project, this chapter describes the implementation of the laboratory system. First the software development is described, the purpose of the software and how it is incorporated in the arrangement. Other aspects dealt with here are the hardware connection, the communication features and the RS-232 connection for the robot. The chapter explains how the implementation functions, the details of the parameter settings and required preparations that had to be made to allow the systems to work properly.

Visual C++ is one programming language that supports serial port programming and it is used here for the interface with the RS-232 ports. The visual C++ program also has a wide variety of help-files that consist of example programs that may be used by the programmer. Visual C++ supports GUI’s (Graphic User Interface) and windows based programming. Using GUI’s helps to develop windows based software faster and more reliably than writing the source code from scratch.
4.2 Software Development

One of the first issues that needed to be dealt with in the implementation of the laboratory
prototype was establishing the communication between the robot controller and a
personal computer.

The development of the communication software was undertaken in several stages. In the
first stage, the hardware connection needed to be verified by sending simple robot
commands such as initialization of the RS-232 link and turning on the power for the
servomotors that actuate the robot joints. The initial software testing began with a C++
program, developed in Visual C++, which sent a string of text characters from one
computer to another using a null modem cable between the two computers [33, 36 and
49]. After the data communication was established between the two computers, one of
the computers was replaced with the laboratory robot prototype and the test procedure of
sending simple commands, such as turning on and off the servo power, started.

The movements of the robot are, in normal direct operation, manually controlled by
means of the robot’s programming pendant. In remote operation, via the RS-232 link to
the PC, these movements are achieved by sending a command that corresponds to the
equivalent pendant command.

The C++ program contains basics features needed for serial communications between
two computers or other devices that support serial interface, such as the initialization,
opening and closing of the communication ports [34, 35, 43, 44 and 52]. A software
called MotoCom32 was obtained from a commercial supplier, Robotic Automation, for handling the data-transmission between a host computer and the robot. This software has library functions that had to be implemented in the software program for sending commands to the robot controller, so that data sent to the controller from a personal computer is packaged according to the format required by the robot controller for interpretation and execution [53]. The library functions are also capable of providing feedback information to a personal computer on the status of the robot.

By modifying the program that sends text strings between the two computers, a new program was created that was able to send commands to the laboratory robot. The commands were simple move and status commands. The library files in the MotoCom32 package were included in the developed software allowing the robot controller to correctly interpret the received data from the PC. These library files need to reside in the same directory as the executable program for handling of the data transmission process. The first program was only able to turn the servo power on and off without including any physical movements and its purpose was for testing the interaction between the MotoCom32 software and the RS 232 connection to the robot. Implementation of some simple move operations of the robot followed this initial testing procedure. The first program, with the library functions from MotoCom32 included, moved the robot in linear cartesian mode. A problem occurred using this linear mode where the operator has to provide all joint coordinates, that is, the values of each rotary joint while the aim has been to create software that is as simple as possible to use.
The program was modified so that instead of the operator needing to know all the joint angles a click of a button would be sufficient to set the direction of the movement. After various experiments with different move-patterns for the robot, an improved version of the software was written, which permitted movement of the robot with an incremental value in the desired direction when the operator pressed a specific set of keys on the keyboard. In this version, a movement upwards requires ‘A’ to be pressed on the keyboard while a movement downwards requires a ‘B’.

The experiments described above were all done with a direct pc/robot connection and the programs ran in ms-dos mode. Before implementing the modems, LAN and the videoconferencing package for the remote operation mode, a simulation of the modem and LAN connections was performed.

For this, a null modem cable was used for testing the system where both computers were located in the same room. The computer acting as the host site was connected to the robot with the custom made RS-232 cable. The second computer acted as the computer at the remote site. The remote computer sent data to the computer at the host site. The software on this computer was continuously reading the serial port for any incoming data sent from the remote site. When it received data, the software tried to match it with one of many switch statements within the software source code. If any of the incoming data matched one of the statements, the case body of that statement was executed and the robot controller received a command. The command received could be anything from turning the servo power on or off, reading the position or a movement command.
After this, simulation of the data transmission between the host site and remote site was completed where the modems were not incorporated and the videoconferencing package was not used. Until now the data transmission only went through serial cables connecting the computers and the robot physically together. With the implementation of the modems, the computers were able to dial each other up and make a remote connection between the host and remote sites.

The remote site was now chosen to be an office located two floors and about 50 meters from the host site the laboratory where all the previous testing and experiments were done.

A computer program was developed that used a modified version of previously written software that supported serial transmission in order to send simple AT commands to the modem [48]. The AT commands are used to control the modem; they are used for the modem to dial a number and for the receiving modem to pick up an incoming call automatically, forming a remote connection. Having experienced some problems with the implementation of the modem in the software, it was decided to use the hyper-terminal program that is available through the Windows 95 software from Microsoft. Using this hyper-terminal program helped to eliminate these problems. It was used to isolate one of the sites, either the remote or the host site, so one could isolate any problems with the developed software and identify the specific modules where the error occurred. The modified software was installed on the computer at the remote site. This software dialed the number to the laboratory where the computer with the hyper-terminal program
running, picked up the incoming call from the remote site. In fact, this was the only function performed by this program, namely dialing the number and waiting for the connection to be made at the host site. It did not transmit any data between the two sites.

Having established automatic dialing, pick-up and connection between the two sites, the development of the software for sending robot commands between the two sites using the modems was begun.

All the software programs mentioned so far run in MS-DOS mode. A Windows based software is considered more user friendly than a MS-DOS based program when it comes to the robot control aspects of the project. Windows based programs equipped with pushbuttons, that the operator could click with a mouse are better suited for use by medical operators. The pushbuttons are labeled with the directions the manipulator is to move, making it easier for the operator to execute manipulator commands.

Additional software was developed, using the MFC program within the Visual C++ compiler [52] instead of MS-DOS based software. The dialog box system in the software is a modified version of a Visual C++ module were the programmer only has to fill in the source code for each of the pushbuttons. Initial problems with simulating operation of all available pushbuttons were overcome by appropriate allocation of global variables.

The user friendly, pushbutton-based dialog box is shown in figure 4.1, where the robot control commands are integrated into simple, high-level user commands. In addition to
pushbuttons for movements, the software also has buttons for controlling the modem, canceling a command, turning the servo power on or off and resetting the robot controller.

![Figure 4.1 The windows based control panel for the robot controller](image)

The final stage of the software development required all elements to operate together. The current setup is: The program at the remote site communicates with the robot through an RS-232 cable allowing control commands to be automatically executed. The new windows based software was installed at the remote site, placing the call to the host site, and sending commands to the host computer. For visual feedback of the position of the manipulator at the host site, a video camera is mounted at the end effector of the manipulator. The CCD camera is part of a video conferencing package, used to send video images over the local area network (LAN), allowing the operator to control the manipulator. Since the laboratory at the present time is not equipped with an ISDN line for testing the videoconferencing package, the LAN network was used instead. Using the...
LAN, the operator was able to see real-time video images as visual feedback from the host site.

Figure 4.2 shows the structure and the data flow of the software package created for this project. First, the initialization and configuration setups are executed by starting the program. Subsequently, the software is ready to send and receive data through the serial port using both modems. The call is placed from the remote site to the host site, and after the connection is established, exchange of data can take place. The data flow continues till the operator at the remote site disconnects the modem and ends the connection with the host site. The data that is transmitted between the remote and host sites, runs in a loop as long as the operator keeps sending data to the host site or does not exit the software. The RS-232 connection between the host computer and the robot controller allows the automatic execution of commands received from the remote site.
Figure 4.2 Flowchart for data transmission between the remote and host locations.
(ACK is the acknowledge signal as in Fig.4.9)
4.3 Hardware Aspects

In order to achieve the objectives outlined in Chapter 3, the hardware components of the system had to be carefully selected to meet the requirements of the project. These requirements include consideration of the data transfer rates of the transmission between the computers through the phone lines as well as the capabilities of the CCD cameras, modems and the video conferencing software [42, 50 and 51]. The implementation requires that the following hardware be integrated.

The four major steps required to install the hardware for the videoconferencing unit are:

- Camera and computer interfacing.
- Mounting camera on manipulator end-effector.
- Installing audio connection.
- Connecting to the LAN network (ISDN for later stages).

The circuit board has Plug-and-Play functionality and was installed into an empty slot in both computers. After these video circuit boards were inserted, the mounting of the rest of the equipment was done. The cameras were plugged into the video circuit boards in the computer and placed at the preferred location.
Figure 4.3 shows the hardware setup used in the laboratory. At the host site the camera is mounted on the end-effector of the manipulator. Currently, it is attached in a frame that is secured to the robot with elastic rubber bands, making it easy to mount and dismount when not in use. Due to the length of the cable being used and the restrictions of the length of the RS 232 cable, the host computer has to be located within 15 feet of the robot controller [37]. If the RS 232 cable is too long, the voltage in the cable may reduce and the signal to noise ratio also drops, causing misreading and errors in the data transmission.

The videoconferencing software is equipped with audio connection, so the remote site is able to talk to the host site. There are two microphones for two-way communication and the computers are equipped with speakers.

The computers are connected to the local area network (LAN), at both sites. The videoconferencing software can be implemented on both LAN and ISDN connections and uses the computer IP address instead of dialing a phone number when it is connected through the LAN. The IP addresses of both computers are put in the dialing properties instead of phone numbers.

The CCD cameras are able to produce live video images that provide visual feedback of the target area of the surgery as well of the position of the manipulator. The camera at the host site has auto-focus, ensuring clear images of the scene that is being monitored are transmitted.
Figure 4.3 The laboratory setup showing both host and remote sites.
4.4 Communications

The hardware implemented in the project provides communication between the host and remote sites.

The communication aspect of the project uses well-known methods such as the RS-232 protocol and hardware handshaking. Between the computers and the modems, standard RS-232 cables are used [37, 38, 39, 40 and 41]. As mentioned earlier, for the direct connection between the robot controller and the host computer a custom made cable had to be used, to match handshaking requirements between the robot and the personal computer. The set up for the RS-232 cable for the robot controller is specified in the user manual.

Communication Features

The system uses the RS-232 standard for the serial interface with each other [45 and 46]. The devices are divided into two groups, the PC is a DTE (Data Terminal Equipment), and the modem and the robot is a DCE (Data Communication Equipment). DCE equipment normally uses DSR (Data Set Ready) as the main handshaking line to signal the DTE that it is powered up and ready to receive transmissions. It can also use CTS (Clear To Send) as a subsidiary handshaking line [35]. By using this information, the programmer is able to monitor whether the manipulator is on-line.
DTE equipment, on the other hand, uses DTR (Data Terminal Ready) as the main handshaking line to signal the DCE that it is ready to receive, and RTS (Request To Send) as a subsidiary handshaking line. By convention, these handshaking lines carry a positive voltage when transmission is enabled and a negative voltage when it is suspended [37]. This version of handshaking is also called hardware handshaking.

In the software, developed for this project, the handshaking is handled by a module called the device control block (DCB) structure [52], see appendix A.7, where with this structure the operator is able to choose a specific setting, depending on the system requirements.

With this DCB the operator can set the baud rate, parity, one or two stop bits and so on. An important aspect is that the settings are the same for both devices, the computer and the robot. If the settings are not the same, errors will occur during execution.
Figure 4.4 Flowchart for the handshaking signals
Figure 4.4 shows the flowchart for the hardware handshaking procedure. Data is transferred in two directions. The minimum number of lines necessary in two-way communication is three, transmitted data in both directions and signal ground. The addition of one handshaking line in each direction brings the total to five [35]. Figure 4.5 shows the five necessary lines for transmission in two directions.

![Diagram showing hardware handshaking lines](image)

**Figure 4.5** The five required lines for two-way data transmission.

In the early stages of the project, a NULL modem cable was used to check the character transmission between the two computers. A NULL modem cable is a regular RS-232 cable with the handshaking signals crossed. Handshaking signal 2 goes to signal 3, 3 to 2 and so on. After the testing was completed, the NULL modem cable was replaced with the RS-232 cable required for the robot.
**RS-232 Connection to the Robot**

The existing technology on the RS-232 standard is well described in the literature, but the specific connection used in this project in communication with the robot controller is described here for completeness. The robot itself is described in chapter 3.

Figure 4.6 (a) and (b) shows the pin connection that had to be adopted for the communication to the robot controller [47]. This pin connection set-up differs somewhat from the standard RS-232 cables as shown in figure 4.5.

