



Norwegian University of
Science and Technology

High-Level Control System for Remote Controlled Surgical Robots

Haptic Guidance of Surgical Robot

Andreas Nygaard

Master of Science in Engineering Cybernetics

Submission date: June 2008

Supervisor: Tor Engebret Onshus, ITK

Co-supervisor: Øyvind Stavadahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The master work assignment should use the work performed in the project autumn 2007 as a background and a basis for further development and implementation.

The objective of this assignment is to develop a high-level control system in order to increase the degree of autonomy of the present telesurgical systems used in minimally invasive surgery. The system should be able to assist the surgeon during the execution of an operation, guiding his movements and performing some tasks in a semi-autonomous mode.

The approach that the candidate shall follow is Haptic guidance (or virtual fixtures), which allows the surgeon to define motion constraints that depend on the task to be performed. These constraints usually consist in the generation of virtual contact forces that force the surgeon to follow a desired path or avoid movements towards forbidden regions.

The following steps need to be followed and addressed in the final report:

1. Literature study:

Investigate haptic feedback at a generic level, pointing out technological and physiological possibilities and challenges. Explain which of these factors are especially relevant for the present application.

2. System analysis:

Describe the system as proposed by The Interventional Centre, and establish a theoretical model of this system. Analyse the model in light of the findings from point 1, and point out any weaknesses and/or obvious potential improvements in the proposed equipment.

3. Concept and algorithm development:

Extend the concept of linear guidance for needle insertion applications (as treated in a previous project) to a more generic system comprising an input interface for segmented non-linear surfaces from different medical image modalities (e.g. CT or MRI) as forbidden regions.

4. Implementation:

As far as time permits, implement and assess the extended concept from point 3 to comply with available equipment and software platforms.

Advisors:

Ole Jakob Elle and Jordi Cornella, Rikshospitalet University Hospital
Øyvind Stavdahl, ITK

Assignment given: 07. January 2008

Supervisor: Tor Engebret Onshus, ITK

Work conducted at
The Interventional Centre,
Rikshospitalet University Hospital



RIKSHOSPITALET

Abstract

This report considers the work to improve the autonomy of surgical teleoperated systems, by introducing haptic guidance.

The use of surgical robots in surgical procedures have become more common the recent years, but still it is in its infancy. Some advantages when using robots is scalability of movements, reduced tremor, better visualisation systems and greater range of motions than with conventional minimally invasive surgery. On the contrary, lack of tactile feedback and highly unstructured medical environment restricts the use of teleoperated robots to specific tasks within specific procedures.

A way of improving autonomy of the teleoperated system is to introduce predefined constraints in the surgical environment, to create a trajectory or forbidden area, in order to guide the movements of the surgeon. This is often called haptic guidance.

This report introduces the basics of teleoperated systems, with control schemes, models and analytical tools. Algorithms for haptic guidance have been developed, and the entire control and guidance system have been modified and suited for implementation on a real teleoperated system. Theoretical analysis of the position position (PP) control scheme reveals some general stability and performance characteristics, later used as a basis for tuning the real system parameters.

The teleoperated system consists of a Phantom Omni device, from SensAble-Technologies, used as master manipulator, and AESOP 3000DS, from Computer Motions Inc., as the slave manipulator. The control system is implemented on a regular PC, connecting the complete system.

Tests reveal that the slave manipulator is not suited for this task due to a significant communication time delay, limited velocity and inadequate control possibilities. The consequences makes force feedback based on the PP control scheme impossible, and limits performance of the entire teleoperated system.

The guidance system is implemented in two variations, one based on slave positions and one based on master positions. This is motivated to give a performance comparison of variations in position error/tracking between the two manipulators. Slave based guidance appears to be stable only for limited values of the gains, and thus, it generates no

strict constraints. It can be used to guide the operator away from forbidden areas, but is not suitable for high precision guiding. The master based guidance is stable for very high gains, and the guidance have the accuracy to improve the surgeons precision during procedures. In the case of line guidance, the master based guidance gives a deviation of up to $1.3mm$ from the given trajectory.

The work has shown the possibilities of using haptic guidance to improve accuracy and precision in surgical procedures, but among others, hardware limitations give room for several improvements in order to develop a teleoperated system that works.

Preface

This report is submitted as my Masters Thesis at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). The work in this report is based on the pre project done on the same topic, and together they form the ninth and tenth semester of my Masters Degree.

During talks with Ole Jacob Elle and Jordi Cornella at the Interventional Centre and Øyvind Stavdahl and Tor Onshus at NTNU we came up with the idea that my Masters Thesis could be carried out at the Interventional Centre. A research project about high level control systems for remote controlled surgical robots was already going on and a thesis with this theme was suggested. As a part of the already existing research project, this report considers the documentation of the work done about haptic guidance or virtual fixtures for remote controlled surgical robots. Equipment available at the Interventional Centre was in mind when the work was carried out, and the work is done at the Interventional Centre at Rikshospitalet University Hospital. The persons already named have been my advisors for the respective institutions.

I want to thank Tor Onshus for pursuing the time limits and Øyvind Stavdahl for his enthusiasm and inspiring talks. Jordi Cornella deserves acknowledgement for sticking up with my never ending questions and for giving very valuable feedback, and Ole Jacob Elle for motivation and for putting things in perspective. Finally I want to thank my wife Maria, for always encouraging me and believe in me.

Oslo, May 31st, 2008

Andreas Nygaard

Nomenclature

\cdot_m	–	master
\cdot_s	–	slave
\cdot_h	–	human/operator
\cdot_e	–	environment
\cdot_i	–	initial
\cdot_d	–	deviation
s	–	Laplace operator
\mathbf{x}	–	positions in Cartesian space
\mathbf{v}	–	velocities in Cartesian space
\mathbf{f}	–	forces in Cartesian space
$\boldsymbol{\tau}$	–	torque
\mathbf{q}	–	joint angles of slave
$\boldsymbol{\theta}$	–	joint angles of master
\boldsymbol{v}	–	space vectors
\mathbf{p}	–	spatial point
\mathbf{M}	–	inertia matrix
\mathbf{V}	–	Coriolis/centrifugal matrix
\mathbf{N}	–	gravity matrix
\mathbf{J}	–	Jacobian matrix
\mathbf{Z}	–	impedance matrix
\mathbf{Y}	–	admittance matrix
\mathbf{H}	–	hybrid matrix
\mathbf{G}	–	inverse of hybrid matrix
\mathbf{F}	–	transmission matrix
\mathbf{F}^*	–	inverse of transmission matrix
\mathbf{S}	–	scattering operator
\mathbf{I}	–	identity matrix

- P – proportional controller
- PD – proportional-derivative controller
- e – energy
- w – wave variable
- m – mass
- \mathbf{c} – damper constant
- \mathbf{k} – spring constant
- ω – frequency
- ζ – damping ratio
- r – real part
- i – imaginary part
- g – residue

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Surgical Robots - Classification	3
1.2.1	Pros and Cons of Surgical Robots	5
1.3	Human Sensing	7
1.3.1	Kinesthetic Feedback	7
1.3.2	Tactile feedback	8
1.3.3	Design Implications	9
1.4	Haptic Guidance - State of Art	10
1.4.1	Forbidden Regions	10
1.4.2	Trajectory Guidance	11
1.5	Objectives of the Work	12
1.5.1	Outline	12
2	Teleoperated Systems	15
2.1	Basic Description	15
2.2	Modelling the System	17
2.2.1	Human and Environment Models	18
2.2.2	Master-Slave System	19
2.3	Stability Analysis	21
2.3.1	Analysis of the whole system	21
2.3.2	Master-Slave System	21
2.4	Performance	23
2.4.1	Perfect Transparency	24
2.5	Control Schemes	25
2.5.1	Position Position Control	25

2.5.2	Force Position Control	26
2.5.3	4 Channel Control	27
2.5.4	Comparing the Control Schemes	28
2.6	Wave Variables	28
2.7	Practical Issues	29
2.7.1	Obtaining the Hybrid Matrix From a Real System	30
3	Haptic Guidance	33
3.1	The Guidance Loop	33
3.2	Guidance Types	35
3.3	Algorithms	36
3.3.1	3 Constraints - Point	36
3.3.2	2 Constraints - Line	37
3.3.3	1 Constraint - Plane	38
3.3.4	Force Generation	39
3.3.5	Max Distance - Max Force	41
3.4	Object Segmentation	42
4	Kinematics and Dynamics of the Manipulators	45
4.1	Slave Robot	46
4.1.1	Kinematics	47
4.1.2	Dynamics	49
4.2	Master Device	50
4.2.1	Kinematics	51
4.2.2	Dynamics	53
4.3	Manipulator Dynamics in Cartesian Space	54
5	Modelling and Implementing the System	57
5.1	Numerical Evaluation	58
5.1.1	Stability Analysis	59
5.1.2	Performance	61
5.1.3	Time Delay	62
5.2	Control Scheme	63
5.2.1	Tuning the PD Controllers	64
5.2.2	Transformation of Coordinate Frames	65
5.3	Haptic Guidance	66

5.3.1	Features	67
5.4	Implementation Details	68
5.4.1	Master and Slave Control Loop	69
5.4.2	Force and Velocity Generation	70
5.4.3	Main Control Loop	70
6	Experiments and Results	73
6.1	Position Position Control Scheme	73
6.1.1	Master Slave Position Tracking	73
6.1.2	Force Feedback	75
6.2	Slave Based Guidance	76
6.2.1	Force vs Deviation	76
6.2.2	Stability	77
6.2.3	Effect of the Maximum Distance Limit	78
6.2.4	Guidance Modes	78
6.3	Master Based Guidance	80
6.3.1	Comparing Master and Slave Guidance	81
7	Discussion and Conclusions	85
7.1	Hardware	85
7.1.1	Phantom Omni	85
7.1.2	AESOP	86
7.2	Control Scheme	87
7.2.1	Force Feedback	87
7.2.2	Stability and Performance	88
7.2.3	Improving Position Tracking	89
7.3	Haptic Guidance	89
7.3.1	Limitations	90
7.3.2	Slave Based Guidance	90
7.3.3	Master Based Guidance	91
7.4	Conclusion	92
7.4.1	Future Work	93
A	Manipulator Details	101
A.1	Phantom Omni Dynamics	101

B Haptic Guidance Source Code	105
B.1 Initiating the System	105
B.2 Main Control Loop	112
B.3 Force / Velocity Generation	114
B.4 Slave Control	119
B.5 Master Control	122
B.6 Key Hit Function	125

List of Figures

1.1	Minimally Invasive Surgery a) overview and b) inside abdominal cavity	2
1.2	a) The CASPAR robot and b) a microrobot developed by Delft University of Technology (2006) crawling into an intestinal canal	5
1.3	(a) The daVinci robot and (b) the The Zeus robot	6
2.1	Different ways to manipulate an object (Christiansson, 2007)	16
2.2	Two port teleoperation system	17
2.3	Model of (a) the operator and (b) the environment (Christiansson, 2007)	19
2.4	Block diagram of a) a position position and b) a force position controller (Aliaga et al., 2004)	26
2.5	Block diagram of a 4 Channel Controller (Aliaga et al., 2004)	27
2.6	Block diagram of a Wave Variable Communication (Niemeyer and Slotine, 2004)	29
3.1	Implemented PP control scheme with haptic guidance	34
3.2	Haptic guidance according to number and sense of the constraints.	36
3.3	The deviation from a) a line and b) a plane	38
3.4	The frequency response for the different values of ζ	41
3.5	The max distance cylinder	42

4.1	The Lab setup of AESOP, its control system and the external PC	46
4.2	The kinematics of the AESOP	47
4.3	The kinematics of the Phantom Omni	51
5.1	Criteria for (a) absolute stability and (b) passivity	61
5.2	Implemented Position Position control scheme	63
5.3	Implemented PP control scheme with haptic guidance	66
5.4	Opening window when starting the guidance system	68
5.5	Sequence diagram of the communication in the system	71
6.1	Position tracking error between master and slave	74
6.2	Slave based guidance	77
6.3	Properties of slave based line guidance	79
6.4	Master based guidance	80
6.5	(a) Line and (b) plane guidance based on master guidance	82

Chapter 1

Introduction

1.1 Motivation

Some parts of this report are based on or taken from the pre project of Nygaard (2007).

Minimally invasive surgery (MIS), like laparoscopy and endoscopy, is a way of performing surgical procedures by inserting tools through small incisions or natural orifices on the human body, see Figure 1.1. This is a technology developed to accomplish an optimal result for the patient by mainly intervene with the area that is injured and not the surrounding organs, muscles and tissue. The benefit, compared with open surgery, is to minimise the invasion and trauma to the adjacent tissue and muscles. The consequences are reduced recovery time for the patient, the hospital stay gets shorter and the incidence of post-surgical complications and morbidity is reduced.

Drawbacks with MIS are that the tactile feedback is reduced or even non existing, vision is reduced to 2D video monitoring, the operating space is reduced making manipulations more difficult and demands higher precision from the surgeon, and the stiff surgical tools gives less degrees of freedom in manipulation than conventional open surgery. Consequences for the patient may be internal bleedings, infections and even death.

Ways to improve the MIS are continuously developed, and a tendency is the increasing number of robotic systems developed to help the

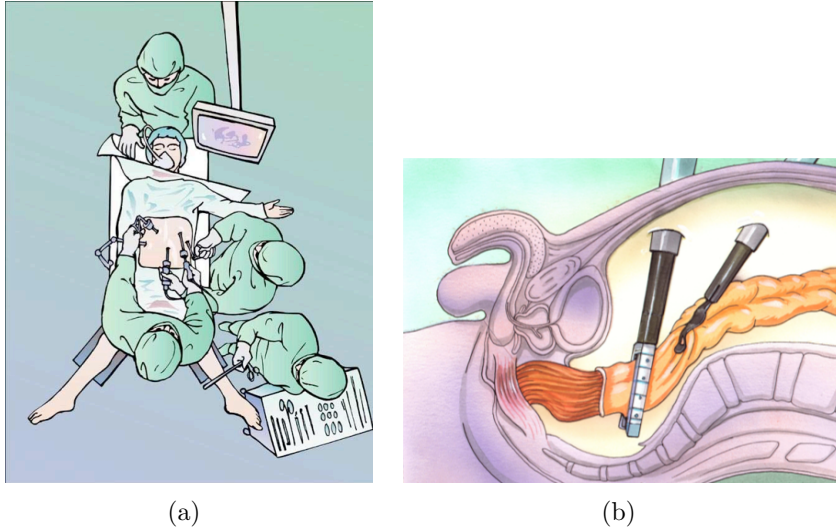


Figure 1.1: Minimally Invasive Surgery a) overview and b) inside abdominal cavity

surgeon during the surgical procedures. One group of surgical robot systems are the teleoperated systems where the surgeon performs the surgical procedure indirectly, by controlling a robot which performs the actual operation. The lack of tactile feedback is one of the most significant drawbacks when using these systems, and to solve this problem researchers try to develop systems which incorporates force feedback, giving the surgeon the ability to sense the environment interacting with the robot. An advantage appears when introducing virtual forces with the objective of guiding the motions of the operator helping him to complete the task successfully. This is also called haptic guidance or virtual fixtures.

Haptic guidance is the topic of this report, and through introducing the basics of teleoperated systems and tools for development and analysis, a teleoperated system with haptic guidance system will be developed, analysed and implemented.

1.2 Surgical Robots - Classification

Despite the fact that robots have been incorporated in most parts of industry there has been little development of robotics in healthcare until some decades ago. Especially robotic surgery is in its infancy. With the introduction of MIS the possibilities and potential of using robotics in surgery became obvious.

A significant number of different robotic applications have been developed and several approaches has been made to classify the various medical robotic devices. One of them is suggested by Elle (2004), and he divides the applications into four categories:

- Robotic assistance
- Remote-/teleoperated manipulators
- Autonomous and image-guided robotic surgery
- Micro-/nanorobotics

The first class is the robot systems that assist in the conventional surgical setups. An example of this is the Aesop (Automated Endoscope System for Optimal Positioning) robot from Computer Motion Inc. (Shew et al., 2003), which is developed to hold the videoscope in minimally invasive surgery and in this way assist the surgeon. It is voice controlled so that the surgeon can control it directly.

Remote manipulators are robotic systems controlled online. The surgeon does not have any physical contact with the patient but performs the procedure through a control console. What makes the distinction between the remote and teleoperated manipulators is the distance from the control console to the surgical device. Telerobotics is mainly remote-operated manipulators controlled through a cable but tests have been carried out from a longer distance through a telecommunication network. An example of the last case was an surgical operation performed overseas with the Zeus robot system, with the surgeon placed in New York and the patient in Strasbourg (IRCAD et al., 2001). Both Zeus and the da Vinci robot is mainly controlled remotely by cable,

but in this report we use teleoperated system about this category in general.

Autonomous robotic systems perform surgery by themselves by executing predefined commands and procedures. Some of the systems does not care about the surgical environment while others adapt to the varying surgical environment by processing sensory data from for example vision, infrared, ultrasound, laser or contact sensation devices. The latter case is referred to as image-guided robotic systems. CASPAR (Computer Assisted Surgical Planning and Robotics) developed by Orto MAQUET GmbH is an image-guided robotic system which prepares for hip prosthesis by performing the milling of the femur following a predefined path computed on the basis of CT images (Petermann et al., 2000), see Figure 1.2.