<table>
<thead>
<tr>
<th>FG</th>
<th>Protective ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>Sending data</td>
</tr>
<tr>
<td>RD</td>
<td>Receiving data</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to send</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to send</td>
</tr>
<tr>
<td>SG</td>
<td>Signal ground</td>
</tr>
<tr>
<td>CD</td>
<td>Carrier detect</td>
</tr>
<tr>
<td>DTR</td>
<td>Data terminal ready</td>
</tr>
</tbody>
</table>

**Figure 4.6 (a)** RS-232 configurations for the Robot controller.
Figure 4.6 (b) RS-232 configurations for the Robot controller.

An important issue is the length of the cable not being too long otherwise the voltage levels fall outside the permitted boundaries. The maximum length of an RS-232C cable is generally 15 feet; in this project the robot and the PC are connected together with a 5 feet long RS 232 cable.

A breakout box was used to allow easier troubleshooting the connection with a loop between the devices. The breakout box is equipped with LED indicators that show whether each line is high or low, thus identifying handshaking problems [47].

The data transfer rate, also known as the baud rate, must be the same for both communication devices. The most common rates (in bps) range from 110 to 19200, and for this project 9600 bps was selected since this is the maximum baud rate the robotic
device is capable of handling. A higher baud rate would give rise to errors in the data transmission.

The data transmission function is divided into three parts as shown below:

1. DCI (Data Communication Interface)
2. Stand-alone function
3. Host control function.

For this project, the robot controller has to be set to Host Control Mode for communicating with the personal computer. Figure 4.7 shows the connection between the host computer and the robot; please refer to Appendix A4 for the instructions on this procedure.

![Figure 4.7 Robot controller setup in Host Control Mode](image)
The operator can check that the robot controller is in the correct mode by entering the remote display command on the programming pendant, which is shown in Figure 4.8. The screen that appears should then show the words 'command mode'.

![Remote Display Diagram]

Figure 4.8. Remote display, showing command mode operation setting [47].

“COMMAND MODE” shows the user that the robot is in the correct mode for remote operation. “CURR”, “PREV” and “DISP” shows the status of the robot in remote mode. These change when commands are sent to the robot controller. If the last command sent to the robot controller is an ‘IMOV’ command, the word ‘IMOV’ appears beside ‘PREV’. The YASNAC MRCII has one serial interface (RS-232C) port where a parameter has to be switched on to enable the port. The user has to enter local mode to do this, please see appendix A4 for further details for the setup and display.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents and Set Value</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS000</td>
<td>Standard port 1 (Playback box)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Protocol specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: NON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: System reservation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: BSC LIKE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(data transmission function)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: FC1</td>
<td></td>
</tr>
</tbody>
</table>

For parameter settings, please refer to Appendix A.5.

### 4.5 Data Transmission

Before the data transmission between the host computer and the robot controller may start, all the settings and parameters have to be correctly set. Without the correct settings no action will result if the operator tries to execute a command to the robot. Also the manipulator may do something unexpected if the wrong settings are chosen.

Figure 4.9 shows how the transmission from the host computer to the robot controller proceeds. If the host control function is ON, the system is ready to receive. The ENQ code is sent from the computer to establish a data link. After establishing the data link, the data is sent from the computer. When the transmission is completed, the computer is ready to receive. Having established the data link a response to the sent data from the computer is received from YASNAC MRCII to terminate the transmission.

The header number and the sub-code number distinguish the type of data [47].
Figure 4.9 Data transmission protocol between the robot controller and the personal computer.

Robot Control Function

The previous section dealt with the data-transmission from the host computer to the robot controller. There is an important part of this transmission process that needs some special attention. The commands to the robot controller have to be sent in a single block. If not,
the operator may experience some problems with the transmission to and from the robot controller.

The transmission of the commands proceeds as follows. The host control function has to be activated. After the data link is established, see Figure 4.9, commands are sent. The header number, see Appendix A.3, distinguishes commands and file data. Figure 4.10 shows how a command to the robot controller must be in a single, complete block. After each transmission is completed, the computer must get ready to receive messages. The YASNAC MRCII sends out the ENQ to establish a data link. After the data link is established, the YASNAC MRCII sends out a response to the command and terminates transmission.
Command format:

\[
\text{SOH} \quad 01,000 \quad \text{STX} \quad \text{COMMAND Data1, Data2, Data3} \quad \text{CR} \quad \text{ETX} \quad \text{BCC}
\]

Response format:

1.

\[
\text{SOH} \quad 90,000 \quad \text{STX} \quad \{0000, \text{or error code}\} \quad \text{CR} \quad \text{ETX} \quad \text{BCC}
\]

0000 : Normal completion

Error code : Number with four digits other than 0000. In case of smaller than 1000, 0 is added before the number.

\[
\text{SOH} \quad 90,001 \quad \text{STX} \quad \text{Data1, Data2…, DataN} \quad \text{CR} \quad \text{ETX} \quad \text{BCC}
\]

**Figure 4.10** Command and response blocks for the robot controller

### 4.6 Modems

Modems are used in this project to transmit robot commands from the remote site to the host site and vice versa. The modems used are 56kbps-fax modems. The modems transform the data received from a computer in serial form into a form suitable for transmission through the telephone system, and transform it back to serial form providing a convenient method of transmitting data between two sites.
The modem

Figure 4.11 shows how the modem is set up. The modems used in this project are external modems and they come with all the necessary connection accessories. The use of an external modem is preferable for the testing, as the LED indicators on the modem can be observed during normal operation.

![Diagram of modem setup]

Figure 4.11. An example of a modem setup

Implementing the modems into the custom made software is possible if the software is able to transmit data by means of its serial port. The modems used in this project are Hayes compatible, so they can be controlled by using various AT commands for performing the actions the operator wishes to be executed. The operator types the AT command in a data string and when the program executes, following the corresponding pushbutton being activated, it sends these AT commands to the modem.
By using AT commands, one can activate the modem to dial a number, answer an incoming call automatically or hang up the line.

Note that hardware handshaking signals (DTR, DTS, and so on) are also used to control communications between the computer and the modem. These handshaking signals are not passed along the phone line to the remote modem and computer, they are used for communicating between the modem and the computer. To enable handshaking between the two computers using modems, software handshaking must be used.

A helpful tool in troubleshooting with the modems is the Hyper-terminal program that is available in Windows 95 and later versions. As previously described, the programmer can eliminate problems occurring during the testing procedure by using the Hyper-Terminal program during the software development stage. The operator may use this program to dial a number while at the other location the communication package written for this project automatically establishes the data link via the modem. If a data link is not established using these two software packages, one may use the Hyper-terminal program at both locations to isolate the problem.

An important consideration is that the modems are properly installed during the testing and development phases. Both software programs may work correctly, but if the modems are not installed properly the operator will receive corrupted data, with the risk that commands sent may not be executed or uncontrolled movements of the manipulator may occur. To eliminate the problem it is important to ensure that the proper drivers are
installed for the modems; usually these are supplied with the modem when purchased [48].

The modems implemented in the project are used only for transmitting commands to the robot controller from the remote site to the host site and not for the videoconferencing communications. The video communications is implemented via the LAN network.
CHAPTER 5

Vision Communications

5.1 Introduction
The operator uses visual feedback from the host site to be able to control the manipulator from the remote site. To achieve this visual feedback, a videoconferencing package is implemented in this project. This chapter describes the videoconferencing unit used in the project for transmitting live video-images between the host site and the remote site. The video-images originate from a CDD camera attached to the end-effector of the manipulator. Since the videoconferencing package is capable of sending real-time images between the host and remote sites, the operator at the remote site is able to follow the ongoing procedures at the host site as they occur. Videoconferencing (VC) is the combination of dedicated audio, video and communications networking technology for real-time interaction and is often used by groups of people who gather in specific settings to communicate with other groups of people. Desktop videoconferencing combines personal computing with the same functions to provide real-time interaction from a typical personal computer and the available packages are relatively low cost.
5.2. Video Conferencing

The videoconferencing software used in the project was obtained as it satisfies the main requirements of the project. The key-features of this software are two-way live action, color video transmission capable of using both ISDN and LAN for transmission, easy installation and have a user-friendly interface. The package comes with CCD cameras and the necessary VGA add-on boards for the computers. The minimum system requirements are a Pentium computer and ISDN or LAN connection.

The videoconference package is able to send live video images between the host site and the remote site. Using this technology, the surgeon at the remote site will be able to follow the procedures at the host site in real time.

Important Features

An important issue regarding the videoconferencing unit is the quality of the video images that are being transmitted between the two sites. Poor image quality may create problems for the surgeon(s) at the remote location, since he or she would then not be able to follow the surgical procedures properly. There are different aspects that may contribute to poor quality video images. One is that the software is not capable of transmitting video images with high enough resolution or enough frames per second for the viewing of real time video. The connection between the two sites must be interference free as well.

Time delays in the transmission of video images must also be kept to a minimum for real-time viewing. The software and the communication lines must be able to cope with the
large amount of data the video-transmission needs for obtaining real-time images. The audio part of the videoconferencing is also important for voice communications between surgeons. The package used in this thesis supports audio contact between the two sites allowing simultaneous voice and vision linking.

*Video Compression Techniques*

To be able to achieve real-time transmission of video images using a video conferencing unit, data compression techniques must be implemented in the videoconference software. As mentioned in Chapter 2, an image of 640x480 pixels requires too much data to be sent over an ISDN line of 128 kbps with a rate of 24 frames per second for real-time video, without data compression techniques.

Since the early 1980’s, new and improved data compression techniques have been developed, such as the H.120, H.130, H.320 and H.323 to mention a few [57]. The videoconferencing unit used in this thesis uses the protocol called H.320. This technique is a hybrid process combining a number of different data compression techniques. Table 5.1 gives an overview of the various protocols making up the H.320.
Table 5.1 Video compression protocols included in H.320

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.261</td>
<td>A video coding algorithm for the compression of signals at data rates from 64 kbits/s to 1920 kbits/s.</td>
</tr>
<tr>
<td>G.711, 722, 726 and 728</td>
<td>A series of algorithms for the compression of audio signals at data rates from 16 kbits/s to 64 kbits/s.</td>
</tr>
<tr>
<td>H.221</td>
<td>Specifies the frame structure for multiplexing video, audio and data into a single bit stream.</td>
</tr>
<tr>
<td>H.230 and 242</td>
<td>Specify the handshaking protocols between H.320 compliant equipment.</td>
</tr>
<tr>
<td>H.233</td>
<td>Allows manufacturers to select from three methods of encryption in their H.320 compliant equipment: DES, used in the US; SEAL used in Japan; and BCRYPT, used in the UK.</td>
</tr>
</tbody>
</table>

To be able to understand the extent of data required to be transferred for achieving real-time video, the H.261 shall be examined in a little more detail.

H.261 is a video-coding standard published by the ITU (International Telecom Union) in 1990. It is designed for data rates that are multiples of 64 Kbit/s, and is sometimes called p x 64 Kbit/s (p is in the range of 1-30) [57]. These data rates suit ISDN lines, for which this video codec was designed.

One of the major problems in defining an international standard for video conferencing was the fact that there are two different video and television standards. One of these is NTSC in North America and Japan that uses 525 lines per interlaced frame at 30 frames per second. The other standard used by most of the other countries has 625 lines per interlaced frame at 25 frames per second. To eliminate the problem of interoperability among systems with different formats, a new common intermediate format (CIF) was
adopted. Both the 625 and the 525-line systems need to include pre- and post processing modules to convert to and from CIF.

CIF is a non-interlaced format. It is based on 352 pixels per line, 288 non-interlaced lines per frame at 30 frames per second. For low bit-rate applications, in addition to CIF, video coders may also use a quarter-CIF (QCIF) format, which has half the number of pixels and lines required for CIF. Support for CIF coding and decoding is optional, however, all coders must be able to operate using QCIF. This format, CIF and QCIF, is called the H.261 video-coding standard and it is developed for use with ISDN [57].

The videoconferencing unit used in this thesis supports the H.320, which includes the H.261 video-encoding standard. Table 5.2 shows an overview of the image size and resolution that is achievable with this unit.

**Table 5.2.** Frame rates achievable using Eye Link video conferencing software.

<table>
<thead>
<tr>
<th>Format</th>
<th>Resolution</th>
<th>Frames per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCIF (Quarter Common Intermediate Format)</td>
<td>176x144 pixels, 30 frames per second</td>
<td></td>
</tr>
<tr>
<td>CIF (Common Intermediate Format)</td>
<td>352x288 pixels, 26 frames per second</td>
<td></td>
</tr>
</tbody>
</table>

Using the H.261 video-encoding protocol, one can compress the images to a bit stream that suits ISDN protocols. If in H.261 the compression phase of the processing is left out, the CIF and QCIF images will constitute too much data for an ISDN line of 128 kbit/s to
transfer at a rate suitable for real-time video. Table 5.3 shows an overview of QCIF and CIF in uncompressed bit rates [58].

<table>
<thead>
<tr>
<th>Picture Format</th>
<th>Luminance Pixels</th>
<th>Luminance Lines</th>
<th>H.261 Support</th>
<th>Uncompressed Bitrate (Mbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 frames/s</td>
</tr>
<tr>
<td>QCIF</td>
<td>176</td>
<td>144</td>
<td>Yes</td>
<td>2.0</td>
</tr>
<tr>
<td>CIF</td>
<td>352</td>
<td>288</td>
<td>Optional</td>
<td>8.1</td>
</tr>
</tbody>
</table>

5.3 Communication Lines and Videoconferencing

Video conferencing units supporting data compression techniques, such as the H.261 are capable of transmitting real-time video images from one location to another. The ISDN is one of the networks that is developed specifically to handle such data compression techniques [57].

ISDN stands for Integrated Services Digital Network. Its best advantages come from the fact that ISDN is purely a digital form of networking, through the whole process from sender to receiver. This makes ISDN capable of handling more data, more cleanly and effectively than existing analog telephone technology, while also opening the door for more advanced digital forms of communication. In contrast to the 1960’s analogue Plain Old Telephone Service (POTS), ISDN provides faster, more reliable service and has more features than has been available previously. Integrated Services Digital Network
(ISDN) has become an extension of the telephone system allowing the transmission of signals to be in digital format.

**Different Forms of ISDN**

ISDN has been available for a decade and is widely available around the world, although not uniformly. Table 5.4 shows the services and primary characteristics of ISDN and lists of the various channels available for the transmission of video images and other data.

<table>
<thead>
<tr>
<th>CHANNEL TYPE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>D channel</td>
<td>(Device channel) used for transfer of signaling information between the user and the network and for packet transmission.</td>
</tr>
<tr>
<td>B channel</td>
<td>(Bearer channel) used for data transmission over the local loop.</td>
</tr>
<tr>
<td>H channel</td>
<td>(Higher rate channel) used for services that need higher transmission rates than a single B-channel.</td>
</tr>
<tr>
<td>B-ISDN</td>
<td>(Broadband ISDN) channels enable applications requiring speeds higher than those defined for H channels. B-ISDN standards allow transmission rates as high as 622.08 Mbps.</td>
</tr>
</tbody>
</table>

The videoconferencing unit used in this thesis, and also in the future clinical implementation, uses the Basic Rate Interface ISDN (BRI- ISDN), referred here to as ISDN and includes two full duplex B channels and one D channel. The B (representing signal bearer) channel is either 56 Kbps or 64 Kbps; it carries voice, video images and audio. The D (data) channel has a bit rate of 16Kbps; it carries service request messages and signal information that describes the application to be transmitted.
The two B channels of ISDN can be multiplexed for a full 128 Kbps of bandwidth, which is significantly faster than the 56 Kbps speed offered by high-speed modems over analog telephone lines. The BRI-ISDN provides the lowest bandwidth for real-time videoconferencing [54]. There are also other combinations of ISDN available that offer wider bandwidth than the BRI, but they will not be discussed here and for further details about other ISDN options, please see reference [54].

For the testing procedures within the laboratory, the LAN connection is used for simulation and testing instead of an ISDN connection. The ISDN network is not available at the time of writing this thesis. The LAN connection provides broader bandwidth than the BRI-ISDN, so all the video images sent are in real-time.

ATM (Asynchronous Transfer Mode)

ATM is a broadband switching technique for use with a desktop through the WAN (Wide Area Network) and is the latest form for digital packets, becoming popular as an information transmission technology. It is a set of international standards, established by the ATM Forum, for sending large amounts of voice, data and video information simultaneously over a single network at speeds up to a thousand times faster than previously possible [31].
Features of ATM

ATM is based on the concept of fast packet switching and it grew out of work done on Broadband Integrated Services Digital Network (B-ISDN). It combines the best of packet switching, which is designed for data, and circuit switching, used to support delay sensitive applications such as voice and video transmission.

Unlike traditional token ring and Ethernet technologies, ATM utilities have short fixed length packets called cells. There are 53 bytes in each cell. Figure 5.1 shows an example of an ATM network, where connections are virtual circuits in which bandwidth is allocated dynamically, on demand, when it is needed. This form of networking can transport virtually every electronic communications format from e-mail to telephone calls to movies.

![ATM network structure](image)

**Figure 5.1** ATM network structure
The network processes ATM cells one at a time. The contents of the routing fields in the header are used to direct the cell on to the next leg of its journey. This means that adjacent cells need have no relationship with each other. They can carry different types of data and originate from different places/ equipment. This freedom to multiplex cells results in flexible bandwidth.

An ATM network regardless of the bandwidth required by each service can accommodate all services. The target cell loss probability in an ATM network is less than one lost cell in a thousand million (one in $10^9$). Because of its special ability to multiplex sources with different data rates (surgical-robot command, voice/video link, stored and real time medical images etc), and its low cell loss rate, ATM is an ideal technology for telesurgery applications. ATM is capable of providing broader bandwidth than ISDN, but ISDN has the advantage that it is far more common and cheaper.
CHAPTER 6

Path Control

6.1 Introduction

Although not yet implemented in the robot control system for automatic operation, a method of controlling the position and orientation of the camera during the monitoring process is described here. It is proposed that in the final version of the prototype system, the position control of the robot end-effector be achieved by the means of a joystick. The signals from the joystick are to be used for sending robot linear move commands that position the end-effector but in this type of move the orientation does not change with respect to the base coordinate system. However, in the final implementation the camera has to be always directed towards the target and therefore its orientation, together with the end-effector to which it is fixed, has to be continually updated during these moves.

To achieve this, an automated process needs to be formulated such that the linear movement of the end-effector is accompanied by an automatic re-alignment of the camera axis so that the target is never out of view. Furthermore, the image that is transmitted to the operator must remain horizontal and not be rotated. In order to adjust the orientation of the camera attached to the end effector after each incremental linear movement of the manipulator so that it is always directed towards the required location of the surgical procedure and oriented horizontally, another command needs to be implemented to perform this action. It is envisaged that this command will be able to execute at very short time intervals where the overall effect is that the camera
remains pointed at the site of interest automatically while the use of the joystick is continuously changing the position to the robot [59].

To calculate the new angles necessary for realigning the end-effector a set of rotation matrices needs to be determined. Although the position of the manipulator can be obtained at each instant automatically, the method used to verify the approach described here is not automated. The testing and development for the path control was performed manually in the laboratory in order to demonstrate the validity of the process.

The position movements in the current prototype is achieved by sending the MOVJ robot command, which simply translates the end-effector in straight line movements, moving one joint at the time, but also results in the camera losing visual contact with the target area. The move from the Initial position to Position 1 in figure 6.1 shows the camera orientation in both positions. As well as being pointed towards the target area, the camera has to transmit an image that is not rotated away from the horizontal. The re-alignment to account for both these aspects is performed manually at the present time where a windows-based program with a pushbutton dialog box is the interface system between the operator and the robot controller.

6.2 Theory

The process of re-aligning of the end-effector, with the camera attached to it, while not rotating the video image away from the horizontal, requires a datum or calibration position to be determined first. This calibration position provides the initial direction
and position of the target area, from which the subsequent camera directions can be calculated.

This consists of the location of both the robot and the target. For the verification of the procedure in the laboratory experiments, the calibration position is represented by the cylindrical object on the table shown in figure 6.4 and is also represented by the calibration point in figure 6.1. The vector $a^0$ locates this calibration point and its parameter values were measured manually for the purpose of the laboratory testing procedure, but in the clinical implementation in the future, the calibration process can be made to be less cumbersome.

Figure 6.1 The setup for calculating the re-alignment of the end-effector after a MOVL command.
The manipulator operates in two different coordinate systems, the tool coordinate axes and the robot base axes. The tool coordinate axes are for orienting the end-effector, while the base coordinate axes represent the reference frame of the robot. Figure 6.2 shows the two coordinates systems as they relate to the robot and the end-effector. Using the programming pendant of the robot, the operator can ascertain the coordinates and wrist angles of the end-effector relative to the base coordinate system of the robot. The values obtained by this process are then used to determine the rotation angles required to re-align the camera-axis. This axis corresponds to one of the tool axes at any given position of the end-effector. These rotations angles, \( \theta \), \( \phi \) and \( \alpha \), that are needed to re-align the end-effector are the parameters of the corresponding rotation matrices, \( R_X \), \( R_Y \) and \( R_Z \), which are described later.

Figure 6.2 The base axes and the tool axes [47]
To re-align the camera (the camera is mounted to the end-effector so that its axis coincides with the 6th axis of the robot), the orientation of the z tool axis at both the initial position and at position 1 is first calculated from the geometry in figure 6.1. This orientation, however, has to be converted to the base coordinate system since all robot commands are sent in this form. This conversion is equivalent to calculating the inverse kinematics.

Having manually adjusted the orientation of the camera so that the calibration point is in the center of its field of view, the vectors $a^0$, $p^0$ and $d^0$ are all known: $a^0$ is assumed to be the known location relative to the base coordinate system $X_B$, $Y_B$, $Z_B$, while $p^0$ is acquired by reading the X, Y and Z coordinates of the end-effector position from the robot controller. This last step can be automated in a future implementation of the control software.

The calibration point coordinates, $a^0$, were measured manually from the robot origin position. Knowing the location of the calibration point and the orientation of the end-effector axis, one can calculate the direction vector, $d^0$, of the end-effector. Moving the robot now from the initial position, with a MOVL command, changes the position of the end-effector to position 1 in figure 6.1. The target will now no longer be in the center view of the camera attached to end-effector. For re-aligning the camera back to the target object, rotation matrices, $R_X$, $R_Y$ and $R_Z$, need to be formulated, which contain the corresponding rotation angles $\theta$, $\phi$ and $\alpha$. 
Thus, \( d^0 \), which is the required orientation of the camera axis at the initial position, is given by:

\[
d^0 = a^0 - p^0
\]  

(1)

At the initial position, the orientation of the camera axis may also be obtained by reading the orientation values from the pendant simultaneously with the X, Y and Z coordinates as described earlier. The calculated values of \( d^0 \) can thus be checked with those obtained from the pendant. These angular parameters represent the rotation angles in the transformation matrices needed to transform the base coordinate system to the tool coordinate system. The two coordinate systems are shown in figure 6.2.

The transformation, \( T \), between the two coordinate systems, is the matrix product of three consecutive rotations about the \( Z_B \)-axis, the \( Y_B \)-axis and the \( X_B \)-axis such that:

\[
T = R_z \cdot R_y \cdot R_x = \begin{bmatrix}
C\alpha & -S\alpha & 0 \\
S\alpha & C\alpha & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \cdot \begin{bmatrix}
C\phi & 0 & S\phi \\
0 & 1 & 0 \\
-S\phi & 0 & C\phi \\
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 \\
0 & C\theta & -S\theta \\
0 & S\theta & C\theta \\
\end{bmatrix}
\]  

(2)

where, \( \alpha, \phi \) and \( \theta \) are the rotation angles, respectively, while ‘\( C \)’ in \( C\theta \) and ‘\( S \)’ in \( S\theta \) represent cosine and sine respectively. The rotations in \( T \) are about successive moving axes and thus each rotation matrix post-multiplies the previous one. Combining the rotations in (2) gives:

\[
T = \begin{bmatrix}
C\phi C\alpha & -S\alpha C\theta + C\alpha S\phi S\theta & S\alpha S\theta + C\alpha S\phi C\theta \\
S\alpha C\phi & C\alpha C\theta + S\alpha S\phi S\theta & -C\alpha S\theta + S\alpha S\phi C\theta \\
-S\phi & C\phi S\theta & C\phi C\theta \\
\end{bmatrix}
\]  

(3)
Thus from (3), if $\mathbf{V}_B$ is a vector expressed in the base coordinate system and $\mathbf{V}_T$ is the same vector in the tool coordinate system (taking only vector directions into considerations) then:

$$\mathbf{V}_B = \mathbf{T} \mathbf{V}_T$$

Also, the columns of $\mathbf{T}$ are the $X_T$, $Y_T$ and $Z_T$ axes expressed in the base coordinate system

$$\text{Thus, } X_T = \begin{bmatrix} C\alpha C\phi \\ S\alpha C\phi \\ -S\phi \end{bmatrix} \text{ and } Z_T = \begin{bmatrix} S\alpha S\theta + C\alpha S\phi C\theta \\ -C\alpha S\theta + S\alpha S\phi C\theta \\ C\phi C\theta \end{bmatrix} \quad (4)$$

Returning to equation (1), the unit vector of $\mathbf{d}^0$ represents the camera axis i.e $Z_T$ at the initial position. Hence, in the initial position where the camera is aligned towards the target object $\theta$, $\phi$ and $\alpha$ are obtained directly from the programming pendant of the robot controller. Also, $Z_T$ from (4) can be verified with the calculated $\mathbf{d}^0$ in (1) since these are equivalent. Now, assume that the robot has been moved linearly from the initial position in figure 6.1 to a nearby position, position 1. In this new position the camera axis is still oriented in the direction specified by $\mathbf{d}^0$ since the move was a linear one only.