The final class is micro-/nanorobotics, which is a result of the evolution and size reduction of electronics, materials, machinery, sensors, micromotors and others making it possible to produce intelligent and microscopic robots. This opens for a new way of performing surgical procedures. Systems in this class varies from multifunctional instrument heads carried by flexible and intelligent endosystems used for endoscopic surgery (Schurr et al., 1998), and to remote controlled micromachines able to crawl around into anatomic lumens (Flynn et al., 1998), see Figure 1.2.

It would be natural to include another category, Synergetic/hands-on robot, which is a combination of remote-/teleoperated manipulators and autonomous and image-guided robots (Taylor and Stoianovici, 2003).

As already described several robot systems are developed and being developed. This project deals with a combination of robots belonging to the class of remote-/teleoperated manipulators and autonomous/semi autonomous/shared autonomy robots. As earlier mentioned two examples are the Zeus system developed by Computer Motion Inc. and the da Vinci developed by Intuitive Surgical Inc. Both systems are master-slave robots with multiple arms controlled from a remote console. One arm is dedicated to control a 3D camera and the others is used for manipulating the surgical tools. At the master console of the Zeus system a monitor and polarised goggles visualises in 3D the operating

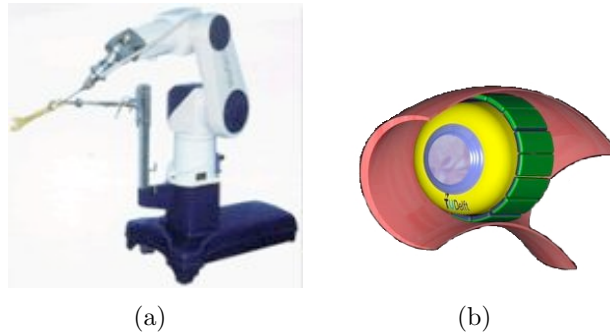


Figure 1.2: a) The CASPAR robot and b) a microrobot developed by Delft University of Technology (2006) crawling into an intestinal canal

environment from the feedback of a AESOP voice activated camera. Instrument handles control two arms of the robot replicating the left and the right hand of the surgeon. The arms can utilise conventional endoscopic tools or end-effectors with a flexible wrist technology called Micro-Wrist giving 7 degrees of freedom (DOF) inside the operating room. The Zeus surgical system is no longer on sale.

The da Vinci system uses a binocular like monitor to generate real 3D high definition visualisation from a 3D endoscopic camera. With two instrument handles and foot pedals the surgeon can control the robot arm holding the camera and three other robotic arms controlling the surgical tools creating the illusion of being the extension of the instrumental handles. As with the Zeus system the da Vinci system make use of conventional endoscopic tools but has also the ability of using a flexible end-effector with the EndoWrist technology giving 7 DOF (Lanfranco et al., 2004; Cepolina and Michelini, 2004; Tavakoli et al., 2005).

1.2.1 Pros and Cons of Surgical Robots

Though the technology of robotic surgery is in its infancy, it is evolving and shows a lot of potential. Compared to conventional MIS the robot assisted surgery can give several improvements. The high definition 3D images that the da Vinci system generates are of a excellent

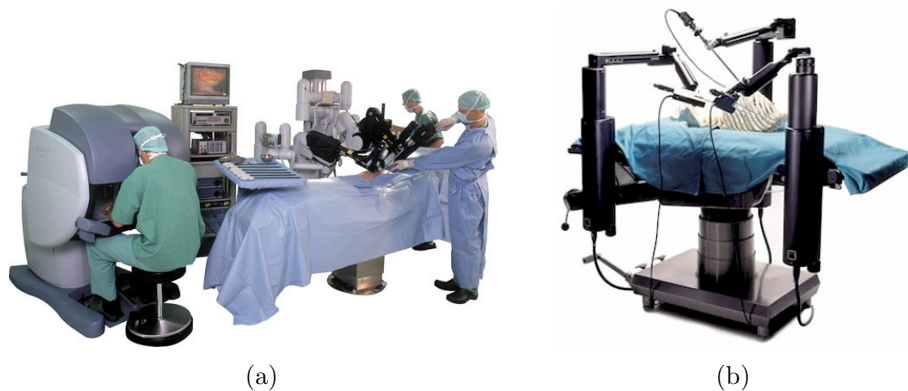


Figure 1.3: (a) The daVinci robot and (b) the The Zeus robot

quality and exceeds by far the comparable visualising systems used in conventional MIS. It provides directly controllable stable visual field with increased resolution and depth of field. The 7 DOF end effectors gives new abilities of manipulating the instruments and in a narrow and limited operating environment it may contribute to making new procedures possible. The system has the ability to reduce physiological tremor and scale motions making microscopic operations possible. Another important advantage is to reduce fatigue of the surgeon by its ergonomically designed workstation.

There are several drawbacks with robot assisted surgery. The robotic system is very expensive and requires a high start-up cost. It may also require extra staff to operate and as the systems get more complex technical expertise is required to maintain the system. The size of the system may demand special operating room to fit and this makes an extra cost to accomplish. The disadvantages mentioned are about the exterior and economical problems to account for when evaluating the use of robots in surgery. On the other hand there are technological challenges that makes these systems disadvantageous. Absence of touch and haptic sensation makes the surgical procedures less intuitive and it is not possible to examine tissue by palpation or tactile perception. The lack of feedback gives the robot no ability to exploit artificial intelligence or perform any kind of judgement which is desir-

able to make the robot semi-autonomous and able to ease the surgeon of simple tasks (Lanfranco et al., 2004; Tavakoli et al., 2005).

1.3 Human Sensing

When a human performs a manipulation of something, extensive information processing is done in order to control the action. The information comes from impulses generated by the sensor system which consists of the eyes, the ears and sensory receptors within the muscles, tendons, joints and skin (Rosenbaum, 1990). Proprioceptive information related to movement (such as gripping) may be called *kinesthesia*, while *tactile* feedback concerns collecting information from the world around us (as the sense of touch). These groups work together while providing information about the human body and the surroundings and both are lost when using robot manipulators (Ottermo, 2006).

1.3.1 Kinesthetic Feedback

Within the muscles some sensory nerve fibres are wrapped around central parts of the muscle which respond to length variations. When stretched, these sensors send nerve impulses to the spinal cord. A contracted muscle results in diminishing sensor activity, and a stretched muscle sends impulses to trigger reflex responses. Consequently, a person can sense and respond to an extended limb.

In the same way sensory fibres are wrapped around the tendons connecting muscles to bone. But these respond by activating when the muscles are contracting, and the activation threshold is quite low, actually they respond to induced muscle tension of a tenth of a gram or less (Rosenbaum, 1990).

Also within the joints connecting the limbs of the body sensory fibres are present. The role of these sensors are undecided. Early research reported that the sensors respond to different joint angles, but tend to adapt slowly, continuing to fire for a long time after a joint angle is assumed (Skoglund, 1956). Later, the joint sensors are reported to adapt quickly but only respond to extreme joint angles (Grigg, 1976;

Clark and Burgess, 1975) which implies that the sensors work as a warning when human limbs are stretched to its limits.

1.3.2 Tactile feedback

When movements are made, the fingers touch some object or a limb is rubbed against something, a sense of displacement or touch is created. This is due to the sensory receptors placed in the skin, also called *mechanoreceptors*, which respond to mechanical deformation of the skin. There are also sensors in the skin responding to temperature and pain (Rosenbaum, 1990). The sense of touch, temperature and pain are also called tactile feedback. The sense of movement belongs to the previous category, kinesthesia, but is tightly connected to the sense of touch since the same sensors provide both sensory experiences.

As stated earlier, these senses may lead to reflex responses. The information from the sensors are sent to the spinal cord where the first "processing" is taking place and in some cases a muscle response is triggered directly from here. The connected system is also called the *sensor and motor control system* (Rosenbaum, 1990). This results in a very low response time. An example of this is when a person lifting a small object, the grip may tighten 80 msec after the object has slipped very slightly (Johansson and Westling, 1988).

The sensitivity of touch is very high. Though there are individual differences, tests have shown that the threshold for sensing touch is approximately 80 mg at the finger and 150 mg at the palm (Sherrick and Craig, 1982). The ability to discriminate two simultaneous stimulus points at the skin is limited to 2.5 mm between the points at the finger and 11 mm at the palm (Shimoga, 1993). The lower time limit to detect and discriminate two consecutive stimuli is reported to be about 5 msec. The corresponding time limit for the eye is 25 msec. Vibration is also detectable by the sensors, and the highest sensitivity for vibration is for frequencies between 200-250 Hz (Burdea, 1996).

To measure the rapidity in which the human can respond one can talk about bandwidth. As for tactile and kinesthetic feedback the bandwidth refers to the frequency of which the stimuli are sensed (input), while for motor control the bandwidth refers to the rapidity in which

the human can respond (output) to the sensed stimuli. The input bandwidth is much higher than the output, which means that the stimuli is sensed much faster than we can respond to them (Burdea, 1996). Shimoga (1992) showed that the bandwidth of the sensor and motor control loop of the hand and the fingers are about 5-10 Hz. By comparison the kinesthetic sensing has a bandwidth of 20 to 30 Hz, and tactile sensing has about 0 to 400 Hz bandwidth.

1.3.3 Design Implications

The reason for going in depth on this topic is the fact that it is an advantage to know the characteristics of the human sensing system in order to develop an optimal haptic system. There are a lot of literature about the anatomy and physiology of the human body and studies giving approximate values for various parameters which can be utilised in the development process. On the other hand it would not be wise to use these results blindly in an ideal haptic setting because the settings and the experimental setups for providing the parameter values most likely will vary significantly in real life. Thus, the literature and the studies should be used as guidelines for a haptic system.

From the discussion above, we see that the human sensing system is sensitive to very small tactile forces, which implies that the force range is large. This is an advantage when designing a haptic system because small forces can be used to give force cues. On the other hand to recreate a rigid or hard contact large forces are required which will cause user fatigue. Eventually, it is a compromise between making a realistic feedback system and minimising user fatigue.

Another point is the frequency of force generation. We see that the bandwidth of the sensor and motor control loop of the fingers are about 5-10 Hz, while the kinesthetic sensing has bandwidth of 20-30 Hz and the tactile sensing has a bandwidth of up to 400 Hz. Rosenberg (1995) recommend a force feedback bandwidth of at least 50 Hz to generate a natural force feedback. However Howe and Kontarinis (1992) show that even 8 Hz is sufficient, and increasing the bandwidth up to 32 Hz gives no significant advantages. SensAble Technologies (2007) recommends to use an update frequency of 500-1000 Hz to give their haptic device

a realistic performance. In teleoperated systems, time delays are usual, and in such cases the bandwidth of the haptic will be limited by the communication channel.

The force feedback device will act on both tactile and kinesthetic sensors in combination, and thus the requirements must consider all the aspects of the two categories.

1.4 Haptic Guidance - State of Art

1.4.1 Forbidden Regions

An important objective of creating haptic guidance is the possibility of making forbidden regions or no-go zones within the operating space where the robot arm is not able to move. This can be vulnerable organs and tissue which is desirable that the surgeon does not touch with the tools or the robotic arm. By using computed tomography (CT) or magnetic resonance (MR) images, one can define the forbidden regions ahead of the operation. In areas of approximately static conditions, like orthopedic operations, this may be satisfactory. On the other hand, surgery in non static surroundings, like gastroscopy, this is not the case because all the organs float around without static geometry and coordinates. What may be a solution to this is to make use of some real time images or data, for example images from CT or ultrasound, to recalculate the constraints in real time.

Li et al. (2007) worked out a spatial motion constraints approach to assist the surgeon performing sinus surgery with a surgical robot. With a CT image as basis, they predefine anatomic-based constraints and 'no fly zones' and by using an optimisation algorithm they calculate a smooth trajectory for moving the robot arm. By modelling the surgical tool and shaft as one or more cylinders a collision detecting algorithm prevent the tool and the tool tip from colliding with the boundary constraints. As the method is developed motivated by endoscopic sinus surgery the operating room can be described as static and consequently the constraints can be predefined using a CT image.

Another project is where Marayongl et al. (2003) have given a so-

lution for a geometric assistance mode for a surgical robot. The idea is to use translational and rotational fixtures as control law for movement inside a limited area and by doing this helping the surgeon to perform high precision tasks. Within a very limited task context they have shown how to translate a closed loop control algorithm into a virtual fixture.

1.4.2 Trajectory Guidance

Haptic guidance may also be used to guide the motion of the operator and not only prevent motion from forbidden regions. In many situations the human accuracy and precision is not good enough to perform microscopic or high precision procedures. If then the robot works in a semi-autonomous way, the robot can guide the surgeon and even perform some of the tasks autonomously. The advantage is that precision and accuracy increases making even the most microscopic procedures possible. The physics behind the trajectory guidance is based on giving force cues to prevent the guided tool to deviate from the desired trajectory. This is the inverse of the force utilised to prevent the guided tool to enter into a forbidden region.

Retinal vein cannulation is a procedure where a needle is inserted into a vein of approximately 100 microns in diameter. The needle has a diameter of 20-50 microns. In this situation there are almost no tactile feedback and depth vision is limited. This is a procedure almost impossible for a human to accomplish because of tremor and the small scale situation. Marayongl et al. (2003) argues that a robot assisting the surgeon can provide tremor free, steady and scaled motions. They have developed a series of surgical systems to improve speed and precision, among them haptic guidance based on virtual constraints. This is an example of procedure where haptic guidance would ease the task even more. With constraints the robot could be predefined to move linearly into and along the vein. Clearly this will provide a benefit compared to conventional procedures.

Another procedure in which haptic guidance would be beneficial is when performing percutaneous biopsy. The access to the target may be limited due to various factors like limited space and difficult angulated

access. In this situation a robotic system can improve accessibility and provide stable access, increased precision and accuracy, and accurate needle guidance. Kettenbach et al. (2004) have addressed this problem by developing a CT directed haptic guidance system for performing biopsies. They combine virtual reality display and haptic guidance to provide assistance in medical procedures.

1.5 Objectives of the Work

The objective of this work is to bring the teleoperated surgical systems a step further in being semi autonomous. A way of doing this is not only to include force feedback but also virtual fixtures or haptic guidance, which will be studied and developed further in this report.

The main goal is to develop and implement a teleoperated system with haptic guidance incorporated on available equipment to show how the system can help and improve the quality of a given task. The guidance system shall give the ability to be guided along trajectories and limit the operating environment by forbidden areas and no-go-zones. The operator shall be able to decide the strictness of the constraints and which mode to use.

The development of the system are based on known literature and theories and address issues concerning the performance of the system.

1.5.1 Outline

Chapter 1 gives some motivation for this report by looking into Minimally Invasive Surgery, robots in general, human sensing and haptic guidance systems in surgical context.

The task of developing a haptic guidance system on a teleoperated system includes many topics concerning the teleoperated system. *Chapter 2* gives a review of issues like different control schemes, performance and stability of these types of systems.

In *Chapter 3* some more details of haptic guidance and algorithms for generating virtual forces are presented.

Chapter 4 outlines the details of the different parts of the system, like the kinematics and dynamics of the master and slave robot.

While *Chapter 5* gives the features of the implemented guidance system and the implementation details of the source code.

Chapter 6 leads the reader through tests and analyses of the system, before the results will be discussed and concluded in *Chapter 7*.

Chapter 2

Teleoperated Systems

This chapter will present the basics of teleoperated systems, both at an intuitive and a mathematical level. Framework for analysing, tuning and controlling a teleoperated system will also be introduced.

2.1 Basic Description

The mission of a teleoperated system is to perform operations in an environment or context where it is not suitable or desirable for a human to perform the task. Reasons might be lack of space or accessibility, harsh or hazardous environment, environment not suited for humans or simply scaling motions to increase achievable force or reduce motions. Figure 2.1 gives an intuitive understanding of a teleoperated system. The desired action to be performed is to pick up an object, which we easily can do with our hand (Figure 2.1a). An alternative approach is to use some tool to do it, like a pair of pincers (Figure 2.1b). The pincers are a handle and an end-effector mechanically connected to copy the functionality of the hand. The teleoperated system also craves for a functionality like the hand with the handle and the end-effector, but the connection between these two parts are different from the pincers (Figure 2.1c). The mechanical connection is replaced by a master device to interact with the human through the handle and a slave device interacting with the object through the end-effector. Between the mas-

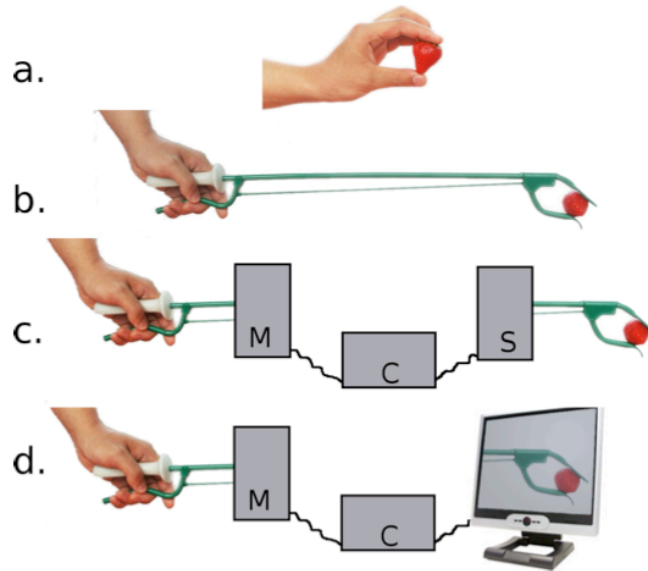


Figure 2.1: Different ways to manipulate an object (Christiansson, 2007)

ter and the slave a communication channel provides the contact that gives the desired functionality of the two robots. An advantage when using teleoperated systems is that the distance between the operator and the object can be large. The master-slave system may also be used to manipulate objects in virtual environments, then the end-effector and the environment are modelled virtually (Figure 2.1d).