Having moved the end-effector a certain distance in a straight line, the re-orientation of the camera is required by calculating the new rotation angles, $\theta$, $\phi$ and $\alpha$ for aligning the camera once again towards the target point. From Figure 6.1, the new required direction, $\mathbf{d}^1$, can be determined from

$$\mathbf{d}^1 = a^0 - p^1 \quad (5)$$
where \( p^1 \) is the position coordinates of the tool center-point, the end of the camera lens, acquired from the pendant, and \( a^0 \) is already known. Hence, the unit vector \( \hat{d}^1 \) can be calculated and equated to \( Z_T \), the new required orientation.

\[
Z_T = \begin{bmatrix}
S\alpha S\theta + C\alpha S\phi C\theta \\
-C\alpha S\theta + S\alpha S\phi C\theta \\
C\phi C\theta
\end{bmatrix} = \hat{d}^1
\]  

(6)

Equation (6) can be solved for only two of the three unknown rotation angles since the three scalar equations in (6) are not all independent. Another equation, representing the requirement that the image should not be rotated as the camera moves from one position to the next, can be obtained as follows. From figure 6.3, where the camera is moved from a generalized position 1 to another arbitrary position 2 the condition that the image does not rotate can be considered to be equivalent to the case where there is no rotation about the camera axis, the \( Z_T \) axis, as the robot moves. This condition can be stated as one where the angle between the projections of the \( X_T \) and \( Z_T \) axes onto the \( X_B Y_B \)-plane remains constant during the movement of the robot between the two positions.
The required condition for non-rotation of the image can then be written as

$$\cos \beta = \frac{(X_T)_{xy} \cdot (Z_T)_{xy}}{|X_T|_{xy} \cdot |Z_T|_{xy}} = \text{constant}$$  \hspace{1cm} (7)$$

where \((X_T)_{xy}\) and \((Z_T)_{xy}\) are the projected vectors of \(X_T\) and \(Z_T\) respectively and \(\beta\) is the angle between these projected vectors. In the initial position the angle is shown in figure 6.3 as \(\beta\) while in the next position it is \(\beta'\). For the condition to be satisfied \(\beta' = \beta\).
From (4) it can be obtained:

\[ (X_\gamma)_{xy} = \begin{bmatrix} C\alpha C\phi \\ S\alpha C\phi \\ 0 \end{bmatrix} \quad \text{and} \quad (Z_\gamma)_{xy} = \begin{bmatrix} S\alpha S\theta + C\alpha S\phi C\theta \\ -C\alpha S\theta + S\alpha S\phi C\theta \\ 0 \end{bmatrix} \]  

(8)

and then

\[ |(X_\gamma)_{xy}| = C\phi \quad \text{and} \quad |(Z_\gamma)_{xy}| = \sqrt{S\theta^2 + C\theta^2 S\phi^2} \]  

(9)

By substituting (8) and (9) into (7) gives the following:

\[ \cos \beta = \frac{C\phi S\phi C\phi}{C\phi \sqrt{S\theta^2 + C\theta^2 S\phi^2}} = \frac{1}{\sqrt{T\theta^2 + 1}} \]  

(10)

where \( T\theta = \tan \theta \)

In the initial position \( \theta, \phi \) and \( \alpha \) are known and therefore (10) provides the value of \( \beta \) where \( \beta' \) has to be the same for non-rotation of the camera image, thus

\[ \cos \beta' = \frac{1}{\sqrt{T\theta^2 + 1}} \]  

(11)

provides the third equation, together with (6), for solving for \( \theta, \phi \) and \( \alpha \) for position 1 in figure 6.1.
The solution for the three rotation angles for position 1 can be summarized as follows:

In equation (6) let

\[
\hat{a} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\]

and let \( e = \cos \beta \)

Then from (6) \( \frac{c}{C \theta} \) and substituting into (10), gives the following

\[
e^2 = \frac{1}{\frac{T \theta^2}{S \phi^2} + 1} \iff \frac{T \theta^2}{S \phi^2} = \frac{1}{e^2} - 1
\]

and thus \( \frac{S \theta^2}{C \theta^2 (1 - C \phi^2)} = \frac{1}{e^2} - 1 \).

Now let \( f = \frac{1}{e^2} - 1 \) and then \( f = \frac{1 - C \theta^2}{C \theta^2 (1 - \frac{c^2}{C \theta^2})} \) resulting in

\[
\cos \theta = \pm \sqrt{\frac{1 + fe^2}{1 + f}}
\]

and

\[
\cos \phi = \frac{c}{\cos \theta}
\]

The angles \( \theta \) and \( \phi \) can then be determined from (12) and (13)

Now the scalar equations from (6) can be written as

\[
C \alpha C \theta S \phi + S \alpha S \theta = a \]

(14)

\[
S \alpha C \theta S \phi - C \alpha S \theta = b
\]

(15)

\[
C \theta C \phi = c
\]

(16)
Multiply (14) by \( \cos \alpha \) and (15) by \( \sin \alpha \) and then add the two resulting equations to get

\[
C \delta \Phi (C \alpha^3 + S \alpha^2) = aC \alpha + bS \alpha
\]  

(17)

And now divide (17) by (16) resulting in

\[
\tan \phi = \frac{a}{c} \cos \alpha + \frac{b}{c} \sin \alpha
\]  

(18)

Equation (18) can be solved for \( \alpha \) since \( \phi \) is already found in (13).

Let \( \frac{a}{c} = D \sin \gamma \) and let \( \frac{b}{c} = D \cos \gamma \), where \( D \) is a constant and \( \gamma \) is an angle. Thus

\[
\tan \gamma = \frac{a}{b} \quad \text{and} \quad D = \frac{(a)^2}{(c)^2} + \frac{(b)^2}{(c)^2}
\]  

(19)

and equation (18) can be written as

\[
\frac{\tan \phi}{D} = \sin \gamma \cos \alpha + \cos \gamma \sin \alpha = \sin (\gamma + \alpha)
\]

and the solution for \( \alpha \) is given by

\[
\alpha = \sin^{-1} \left\{ \frac{\tan \phi}{D} \right\} - \gamma
\]  

(20)

Equations (12), (13) and (20) provide the solution to the required angles \( \theta, \phi \) and \( \alpha \) for aligning the camera axis to point towards the specified target at position 1 while at the same time ensuring that the image transmitted by the camera is not rotated about its own axis. The equations give multiple answers, so the control program has to choose the correct set of angles before sending the re-aligning command back to the robot controller. Doing the calculations, say, every half a second solves this problem by allowing the program to choose the angles that are closest to the previously calculated set of angles. In this way, the control program continuously updates the orientation of the manipulator.
The description above provides a strategy for guiding the robot, with the camera attached to its end-effector, from point to point while the orientation of the camera is automatically updated to keep pointing towards the monitored location during a surgical procedure.

This strategy and the calculations presented above have been verified by manually moving the robot through a number of points and determining the orientation angles as described by equations (12), (13) and (20).

Two positions of the robot were used to verify the above method. The calculations are shown in the next section.

6.3 Numerical verifications for the re-orientation of the camera

By using inverse kinematics and the information of the positions for the manipulator obtained in the laboratory setup, the angles needed to re-align the end effector of the manipulator so that it is directed towards the target could be calculated.

It can be seen from equation (12), (13) and (20), that there will be several possible solutions to the angles, but by comparing these angles with the angles from the previously known position a correct set of solutions for the new angles can be identified. For the specific numerical example used for the verification of the process, the coordinates of the calibration point were measured as follows.
Calibration point:

\[ X = -11.4 \text{ mm} \]
\[ Y = -697.3 \text{ mm} \]
\[ Z = -297.6 \text{ mm} \]

Coordinates obtained from the programming pendant for the initial position were:

\[ X = 188.9 \text{ mm} \]
\[ Y = -310.1 \text{ mm} \]
\[ Z = 114.2 \text{ mm}. \]

And the orientation angles were:

\[ \theta = -136.00^\circ \]
\[ \phi = -19.90^\circ \]
\[ \alpha = 160.00^\circ \]

By substituting these angles into equation (4), we get:

\[
X_T = \begin{bmatrix}
-0.88 \\
0.32 \\
0.34 
\end{bmatrix}
\] (21)

And from equation (5):

\[
d^1 = \begin{bmatrix}
-11.4 \\
-697.3 \\
-297.6 
\end{bmatrix} - \begin{bmatrix}
188.90 \\
-310.10 \\
114.20 
\end{bmatrix} = \begin{bmatrix}
-200.26 \\
-387.17 \\
-411.84 
\end{bmatrix} \text{ mm} \] (22)
and the corresponding unit vector, \( \hat{d}^1 \), is given by:

\[
\hat{d}^1 = \frac{1}{599.68} \cdot \left[ \begin{array}{c}
-200.26 \text{mm} \\
-387.17 \text{mm} \\
-411.84 \text{mm}
\end{array} \right] = \left[ \begin{array}{c}
-0.33 \\
-0.65 \\
-0.69
\end{array} \right] = \left[ \begin{array}{c}
a \\
b \\
c
\end{array} \right]
\] (23)

For a check, the angles for position 1 can be calculated from (21) and (4) as:

\[-\sin \phi = 0.34 \iff \phi = -19.9^\circ \text{ or } -160.0^\circ\]

\[\cos \theta = \frac{-0.69}{\cos \phi} \iff \theta = 137.2^\circ \text{ or } -137.2^\circ\]

\[\sin \alpha = \frac{0.32}{\cos \phi} \iff \alpha = 21.2^\circ \text{ or } 158.8^\circ\]

Comparing these with the original values, the solutions \(-19.9^\circ, -137.2^\circ\) and \(158.8^\circ\), are close to the values read from the pendant obtained values, \(-19.9^\circ, -136.0^\circ\) and \(160.0^\circ\).

To verify the re-orientation calculations, the robot is moved with a MOVL command to a new location, position 1 as in figure 6.1.

At the new location, where the target point no longer is in the view of the camera, the position coordinates, X, Y and Z are obtained from the robot pendant. After this information is collected, the calculations for obtaining the new angles required to realign the camera mounted at the end effector realign can begin.
The known parameters for both positions are listed in Table 6.1.

**Table 6.1 Coordinates and angles for calibration position and position 1**

<table>
<thead>
<tr>
<th>Calibration position</th>
<th>Position 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X = -12.88 mm</td>
<td>X = 233.50 mm</td>
</tr>
<tr>
<td>Y = -500.13 mm</td>
<td>Y = -500.14 mm</td>
</tr>
<tr>
<td>Z = 28.78 mm</td>
<td>Z = 28.78 mm</td>
</tr>
</tbody>
</table>

Using equation (12), (13) and (20), the following solutions for the angles are obtained for position 1:

\[
\begin{align*}
\theta &= -136.00° \\
\phi &= 0.78° \\
\alpha &= 126.00°
\end{align*}
\]

Comparing these with the values of θ, φ and α for the initial position and choosing the set of angles that are closest to these will give the correct orientation angles for the re-aligned camera.

By sending move commands that effectively rotate the end-effector according to these angles, together with the position X, Y and Z coordinates, the camera will be re-aligned so that the target comes into view and is horizontally oriented. The camera view in the two positions of the robot are shown in figures 6.4 and 6.5.
Figure 6.4 The robot at its calibration position with the camera attached to it and aligned for viewing the target at the calibration point.

Figure 6.5 shows the robot with the camera after a linear move command as well as the re-aligning command with the new orientation angles being sent to it. It can be seen that it re-aligns so that the camera points towards the target.
Figure 6.5 The camera re-aligned after a MOVL command

Figure 6.6 shows the image of the target as seen through the camera attached to the end-effector. The first image is the robot at the position shown in figure 6.4 and the second image shows when the camera is re-aligned after the robot has moved as shown in, figure 6.5
Figure 6.6 The re-alignment procedure seen through the camera lens.