A teleoperated system consists of five interacting subsystems: operator, master device, communication channel, slave device and environment (Lawrence, 1993). The master and slave devices make the teleoperator, and the operator and the environment are the external elements interacting with the teleoperator. The communication between the different subsystems is exchanging forces, velocities and positions. Taking as an example the da Vinci robot, the surgeon is the operator that manipulates the control console, which is the master, the robot tools are the slave that manipulates the organs of the patient which are the environment.

The task of the teleoperator is to transmit and execute the move-

ments commanded by the operator in a way that feels normal. An ideal teleoperator behaves in a manner which makes itself invisible for the operator and the environment (Lawrence, 1993), in other words the teleoperator subsystem in Figure 2.1c is removed so the operator manipulates the environment directly. This is impossible due to various reasons that can be summarised by the inimitable human body. The human body is very complex, the sensory system has a large bandwidth and can sense microscopic changes of texture, hardness, shape or movement (McLaughlin et al., 2001) and combined with a very fast kinaesthetic response the requirements of the teleoperator is too high for the present technology.

2.2 Modelling the System

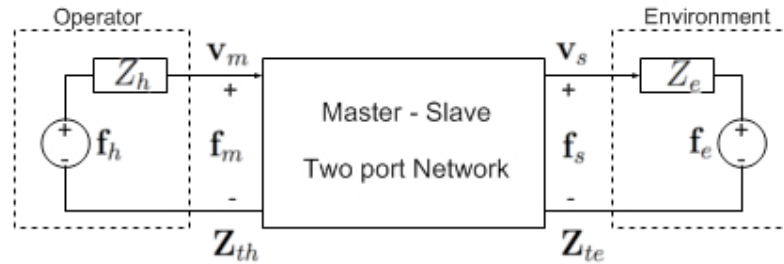


Figure 2.2: Two port teleoperation system

The teleoperation system can be modelled as two-port model (Niemeyer and Slotine, 1991), and analysed as an electrical circuit where velocities and forces represent intensities and voltages, as it is shown in Figure 2.2, and then the following relations are obtained. The teleoperator interface transmits forces \mathbf{f} and velocities \mathbf{v} between the local operator and the remote task which is the same as the environment. When the teleoperator is in contact with the remote task the slave velocities \mathbf{v}_s and forces \mathbf{f}_s are related by the impedance \mathbf{Z}_e and the forces \mathbf{f}_e of the slave environment. \mathbf{Z}_h and \mathbf{Z}_e represents the dynamic of the operator hand and the environment respectively, which relates

the force the operator must apply in order to move the arm and the master device. \mathbf{Z}_{th} represents the total impedance the operator feels when using the system. Which means both the impedance of the environment and of the telemanipulated system. In the same way, \mathbf{Z}_{te} is the total impedance of both the human operator and the system, seen from the environment. The dynamics of the master and slave robots are represented by the impedances \mathbf{Z}_m and \mathbf{Z}_s , which is within the master-slave two port network box in Figure 2.2.

$$\mathbf{f}_s = \mathbf{f}_e + \mathbf{Z}_e \mathbf{v}_s \quad (2.1)$$

and the relation of the forces and position on the master side, where \mathbf{Z}_h is the impedance of the operator, can be expressed

$$\mathbf{f}_m = \mathbf{f}_h - \mathbf{Z}_h \mathbf{v}_m \quad (2.2)$$

2.2.1 Human and Environment Models

Neither the human operator or the environment are parts of a teleoperated system. However, they are both mechanically connected to the system, and thus influences the performance and characteristics of the connected teleoperated system. Both the environment and the operator are unpredictable in behaviour and are difficult to model, but in general they both affect the stability of the entire system, so it is beneficial if they are included in stability analysis. It is common to model the operator and the environment with a impedance model, $Z = ks + c$, where the impedance is characterised by a mass-spring-damper with stiffness k , and damping c , see Figure 2.3. The human operator plays a significant role for the stability of the system, so usually when being modelled the damping characteristics are intensified. The environment varies a lot more, from free movement to hard contact, and therefore it is difficult to make a linear model complying for all cases. It is useful to look at extreme values, maximum and minimum values, and then test the stability in these cases.

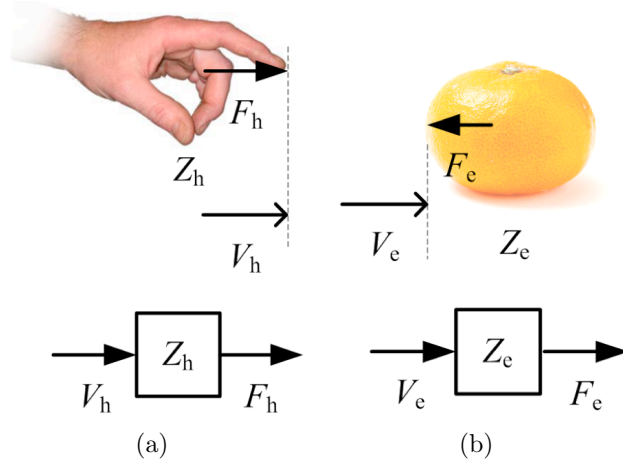


Figure 2.3: Model of (a) the operator and (b) the environment (Christiansson, 2007)

2.2.2 Master-Slave System

The characteristics of the two-port system can be analysed by using six matrices which relates the forces and positions on the master and slave side (Aliaga et al., 2004):

$$\begin{bmatrix} \mathbf{f}_m \\ \mathbf{f}_s \end{bmatrix} = [\mathbf{Z}] \begin{bmatrix} \mathbf{v}_m \\ \mathbf{v}_s \end{bmatrix} \quad \text{Impedance matrix} \quad (2.3)$$

$$\begin{bmatrix} \mathbf{v}_m \\ \mathbf{v}_s \end{bmatrix} = [\mathbf{Y}] \begin{bmatrix} \mathbf{f}_m \\ \mathbf{f}_s \end{bmatrix} \quad \text{Admittance matrix} \quad (2.4)$$

$$\begin{bmatrix} \mathbf{f}_m \\ -\mathbf{v}_s \end{bmatrix} = [\mathbf{H}] \begin{bmatrix} \mathbf{v}_m \\ \mathbf{f}_s \end{bmatrix} \quad \text{Hybrid parameter matrix} \quad (2.5)$$

$$\begin{bmatrix} \mathbf{v}_m \\ \mathbf{f}_s \end{bmatrix} = [\mathbf{G}] \begin{bmatrix} \mathbf{f}_m \\ -\mathbf{v}_s \end{bmatrix} \quad \text{Inverse of } \mathbf{H} \text{ matrix} \quad (2.6)$$

$$\begin{bmatrix} \mathbf{f}_m \\ \mathbf{v}_m \end{bmatrix} = [\mathbf{F}] \begin{bmatrix} \mathbf{v}_s \\ \mathbf{f}_s \end{bmatrix} \quad \text{Transmission matrix} \quad (2.7)$$

$$\begin{bmatrix} \mathbf{v}_s \\ \mathbf{f}_s \end{bmatrix} = [\mathbf{F}^*] \begin{bmatrix} \mathbf{f}_m \\ \mathbf{v}_m \end{bmatrix} \quad \text{Inverse of } \mathbf{F} \text{ matrix} \quad (2.8)$$

Any of the matrices listed above contains the characteristics of the system. If one of the matrices are known, all of the others can be obtained.

The most popular matrix to evaluate is the hybrid matrix, (2.6), whose entries are

$$\begin{bmatrix} \mathbf{f}_m \\ -\mathbf{v}_s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \mathbf{v}_m \\ \mathbf{f}_s \end{bmatrix} \quad (2.9)$$

$$h_{11} = \left. \frac{\mathbf{f}_m}{\mathbf{v}_m} \right|_{\mathbf{f}_s=0} \quad \text{Unconstrained impedance} \quad (2.10)$$

$$h_{12} = \left. \frac{\mathbf{f}_m}{\mathbf{f}_s} \right|_{\mathbf{v}_m=0} \quad \text{Force tracking} \quad (2.11)$$

$$h_{21} = \left. \frac{\mathbf{v}_s}{\mathbf{v}_m} \right|_{\mathbf{f}_s=0} \quad \text{Position tracking} \quad (2.12)$$

$$h_{22} = \left. \frac{\mathbf{v}_s}{\mathbf{f}_s} \right|_{\mathbf{v}_m=0} \quad \text{Contact admittance} \quad (2.13)$$

Parameter h_{11} can be interpret as the **unconstrained impedance** of the master, which means the damping and inertia the operator feels when moving the master robot and the slave robot is unconstrained, desired as low as possible. h_{21} is the **velocity tracking** during unconstrained motion and measures the ability of the slave robot to copy the motion of the master robot, if a integrator part is included it also applies for position tracking. This should tend to one (if no motion scaling is desired) with infinite bandwidth. For contact tasks h_{12} relates to the **tracking of forces**. For tracking of positions during contact tasks, h_{22} is defined as the **contact admittance**. The minus sign for \mathbf{v}_s origins in the electric circuit analogy where the positive currents are going into the network.

2.3 Stability Analysis

2.3.1 Analysis of the whole system

If there exist a model of the dynamics of the operator and the environment some known methods may be used to provide stability. Both the Nyquist and the Lyapunov stability concepts gives ability to tune the system into stability by manipulating various parameters of the system. The advantage is that the conditions are not conservative and the stability can be established by placing the poles of the system. One of the disadvantages is that the operator and environment must be modelled, and the results depend on the accuracy of these models.

Another way to analyse the stability of a system is by using the Root Locus method (Franklin, 1993). It shows how the pole placements changes when altering the parameters and the characteristics of the system. The results are presented in a root locus graph. As long as the poles are placed in the left half plane the system is stable. In this way the stability can be analysed systematically and the effect of the various parameters on the stability can be discovered. On the other hand, only one parameter can be analysed at once which may give difficulties analysing the overall system.

2.3.2 Master-Slave System

If the models of the operator and the environment are not known, assumptions must be taken to simplify the total model which results in larger uncertainties so the tools to analyse the system must be more conservative to be true for all cases. Some of the advantages by using these tools is that there are no need for a model of the operator and the environment, but on the contrary, the disadvantages are that the stability conditions are conservative.

Absolute Stability

One way to ensure stability is by using *Llewellyn's criterion for absolute stability* (Haykin1970). For any passive (does not add any energy to the

system) operator or environment the necessary and sufficient conditions for absolute stability in means of the hybrid elements and its real parts ($r_{ij} = \text{real}(h_{ij})$) are:

- $h_{11}(s)$ and $h_{22}(s)$ have no poles in right half plane
- Any poles of $h_{11}(s)$ and $h_{22}(s)$ on the imaginary axis are simple with real positive residues
- for all real values of ω

$$\begin{aligned} r_{11}(j\omega) &\geq 0 \\ r_{22}(j\omega) &\geq 0 \\ 2r_{11}r_{22} - r_{12}r_{21} - |h_{12}h_{21}| &\geq 0 \end{aligned}$$

If some of the conditions is not satisfied the system is potentially unstable.

Passivity and the Scattering Operator

A way to provide stability is to make the system passive. Niemeyer and Slotine (2004) defines passivity as

$$\int_0^t e_{input} d\tau = \int_0^t \mathbf{v} \mathbf{f} d\tau \geq -e_{store}(0) \quad \forall t \geq 0 \quad (2.14)$$

Where e_{input} is the energy flowing into the system and e_{store} is the energy initially stored in the system. This means that the system is passive if the energy going out of the system is limited to the energy going into the system plus the stored energy.

Haykin provided some necessary and sufficient conditions for passivity in terms of the hybrid matrix elements h_{ij} , referred to as the *Raisbeck's passivity criterion* (Christiansson, 2007):

- None of the h_{ij} has poles in the right half plane

- Any poles on the imaginary axis are single, and the residues g_{11} , g_{12} , g_{21} and g_{22} of the h_{ij} elements satisfies the following conditions:

$$\begin{aligned} g_{11} &\geq 0 \\ g_{22} &\geq 0 \\ g_{11}g_{22} - g_{12}g_{21} &\geq 0 \quad \text{with } g_{12} = g_{21}^* \end{aligned}$$

- The real (r_{ij}) and imaginary (i_{ij}) parts of the hybrid elements satisfies for all ω :

$$\begin{aligned} r_{11} &\geq 0 \\ r_{22} &\geq 0 \\ 4r_{11}r_{22} - (r_{12} + r_{21})^2 - (i_{12} - i_{21})^2 &\geq 0 \end{aligned}$$

As we can see the *Absolute Stability* and *Passivity* criterions are very alike, and the essential difference lies in the last of conditions three.

Another way to analyse the passivity of the system is to use the scattering operator. In terms of the hybrid matrix the scattering operator is defined (Anderson and Spong, 1988)

$$\mathbf{S}(s) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} (\mathbf{H}(s) - \mathbf{I})(\mathbf{H}(s) + \mathbf{I})^{-1} \quad (2.15)$$

To ensure passivity of the system the scattering operator must be $\|\mathbf{S}(j\omega)\| \leq 1$ for all ω .

2.4 Performance

The performance of the teleoperator is a matter of transparency and stability. Transparency represents the 'degree of invisibility' of the system, if the system is perfect transparent the operator sense the same as he/she would do directly without the teleoperator. The paradox in the context of teleoperation is that high transparency leads to marginally stable systems and high stability leads to poor transparency, so the

performance of the system is a compromise between stability and transparency and the performance is thus limited by the stability (Lawrence, 1993). First the system needs to be stable and then make it as transparent as possible.

2.4.1 Perfect Transparency

As mentioned above good transparency is one the most important tasks when developing teleoperator systems. And a definition of perfect transparency is that the human operator feels the same forces and velocities at the master device as if manipulating the environment directly (Yokokohji and Yoshikawa, 1994), which implies

$$\mathbf{f}_m = \mathbf{f}_s \quad (2.16)$$

$$\mathbf{v}_m = \mathbf{v}_s \quad (2.17)$$

And the resulting hybrid matrix will be

$$\mathbf{H}(s) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (2.18)$$

Another way to analyse the transparency is to look at the transmitted impedance. When using a teleoperated system the operator feels not only the impedance of the environment but also the dynamics of the system. The transmitted impedance from the remote side to the human operator side includes both the environment and system impedance, and is derived from Equation (2.1) and (2.9)

$$\mathbf{Z}_{th} \equiv \left. \frac{\mathbf{f}_m}{\mathbf{v}_m} \right|_{\mathbf{f}_e=0} = \frac{h_{11} + (h_{11}h_{22} - h_{12}h_{21})\mathbf{Z}_e}{1 + h_{22}\mathbf{Z}_e} \quad (2.19)$$

The transmitted impedance from the master side to the remote environment can in the same way be written

$$\mathbf{Z}_{te} \equiv \left. \frac{\mathbf{f}_s}{-\mathbf{v}_s} \right|_{\mathbf{f}_h=0} = \frac{h_{11} + \mathbf{Z}_h}{(h_{11}h_{22} - h_{12}h_{21}) + h_{22}\mathbf{Z}_h} \quad (2.20)$$

Though the hybrid matrix may become very complicated if significant dynamics is present some general principles can be derived from Equation (2.20)

- Perfect transparency ($\mathbf{Z}_{th} \equiv \mathbf{Z}_e$ and $\mathbf{Z}_{te} \equiv \mathbf{Z}_h$) requires that $h_{22} = h_{11} = 0$ and $h_{12}h_{21} = -1$
- For any nonzero contact admittance, hard surfaces will be felt as much softer and the slave's position will be sensitive to external forces

2.5 Control Schemes

Several control schemes are developed to deal with the challenges named above to control master and slave in a teleoperation system. All of them have various characteristics and gives good results for some aspects of the teleoperated system, but it is difficult to achieve all desired properties. To decide what control scheme to use in a specific application the following questions would be appropriate to ask (Lawrence, 1993):

- What degree of transparency is necessary to accomplish a given set of teleoperation tasks?
- What degree of transparency is possible?
- What are suitable teleoperator architectures and control laws for achieving necessary or optimal transparency?

Some of the most common schemes will be presented here.

2.5.1 Position Position Control

This is a very simple scheme where the only information exchanged between master and slave is the position, and as this indicates the forces are estimated on the bases of the position errors. Some advantages of using this scheme are the simplicity, no need for force sensors and fast response in means of calculation time needed. But this is on the expense of accuracy and less realistic force sensation.

The position of the master (\mathbf{x}_m) is used as the reference trajectory for the slave, and the position of the slave (\mathbf{x}_s) is used to generate the forces in which the master is reflecting to the operator. This symmetric

scheme use PD controllers both at the master and slave side which acts like a spring with stiffness \mathbf{k} and a damper with constant \mathbf{c} ($PD = \mathbf{k} + s\mathbf{c}$). PD_m control the force to apply through the haptic based on the deviation between \mathbf{x}_m and \mathbf{x}_s , and PD_s attempt to follow the trajectory given by \mathbf{x}_m . Figure 2.4a) shows the setup of the position-position control scheme, here the velocities are used so the positions are the integrated velocities.

2.5.2 Force Position Control

This is a more intuitive control scheme thus the reflected forces are measured real forces. The slave diagram corresponds to the previous scheme in which the PD_s controller tries to follow the trajectory given from the master, see Figure 2.4b). The difference from the position position scheme is that the information reflected from slave to master is real forces measured in the contact between slave and environment, and at the master side this contact forces are scaled by a constant \mathbf{k} . This requires the installation of a force sensor at the slave tool tip.

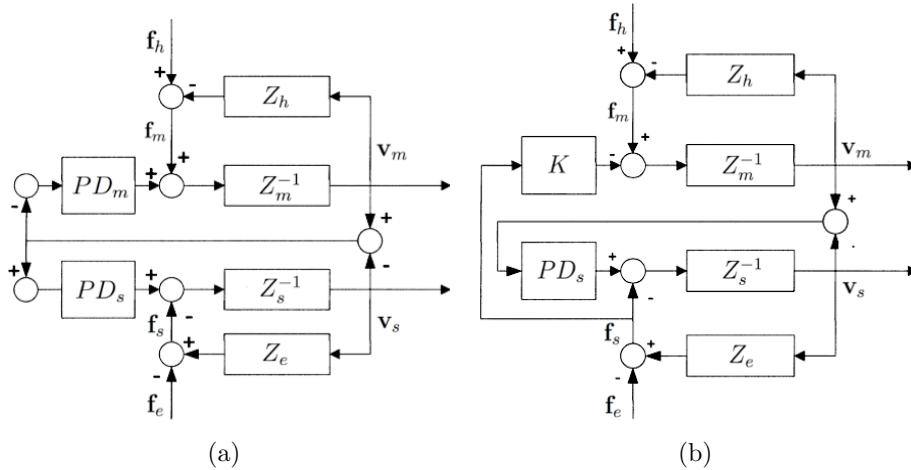


Figure 2.4: Block diagram of a) a position position and b) a force position controller (Aliaga et al., 2004)

The advantage of this algorithm compared to the position position

scheme is that the force feedback is a result of real measurements, so the force felt at master side is more realistic. On the other hand, by measuring the forces significant noise is introduced, but may be taken care of by a filter.

2.5.3 4 Channel Control

This scheme utilises both forces and positions at master and slave side. In other words, it uses all the information available in a telemanipulated system. Thus it requires position and force sensor to be installed both at master and slave side. The name denotes the four communication channels exchanging the position and force information in both directions.

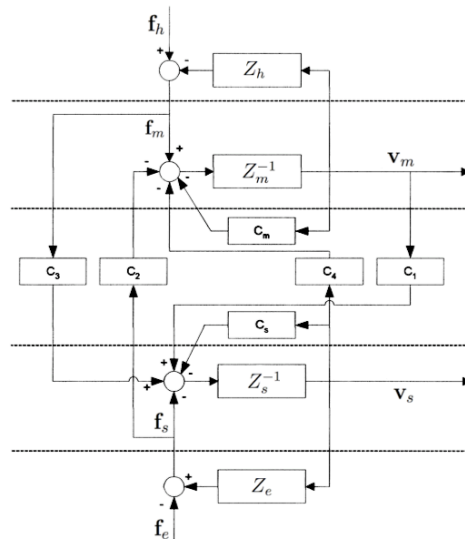


Figure 2.5: Block diagram of a 4 Channel Controller (Aliaga et al., 2004)

Lawrence (1993) presents a recipe on how to tune the parameters of the hybrid matrix, derived from the 4 channel scheme, to obtain optimal performance.

2.5.4 Comparing the Control Schemes

The three control schemes presented in this report are the most commonly used within teleoperated systems, but they have very different characteristics, introducing advantages and drawbacks at different levels.

The position position (PP) control scheme is one of the first control schemes documented for use in teleoperated systems. It is very simple both in structure and functionality, and requires no force sensor to produce force feedback. But it is difficult to tune the scheme to give realistic force feedback both at no contact movement and hard contact movement.

The force position (FP) control scheme is equally simple as the PP scheme, but the force feedback is based on real force measurement at the slave side. This gives a more realistic force feedback, easier to tune for various contact levels. But the large number of components in the inner control loop implies a important phase loss, which is one of the reasons for the stability problems of this scheme (Christiansson, 2007).

The 4 channel control scheme introduces the use of both position and force to control each manipulator. Lawrence (1993) shows that this scheme improves both stability and performance of a telemanipulated system, due to the increased information available at both sides. He also argues how the scheme can compensate for time delays. A drawback is the need for force measurements at both master and slave side.

2.6 Wave Variables

When significant time delays occur in the teleoperated system wave variables can be used to ensure stability (Niemeyer and Slotine, 2004). This is a way of encoding the information to be sent in the communication channel of a teleoperated system in means of power and energy. The wave variables are applicable to nonlinear systems and can handle large uncertainties which makes it suitable for interaction with real physical environments. This is a technique that can be used together with all the control schemes presented in this report.

At each side of the communication channel the signals are transformed into a pair of wave variables ($\mathbf{w}_f, \mathbf{w}_b$).

$$\mathbf{w}_f = \frac{b\mathbf{v} + \mathbf{f}}{\sqrt{2b}} \quad \mathbf{w}_b = \frac{b\mathbf{v} - \mathbf{f}}{\sqrt{2b}} \quad (2.21)$$

where \mathbf{w}_f denote the forward or right moving wave, while \mathbf{w}_b denotes the backward or left moving wave. The force \mathbf{f} and velocity \mathbf{v} variables may be replaced by any other effort or flow pair. To decode the variables at the other side some manipulation of Equation (2.21) gives us the result on the original form.

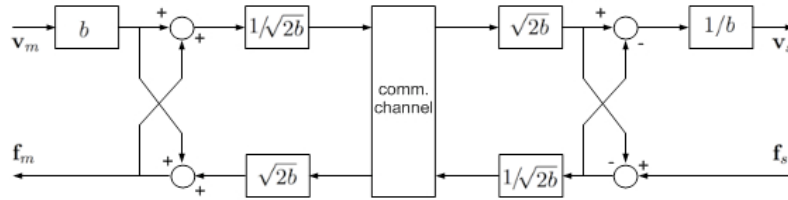


Figure 2.6: Block diagram of a Wave Variable Communication (Niemeyer and Slotine, 2004)

Some of the motivation for using wave variables is the guarantee of stability, which is also what makes this technique very conservative. As the performance of the system is a compromise between transparency and stability, the transparency when using wave variables are not the best. The stability is preserved by the passivity of wave variables, which is ensured by the condition that more energy has to be put into a system than that may come out of the system.

2.7 Practical Issues

To be able to model and analyse a teleoperated manipulator some assumptions needs to be taken. All the models presented in this report are linear and time invariant thus this is most common in the literature and the linearized models often gives good enough results.

All signals are modelled in the Laplace domain and thus the position and velocity contains the same information only a factor difference. The impedance is usually defined as the force/velocity relationship but as most velocity decoders actually measures position and the models are linear so the difference between position and velocity is irrelevant the impedance in this report is defined as force/position.

All models are for one degree of freedom manipulators, but are implemented on a system with three degrees of freedom under the assumption that the three axes are independent of each other.

With the analytical tools presented here it is possible to analyse theoretical how systems will behave and how to tune them, but it is practically difficult to apply these to the real systems, and it is difficult to determine the \mathbf{H} -matrix for the real system with 3 degrees of freedom.

2.7.1 Obtaining the Hybrid Matrix From a Real System

Though it is practically difficult, Aliaga et al. (2004) has shown how the hybrid elements can be retrieved from a real system with two degrees of freedom, by performing some tests on each degree of freedom separately.

Unconstrained Movement Test

Unconstrained motion means the free motion of the slave robot, mathematically speaking $\mathbf{f}_s = 0$, which reduces the hybrid matrix parameters of interest to h_{11} and h_{21} .

The test is carried out by the operator moving the master robot while the slave robot tries to follow the movements best as possible. In this way the \mathbf{v}_m , \mathbf{v}_s and \mathbf{f}_m can easily be acquired. By analysing the data in the frequency domain the transfer function of h_{11} and h_{21} can be obtained, in other words the free motion impedance and the position tracking of the slave. We see that these two parameters can be evaluated both mathematically and experimentally without any additional preparation of the system.

Hard Contact Test

To analyse the to other parameters experimentally we need to add force sensors to the system in order to measure the force at the master and slave side. As we see from Equation (2.13) these parameters can be analysed by setting $\mathbf{v}_m = 0$. This means that the master robot must be fixed while a force is exerted onto the slave robot. Physically this is difficult to carry out and not very intuitive, a better condition would be if the slave were fixed, $\mathbf{v}_s = 0$, because the significance would be clearer and the experiment easier to perform.

Chapter 3

Haptic Guidance

Teleoperated systems with haptic feedback allows the operator to feel the forces acting between the slave manipulator and the environment. Introducing virtual forces based on virtual constraints, can add a guidance functionality to the system. By using constraints based on a pre-operative plan, the virtual forces can guide the operator away from critical areas avoiding to damage vulnerable organs and tissue, or it can guide the operator along a path or trajectory which gives an optimal result (Cornella et al., 2008b). Several tests have proven haptic guidance to improve performance of teleoperated systems increasing speed and precision, reducing operator workload and reducing effects of time delays (Rosenberg, 1993; Sayers and Paul, 1994).

The distinction between haptic feedback and guidance is that the generated force of haptic feedback is a result of real contact forces while the forces generated of haptic guidance is due to the virtual touch of some predefined constraints. Still the characteristics and properties of haptic feedback systems applies, but the values are generated on the basis of the virtual contact.

3.1 The Guidance Loop

In the case of the position position control scheme, a guidance system can be incorporated to the teleoperated system like in Figure 3.1. We

see that the reference error to the guidance controller is equal to the error between the desired trajectory \mathbf{x}_d and the slave position \mathbf{x}_s . We call this slave based guidance. We have integrated the output from the manipulators to positions, which will be used when implementing the system. The force generated of the guidance system is added to

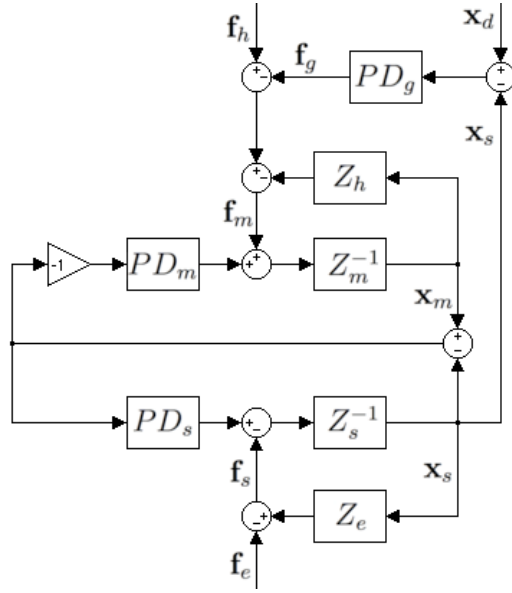


Figure 3.1: Implemented PP control scheme with haptic guidance

the total amount of force which are exerted through the master device, actually the guidance system runs as an external loop on the top of the control system for the telemanipulated system. This means that if we find the closed loop transfer function for the guidance system, we can analyse the entire system. With \mathbf{x}_d as the input and \mathbf{x}_s as output of the guidance system we find the following transfer function

$$h_g = \frac{\mathbf{x}_s}{\mathbf{x}_d} = -1 - \frac{Z_m}{h_{21}} PD_g - \frac{PD_m Z_s + 2PD_m PD_s}{PD_s PD_g} \quad (3.1)$$

All the characteristics of the four hybrid parameters are represented in this transfer function, but we see that one of them are recognised as a

part of this equation, and that is the position tracking transfer function h_{21} . From chapter 5.1 we know that the ideal position tracking, $h_{21} = -1$, giving $PD_s = -0.5Z_s$. Putting this information into equation (3.1)

$$h_g = \frac{Zm}{PD_g} - 1 \quad (3.2)$$

we see that the guidance transfer function only depends on the impedance of the master and the value of the guidance controller. The part of the slave control does not affect the guidance, which is clear since the definition of perfect position tracking say that $\mathbf{x}_s = -\mathbf{x}_m$ and thus the guidance loop becomes a closed loop around the master control loop, we call this master based guidance. This can be implemented by using the master position \mathbf{x}_m as basis for the reference error, instead of using the slave position \mathbf{x}_s . Reasons for the position tracking to deviate from the ideal value are the dynamics of the system and time delay in communication between the various parts of the system.

3.2 Guidance Types

The haptic guidance can be divided into three categories depending on how the constraints are formulated. Considering the position of a point in a 3D space, if the constraints are given for all three degrees of freedom (DoF), the guidance is related to a point. With two DoF constraints a line is formed as basis for the guidance. The last case is where the constraints only is given for one DoF, which forms a surface. In addition to the geometrical considerations the haptic forces used to render the guidance can be either attractive or repulsive. If the force is attractive the operator is driven towards the constraints so the zero point is when the haptic is situated at the constraint, this is the case of a guidance along a path. In this situation there will not be generated any forces. In the other case where the forces are repulsive the operator is driven away from the constraints. Preventing the haptic to enter a forbidden area is an example of this situation. Figure 3.2 summarises the different types of constraints. The forces are increasing when approaching the forbidden area, and decreasing when being far away from the area (Nuño and Basañez, 2006).

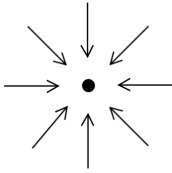
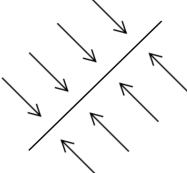
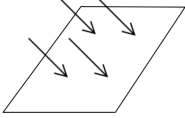
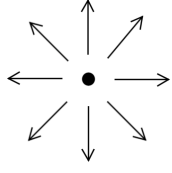
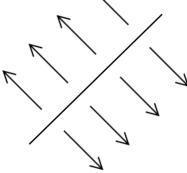
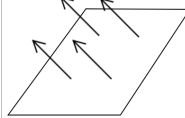
	3 Motion Constraints	2 Motion Constraints	1 Motion Constraint
Attractive forces (guidance)			
Repulsive forces (no-go zones)			

Figure 3.2: Haptic guidance according to number and sense of the constraints.

3.3 Algorithms

As stated above the guidance can be divided into three different categories according to the number and sense of constraints. When the guidance type is chosen the next will be to find the constraints. In the case of line guidance, the operator follows a line and when deviating from this line an action needs to be taken. The action will be in form of forces acting through the haptic device to guide the operator back to the line. The sense of guidance are decided on the bases on the deviation from the line and some control law.

3.3.1 3 Constraints - Point

The simplest case for computing the deviation is when there is a point constraint, i.e, the constraints are given in the 3 degrees of freedom. In Cartesian space the deviation can be calculated by using vectors. The advantage is that both the deviation and the direction is found at once and can then be used directly to generate three dimensional forces.

Let $\mathbf{x}_i = [x_i, y_i, z_i]^T$ represent the spatial initial point and $\mathbf{x}_s = [x_s, y_s, z_s]^T$ denote the present spatial position of the slave, then the deviation and direction between the two points be described as

$$\mathbf{v}_d = \mathbf{x}_i - \mathbf{x}_s \quad (3.3)$$

where $\mathbf{v}_d = [v_{dx}, v_{dy}, v_{dz}]^T$. The direction of \mathbf{v}_d is pointing towards \mathbf{x}_i because the guidance forces are to guide the operator towards the initial point, and thus ease the further calculation of forces. In the case of no-go zones the forces are to be repulsive, then the sign of \mathbf{v}_d must be switched.

3.3.2 2 Constraints - Line

When the constraint is a straight line, see Figure 3.3a, the constraints are given in 2 degrees of freedom, and the calculation of the deviation needs some more step. The following developments are necessary:

A straight line can be modelled as

$$\mathbf{x}_d = \mathbf{x}_i + \lambda \mathbf{v} \quad (3.4)$$

where $\mathbf{x}_d = [x_d, y_d, z_d]^T$ and $\mathbf{v} = [x_v, y_v, z_v]^T$. \mathbf{x}_i represent the coordinate of the initial point of the line, \mathbf{x}_d is the coordinate for the desired point, \mathbf{v} is a unit vector that gives the spatial direction of the line and λ is the length between the points \mathbf{x}_i and \mathbf{x}_d on the line.

The important parameters here is \mathbf{x}_i , that gives the initial position of the line, and the vector \mathbf{v} that gives the direction of the line. By basic geometric considerations one can use these two parameters to find the distance from an arbitrary point to the given line. If the position of the tool tip on the slave arm is \mathbf{x}_s , a vector can be found between \mathbf{x}_i and \mathbf{x}_s .

$$\mathbf{v}_s = \mathbf{x}_s - \mathbf{x}_i \quad (3.5)$$

When vectors \mathbf{v} and \mathbf{v}_s are known the length of the vectors and the angle between them can be used to find the shortest distance \mathbf{v}_d from the slave tool tip to the line, which is parallel to the normal of the line.

$$\mathbf{v} \cdot \mathbf{v}_s = |\mathbf{v}| |\mathbf{v}_s| \cos \alpha \quad (3.6)$$

$$|\mathbf{v}_l| = |\mathbf{v}_s| \cos \alpha \quad (3.7)$$

where \mathbf{v}_l is a line parallel to \mathbf{v} , and substituting these equations gives the magnitude

$$|\mathbf{v}_l| = \frac{\mathbf{v} \cdot \mathbf{v}_s}{|\mathbf{v}|} \quad (3.8)$$

Since $\lambda = |\mathbf{v}_l|$ we know the point \mathbf{x}_d and can then calculate the deviation vector

$$\mathbf{v}_d = \mathbf{x}_d - \mathbf{x}_s \quad (3.9)$$

The vector \mathbf{v}_d represents the deviation of the tool tip from the given trajectory as in this case is a straight line. This vector gives the magnitude and the direction towards the trajectory.