In this chapter a method for automatically realigning the camera to always point towards a fixed target position is presented. The technique described is implemented in a manual manner for the current laboratory prototype. To develop this manual procedure for automated application, the remote program would need to be modified. A numerical example showing the steps required for the realignment procedure has also been presented in this chapter.
CHAPTER 7

Conclusion and Discussion

7.1 Summary

A research project on the development of a remotely operated robotic system for visual monitoring of surgical procedures is presented in this thesis. The aim of the project has been to develop a remotely operated system to be used for interactive visual monitoring of surgical procedures where the specialist surgeon cannot be physically present. This thesis describes how a mechanical system consisting of commercially available devices, including an electrically driven robot and equipped with a video camera is implemented for this task. The intention is that the system becomes the basis for a custom-built system that can be mounted in an operating theatre or surgical clinic in country regions, where the need for specialist consultation is difficult to access quickly. Especially in the Australian context where there are large distances between centers, the capability provided by the proposed system for monitoring and advising through a vision and voice link is a new development.

The benefits of such a remotely operated robotic device, especially in countries where surgical expertise is concentrated in city areas and where there are great distances between centers, include the elimination of the need to transport patients over large distances to have certain procedures performed in city hospitals. In addition, the outcomes of the whole project are expected to have downstream, spin-off application in surgical education and training.
The main difference between this project and other projects within the area of teleoperated surgery and telemedicine is that this system is designed in its final version of the prototype not to infringe with the workspace of the performing surgeon. This is in contrast with the applications where manipulators are used in laparoscopic surgery and neurosurgery and are located within the same working space as the performing surgeon. The aim for this project was to make a remote manipulator unit that does not interfere with the surgeons work volume but the manipulator should still be able to monitor the surgical procedure.

7.2 Project Outcomes

The teleoperated robotic system described in this thesis, is a laboratory prototype for demonstrating the feasibility of visual monitoring and consultation during surgical procedures at remote locations. The system employs a commercially available videoconferencing package for the vision control of the manipulator. A CCD camera is mounted on the end-effector of the manipulator for transmitting video images to the remote location for visual feedback. The videoconferencing unit runs via the LAN network the reason for this being that the LAN provides enough bandwidth for real-time images and ISDN was not available in the laboratory setup at the time of writing. Commands for the robot, such as move or status commands, are transmitted using high-speed modems connected to the telephone lines.
As part of the project a control system was devised for operating the manipulator so it could be controlled from a remote location. The control system was developed using Visual C++® and for establishing the software interface with the manipulator. In making the control system as user friendly as possible, a windows based dialog box was created for handling the transmission of commands to the robot.

The dialog box incorporates modules from the Graphic User Interface (GUI) library in Visual C++. The dialog box also handles the modem interface, including such functions as dialing the number to the host location and ending the transmission (hang up) of commands to the robot controller. When the dialog box is active, it displays two identical control boards. One is for rough and high-speed movements, the other is for finer and slower movements of the manipulator when the target object is in view.

Automated visual tracking of the target area, namely the area of interest during a surgical procedure, can be achieved by adapting the process developed in this thesis. The image control method developed in this thesis, and that ensures that the camera is continually aligned towards the target area and the image is not rotated away from the horizontal, has been verified by manual experiments.
7.3 Recommendations

The results of this project and the knowledge gained in carrying out the research are intended to provide the basis for further work. Some recommendations for developing the laboratory system into a fully functional clinical system is outlined here.

The system is fully operational, but it has to go through some improvements before it can be used in a surgical theatre. The system has yet to be tested under real surgical conditions with human subjects, and safety is a key issue here. This will require a custom-designed manipulator to be constructed to ensure non-interference with the space occupied by the surgeon during the visual monitoring. The robot used as the laboratory prototype is not suited for the task since it is an industrial robot originally designed for welding operations.

The LAN will need to be replaced with an ISDN connection. The ISDN connection will handle the transmission of the video images and the robot commands. Using video compression techniques (h.321), the ISDN provides enough bandwidth (128 kbps [54]) to allow real-time images to be transmitted for the visual feedback of the manipulator and for the movement control.

One of the most important aspects of this project is to have a control system that is simple, reliable and easy to use. The windows based program developed in this thesis provides a user-friendly interface, but the implementation of a joystick may make the controls easier and more intuitive for the operator than it is at the present time. An
automatic path control system may also be implemented when using the joystick. This control system will ensure that the end effector of the robot, where the CDD camera is mounted, always points towards the area where the surgical procedure takes place. With the help of this path control system, it will be easier for the operator to control the movements of the manipulator. It will also ensure that the end effector with the attached camera does not rotate about its lens axis so the video images remain horizontal throughout the movement. A laser pointer may also be added to the manipulator, and could be remotely controlled to allow identification of specific areas during consultation between surgeons.
References:


35. Gofion P W, Mastering serial communication, 2nd edition,


47. Yaskawa, MRC instructions, Yaskawa.


54. Ed Tittel, Steve James, David Picitello, Lisa Phifer, ISDN clearly explained, 2\textsuperscript{nd} edition, AP professional, 1997.


Appendix A.1

C++ source code for terminal program

The computer programs developed for this thesis are written in Visual C++, and they are included here in full. The program "terminal program" performs the handshaking between the robot controller and the host computer. In addition to this, the software executes the commands the host site receives from the remote site through the modem connection.

```cpp
// include <conio.h>
# include <iostream.h>
# include <stdio.h>
# include <windows.h>
# include <winbase.h>
// Declarations of variables.
int n, anything;
int nId;
int rc;
int numclosed;
int rNumFlushed;
char *projectName = "project1";
unsigned short shortroom;
double *p;
char ch = 'q';
char cChar[15] = "ATSO=5n"; // AT commands to modem.
char bData[20]; // used for reading data
char *ryper = "V";
double spd = 10;
double axith[12];
char *filename = "ROBOT";
short toolnum = 0;
// Declarations of the DCB structure
DCB dcb;
HANDLE hCom;
DWORD dwError;
ULONG Count = 20;
BOOL fSuccess;
BOOL fWriteStat;
OVERLAPPED or;
OVERLAPPED ow;
ULONG hCount;
ULONG hCount2 = 50;
int main()
{

//Flush the buffers and close all open COM ports
numflushed = _flushall();
printf("There were %d streams flushed\n", numflushed);
}
numclosed = fcloseall();
printf("Number of files closed by fcloseall %d\n", numclosed);

// Creates a handle to communicate with the serial port
hCom = CreateFile("COM2", GENERIC_READ,
GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_FLAG
OVERLAPPED, NULL);
if (hCom == INVALID_HANDLE_VALUE)
{
dwError = GetLastError();
cout << "Create Handle failed" << endl;
cout << dwError << endl;
}
// fills in a device-control block dcb
fSuccess = SetCommState(hCom, &dcb);
if (!fSuccess)
{
cout << "Setting up the dcb failed" << endl;
cout << dwError << endl;
}
else
{
cout << "Setting the dcb structure success" << endl;
}
// DCB structure
dcb.BaudRate = 9600;
dcb.ByteSize = 8;
dcb.Parity = NOPARITY;
dcb.StopBits = ONESTOPBIT;
// function configures a communications device according
// to the specifications in a device-control block (a DCB structure).
fSuccess = SetCommState(hCom, &dcb);
if (!fSuccess)
{
cout << "Setting the dcb failed" << endl;
cout << dwError << endl;
}
else
{
cout << "Setting the dcb success" << endl;
}
// Initializes the communications parameters
// for a specified communications device.
fSuccess = SetupComm(hCom, 1024, 256);
if (fSuccess)
{

```
cout << "initialize the communications parameters failed" << endl;
  cout << "dwError" << endl;
}
else {
  cout << "communications parameters initialized" << endl;
  //Sets the state of the event to signaled, releasing any waiting threads
  if (Success) {
    cout << "SetEvent failed" << endl;
    cout << "dwError" << endl;
  } else {
    cout << "SetEvent success" << endl;
  }
  //provides a high performance test operation that can be used to
  //poll for the completion of an outstanding I/O operation.
  if (Success) {
    cout << "HasOverlappedIoCompleted failed" << endl;
    cout << "dwError" << endl;
  } else {
    cout << "HasOverlappedIoCompleted success" << endl;
  }
}
//This is the end of the communication initializing section
//Read and write operations begin here.
//Sends AT commands to modem, makes it go off hook and turns echo off
//Opening of transmission port for Robot using com1
anything = BscOpen(project1, 1);
if (anything != -1) {
  cout << "failed to open COM port" << endl;
  BscClose();
  cout << "COM port closed" << endl;
} else {
  cout << "COM1 port open" << endl;
  //Settings for COM1 port for robot
  rsc = BscSetCom(0, 1, 9600, 2, 8, 0);
  if (rsc != 0) {
    cout << "Error in setting the parameters for COM1 port" << endl;
    BscClose();
    cout << "COM port closed" << endl;
  } else {
    cout << "Settings success" << endl;
  }
  //Connection to com1 for robot
  if (rsc != 0) {
    cout << "Connection to COM port failed" << endl;
    BscClose();
    cout << "COM port closed" << endl;
  } else {
    cout << "connection com1 success" << endl;
  }
  //reads what mode the robot is in
  if (anything != 0) {
    cout << "Robot in Teach mode" << endl;
    cout << "Robot must be in Play Mode" << endl;
  } else {
    cout << "Robot in Play Mode" << endl;
    cout << "Robot in correct Mode" << endl;
  }
  //specifies a set of events to be monitored for a communications device.
  if (Success) {
    cout << "SetCommMask failed" << endl;
    cout << "dwError" << endl;
  } else {
    cout << "SetCommMask success" << endl;
  }
  //Reads com port for data till escape is pressed.
  char cChar[1] = "ATSO1v";
  cout = strlen(cChar);
  if (Success)
    cout << "AT commands not sent" << endl;
    cout << "dwError" << endl;
  else {
    cout << "Autoanswer on" << endl;
    //reads from modem1.
    /*
      char dData[15];
      int length = strlen(dData);
      if (Success = ReadFile(hCom, dData, hCount, &hCount, &or));
      if (Success)
        dwError = GetLastError();
    */
    cout << "dwError" << endl;
  } else {
    cout << "response from executed AT command" << endl;
    for (int i = 0; i <= length; i++)
      cout << dData[i];
  }
  char cData[1] = "ATE0v";
  cout = strlen(cData);
  WriteStrm = WriteFile(hCom, cData, Count, &Count, &or);
  if (Success)
    cout << "Writefile failed" << endl;
  else {
    cout << "Echo off" << endl;
  }
}
// reads from modem2.
char cData[48];
int Count=0;
if(Count<48)
  cout<<"Count<<endl;
if(Success=ReadFile(hCom,cData,Count,&hCount,&&or);)
  cout<<"Count<<endl;
if(Success)
  dwError=GetLastError();
  cout<<"dwError<<endl;
  else
  cout<<"response from executed AT command2"<<endl;
  int i=0;
  int i=len(i++)
  cout<<"Data[i];
  if(Success="flushall());
  
  cout<<"read finished"<<endl;
  cout<<"Count<<endl;
  //CharData[50];
  int Count=len(eData);
  fSuccess=ReadFile(hCom,eData,Count,&hCount,&&or);)
  if(Success)
  dwError=GetLastError();
  cout<<"dwError<<endl;
  else
  cout<<"carrier signal detected"<<endl;
  cout<<"Count<<endl;
  char gData[50];
  int C=len(gData);
  fSuccess=ReadFile(hCom,exData,Count,&hCount,&&or);)
  if(Success)
  dwError=GetLastError();
  cout<<"dwError<<endl;
  else
  cout<<"Airing indicator"<<endl;
  cout<<"Count<<endl;
  char hData[50];
  int Count=len(hData);
  fSuccess=ReadFile(hCom,exData,Count,&hCount,&&or);)
  if(Success)
  dwError=GetLastError();
  cout<<"dwError<<endl;
  else
  cout<<"carrier signal detected"<<endl;
  cout<<"Count<<endl;
  /*CharData[10];
  int Count=len(bData);
  int max3=len(bData);
  for(int i=0; i<=max;i++)
  cout<<"bData[1];
  switch(*bData)
  // on the servo power case 'a':
  numflushed=_flushall();
  if(!numflushed)
  cout<<"Servo power on"<<endl;
  else
  cout<<"Servo power off"<<endl;
  
  //Flush the buffers and close all open COM ports
  numflushed=_flushall();
  numclosed=_fcloseall();
  printf("Number of streams closed %d", numflushed);
  printf("Number of files closed by _fcloseall: %d", numclosed);
  
  break;
  // on the servo power off case 'b':
  numflushed=_flushall();
  anything=BxServoOff(0);
  if(!anything)
  cout<<"Servo off"<<endl;
  cout<<"anything<<endl;
  