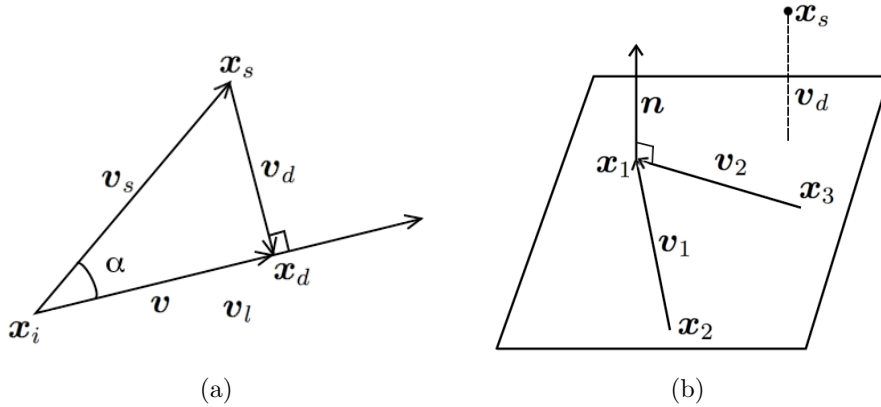


Figure 3.3: The deviation from a) a line and b) a plane

3.3.3 1 Constraint - Plane

When the constraint is a plane, i.e., the constraint is just in 1 degree of freedom, the deviation can also be computed directly from the equation of the plane. We have three points \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 which forms a plane. By creating two vectors \mathbf{v}_1 and \mathbf{v}_2 in the plane and taking cross product

of them we find the normal \mathbf{n} to the plane, see figure 3.3b:

$$\mathbf{v}_1 = \mathbf{x}_1 - \mathbf{x}_2 \quad (3.10)$$

$$\mathbf{v}_2 = \mathbf{x}_1 - \mathbf{x}_3 \quad (3.11)$$

$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2 \quad (3.12)$$

Then the standard plane equation can be found

$$(\mathbf{x} - \mathbf{v}_1)\mathbf{n} = Ax + By + Cz + D = 0 \quad (3.13)$$

where

$$A = n_1$$

$$B = n_2$$

$$C = n_3$$

$$D = -(n_1v_{11} + n_2v_{12} + n_3v_{13})$$

If the position of the slave is \mathbf{x}_s we can then find the deviation from this point to the plane

$$d = \frac{Ax_{x_s} + By_{x_s} + Cz_{x_s} + D}{\sqrt{A^2 + B^2 + C^2}} \quad (3.14)$$

and finally the deviation vector \mathbf{v}_d can be found by multiplying the normalised \mathbf{n} with the deviation d

$$\mathbf{v}_d = \|\mathbf{n}\| d \quad (3.15)$$

3.3.4 Force Generation

When the deviation is found, one needs to select a strategy for decreasing this deviation, some forces has to be exerted through the haptic device. The most common control law to use is a spring-damper model. It is simple and intuitive but yet an effective way of controlling deviations of oscillating systems. These equations are one dimensional.

$$m\ddot{x} + \underbrace{c\dot{x}}_{\text{damper}} + \underbrace{kx}_{\text{spring}} = 0 \quad (3.16)$$

Equation (3.16) is the classic expression of the mass-spring-damper model, where c is the damper gain and k is the spring gain. By rearranging the equation we find two very important parameters ω_0 and ζ :

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2x = 0 \quad (3.17)$$

where

$$\omega_0 = \sqrt{\frac{k}{m}} \quad (3.18)$$

$$\zeta = \frac{c}{2\sqrt{km}} \quad (3.19)$$

ω_0 is called the undamped natural frequency. This is the frequency which the system will oscillate if no damping occurs. The damping ratio, ζ , characterises the frequency response of a ordinary second order differential equation. Equation 3.16 can be transformed into the frequency domain and is then written

$$h(s) = \frac{1/k}{1 + 2\zeta\frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2} \quad (3.20)$$

By assuming a solution $x = e^{\gamma t}$ equation (3.17) can be solved by substituting this solution of x back into the equation.

$$\gamma^2 + 2\zeta\omega_0\gamma + \omega_0^2 = 0 \quad (3.21)$$

solving for γ

$$\gamma = \omega_0(-\zeta \pm \sqrt{\zeta^2 - 1}) \quad (3.22)$$

The behaviour of the oscillating system depends on ω_0 and ζ and the frequency response can be split in three groups; critical-, over- and under-damping.

- $\zeta > 1$ Over damping
- $\zeta < 1$ Under damping
- $\zeta = 1$ Critical damping

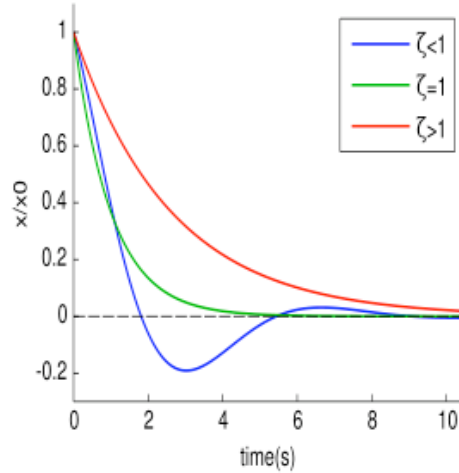


Figure 3.4: The frequency response for the different values of ζ

Over damping means that γ have two real solutions and the frequency response will not have any oscillations but closes in on the steady state directly. In the case of under damping γ will also have two solutions but they are both complex, and the result is some oscillation before closing in to the steady state. The final case is often the most desired case when controlling systems and it is the critical damping. The result of this is only one real solution of γ , and the response, as for over-damping, closes directly into the steady state. The reason why this is a more desirable situation is that the response closes into the steady state faster then if system is over-damped.

3.3.5 Max Distance - Max Force

To give the operator the ability to decide the softness/hardness of the guidance a maximum distance $v_{d_{max}}$ is provided. The meaning of this is that the maximum exertable force of the haptic device will be used when the slave robot deviates a distance equivalent to the maximum distance from the guidance trajectory. In other words the max distance is the same as the radii in a cylinder wrapping the guidance trajectory

(or radii in a sphere or distance to a plane), see figure 3.5.

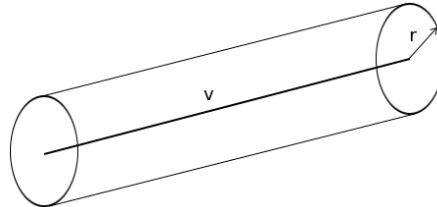


Figure 3.5: The max distance cylinder

3.4 Object Segmentation

One of the objectives with haptic guidance is to avoid forbidden regions, like the plane repulsive guidance. But to give a real functionality the guidance needs to relate to other structures than the plane. Take as an example an organ which is not desired to touch, then it would be useful to make a no-go-zone out of the geometry of the organ, so the walls of the organ represent the guidance constraints. In order to make this the organ needs to be identified as a spatial and volumetric object. Image or object segmentation is a way of partitioning a digital image into multiple parts to simplify or convert an image into a format that is easier to analyse. The result is a set of regions that covers the entire object, which can be one, two or three dimensional. The challenge is to analyse the large amount of data which is created in a effective way.

The Phantom Omni API provides a fully developed feature for handling various types of shapes. It has some ready-to-run objects which can be used to manipulate virtually by haptic interaction, or by using functions provided one can input new objects. This functionality is optimised for virtual interaction purposes and thus a easy way to include real objects in the haptic system (SensAble Technologies, 2005).

At The Interventional Center a system for tracking collisions between digital segmented objects is developed (Morvan et al., 2008), and by introducing the virtual object to avoid and the surgical tool, forces can be generated on the basis of the distance v_d between these.

The use of real objects as forbidden regions will not be discussed any further in this report, but is a topic for future enhancements.

Chapter 4

Kinematics and Dynamics of the Manipulators

The study and implementation of teleoperated systems requires the impedance model of the different parts of the system to be known. When the kinematics and dynamics of slave and master robot differs from each other a common reference frame needs to be found, in this case the Cartesian space is chosen. In this chapter the steps for acquiring the details of the teleoperated system is presented. The described guidance system is analysed and implemented on the available equipment at The Interventional Center. As a master device the Phantom Omni haptic device is utilised, and the AESOP 3000DS works as the slave. The communication and control loops are implemented on a regular PC.

Some of the work in this chapter have been to derive the kinematic equations for the Phantom Omni, and to derive the basis for the space transformations for the two manipulators needed in the teleoperated system. A significant amount of time are spent on simulating the dynamics of the system, in order to do theoretical analysis. But it appeared to be difficult due to complexity of the models.

4.1 Slave Robot

The Interventional Centre has the access to an AESOP 3000DS (Automated Endoscope System for Optimal Positioning) robot from Computer Motion Inc. Originally this was a robot with 7 DOF intended to hold the camera in minimally invasive surgery, and it is also a part of the Zeus system where the operation solely is done by the robot system, directed by a surgeon. The robot can be controlled by foot pedals, a hand controller or voice commands. The robot is controlled by an internal control system, and with a RS232 serial interface it is possible to communicate with the control system, reading and sending both joint and end-effector positions/velocities. The communication introduces a time delay of 25 ms for sending and receiving information, a total of 50 ms for a communication cycle. In other words the maximum update frequency of the control loop is 20Hz. This will introduce a significant time delay of external control loops which will probably be the largest challenge using this robot system.

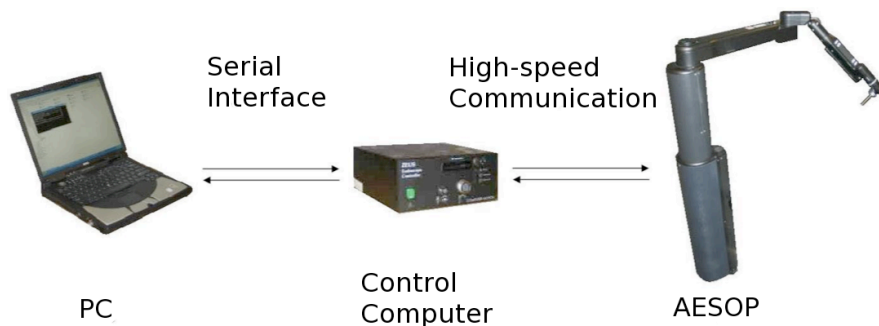


Figure 4.1: The Lab setup of AESOP, its control system and the external PC

To communicate with the AESOP system, some commands are provided by Computer Motion. The commands gives ability to send control, status, trajectory, move and screen relative commands. But the most important commands which are used in this report are

- *MoveAllJointsContinuous()*: directs the robot joints to move at

specific velocities, respectively according to a joint velocity given in an array.

- $JtPosReq()$: returns the angle of the joints.

In the control schemes presented in chapter 2.5 a force control scheme is suggested for both master and slave robot, but to control the AESOP velocities will be used.

4.1.1 Kinematics

This is a robot with 7 DOF with 4 active joints, 2 passive joints and 1 that is fixed manually. The 4 active joints gives 3 translational DOF and 1 rotational DOF. The reasons for this configuration are safety

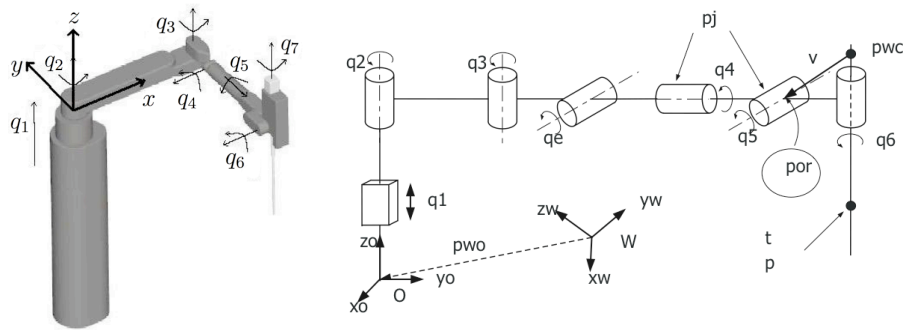


Figure 4.2: The kinematics of the AESOP

issues. The point where the endoscopic tool enters the body, called the trocar point, constraints the movements of the robot. What is very important is to not exert any force onto this point. The simplest and safest way to achieve this is to use two passive joints whose angle depends on the trocar point position, see figure 4.2. As a result of these aspects there are only 4 DoF available inside the body (Cornella et al., 2008a).

Figure 4.2 show the kinematics of the AESOP robot. Joint 1 is a translational joint, while joints 2 to 6 are rotational. Joint 4 and 5 are the passive joints, and q_e is the manually fixed joint. Link 2 (L_2) is

the link between joint 2 and joint 3, L_{31} is the link between joint 3 and q_e and L_{32} is the link between q_e and joint 5. The ranges of the joints are described in the AESOP user's guide Computer Motion Inc. (1999) and the lengths of each link were determined experimentally as:

$$\begin{aligned} L_2 &= 384.4879 \text{ mm} \\ L_{31} &= 76.7056 \text{ mm} \\ L_{32} &= 304.8001 \text{ mm} \\ L_3 &= L_{31} + L_{32} \cos q_e \text{ mm} \end{aligned}$$

q_e is the angle of the fixed joint of link 3. To find the Cartesian coordinates of the tip of the robot arm *forward kinematics* is used, and the formulas are as follows:

$$x_s = L_2 \cos q_2 + L_3 \cos (q_2 + q_3) \quad (4.1)$$

$$y_s = L_2 \sin q_2 + L_3 \sin (q_2 + q_3) \quad (4.2)$$

$$z_s = L_{32} \sin q_e + q_1 \quad (4.3)$$

where q_1 , q_2 and q_3 is the angle of joint 1, 2 and 3 respectively. Note that the forward kinematics considering the tip of the robot does not depend on the position of the passive joints, since these joints only affect the position of the endoscopic tool inside the body. These joints can introduce errors in the kinematics chain and for this reason they are not considered at this moment. Methods to improve the accuracy of the robot considering these joints are developed at The Interventional Centre (Cornella et al., 2008a).

The velocity and acceleration in Cartesian and joint space can be related through the Jacobian matrix $\mathbf{J}_s(\mathbf{q}_s) \in R^{3 \times 3}$ which depends on the joints angles of the robot as follows.

$$\mathbf{v}_s = \mathbf{J}_s \dot{\mathbf{q}}_s \quad (4.4)$$

$$\dot{\mathbf{v}}_s = \dot{\mathbf{J}}_s \dot{\mathbf{q}}_s + \mathbf{J}_s \ddot{\mathbf{q}}_s \quad (4.5)$$

The Jacobian elements are

$$\mathbf{J}_s = \begin{bmatrix} \dot{j}_{s1,1} & \dot{j}_{s1,2} & \dot{j}_{s1,3} \\ \dot{j}_{s2,1} & \dot{j}_{s2,2} & \dot{j}_{s2,3} \\ \dot{j}_{s3,1} & \dot{j}_{s3,2} & \dot{j}_{s3,3} \end{bmatrix} \quad (4.6)$$

$$\begin{aligned}
j_{s_{11}} &= 0 \\
j_{s_{12}} &= -L_2 \sin(q_2) - L_3 \sin(q_2 + q_3) \\
j_{s_{13}} &= -L_3 \sin(q_2 + q_3) \\
j_{s_{21}} &= 0 \\
j_{s_{22}} &= L_3 \cos(q_2 + q_3) + L_2 \cos(q_2) \\
j_{s_{23}} &= L_3 \cos(q_2 + q_3) \\
j_{s_{31}} &= 1 \\
j_{s_{32}} &= 0 \\
j_{s_{33}} &= 0
\end{aligned}$$

4.1.2 Dynamics

The general form of the equation of motion for robots and manipulators are

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (4.7)$$

where $\mathbf{M} = \mathbf{M}^T \in R^{3 \times 3}$ is the inertia matrix, $\mathbf{V} \in R^{3 \times 3}$ is the Coriolis and centrifugal forces, and $\mathbf{N} \in R^3$ represent the gravity and other forces acting on the joints. $\mathbf{q} = [q_1, q_2, q_3]^T \in R^3$ is the vector of joints and $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T \in R^3$ is the vector of torques acting on the joints.

As mentioned above the AESOP system runs an internal control system which control the motor torques. It is not known what controllers that have been used here, and the only way of controlling the robot externally is by using the control commands provided by Computer Motion Inc. (1999). As a result the elements of the dynamic model matrices are not known, and the system must be seen as a black box system whose dynamics are unknown. Cornella (2007) has found the response of each joint independent of the others by using a velocity input and measuring velocity output to discover the transfer function which represents the joint dynamics, here in matrix form

$$\begin{bmatrix} \dot{q}_{1o} \\ \dot{q}_{2o} \\ \dot{q}_{3o} \end{bmatrix} = \begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{bmatrix} \begin{bmatrix} \dot{q}_{1i} \\ \dot{q}_{2i} \\ \dot{q}_{3i} \end{bmatrix} \quad (4.8)$$

The transfer functions can be written as first order transfer functions:

$$T_1(s) = \frac{0.98}{0.065s + 1} \quad (4.9)$$

$$T_2(s) = \frac{0.98}{0.1s + 1} \quad (4.10)$$

$$T_3(s) = \frac{0.98}{0.11s + 1} \quad (4.11)$$

where $T_1(s)$, $T_2(s)$ and $T_3(s)$ represent joint 1, 2 and 3, respectively.