  //Flush the buffers and close all open COM ports
  numflushed=_flushall();
  numclosed=_fcloseall();
  printf("Number of streams closed %d", numclosed);
  printf("Number of files closed by _fcloseall: %d", numclosed);
  
  break;
  // Moves the robot in the negative x-direction case 'c':
  

p=axis6;
anything=BscLsLoc(0,1,1,rcnf,p);
if(anything==1)
{
cout<<"Position read failed"<<endl;
BscClose(0);
exit(0);
}
else
{
cout<<"pos read success"<<endl;
cout<<"x-pos="<<p[0]<<endl;
cout<<"y-pos="<<p[1]<<endl;
cout<<"z-pos="<<p[2]<<endl;
cout<<"Tx-pos="<<p[3]<<endl;
double pos0=p[0];2000;
double pos2=p[1];
double pos3=p[2];
double pos4=p[3];
double pos5=p[4];
double pos6=p[5];
char *type="V";
double spd=10;
char *frame="ROBOT";
short toolno=0;
double *p1;
double
axis7[12]={pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0};
p1=axis7;
anything=BscPmovj(0,0,0,0,0,0,p1);
if(anything==0)
{
cout<<"anything"<<endl;
}
else
{
cout<<"move failed"<<endl;
}
break;
//Moves the robot in the negative x-direction. case 'v':
p=axis6;
anything=BscLsLoc(0,1,1,rcnf,p);
if(anything==1)
{
cout<<"Position read failed"<<endl;
BscClose(0);
exit(0);
}
else
{
cout<<"pos read success"<<endl;
cout<<"x-pos="<<p[0]<<endl;
cout<<"y-pos="<<p[1]<<endl;
cout<<"z-pos="<<p[2]<<endl;
cout<<"Tx-pos="<<p[3]<<endl;
double pos0=p[0];10000;
double pos2=p[1];
double pos3=p[2];
double pos4=p[3];
double pos5=p[4];
double pos6=p[5];
char *type="V";
double spd=10;
char *frame="ROBOT";
short toolno=0;
double *p1;
double
axis7[12]={pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0,0};
p1=axis7;
anything=BscPmovj(0,0,0,0,0,0,p1);
if(anything==0)
{
cout<<"anything"<<endl;
}
else
{
cout<<"move failed"<<endl;
}
break;
// Moves the robot in the positive x-direction, slow
case 'F':
    p=axis6;
    anything=BscslLoc(0,1,&config,p);
    if (anything==1)
    {
        cout<<"Position read failed"<<endl;
        BscClose(0);
        exit(0);
    }
    else
    {
        cout<<"pos read success"<<endl;
        cout<<"x-pos"<<crr[0]"y-pos"<<crr[1]"z-pos"<<crr[2]"ty-pos"<<crr[3]"tx-pos"<<crr[4]"tx-pos"<<crr[5]"p0=1;10000;
        double pos2=p[1]; double pos3=p[2]; double pos4=p[3]; double pos5=p[4];
        double pos6=p[5];
        char *type="V";
        double spd=10;
        char *framename="ROBOT";
        short toolno=0;
        double *p1;
        double axis7[12]= {pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0};
        p1=axis7;
        anything=BscPMovj(0,0,0,p1);
        if (anything==0)
        {
            cout<<"<null><anything><endl;}
            else
            {
                cout<<"move failed"<<endl;
            }
        break;
        // Moves the robot in the positive y-direction, fast
        case 'G':
    p=axis6;
    anything=BscslLoc(0,1,&config,p);
    if (anything==1)
    {
        cout<<"Position read failed"<<endl;
        BscClose(0);
        exit(0);
    }
    else
    {
        cout<<"pos read success"<<endl;
        cout<<"x-pos"<<crr[0]"y-pos"<<crr[1]"z-pos"<<crr[2]"ty-pos"<<crr[3]"tx-pos"<<crr[4]"tx-pos"<<crr[5]"p0=1;10000;
        double pos1=p[0];
        double pos2=p[1]+20000;
        double pos3=p[2]; double pos4=p[3]; double pos5=p[4];
        double pos6=p[5];
        char *type="V";
        double spd=10;
        char *framename="ROBOT";
        short toolno=0;
        double *p1;
        double axis7[12]= {pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0};
        p1=axis7;
        anything=BscPMovj(0,10,0,p1);
        if (anything==0)
        {
            cout<<"<null><anything><endl;}
            else
            {
                cout<<"move failed"<<endl;
            }
        }
break; //moves in negative y-dir, slow case 'i':
    
    p=axis6;
    anything=BasciLoc(0,1,&confg,p);
    if (anything==-1)
    {
        cout<<"Position read failed"<<endl;
        BscClose(0);
        exit(0);
    }
    else
    {
        cout<<"pos read success"<<endl;
        cout<<"x-pos="<<cp[0]<<endl;
        cout<<"y-pos="<<cp[1]<<endl;
        cout<<"Tx-pos="<<cp[2]<<endl;
        cout<<"Ty-pos="<<cp[3]<<endl;
        cout<<"Tz-pos="<<cp[4]<<endl;
        double pos6=p[0];
        double pos2=p[1]*10000;
        double pos3=p[2];
        double pos4=p[3];
        double pos5=p[4];
        double pos6=p[5];
        char *vtype="V";
        double spd=10;
        char *framenme="ROBOT";
        short toolno=0;
        double *p1;
        double axis7[12]=(pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0);
        p=axis7;
        anything=BasciMov(0,5,0,p1);
        if (anything==0)
        {
            cout<<"<anything<"<<endl;
            }
        else
        {
            cout<<"move failed"<<endl;
            }
        }
    }

break; //negative z-dir, fast case 'k':
    
    p=axis6;
    anything=BasciLoc(0,1,&confg,p);
    if (anything==-1)
    {
        cout<<"Position read failed"<<endl;
        BscClose(0);
        exit(0);
    }
    else
    {
        cout<<"pos read success"<<endl;
        cout<<"x-pos="<<cp[0]<<endl;
        cout<<"y-pos="<<cp[1]<<endl;
        cout<<"z-pos="<<cp[2]<<endl;
        cout<<"Tx-pos="<<cp[3]<<endl;
        cout<<"Ty-pos="<<cp[4]<<endl;
        cout<<"Tz-pos="<<cp[5]<<endl;
        double pos1=p[0];
        double pos2=p[1]*100000;
        double pos3=p[2]*200000;
        double pos4=p[3];
        double pos5=p[4];
        double pos6=p[5];
        char *vtype="V";
        double spd=10;
        char *framenme="ROBOT";
        short toolno=0;
        double *p1;
        double axis7[12]=(pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0);
        p=axis7;
        anything=BasciMov(0,10,0,p1);
        if (anything==0)
        {
            cout<<"<anything<"<<endl;
        }
        else
        {
            cout<<"move failed"<<endl;
        }
    }
cout<<"move failed"<<endl;
}

break;
//MOVES wrist left, slow
case 'r':
    p=axis6;
    anything=BscLoc(0.1,&rconf,p);
    if(anything==1)
        cout<<"Position read failed"<<endl;
    BscClose(0);
    exit(0);
    else
        cout<<"pos read success"<<endl;
    cout<<"x-po"<<p[0]<<endl;
    cout<<"y-po"<<p[1]<<endl;
    cout<<"z-po"<<p[2]<<endl;
    cout<<"Tx-po"<<p[3]<<endl;
    cout<<"Tz-po"<<p[5]<<endl;
    double pos1=p[0];
    double pos2=p[1];
    double pos3=p[2];
    double pos4=p[3]-10000;
    double pos5=p[4];
    double pos6=p[5];
    char *type="V";
    double spd=10;
    char *frame=MOVJ(0.5,0,p1);
    if(anything==0)
        cout<<"<<anything<<endl;
    else
        cout<<"move failed"<<endl;
}
break;
//moves the wrist up
case 'u':
    p=axis6;
    anything=BscLoc(0.1,&rconf,p);
    if(anything==1)
        cout<<"Position read failed"<<endl;
    BscClose(0);
    exit(0);
    else
        cout<<"pos read success"<<endl;
    cout<<"x-po"<<p[0]<<endl;
    cout<<"y-po"<<p[1]<<endl;
    cout<<"z-po"<<p[2]<<endl;
    cout<<"Tx-po"<<p[3]<<endl;
    cout<<"Tz-po"<<p[5]<<endl;
    double pos1=p[0];
    double pos2=p[1];
    double pos3=p[2];
    double pos4=p[3]-20000;
    double pos5=p[4];
    char *type="V";
    double spd=10;
    char *frame=MOVJ(0.1,0,p1);
    if(anything==0)
        cout<<"<<anything<<endl;
    else
        cout<<"move failed"<<endl;
}
{ cout<<"move failed"<<endl; } break; //moves wrist down, slow case 'v': { p=axis6; anything=BsciL0c(0,1,&conLp); if (anything==1) { cout<<"Position read failed"<<endl; BsciClose(0); exit(0); } else { cout<<"pos read success"<<endl; cout<<"x-pos="<<p[0]<<endl; cout<<"y-pos="<<p[1]<<endl; cout<<"z-pos="<<p[2]<<endl; cout<<"Tx-pos="<<p[3]<<endl; cout<<"Ty-pos="<<p[4]<<endl; cout<<"Tz-pos="<<p[5]<<endl; double pos1=p[0]; double pos2=p[1]; double pos3=p[2]; double pos4=p[3]; double pos5=p[4]-10000; double pos6=p[5]; char *type="V"; double spd=10; char *filename="ROBOT"; short toolno=0; double *p1; double axis7[12] = {pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; p1=axis7; anything=BsciPmov(0.5,0.0,p1); if (anything==0) { cout<<""<<endl; } else { cout<<"move failed"<<endl; } } break; //moves wrist up, slow case 'V': { p=axis6; anything=BsciL0c(0,1,&conLp); if (anything==1) { cout<<"Position read failed"<<endl; BsciClose(0); exit(0); } else { cout<<"pos read success"<<endl; cout<<"x-pos="<<p[0]<<endl; cout<<"y-pos="<<p[1]<<endl; cout<<"z-pos="<<p[2]<<endl; cout<<"Tx-pos="<<p[3]<<endl; cout<<"Ty-pos="<<p[4]<<endl; cout<<"Tz-pos="<<p[5]<<endl; double pos1=p[0]; double pos2=p[1]; double pos3=p[2]; double pos4=p[3]; double pos5=p[4]-10000; double pos6=p[5]; char *type="V"; double spd=10; char *filename="ROBOT"; short toolno=0; double *p1; double axis7[12] = {pos1,pos2,pos3,pos4,pos5,pos6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; p1=axis7; anything=BsciPmov(10.0,0,p1); if (anything==0) { cout<<""<<endl; } } 

114
coun<< "<<anything()<<endl;
} else {
    coun<<"move failed"<<endl;
}
}
break;
//turns tool CW
case 'z':
{
    p=axis5;
    anything=BscSlm();
    if (anything==1) {
        coun<<"Position read failed"<<endl;
        BscClose();
        exit(0);
    }
} else {
    coun<<"pos read successs"<<endl;
    coun<<"x-pos="<<pos[0]<<endl;
    coun<<"y-pos="<<pos[1]<<endl;
    coun<<"z-pos="<<pos[2]<<endl;
    coun<<"Tx-pos="<<pos[3]<<endl;
    coun<<"Ty-pos="<<pos[4]<<endl;
    coun<<"Tz-pos="<<pos[5]<<endl;
    double pos1=pi[0];
    double pos2=pi[1];
    double pos3=pi[2];
    double pos4=pi[3];
    double pos5=pi[4];
    double pos6=pi[5]+10000;
    char *type="V";
    double spid=10;
    char *frame="ROBOT";
    short toolno=0;
    double *p1;
    double axis7[2]=pos1,pos2,pos3,pos4,pos5,pos6,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0;
    p=axis7;
    anything=BscPMovj(0.5,0.5,0.5);
    if (anything==1) {
        coun<<"<<anything()<<endl;
    } else {
        coun<<"move failed"<<endl;
    }
}
break;
//resets an error
case 'x':
{
    anything=BscReset();
    if (anything==1) {
        coun<<"reset failed"<<endl;
        re=BscGetError();
        cout<<re<<endl;
    }
} break;
// cancels an error
case 'y':
{
    anything=BscCancel();
    if (anything==1) {
        coun<<"failed"<<endl;
    }
}
Appendix A.2

C++ code for the dialog box

This is the source code for the dialog box, written using Visual C++. The dialog box is the control unit at the remote site, sending the commands the operator would like to execute to the host site.

```
// Final2Dlg.cpp : implementation file
//
#include "stdafx.h"
#include "Final2.h"
#include "Final2Dlg.h"
#include "comn.h"
#include "oostreem.h"
#include "windows.h"
#include "io.h"

#include "afxwin.h"

#endif _DEBUG

define new DEBUG_NEW

#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

ULONG count=12;
OVERLAPPED ov;
BOOL fWriteStat;
HANDLE hCom;
DCB dcb;
BOOL fSuccess;

// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:

CAboutDlg();

// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX },
//}}AFX_DATA

//ClassWizard generated virtual function overrides

//{{AFX_VIRTUAL(CAboutDlg)
protected:

template void DoDataExchange(CDataExchange* pDX);
//}}AFX_VIRTUAL

// Implementation
protected:

//{{AFX_MSG(CAboutDlg)
//}}AFX_MSG

DECLARE_MESSAGE_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{

//}}AFX_DATA_INIT

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{

CDlg: :DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAboutDlg)
//}}AFX_DATA_MAP

}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//}}AFX_MSG_MAP

END_MESSAGE_MAP()

```

// CFinal2Dlg dialog

CFinal2Dlg: :CFinal2Dlg(CWnd* pParent /*=NULL*/) : CDialog(CFinal2Dlg::IDD, pParent)
{

//}}AFX_DATA_INIT

BEGIN_MESSAGE_MAP(CFinal2Dlg, CDialog)
//}}AFX_MSG

END_MESSAGE_MAP()

```
ON_BN_CLICKED(IDC_BUTTON6, 
ON_BN_CLICKED(IDC_BUTTON31, 
ON_BN_CLICKED(IDC_BUTTON32, 
ON_BN_CLICKED(IDC_BUTTON7, 
ON_BN_CLICKED(IDC_BUTTON10, 
ON_BN_CLICKED(IDC_BUTTON9, 
ON_BN_CLICKED(IDC_BUTTON22, 
ON_BN_CLICKED(IDC_BUTTON8, 
ON_BN_CLICKED(IDC_BUTTON11, 
ON_BN_CLICKED(IDC_BUTTON20, 
ON_BN_CLICKED(IDC_BUTTON23, 
ON_BN_CLICKED(IDC_BUTTON9, 
ON_BN_CLICKED(IDC_BUTTON12, 
ON_BN_CLICKED(IDC_BUTTON21, 
ON_BN_CLICKED(IDC_BUTTON24, 
ON_BN_CLICKED(IDC_BUTTON13, 
ON_BN_CLICKED(IDC_BUTTON16, 
ON_BN_CLICKED(IDC_BUTTON25, 
ON_BN_CLICKED(IDC_BUTTON28, 
ON_BN_CLICKED(IDC_BUTTON14, 
ON_BN_CLICKED(IDC_BUTTON17, 
ON_BN_CLICKED(IDC_BUTTON26, 
ON_BN_CLICKED(IDC_BUTTON29, 
ON_BN_CLICKED(IDC_BUTTON15, 
ON_BN_CLICKED(IDC_BUTTON18, 
ON_BN_CLICKED(IDC_BUTTON27, 
ON_BN_CLICKED(IDC_BUTTON30, 
ASSERT(IDM_ABOUTBOX < 0xF000); 
CMenu* pSysMenu = GetSystemMenu(FALSE); 
if (pSysMenu != NULL)
{
    CString strAboutMenu;
    strAboutMenu.LoadString(IDS_ABOUTBOX);
    if (!strAboutMenu.IsEmpty())
    {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
    }
    // Set the icon for this dialog. The framework
does this automatically
    // when the application's main window is not a
    // dialog
    SetIcon(m_hIcon, TRUE); // Set big icon
    SetIcon(m_hIcon, FALSE); // Set small icon
    // TODO: Add extra initialization here
    return TRUE; // return TRUE unless you set the
focus to a control
}
void CFinal2Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDIalog::OnSysCommand(nID,
LParam);
    }
}
// If you add a minimize button to your dialog, you will need
// to draw the icon. For MFC applications using the
document/view model,
// this is automatically done for you by the framework.
void CFinal2Dlg::OnPaint()
{
    if (lParam)
    {
        CPaintDC dc(this); // device context
for painting
        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
        // Center icon in client rectangle
        int cxIcon =
GetSystemMetrics(SM_CXICON);
        int cyIcon =
GetSystemMetrics(SM_CYICON);
void CFinal2Dlg::OnButton4()
/*
   m_EditCtrl.SetWindowText("Error cancelled");
   char cData[3]="y";
   count=strlen(cData);
   fwriteState=WriteFile(hCom,cData,count,&count,0);
   fSuccess= floseal();
   //TODO: Add your control notification handler code here
*/
}

void CFinal2Dlg::OnButton5()
/*
   m_EditCtrl.SetWindowText("Servo on");
   char cData[3]="a";
   count=strlen(cData);
   fwriteState=WriteFile(hCom,cData,count,&count,0);
   fSuccess= floseal();
   //TODO: Add your control notification handler code here
*/
}

void CFinal2Dlg::OnButton6()
/*
   m_EditCtrl.SetWindowText("Servo off");
   char cData[3]="b";
   count=strlen(cData);
   fwriteState=WriteFile(hCom,cData,count,&count,0);
   fSuccess= floseal();
   //TODO: Add your control notification handler code here
*/
}
void CFinal2Dlg::OnButton32()  
{  
  m_EditCtrl.SetWindowText("Hold function off");  
  char cData[3]="Y";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}  

void CFinal2Dlg::OnButton33()  
{  
  m_EditCtrl.SetWindowText("Moving right with slow movements");  
  char cData[3]="f";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}  

void CFinal2Dlg::OnButton7()  
{  
  m_EditCtrl.SetWindowText("Moving left");  
  char cData[3]="c";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}  

void CFinal2Dlg::OnButton10()  
{  
  m_EditCtrl.SetWindowText("Moving right");  
  char cData[3]="d";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}  

void CFinal2Dlg::OnButton19()  
{  
  m_EditCtrl.SetWindowText("Moving left with slow movement");  
  char cData[3]="e";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}  

void CFinal2Dlg::OnButton20()  
{  
  m_EditCtrl.SetWindowText("Moving forwards with slow movements");  
  char cData[3]="r";  
  count=strlen(cData);  
  fWriteStat=WriteFile(hCom,cData,count,&count, &cow);  
  fSuccess=_fcloseall();  
  // TODO: Add your control notification handler code here  
}
void CFinal2Dlg::OnButton23()
{
    m_EditCtrl.SetWindowText("Moving backwards with slow movement");
    char cData[3]="y";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();

    // TODO: Add your control notification handler code here
}

void CFinal2Dlg::OnButton9()
{
    m_EditCtrl.SetWindowText("Moving up");
    char cData[3]="k";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();

    // TODO: Add your control notification handler code here
}

void CFinal2Dlg::OnButton12()
{
    m_EditCtrl.SetWindowText("Moving down");
    char cData[3]="l";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton21()
{
    m_EditCtrl.SetWindowText("Moving up slow");
    char cData[3]="m";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton24()
{
    m_EditCtrl.SetWindowText("Moving down slow");
    char cData[3]="n";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();

    // TODO: Add your control notification handler code here
}

void CFinal2Dlg::OnButton13()
{
    m_EditCtrl.SetWindowText("Wrist left");
    char cData[3]="o";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton16()
{
    m_EditCtrl.SetWindowText("Wrist right");
    char cData[3]="p";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton25()
{
    m_EditCtrl.SetWindowText("Wrist right slow");
    char cData[3]="q";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton28()
{
    m_EditCtrl.SetWindowText("Wrist left slow");
    char cData[3]="r";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}

void CFinal2Dlg::OnButton14()
{
    m_EditCtrl.SetWindowText("Wrist up");
    char cData[3]="s";
    count=strlen(cData);
    fWriteStat=WriteFile(hCom,cData,count,&count,&ow);
    fSuccess=\_fcloseall();
}
} void CFinal2Dlg::OnButton17()
{    m_EditCtrl.SetWindowText("Wrist down");
    char cData[3]="t";
    count=strlen(cData);
    fwrite=cFileWrite(hCom,cData,count,&count,
&row);
    fSuccess=_fcloseall();
}
}
void CFinal2Dlg::OnButton26()
{    m_EditCtrl.SetWindowText("Wrist down");
    char cData[3]="u";
    count=strlen(cData);
    fwrite=cFileWrite(hCom,cData,count,&count,
&row);
    fSuccess=_fcloseall();
}
void CFinal2Dlg::OnButton29()
{    m_EditCtrl.SetWindowText("Wrist up");
    char cData[3]="v";
    count=strlen(cData);
    fwrite=cFileWrite(hCom,cData,count,&count,
&row);
    fSuccess=_fcloseall();
}
void CFinal2Dlg::OnButton15()
{    m_EditCtrl.SetWindowText("Turning camera
CCW");
    char cData[3]="w";
    count=strlen(cData);
    fwrite=cFileWrite(hCom,cData,count,&count,
&row);
    fSuccess=_fcloseall();
}
void CFinal2Dlg::OnButton18()
{    m_EditCtrl.SetWindowText("Turning camera
CW");
    char cData[3]="z";
    count=strlen(cData);
    fwrite=cFileWrite(hCom,cData,count,&count,
&row);
    fSuccess=_fcloseall();
}
void CFinal2Dlg::OnButton27()
## Appendix A.3

### ASCII code chart

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Key</th>
<th>Use in C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>(NULL)</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Ctrl A</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Ctrl B</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>Ctrl C</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>Ctrl D</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Ctrl E</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Ctrl F</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Ctrl G (Beep)</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Backspace</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Tab</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Ctrl J</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Ctrl K (Linefeed)</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Ctrl L (Form Feed)</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Enter (Carriage Return)</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Ctrl N</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Ctrl O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Ctrl P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Ctrl Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Ctrl S</td>
</tr>
</tbody>
</table>
Appendix A.4

Host control setup for the Robot controller

Appendix A4 shows how to set up the robot controller in host control, making it able to communicate with an external personal computer.

While holding down [CUSTOMER] on the programming pendant, turn ON the main power of YASNAC MRC. The system starts up in maintenance mode and the screen on the left appears.

Move the cursor to "SETUP SYSTEM" and depress [ENTER].
In the above display, "IO" indicates I/O remote and "COMMAND" indicates command remote.

When "USED" is selected for IO, I/O remote becomes validated; operation from external I/O is enabled with the PBOX remote key depressed. When "NOT USED" is selected, operation from external I/O is not enabled.

When "USED" is selected for COMMAND, the host control function becomes validated; operation from external I/O is enabled with the PBOX remote key depressed. When "NOT USED" is selected, the host control function cannot be used.

When "USED" is selected for PP/PBOX, PP/PBOX operation is enabled even in the remote mode. When "NOT USED" is selected, operation from PP/PBOX is prohibited. However, emergency stop, hold key or remote key can be operated even in the prohibited status.
When the remote status is entered by selecting remote by using the PBOX remote key or external I/O, you can verify how the above settings are in the following remote display.

\[
\text{DISP} \rightarrow \text{DIAG} \rightarrow \text{MORE} \rightarrow \text{REMOTE}
\]

\[
\begin{array}{c}
\text{REMOTE} \\
\hline
\text{COMMAND MODE} \\
\hline
\text{CURR} : \\
\text{PREV} : \\
\text{DISP} : \\
\hline
\text{QUIT}
\end{array}
\]

There are four types of messages as follows:
- "REMOTE MODE NOT SPECIFIED": Remote status is not entered.
- "I/O MODE": I/O remote status is entered.
- "COMMAND MODE": Command remote status is entered.
- "I/O AND COMMAND MODE": I/O remote and command remote status is entered.

Select "USED" or "NOT USED" for each item and depress \(\text{ENTER}\). The message on the left appears.

**CUSTOMER OPTION**

**REMOTE FUNCTION**

<table>
<thead>
<tr>
<th></th>
<th>I/O</th>
<th>COMMAND</th>
<th>PNP/PPBX</th>
</tr>
</thead>
<tbody>
<tr>
<td>USED</td>
<td>USED</td>
<td>USED</td>
<td>USED</td>
</tr>
<tr>
<td>NOT USED</td>
<td>NOT USED</td>
<td>NOT USED</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>

**INITIALIZE RELATED FILES**

<table>
<thead>
<tr>
<th>ARE YOU SURE?</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL BY (\leftarrow\rightarrow)/CONFIRM BY (\text{ENTER})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Move the cursor to "YES" and depress \(\text{ENTER}\) to change the setting. To cancel it, move the cursor to "NO" and depress \(\text{ENTER}\).
Appendix A.5

Parameter settings for the Robot controller

Appendix A.5 shows the parameter setup for the robot controller and the initial values for these. The parameters are to be set to match the external personal computer for serial communication between the two.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contents and Set Value</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS030</td>
<td>Data bit number 7: 7 (bit) 8: 8</td>
<td>8</td>
</tr>
<tr>
<td>RS031</td>
<td>Stop bit number 0: 1 (bit) 1: 1.5 2: 2</td>
<td>0</td>
</tr>
<tr>
<td>RS032</td>
<td>Parity specification 0: No specification 1: Odd parity 2: Even parity</td>
<td>2</td>
</tr>
<tr>
<td>RS034</td>
<td>Timer A Sequence monitoring timer. Serves as protection against invalid or no response. Unit: 0.1 sec (range: 0 to 100)</td>
<td>30</td>
</tr>
<tr>
<td>RS035</td>
<td>Timer B Text reception monitoring timer. Serves as protection against no response of text end character. Unit: 0.1 sec (range: 0 to 255)</td>
<td>200</td>
</tr>
<tr>
<td>RS036</td>
<td>Retry 1 Number of resending of a sequence character at an invalid or no response. (range: 0 to 30)</td>
<td>10</td>
</tr>
<tr>
<td>RS037</td>
<td>Retry 2 Number of resending of a text at a block check error (reception of NAK). (range: 0 to 10)</td>
<td>3</td>
</tr>
<tr>
<td>RS038</td>
<td>Block check method 0: Check sum</td>
<td>0</td>
</tr>
</tbody>
</table>
Appendix A.6

Specifications for the video conferencing software

Appendix A.6 shows the different settings and protocols the video conferencing software is able to handle and uses for the transmission of video images between the remote and host site.

- Selectable composite video inputs: NTSC or PAL.
- Image control: speed, contrast, brightness, hue, and saturation.
- Composite video output: NTSC or PAL.
- H.320 standard CODEC.

- Line In and Out ports.
  Basic Module: Microphone in and Speaker out; Expand-5 Module: Microphone in and Speaker out, Phone Set I/O; Lin in and Speaker out; Expand-6 Module: Microphone in and Speaker out, Phone Set I/O.
- Variable compression standard: G.711, G.728.

PCI Video Overlay Support:

- Support scalable video window to any size.

Frames Rates:

- ISDN 2B channel (128 kbps).
- QCIF (Quarter Common Intermediate Format, 176 x 144 pixels) 30 frames per second.
- CIF (Common Intermediate Format, 352 x 288 pixels) 26 frames per second.

Temperature Requirement

- 10° C to 50° C.

Power Requirement

- 5 V ± 5% 2A.
- 12 V ± 100 mA.
Appendix A.7

The DCB structure

The DCB itself looks like this. The DCB structure defines the control setting for a serial communications device.

typedef struct _DCB { // dcb
    DWORD DCBlength; //sizeof (DCB)
    DWORD BaudRate; //current baud rate
    DWORD fBinary: 1; //binary mode, no EOF check
    DWORD fparity: 1; //enable parity checking
    DWORD fOutxCtsFlow: 1; //CTS output flow control
    DWORD fOutxDsrFlow: 1; //DSR output flow control
    DWORD fDtrControl: 2; //DTR flow control type
    DWORD fDsrSensitivity: 1; //DSR sensitivity
    DWORD fXContinuonXoff: 1; //XOFF continues Tx
    DWORD fOutX: 1; //XON/xOFF out flow control
    DWORD fInX: 1; //XON/xOFF in flow control
    DWORD fErrorChar: 1; //enable error replacement
    DWORD fNull: 1; //enable null stripping
    DWORD fRtsControl: 2; //RTS flow control
    DWORD fAbortOnError: 1; //abort reads writes on error
    DWORD fDummy2: 17; //reserved
    WORD wReserved1; //not currently used
    WORD XonLimit; //transmit XON threshold
    WORD XoffLimit; //transmit XOFF threshold
    BYTE ByteSize; //number of bits/byte, 4-8
    BYTE Parity

4= no, odd, even, mark, space
BYTE StopBits // 0, 1, 2 = 1, 1.5, 2
char XonChar // Tx and Rx XON character
-char XoffChar // Tx and Rx XOFF character