The assumption that the movements of each joint does not influence the others, is valid only if the reduction factor between the position of the joint and the actuator is high. In this case, the dynamic of each joint can be expressed as the dynamic of the actuator and a small perturbation torque, representing the effect of the other joints. For an electrical actuator the dynamics can be written on the form

$$\tau = (Is + B)sq + \tau_p \quad (4.12)$$

where I and B are inertia and viscous friction of the actuator, respectively, q is the joint angle, τ and τ_p are the motor and perturbation torque, all in the Laplace domain with the Laplace operator s .

4.2 Master Device

As master device for the robot system a Phantom Omni haptic device from SensAble Technologies (2007) are used. It is a joystick that can give the position in six degrees of freedom (DoF), three translational and three rotational, additionally it has two buttons that can be used for various purposes. What distinguishes the haptic device from other control devices is that it has actuators in three joints giving the ability to generate virtual forces in three DoF. With Open haptics, which is an open source API interface for the haptic, distributed by SensAble, one can manipulate the haptic to create forces in three translational DoF which can be used to recreate the sense of touch. The haptic is also a very powerful tool in means of the possibility to generate virtual constraints and surfaces.

The force of the haptic is rendered with a servo loop, where the magnitude and direction of the force is calculated for every cycle. The algorithm for calculating the force and the frequency of the loop decides the quality of the feedback. In the OpenHaptic Toolkit, a programmer's guide provided by SensAble, it is recommended to use a update frequency of at least 500 Hz (SensAble Technologies, 2007). The maximum exertable force of the haptic is at nominal position about 3.3 N. Some important commands with which to manipulate the haptic device are

- *hdGet*: returns the value of the device parameters
- *hdSet*: set forces to act through the device

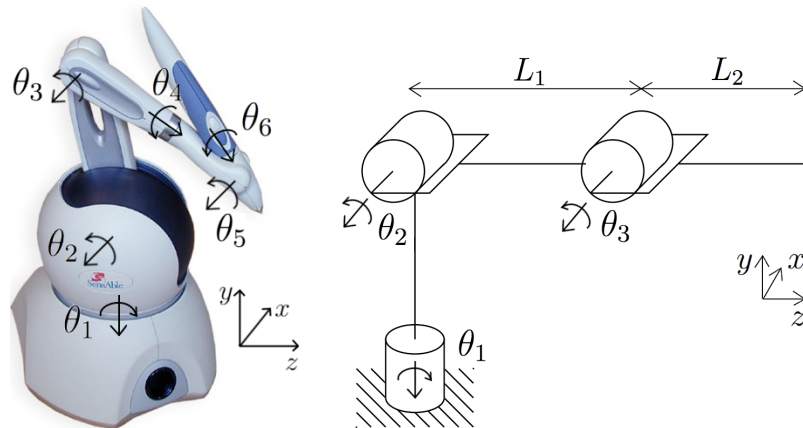


Figure 4.3: The kinematics of the Phantom Omni

4.2.1 Kinematics

The Phantom Omni is a 6 DOF manipulator. All joints are rotational but the joints $\theta_1 - \theta_3$ gives the translational movements for the end-effector while the joints $\theta_4 - \theta_6$ is mounted as a gimbal providing the rotational movements. As mentioned above the Phantom is only actuated in the first three joints, which gives the ability to generate haptic

forces in the translational directions, and thus only the translational positions/directions will be utilised in this report. As a matter of fact the wrist of the Omni has no function thus the Cartesian space coordinates is referred to the end of link two, and the joint angles is measured for the first three joints, so the Omni is actually a three link articulated robot, see Figure 4.3. After employing the Denavit Hartenberg conversion on the manipulator the transformation matrix is found and the kinematics can then be expressed as

$$x_m = -\sin \theta_1 (L_2 \sin \theta_3 + L_1 \cos \theta_2) \quad (4.13)$$

$$y_m = -L_2 \cos \theta_3 + L_1 \sin \theta_2 + k_y \quad (4.14)$$

$$z_m = L_2 \cos \theta_1 \sin \theta_3 + L_1 \cos \theta_1 \cos \theta_2 + k_z \quad (4.15)$$

where

$$L_1 = 133.35 \text{ mm}$$

$$L_2 = 133.35 \text{ mm}$$

$$k_y = 23.35 \text{ mm}$$

$$k_z = -168.35 \text{ mm}$$

The L_1 and L_2 are the lengths of link 1 and 2. Joint 1 and 2 are seen to be the same link and thus there are no link 0. The k_y and k_z are the workspace transformation offset between the origin of the haptic and the first joint.

To transform velocities, accelerations and forces between joint and Cartesian space the Jacobian matrix, $\mathbf{J}_m \in R^{3 \times 3}$, has to be known. It is derived from the forward kinematics equations (4.13) - (4.15) and can be expressed as

$$\mathbf{J}_m = \begin{bmatrix} \dot{j}_{m1,1} & \dot{j}_{m1,2} & \dot{j}_{m1,3} \\ \dot{j}_{m2,1} & \dot{j}_{m2,2} & \dot{j}_{m2,3} \\ \dot{j}_{m3,1} & \dot{j}_{m3,2} & \dot{j}_{m3,3} \end{bmatrix} \quad (4.16)$$

$$\begin{aligned}
\dot{j}_{m11} &= -\cos(\theta_1)(L2\sin(\theta_3) + \cos(\theta_2)L1) \\
\dot{j}_{m12} &= \sin(\theta_1)\sin(\theta_2)L1 \\
\dot{j}_{m13} &= -\sin(\theta_1)L2\cos(\theta_3) \\
\dot{j}_{m21} &= 0 \\
\dot{j}_{m22} &= \cos(\theta_2)L1 \\
\dot{j}_{m23} &= L2\sin(\theta_3) \\
\dot{j}_{m31} &= -L2\sin(\theta_1)\sin(\theta_3) - \sin(\theta_1)\cos(\theta_2)L1 \\
\dot{j}_{m32} &= -\cos(\theta_1)\sin(\theta_2)L1 \\
\dot{j}_{m33} &= L2\cos(\theta_1)\cos(\theta_3)
\end{aligned}$$

4.2.2 Dynamics

By restating equation (4.7) the dynamics can be written on the form

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{N}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \boldsymbol{\tau} \quad (4.17)$$

where $\mathbf{M} = \mathbf{M}^T \in R^{3 \times 3}$ is the inertia matrix, $\mathbf{V} \in R^{3 \times 3}$ is the Coriolis and centrifugal forces, and $\mathbf{N} \in R^3$ represent the gravity and other forces acting on the joints. $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^T \in R^3$ is the vector of joints and $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T \in R^3$ is the vector of torques acting on the joints. As equation (4.17) shows the dynamic equation is expressed in joint space relating the joint angles and its derivatives to the motor torques. See Chapter A.1 in the Appendix for details which have been worked out at the Interventional Centre.

The inertia matrix is found by examining the moment of inertia of each link of the manipulator according to the joint angle, length and mass. The \mathbf{V} matrix consist both of the Coriolis and centrifugal effects which appears for objects moving with a rotating frame of reference. The Coriolis depends both on mass and velocity but is independent of the position. The centrifugal forces depends only of the position and the mass and is always oriented away from the axis of rotation. Gravity forces are included in the \mathbf{N} matrix and depends on the joint angles, lengths and mass of each link. This form of the manipulator dynamics does not include the frictional forces, so when simulating the system it is important to include some frictional effect.

By knowing the physical reaction and data of the manipulator the elements of the matrices can be calculated and if these are known a complete model of the device can be used to design the best suited control algorithms. The values have been found experimentally.

4.3 Manipulator Dynamics in Cartesian Space

The two manipulators used in this setup are of different kinematics and dynamics, to compare them and use them in the same control algorithm a common framework needs to be set. The dynamic model of the Phantom Omni and the AESOP are both in joint space, but the kinematic configuration are dissimilar and therefore it is not possible to link the two manipulators in joint space. Thus, the Cartesian space is chosen to be the shared frame, and the following transformations needs to be applied to transform the dynamic model into Cartesian space

$$\mathbf{f} = \mathbf{M}_C(\boldsymbol{\theta})\ddot{\mathbf{x}} + \mathbf{V}_C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{N}_C(\boldsymbol{\theta}) \quad (4.18)$$

where the relationship between joint space and Cartesian acceleration are developed from the Jacobian matrix, \mathbf{J} , expanded from the kinematics, Equation (4.13) - (4.15).

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\boldsymbol{\theta}} \quad (4.19)$$

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\dot{\mathbf{x}} \quad (4.20)$$

differentiated to obtain

$$\ddot{\mathbf{x}} = \dot{\mathbf{J}}\dot{\boldsymbol{\theta}} + \mathbf{J}\ddot{\boldsymbol{\theta}} \quad (4.21)$$

$$\ddot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\ddot{\mathbf{x}} - \mathbf{J}^{-1}\dot{\mathbf{J}}\dot{\boldsymbol{\theta}} \quad (4.22)$$

The end effector force and joint torques can be related through

$$\boldsymbol{\tau} = \mathbf{J}^T(\boldsymbol{\theta})\mathbf{f} \quad (4.23)$$

The matrices including the physical characteristics of the Phantom Omni from Equation (4.17) can be transformed to Cartesian space by

the following equations

$$\mathbf{M}_C = \mathbf{J}^{-T}(\boldsymbol{\theta})\mathbf{M}(\boldsymbol{\theta})\mathbf{J}^{-1}(\boldsymbol{\theta}) \quad (4.24)$$

$$\mathbf{V}_C = \mathbf{J}^{-T}(\boldsymbol{\theta})(\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - \mathbf{M}(\boldsymbol{\theta})\mathbf{J}^{-1}(\boldsymbol{\theta})\dot{\mathbf{j}}(\boldsymbol{\theta}))\dot{\boldsymbol{\theta}} \quad (4.25)$$

$$\mathbf{N}_C = \mathbf{J}^{-T}(\boldsymbol{\theta})\mathbf{N}(\boldsymbol{\theta}) \quad (4.26)$$

Chapter 5

Modelling and Implementing the System

As the basis for the control of the teleoperated system the position position (PP) scheme presented in chapter 2.5.1 is used. The choice of this scheme is based on the features which the available equipment possesses. There are no force sensor present at the slave side, motivating the use of this scheme in order to remove the need of force measurements. In clinical context, the incorporation of a force sensor introduces challenges both with size and sterility. This argues that a good enough result with a scheme that is not dependent of a force sensor is better than incorporating the sensor to use another control scheme. At the Interventional Centre a PhD research project funded by the Norwegian Research Council is being done to make a dynamic model of a robotic system making it possible to estimate the external forces acting on the robot arm instead of using a sensor to measure them. In that case a sensor is not necessary to generate haptic feedback (Nærum et al., 2008).

To implement the 4 Channel scheme one needs to measure both velocities and forces at both sides of the teleoperated system, which is not possible in our configuration. One force/torque sensor is available so it is possible to implement the force position scheme, but due to time limitations this has not been done.

5.1 Numerical Evaluation

If we analyse the position position control scheme (Figure 2.4a) we can find the equation for the elements of the hybrid matrix, and this can tell us much about the optimal tuning of the various controllers. We start by deriving the expressions for the master and slave velocities

$$\mathbf{v}_m = \mathbf{Z}_m^{-1}(\mathbf{f}_m - PD_m \mathbf{e}) \quad (5.1)$$

$$\mathbf{v}_s = \mathbf{Z}_s^{-1}(PD_s \mathbf{e} - \mathbf{f}_s) \quad (5.2)$$

where

$$\mathbf{e} = \mathbf{v}_m - \mathbf{v}_s \quad (5.3)$$

Equation (5.2) and (5.3) can be solved for \mathbf{v}_m and \mathbf{v}_s which gives us the *position tracking* when $\mathbf{f}_s = 0$

$$h_{21} = \left. \frac{\mathbf{v}_s}{\mathbf{v}_m} \right|_{\mathbf{f}_s=0} = \frac{PD_s}{\mathbf{Z}_s + PD_s} \quad (5.4)$$

Further we see that when substituting equation (5.2) into equation (5.2) for \mathbf{v}_s the *unconstrained impedance* of the master can be found

$$h_{11} = \left. \frac{\mathbf{f}_m}{\mathbf{v}_m} \right|_{\mathbf{f}_s=0} = \frac{(\mathbf{Z}_m + PD_m)(\mathbf{Z}_s + PD_s) - PD_m PD_s}{\mathbf{Z}_s + PD_s} \quad (5.5)$$

If we now set $\mathbf{v}_m = 0$ equation (5.2) through (5.3) will give us the two remaining elements of the hybrid matrix. Substitute equation (5.2) for \mathbf{v}_s into equation (5.2) and the expression for *inverse force tracking* appears

$$h_{12} = \left. \frac{\mathbf{f}_m}{\mathbf{f}_s} \right|_{\mathbf{v}_m=0} = -\frac{PD_m}{\mathbf{Z}_s + PD_s} \quad (5.6)$$

The equation for *contact admittance* reveals itself when setting $\mathbf{v}_m = 0$ in equation (5.2)

$$h_{22} = \left. \frac{\mathbf{v}_s}{\mathbf{f}_s} \right|_{\mathbf{v}_m=0} = \frac{1}{\mathbf{Z}_s + PD_s} \quad (5.7)$$

5.1.1 Stability Analysis

From the equations (5.4) through (5.7) we see that all the hybrid elements have the same denominator, so it is easy to check the two first specifications of the *Absolute Stability* criterion whether any of the matrix elements h_{11} and h_{22} have poles in the right half plane, and if there are poles on the imaginary axis the residues of those must be real and positive. By some manipulation we can find the values of the controllers which satisfies this criterion

$$\text{Denominator: } \mathbf{Z}_s + PD_s \quad (5.8)$$

where

$$\mathbf{Z}_s = \mathbf{c}_{Z_s}s + \mathbf{k}_{Z_s} \quad (5.9)$$

$$PD_s = \mathbf{c}_s s + \mathbf{k}_s \quad (5.10)$$

$$\text{Criterion: } \text{poles} \leq 0 \rightarrow \mathbf{k}_s \geq -\mathbf{k}_{Z_s} \quad (5.11)$$

where s is the Laplace operator, \mathbf{k} and \mathbf{c} are the spring and damper gains, respectively, and we have assumed that the slave impedance have the form $\mathbf{Z}_s = \mathbf{c}s + \mathbf{k}$. If we derive the next criterion for h_{22} whether the real part of $h_{22}(j\omega) \geq 0$ for all real value frequencies, we find exactly the same result as equation (5.11).

It is difficult to get a well-arranged expression which defines the stability limits for h_{11} , but if we assign some numerical values, in one degree of freedom, we can see if the criterions are complied. The models of the manipulators are known, but they are too complex to implement in these analysis, so some generic values will be assigned to show the effect of the analysis. The value of the master impedance is taken from Christiansson (2007), the slave impedance is taken from Cornella (2007), while the controller gains are chosen arbitrary.

$$PD_m = 0.1s + 0.5 \quad (5.12)$$

$$PD_s = 0.1s + 0.1 \quad (5.13)$$

$$PD_g = 0.005s + 0.09 \quad (5.14)$$

$$Z_m = 0.4s + 5 \quad (5.15)$$

$$Z_s = 0.65s + 2 \quad (5.16)$$

For these values the hybrid elements have a single real pole in the left half plane and the first two criterions are complied, in addition we see that equation (5.11) is satisfied.

Next up is to check the stability compared to the frequency, the criterion is that the real part of $h_{11}(j\omega)$ and $h_{22}(j\omega)$ are to positive or equal to zero for all frequencies. With the values assigned we see that

$$\text{Re}(h_{22}(\omega)) \leq 0 \quad \forall \omega \quad (5.17)$$

The last criterion says

$$2r_{11}r_{22} - r_{12}r_{21} - |h_{12}h_{21}| \geq 0, \forall \omega \quad (5.18)$$

which is complied when $\omega \leq 5.71$, see Figure 5.1a. r_{ij} denotes the real part of h_{ij} .

To summarise the results, we see that the system is not absolute stable because of the equations (5.17) and (5.18) which is not positive for all frequencies. The other criterions are complied. This means that the system is potentially unstable.

If we now look at the *Passivity Criterion*, the two first are more or less like the ones in the *Absolute Stability*, and they do still comply. The only difference which affect our selected parameters is the last criterion

$$4r_{11}r_{22} - (r_{12} + r_{21})^2 - (i_{12} - i_{21})^2 \geq 0 \quad (5.19)$$

but this is satisfied only for $\omega \leq 8.52$. So the conclusion remains the same as for the *Absolute Stability*, two out of the three last criterions are not complied, and thus, the system is potentially passive.

Last thing, we will have a look at the scattering operator, which also tells us something about the passivity of the system. When the norm of the scattering operator is less than or equal to 1, the system is passive. Plotting the scattering operator for these parameter values we see that the system is stable for all frequencies below $\omega < 18.3$, see Figure 5.1b.

A conclusion for the position position control scheme with these values assigned, is that it is stable and passive for low frequencies, while for higher frequencies it is potentially unstable.