-char ErrorChar // error replacement character
-char EoffChar // end of input character
-char EvtChar // received event character
WORD wReserved1; // reserved do not use
} DCB
The University of Western Sydney
School of Engineering & Industrial Design

TELEOPERATED SYSTEM FOR VISUAL MONITORING OF SURGERY

by

TORE IDSOE

March 2002

Thesis submitted in fulfilment of the requirements for the degree of Master of Engineering (Honours)
PLEASE NOTE

The greatest amount of care has been taken while scanning this thesis,

and the best possible result has been obtained.
Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. John Gal for his patient supervision, concrete direction and instructive suggestion. His enthusiasm, understanding and cheerful manner have made this work possible.

Many thanks are also given to Dr. Gu Fang and Hisham Darjazini for their help, support and encouragement.

Finally, I am grateful to my family, friends and fellow postgraduate students in this school for their understanding, help and friendship throughout this study.
Statement of Originality

The material contained in this thesis is original and independently performed except where otherwise indicated by references to previous research and publications. No part of this thesis has been submitted for the award of a higher degree or diploma.

Tore Idsoe
Abstract

In this thesis the development of a remotely controlled system used for visual monitoring of surgical procedures at distant locations is described. The system has been developed for laboratory testing, where in the longer term it is to be verified under field conditions. Using existing technology in areas of serial communication and video conferencing in a new configuration, it has been shown that it is possible to achieve such a system. The system is intended to assist in performing complex surgical procedures at remote locations where specialist surgeons are normally unavailable. With the prototype system developed in this thesis, a remotely based general surgeon performing an operation can consult and interact with other specialist surgeons through visual observation and voice communications.

The prototype system forming the subject of this thesis, incorporates a six degree-of-freedom robot with a CCD camera as its end-effector and it is controlled through a Windows based program. Communication between the local and remote sites is implemented by a modem and LAN connection. A testing procedure to verify the efficiency of the monitoring system is also presented in this thesis.

The teleoperated system consists of two computers, a commercially available robot and a videoconferencing unit.
## Contents

Acknowledgements ........................................................................................................ ii  
Statement of Originality ................................................................................................. iii  
Abstract ........................................................................................................................ iv  
Contents .......................................................................................................................... v  
List of figures .................................................................................................................. vii  
List of tables .................................................................................................................... viii  
Terminology ...................................................................................................................... ix  

### CHAPTER 1: Introduction .......................................................................................... 1  
1.1 Aims ....................................................................................................................... 1  
1.2 Background ............................................................................................................ 2  
1.3 Thesis Structure ...................................................................................................... 7  

### CHAPTER 2: Literature Review ............................................................................... 9  
2.1 Introduction ............................................................................................................ 9  
2.2 Robots in Telemedicine ....................................................................................... 10  
2.3 General features of a Teleoperated System ......................................................... 11  
   *Industrial Robots* .................................................................................................. 13  
2.4 Application of Telemanipulators in Surgery ....................................................... 14  
   *Robotic Control Systems* .................................................................................. 17  
2.5 Visual Monitoring in Telemedicine .................................................................... 19  
   *Teleconferencing Application* ........................................................................ 21  
2.6 Data-transmission ............................................................................................... 23  
   *Integrated Services Digital Network* ................................................................ 25  
   *Asynchronous Transfer Mode (ATM)* .............................................................. 26  
   *Transmission Delays* ....................................................................................... 27  
2.7 Training and Simulation, Virtual Reality ............................................................. 29  

### CHAPTER 3: Laboratory Prototype ......................................................................... 31  
3.1 Introduction .......................................................................................................... 31  
   *Overall Setup* .................................................................................................... 31  
3.2 Prototype Sub-Systems ....................................................................................... 33  
   *Robot Manipulator* ......................................................................................... 33  
   *The Computers* ............................................................................................... 35  
   *Modem Interface* ............................................................................................. 36  
   *CCD Cameras* ................................................................................................. 37  

### CHAPTER 4: Implementation of the Laboratory Prototype .................................. 38  
4.1 Introduction .......................................................................................................... 38  
4.2 Software Development ....................................................................................... 39  
4.3 Hardware Aspects ............................................................................................... 47
4.4 Communications

Communication Features .......................................................................................... 50
RS-232 Connection to the Robot .............................................................................. 54

4.5 Data Transmission

Robot Control Function ......................................................................................... 58

4.6 Modems

The Modem ............................................................................................................ 61

CHAPTER 5: Vision Communications ........................................................................ 65

5.1 Introduction ......................................................................................................... 65
5.2 Video Conferencing

Important Features .................................................................................................. 66
Video Compression Techniques ............................................................................... 67

5.3 Communication Lines and Videoconferencing

Different Forms of ISDN ......................................................................................... 70
ATM (Asynchronous Transfer Mode) ....................................................................... 72
Features of ATM .................................................................................................... 73

CHAPTER 6: Path Control .......................................................................................... 75

6.1 Introduction ......................................................................................................... 75
6.2 Theory ................................................................................................................ 76
6.3 Numerical verifications of re-orientation of the camera .................................... 87

CHAPTER 7: Conclusion and Discussion .................................................................. 94

7.1 Summary ............................................................................................................ 94
7.2 Project Outcomes .............................................................................................. 95
7.3 Recommendations ............................................................................................ 97

References ............................................................................................................... 99

Appendices ............................................................................................................... 105

A.1 C++ source code for terminal program at local site ....................................... 105
A.2 Source code for the dialogbox .......................................................................... 116
A.3 ASCII code chart .............................................................................................. 122
A.4 Host control set-up for the Robot controller .................................................... 123
A.5 Parameter settings ............................................................................................ 126
A.6 Video conferencing .......................................................................................... 127
A.7 The DCB structure ............................................................................................ 128
List of Figures

Figure 2.1 A teleoperated robotic system (* LAN used in Laboratory).................. 12

Figure 2.2 General loop.................................................................................. 18

Figure 3.1 The overall setup........................................................................... 32

Figure 3.2 An overview of the Motoman six degrees of freedoms manipulator
used in this project [47]................................................................................ 35

Figure 4.1 The windows based control panel for the robot controller.............. 44

Figure 4.2 Flowchart for data transmission between the remote and host locations
(ACK is the acknowledge signal as in figure 4.9)............................................ 46

Figure 4.3 The laboratory setup showing both host and remote sites............... 49

Figure 4.4 Flowchart for the handshaking signals.......................................... 52

Figure 4.5 The five required lines for two-way data transmission.................... 53

Figure 4.6 (a) RS-232 configuration for the Robot controller.......................... 54

Figure 4.6 (b) RS-232 configurations for the Robot controller....................... 55

Figure 4.7 Robot controller setup in Host Control Mode............................... 56

Figure 4.8 Remote display, showing command mode operation setting [47]........ 57

Figure 4.9 Data transmission protocol between the robot controller and a
personal computer......................................................................................... 59

Figure 4.10 Command and response blocks for robot controller..................... 61

Figure 4.11 An example of a modem setup..................................................... 62

Figure 5.1 ATM network structure.................................................................. 73
Figure 6.1 The setup for calculating the re-alignment of the end-effector after a MOVL command................................. 77

Figure 6.2 The base axes and the tool axes [47]........................................... 78

Figure 6.3 The $Z_T$ axis......................................................................... 83

Figure 6.4 The robot at its calibration position with the camera attached to it and aligned for viewing the target at the calibration point......................... 91

Figure 6.5 The camera re-aligned after a MOVL command........................... 92

Figure 6.6 The re-alignment procedure seen through the camera lens........... 93

List of Tables

Table 5.1 Video compression protocols included in H.320.......................... 68
Table 5.2 Frame rates achievable using Eyelink video conferencing software..... 69
Table 5.3 Picture formats supported using H.261...................................... 70
Table 5.4 ISDN channels and available services........................................ 71
Table 6.1 Coordinates and angles for calibration position and position 1........ 90
**Terminology**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laparoscopic surgery</td>
<td>Special form for surgery</td>
</tr>
<tr>
<td>Null Modem</td>
<td>A RS-232 cable where the handshaking signals are crossed.</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft Foundation Class Library</td>
</tr>
<tr>
<td>LAN/ WAN</td>
<td>Local area network/ wide area network</td>
</tr>
<tr>
<td>ISDN</td>
<td>Integrated services digital network.</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous transfer mode</td>
</tr>
<tr>
<td>Baudrate</td>
<td>Speed of data transmission in bits per second.</td>
</tr>
<tr>
<td>Parity</td>
<td>Very basic check of information integrity.</td>
</tr>
<tr>
<td>DTR</td>
<td>Data terminal ready, handshaking signal</td>
</tr>
<tr>
<td>DTS</td>
<td>Data</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to send</td>
</tr>
<tr>
<td>AT</td>
<td>Hayes commands for modems, ATtention.</td>
</tr>
<tr>
<td>Hayes compatible</td>
<td>Modems that can be controlled using AT commands</td>
</tr>
<tr>
<td>Nissen fundoplications</td>
<td>Stomach surgery.</td>
</tr>
<tr>
<td>T1</td>
<td>A variety of ISDN bandwidth</td>
</tr>
</tbody>
</table>