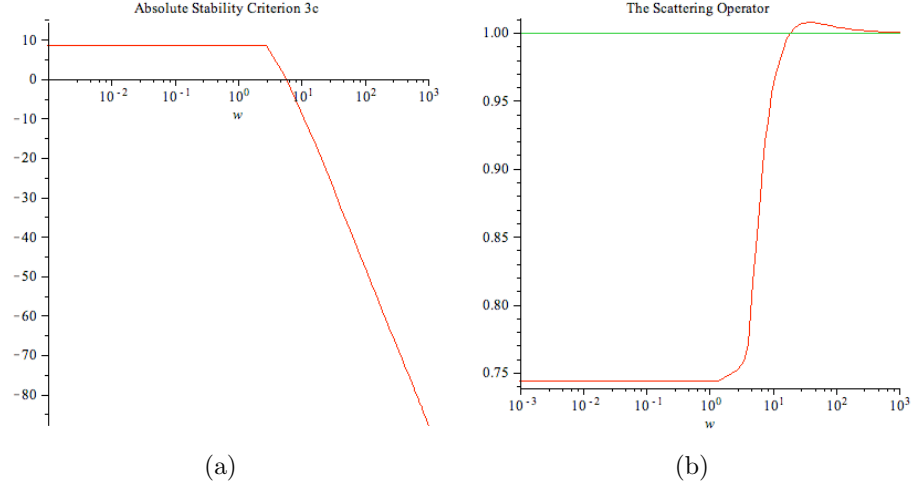


Figure 5.1: Criteria for (a) absolute stability and (b) passivity

5.1.2 Performance

The aim for the performance of the teleoperated system is the ideal values of the hybrid matrix, if we input these values in the equations (5.4) through (5.7) we find the relations between the various parts of the system when the system has perfect transparency.

$$\mathbf{H}(s) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (5.20)$$

This can tell us something about the ideal tuning of the controllers. To get zero master impedance when the slave moves unconstrained we see that the tuning relation between master and slave controller can be expressed as equation (5.21).

$$h_{11} = 0 \quad \rightarrow \quad PD_m = -Z_m - \frac{Z_m}{Z_s} PD_s \quad (5.21)$$

$$h_{12} = 1 \quad \rightarrow \quad PD_m = PD_s + Z_s \quad (5.22)$$

$$h_{21} = -1 \quad \rightarrow \quad PD_s = -0.5Z_s \quad (5.23)$$

In the same way when the inverse of the force tracking is optimal the controllers should be related through equation (5.22). To get optimal

position tracking the slave controller must have a value like in equation (5.23). Actually, the two last equations are enough to tune the controllers of the system optimally. The last hybrid element makes no sense when equal to zero, when there are no contact between slave and environment there are no contact admittance present.

The deviation of the real system hybrid matrix from the ideal hybrid matrix can be used as a performance measure. Take as an example the unconstrained master impedance h_{11} , for low frequencies it has a value of 6, and for higher frequencies it decreases to 5.74. Ideally it should tend to zero. When tuning the gains, tests like this can be performed to ensure a performance which is as good as possible.

A comment to h_{21} , we see that the ideal case, equation (5.23), gives $\mathbf{k}_s = -0.5\mathbf{k}_{Z_s}$, which actually is within the limit of stability and passivity, and thus, the optimal value for this parameter in a telemanipulated system.

5.1.3 Time Delay

All the analysis so far has neglected time delays. Often this is not a realistic assumption in teleoperated systems, due to communication channels which introduces a time delay of some magnitude. Therefore it is of interest to do the analysis with some time delay incorporated. The most common way to model a time delay is to add an exponential part in the equations, in the Laplace domain it is written

$$e^{-\tau s} \quad (5.24)$$

where τ represent the time delay. To represent a time delay in the communication with the slave manipulator one can write

$$\mathbf{Z}_s^{-1} e^{-\tau s} \quad (5.25)$$

When introduced into the simulations, we see that a time delay of 20 ms gives a decrease of maximum frequency for *Passivity Criterion 3c* to 3.13 instead of 8.52, and for the *Absolute Stability* criterion 3c a new maximum is 2.67 in comparison to 5.71. This means that the stability bandwidth decreased significantly. As for the scattering operator, it

shows not to comply the criterion at all when the time delay exceeds $5ms$ for these values.

A time delay will affect the phase of a transfer function by introducing a delay, and a phase delay increases the error between input and output at each instant of time. This means that the position tracking performance will decrease, and the larger the time delay, the more difficult it is to ensure a good performance of the teleoperated system.

5.2 Control Scheme

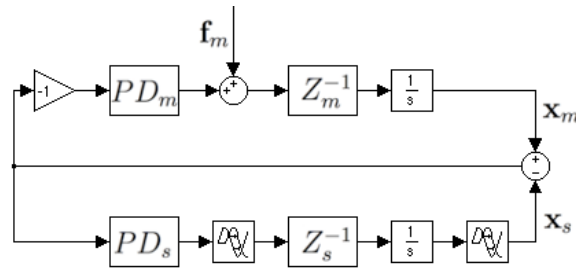


Figure 5.2: Implemented Position Position control scheme

The master-slave system is controlled by the PP scheme, see Figure 5.2. This is a way to connect master and slave in a teleoperated system which gives interaction between the two manipulators and not just one way communication. An advantage is that for later extensions and improvements of the system it is easier to include force feedback.

The teleoperated system has three degrees of freedom (DoF), so in fact there are three PP schemes working, one for each direction. This is under the assumption that the three directions are independent of each other. The way this is implemented is by using a matrix approach giving the ability to tune the specific gains in each axis.

In chapter 2.5.1 the feedback control loop depends on the difference in velocities of the two manipulators. This is due to the impedance control strategy which relates force and velocities. Some changes from this scheme has been done in this system. Instead of using the velocity

error as set point for the control of the manipulators the position error is used. There are mainly two reasons for using the position error instead of the velocity. Firstly it is only possible to read positions from the slave robot so a velocity estimator is needed if velocities are to be used. Secondly, the guidance system depends on positions and thus it is more convenient to use this for both control and guidance purposes. As a matter of fact the velocities measured in robot systems are most often a result from position sensors differentiated by time.

An important effect of using positions is that the error is absolute relating to spatial positions. On the contrary, if the velocity error is used, the relation to spatial positions are relative and will lead to spatial drifting. This is of course not good enough in guidance purposes where the constraints relates to given spatial positions.

Another change done compared to the general PP scheme concerns the input of the slave robot. The only way to control the slave robot is to input joint velocities with which the robot are to move. So the slave controller calculates the velocities which is required to follow the master. Both the slave and master controller are PD controllers.

The transport delay block in figure 5.2 represents the time delay which is present in the communication with the slave. The time delay is approximately $25ms$ for sending and receiving information to the manipulator, which gives a total time delay of $50ms$ for each iteration of the control loop. As indicated earlier this delay limits the frequency of the entire system.

5.2.1 Tuning the PD Controllers

By implementing the control and guidance system in Simulink and Matlab one can do some theoretical analysis and simulations before implementing the real system. In this way the parameters of the system can be tuned by giving various input signals and see the response, overall performance and stability can also be analysed. In order to implement the system the theoretical model needs to be available, which includes the dynamics of both manipulators. The manipulator models are known, so it should be possible to do the implementation, but it appears to be very difficult to implement the model of the Phan-

tom Omni, due to its complexity. Several attempts are made, a lot of changes done, and a series of simplifications are tested out in order to obtain a model which can simulate the real performance of the manipulator, but in vain. The result is no knowledge about the performance before the system is implemented on the real system.

In order to tune the control parameters on the real teleoperated system, trial and error is used. First the position tracking between master and slave are tuned by moving the master while the slave follows unconstrained. The goal is that the slave follows the master as tightly as possible.

$$\mathbf{K}_s = [1.0 \quad 1.0 \quad 1.0] \quad (5.26)$$

$$\mathbf{C}_s = [1.0 \quad 1.0 \quad 0.6] \quad (5.27)$$

We see that the optimal tuning gives high gains, and the same value for the two first axes. The third, which is the z-axis, has a bit lower damping gain than the others to reduce noise.

By continuing the unconstrained motion, one can tune the master impedance. This is supposed to be as low as possible to generate small forces.

$$\mathbf{K}_m = [0.010 \quad 0.010 \quad 0.020] \quad (5.28)$$

$$\mathbf{C}_m = [0.001 \quad 0.001 \quad 0.005] \quad (5.29)$$

This tuning gives small forces for free motion, but not zero. The damping introduces most forces, due to the velocity difference between the two manipulators. It works like a viscous friction effect for the master forcing the operator to not move faster than the slave can handle. Tuning the impedance for hard contact was not possible because of hardware limitations of the slave robot.

5.2.2 Transformation of Coordinate Frames

When using two manipulators in the same system a shared coordinate frame needs to be established to ensure appropriate communication. If the manipulators have the same kinematic and dynamic configuration a shared frame may be the joint space since both robots will behave

similar. For manipulators with dissimilar kinematic configuration this is not the case, but an alternative frame to choose is the Cartesian space.

The master can be manipulated in Cartesian space directly, but the control of the slave takes place in joint space. Forward kinematic equations and the Jacobian matrix evolved in chapter 4.1 can be used to transform the positions and velocities between joint and Cartesian space. Investigating the kinematic configuration of the two manipulators reveals another important issue which is the difference in alignment of the Cartesian axes, see figure 4.2 and 4.3. By using the following transformation we can ensure correct communication. The coordinate frame which is used in the guidance algorithms is the slave coordinate frame.

$$x_m = y_s \quad \dot{x}_m = \dot{y}_s \quad (5.30)$$

$$y_m = z_s \quad \Rightarrow \quad \dot{y}_m = \dot{z}_s \quad (5.31)$$

$$z_m = x_s \quad \dot{z}_m = \dot{x}_s \quad (5.32)$$

5.3 Haptic Guidance

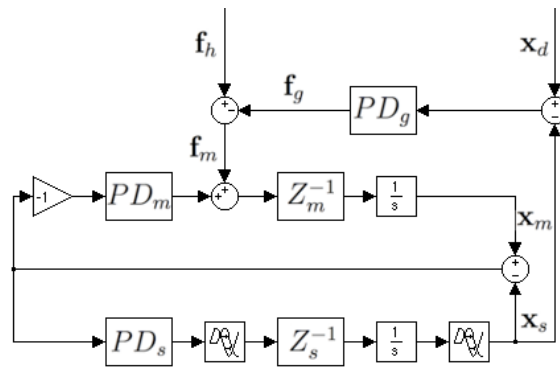


Figure 5.3: Implemented PP control scheme with haptic guidance

The haptic guidance system is implemented as an extension to the

PP control system, see figure 5.3. The inputs to the guidance system are the desired position $\mathbf{x}_d \in R^3$ and the real position $\mathbf{x}_s \in R^3$ of the slave robot. The deviation between these two points are calculated on the basis of a control law chosen by the operator which either can be trajectory guidance or no-go-zone guidance. The trajectory guidance is implemented as the 2 constrained attractive line guidance, while the no-go-zone is implemented as the 1 constrained repulsive plane guidance where the plane represent the surface of the object which is the restricted area. A 3 constrained attractive point guidance is also implemented, but this gives no analogy to real life functionality other than as an example of how the guidance forces are generated. One valuable outcome of the point guidance is to tune the forces in each axis to become as similar as possible within the limits of stability. On the other hand the system get unstable very fast in this mode because the only point where there are no forces generated is in the initial point which due to its size almost never occurs.

The guidance forces \mathbf{f}_g are introduced to the system as an addition to the forces \mathbf{f}_h from the human operator, and together they form the master force \mathbf{f}_m . In most cases these will be of opposite signs since the task of the guidance forces is to correct the movement of the operator. If no guidance is chosen the force contribution \mathbf{f}_g will be zero and the system works as the stand alone PP control system.

5.3.1 Features

When starting the Haptic Guidance program there are no guidance activated, so one can manipulate the master freely. The operator has four modes to use, he/she can continue without any guidance or to choose point/line/plane guidance. The maximum distance in [mm] (figure 3.5) from the guidance trajectory can also be manipulated to increase or decrease softness/hardness of the guidance, see figure 5.4 for instruction keys.

To move the slave robot the button on the haptic must be pushed, and as long as the button remains pushed the slave will follow the haptic device. When the button is released one can move the haptic freely without concern of the robot because it is deactivated, and when

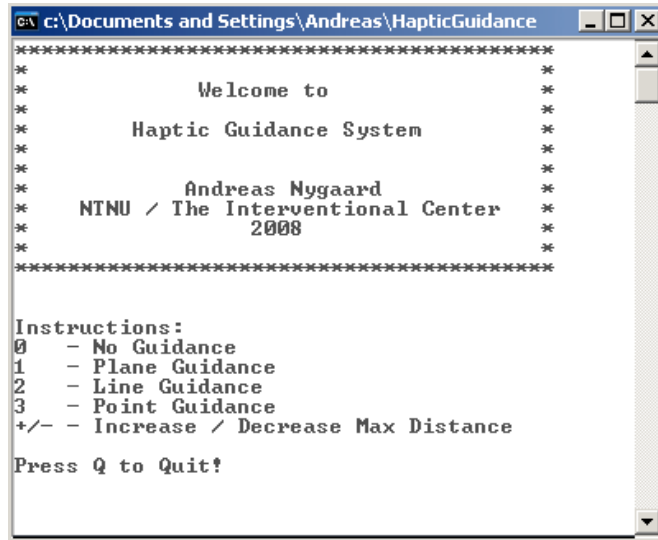


Figure 5.4: Opening window when starting the guidance system

the button is pushed again the robot continues from where it stopped. This feature works like a clutch and is very useful thus the workspace of the haptic device is limited and scaling motions will strengthen this need.

When the line guidance is activated first of all the operator is asked to create a line which will work as the guidance trajectory to follow. The same is for the plane guidance, which ensures the robot to not cross the given plane without force being exerted through the haptic. The operator must mark three points creating a plane before starting.

5.4 Implementation Details

The presented system is implemented on a regular PC in C++, which is supported of both manipulators. In this chapter an overview of the main functions and functionality of the system is presented. The various functions of the system can be grouped on the basis of their working area, and this can be used to give an overview of the structure and to

summarise the communication.

- *Master Control Loop* - control of the Phantom Omni
- *Slave Control Loop* - control of AESOP
- *Force / Velocity Generation* - this is the brain of the haptic guidance system
- *Main Control Loop* - synchronises and control the overall system

5.4.1 Master and Slave Control Loop

Both master and slave are controlled by a separate threads. The AESOP control loop is a black box which we can communicate with through some commands, while the Phantom Omni control loop are specially designed for the guidance system.

As mentioned earlier the recommended update frequency of the loop controlling the haptic device is 1000Hz. A predictable way to establish this is to create a function with a limited number of tasks to run as the control loop of the haptic. This gives the high speed, accuracy and stability needed, thus it is not dependent on external tasks to be done and only a few tasks are to be done each round of the loop. The Phantom Omni API gives the ability to run the control loop of the haptic device in a separate thread, along with a complete thread safe communication set, which ensures synchronised communication between this thread and other threads running. The function *HapticLoop()* is the haptic control loop with the task of reading positions and velocities from the device storing them in a struct supervised by this function, and to put forces onto the device. To retrieve this information and send forces to the haptic control loop from the other thread, two functions are created. *ReadSysParam()* and *WriteSysParam()* gives synchronised access to read from and write to the information struct, respectively. Even if the haptic control loop has a frequency of 1000Hz, the forces will only be changes as often as the force generation algorithm provides new forces.

The control loop of the AESOP requires the use of given commands to communicate with it, and the information it needs and provides are

in an AESOP reference frame. Therefore some functions are created to take care of this. All calculations are done in AESOP Cartesian space frame, but the communication with AESOP are in joint space, so the two functions *AesopCartToJointVel()* and *AesopForwardKinematics()* converts velocities from Cartesian to joint space and converts joint angles into Cartesian positions, respectively.

5.4.2 Force and Velocity Generation

The master and the slave manipulator are controlled by force and velocity, respectively, and to generate these effort variables the algorithms from chapter 3.3 are implemented in the function *CalculateForce()*. The force and velocity are generated from the basis of the position error between the two manipulators plus the guidance constraints. There are implemented three different types of guidance, point, line and plane, which may be chosen among. If a guidance type is chosen the deviation of the slave position compared to the constraint is calculated. Later this deviation and the position error are used to calculate the force/velocity. Finally these effort variables are communicated to the manipulators.

5.4.3 Main Control Loop

The *mainloop()* is the state machine taking care of the overall system. This loop runs until some error occurs or the Quit button is pushed. For each iteration the manipulator variables are read, force and velocity are calculated, it checks if command buttons are pushed and the effort variables are transferred to the manipulators. The frequency of this loop is mainly set by the communication delay with the AESOP system, in other words it runs at 20Hz. The advantage of keeping the haptic control in a separate loop remains with the synchronised communication and the guarantee of stable updates of input and output even if there have been no changes in the force calculation.

The information flow and command timing can be summarised in a sequence diagram where the four objects are the categories presented above. Some details which can be mentioned here is that start and stop procedures occur in the very beginning and at the end of running the

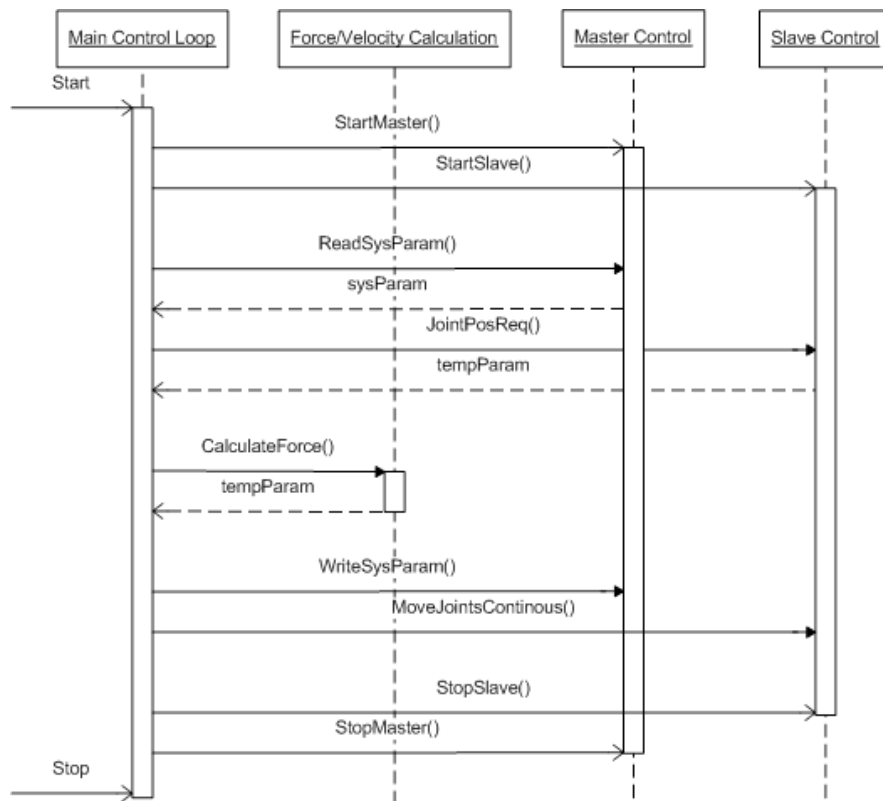


Figure 5.5: Sequence diagram of the communication in the system

program, in between the loop iterates with the steps in figure 5.5. The Slave Control also includes some conversion commands which needs to be used together with the *JointPosReq()* and *MoveJointContinuous()*. There are also a function that checks if some keys are hit and a function that writes to a log for analysing purposes.

Chapter 6

Experiments and Results

6.1 Position Position Control Scheme

The PP control scheme forms the basis for the teleoperated system and plays an important role for the performance of the system. But the PP scheme has mainly two objectives; to make sure that the slave robot follows the master robot as tightly as possible, and to feed back to the master the forces sensed at the slave side in a realistic manner. Both these aspects will be looked closer into through this chapter, and tests will be carried out to discover the properties of the implemented system.

6.1.1 Master Slave Position Tracking

In an ideal teleoperated system the hybrid matrix have the values

$$\mathbf{H}(s) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (6.1)$$

Among others this means that the position tracking $\frac{\mathbf{x}_s}{\mathbf{x}_m} = h_{21} = -1$, which express that the motions of the slave are the exact same as the master, if no scaling of motions are present, and the bandwidth tends to infinity. In this report the teleoperated system are assumed to have a 1:1 scaling of motions in the Cartesian space. These are challenging

requirements for a real life systems and not distinctly possible, but they can work as a good basis for the performance ambitions. Plotting the slave and master position in the same coordinate frame gives us an idea of the tracking performance, see Figure 6.1.

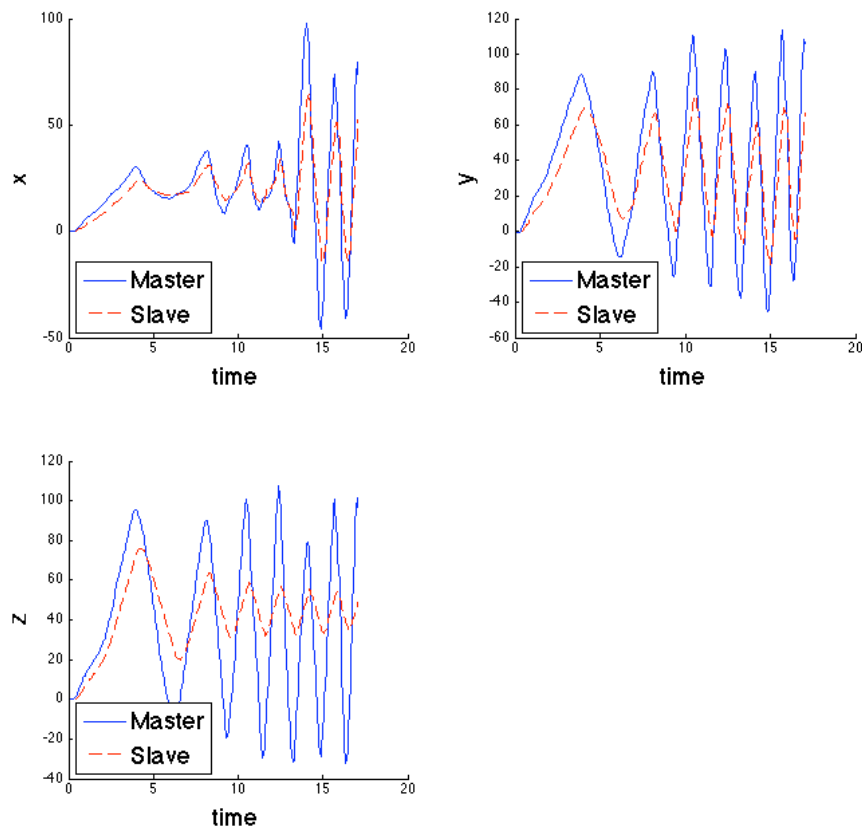


Figure 6.1: Position tracking error between master and slave

We see that for small variations and velocities the slave follows the master fairly well in all axes, but when increasing the frequency, and thus the velocity, the slave slows behind, and especially in the z axis. The fact that the z axis joint is translational while the other two joints are rotational can explain the different behaviour. The velocity of the translational joint is limited compared to the two others. As the fre-

quency of the master oscillations increases the variations or amplitude of the z axis decreases.

6.1.2 Force Feedback

The force feedback depends on the deviation between the two manipulators, under the assumption that if the deviation is large the slave manipulator is obstructed by the environment. This assumption depends on a teleoperated system where the master and slave position tracking error is small, so that the only plausible explanation to a deviation is external interaction with the slave robot.

Taking into account the performance of the position tracking it is a reason suspect that PP control scheme most probably is not the best suited control scheme for this system in means of force feedback, at least it limits the working range of the velocities. To test this out, the *unconstrained movement test* from Chapter 2.7.1 is performed on the teleoperated system. This means that the force from the slave is $\mathbf{f}_s = 0$, and the slave robot is free to move unconstrained. Ideally this shall lead to no forces felt at the master side, which is not the case. When moving the master, forces are felt working towards the movement direction, like sensing a large frictional force. The cause of these forces are the large tracking error between the manipulators. In order to decrease these forces the gains needs to be scaled down closing to zero

$$\begin{aligned}\mathbf{k}_m &= [0.01, 0.01, 0.02] \\ \mathbf{c}_m &= [0.001, 0.001, 0.005]\end{aligned}$$

The other test which is interesting to run on the system is the *hard contact test*. In this case forces are to be generated at the master side when the slave interacts with some physical constraint. The challenge is that the gains are scaled down from the previous test, so most probably they need to be scaled up to a certain degree to give any force feedback. A problem appears to be the slave robot in itself. It is not possible to move the robot out of position, due to security reasons it shuts down when it encounters to high external forces. Another thing is that the

system is very rigid and the joints might have some sort of position control so that it is not possible to move the links out of position. This means that no forces will be generated on the bases of slave position error caused by external effects. If the PP control scheme are to be used with this slave robot a external position tracking system may be used to track the position of the slave and compare this to the internally measured position. A deviation will be present because the links and joints are not absolute rigid. On the other hand the Force Position control scheme might have better conditions to succeed.

A concluding remark to the force feedback in this configuration is that it may play a role as it is implemented now, and that is to limit the operator movement velocity to the accessible velocity of the slave. This may give the advantage of decreasing the position tracking error.

6.2 Slave Based Guidance

The guidance system bases its functionality on the already existing PP control architecture and functionality, and provides an extra feature for the total system which is quite easy to incorporate on the teleoperated system. The performance of the guidance system will depend significantly on the performance of the teleoperated system, a poor performance will be a limitation to the guidance system. Experimental studies of the guidance system therefor needs to be analysed in the light of the total system. In this chapter tests will be carried out to see the performance of the guidance system.

6.2.1 Force vs Deviation

To see the force response compared to the deviation we have plotted them in Figure 6.3c) in the xy plane together with the guidance line. The forces ideally shall act orthogonal to and towards the guidance line and with a magnitude proportional to the deviation but with some damping related to the deviation velocity. As the plot shows us, the forces are pretty close to orthogonal, and the forces are large when the deviation are large. If we look careful we see that the forces are slightly

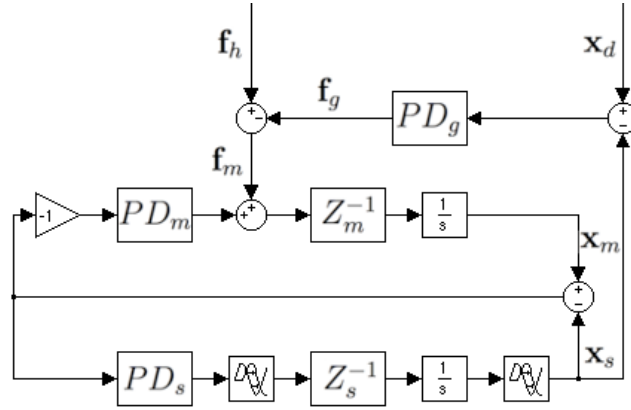


Figure 6.2: Slave based guidance

decreased or increased when the deviation increases or decreases fast compared to the almost static deviation. This proves the damping effect.

6.2.2 Stability

The point guidance shows to be useful when tuning the PD gains due to the intuitive force generation when deviating from the initial point $\mathbf{p}_i = [0, 0, 0]$, and it is easy to compare the tests with each other when trying new gains. After some testing the system demonstrates to be very vulnerable to high gains which makes the system unstable. Another property that reveals is that the operator grip onto the master device plays a significant role. When the operator keeps a firm grip to the master a higher gain can be used than when the grip is soft. In other words the impedance Z_h of the operator decides which values are possible for the gains. A higher damping effect of Z_h gives ability to increase the gains. Since higher gains gives more stiffness of the constraints the operator impedance also decides the possible stiffness.

When the guidance are tested with line and plane constraints, it appears that the gains from the point tests are too conservative, and can therefore be increased. The spring and damper gains are the same for

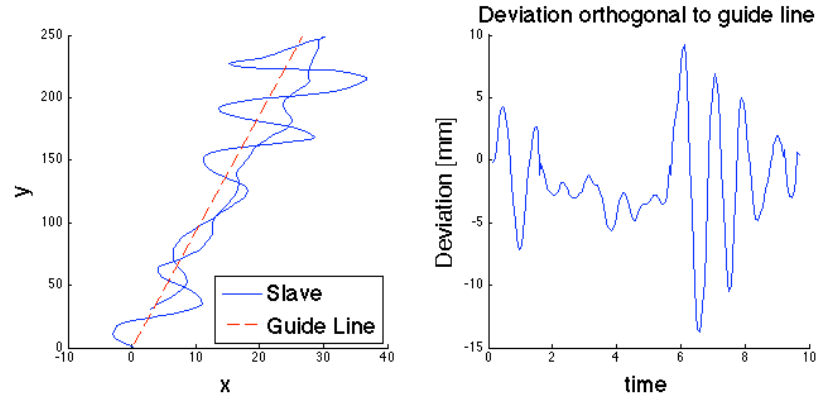
all the guidance types so the difference must be caused by the guidance algorithm in itself. In the case of point guidance there are only one point where there are no guidance forces created, and that is in the initial point $\mathbf{p}_i = [0, 0, 0]$. The reason for the conservative gains might be because the total amount of forces acting in all three axes are larger than for the other guidance types, where only one or two axes are constrained.

6.2.3 Effect of the Maximum Distance Limit

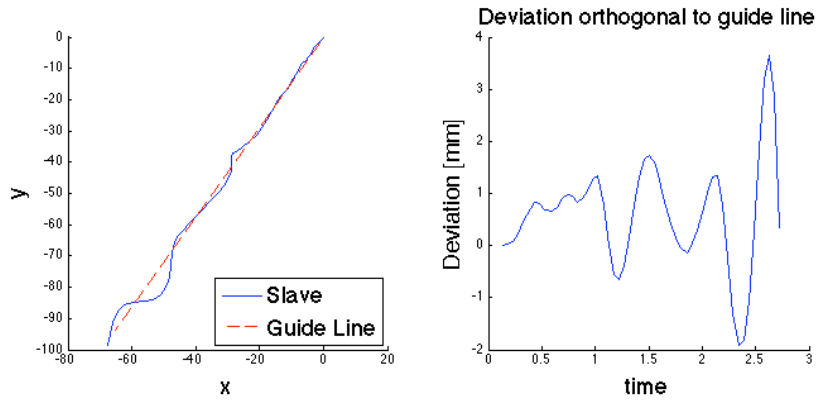
When the maximum distance is increased we see that the the system gets under damped, and oscillations occur in the beginning of the guidance. Figure 6.3a shows the guidance along a line plotted in the xy plane. The slave starts in position zero $\mathbf{p}_i = [0, 0, 0]$ at time $t = 0$ and moves to the end of the line and then turns back again. We see that in the beginning of each movement there are some oscillations which decreases slowly (see the time line). The maximum distance in this case is set to $v_{d_{max}} = 20mm$, and we see that the deviation is less than $v_{d_{xy}} < 14.2mm$. In (b) the $v_{d_{max}} = 5mm$, and we see a different response. In the beginning the slave follows the guide line quite strict, and then suddenly the deviation increases until the system gets unstable. The instabilities are caused by the total gain of the system which has become too high and when the deviation exceeds a certain magnitude the system gets unstable. But within some limit the slave follows the line better when the $v_{d_{max}}$ is low. This result indicates that the $v_{d_{max}}$ plays a significant role in the stability and stiffness of the guidance, and needs to be taken account for when tuning the guidance controller, optimally the controllers should be tuned dynamically compared to the given maximum distance to give the best performance possible in each case.

6.2.4 Guidance Modes

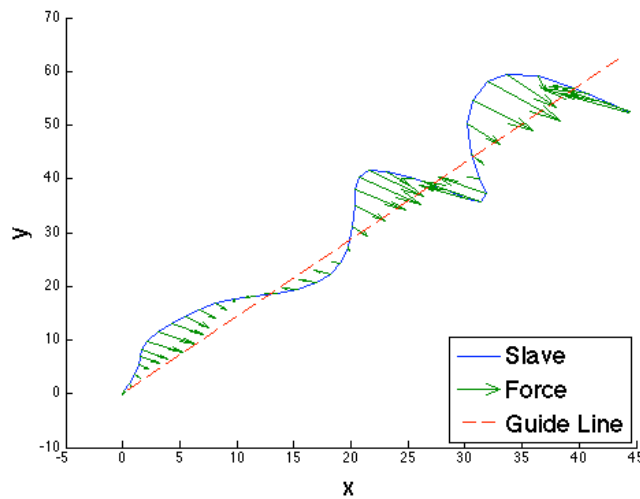
In general we see that when the guidance modes line and plane are utilised the stiffness is not very high. A compromise have been done tuning the gains as stiff as possible within the limits of stability, re-



(a) Guidance with $v_{d_{max}} = 20mm$



(b) Guidance with $v_{d_{max}} = 5mm$



(c) Relation between force and deviation

Figure 6.3: Properties of slave based line guidance

sulting in guidance constraints which are not very strict. Figure 6.3 (a) and (b) shows the variations in level of the maximum distance, a distance level within the limits of stability gives a deviation of up to 10 mm, and still when moving at a maximum distance the force is not sensed as very stiff. This gives a vague sensation of a line, but with too small accuracy to give any benefit in means of improving the precision of the operator movement.

The no-go-zone guidance works better, not because the forces are stiffer, but because the sensation of being within a forbidden area is seems realistic. There are no wall feeling when entering the no-go-zone, but the forces are large enough and scaled in the right direction to tell the operator that he should not be there.

6.3 Master Based Guidance

To see how a guidance system works if time delays are decreased and position tracking of the slave robot is optimal, we implemented the guidance system on the haptic device in a closed loop with the slave robot as a passive slave only following the master, see figure 6.4. We

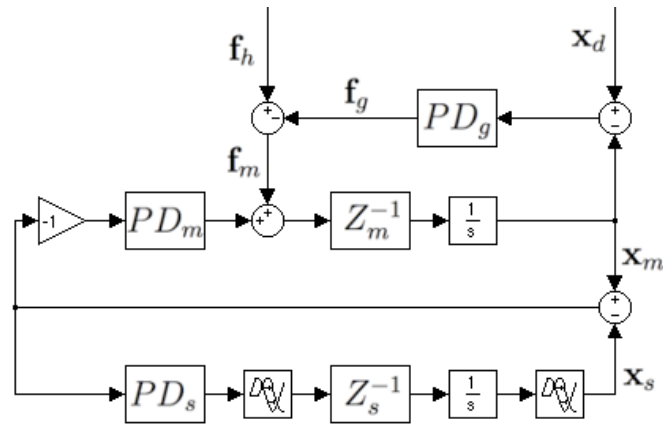


Figure 6.4: Master based guidance

see that the guidance loop is closed and not dependent of the slave at

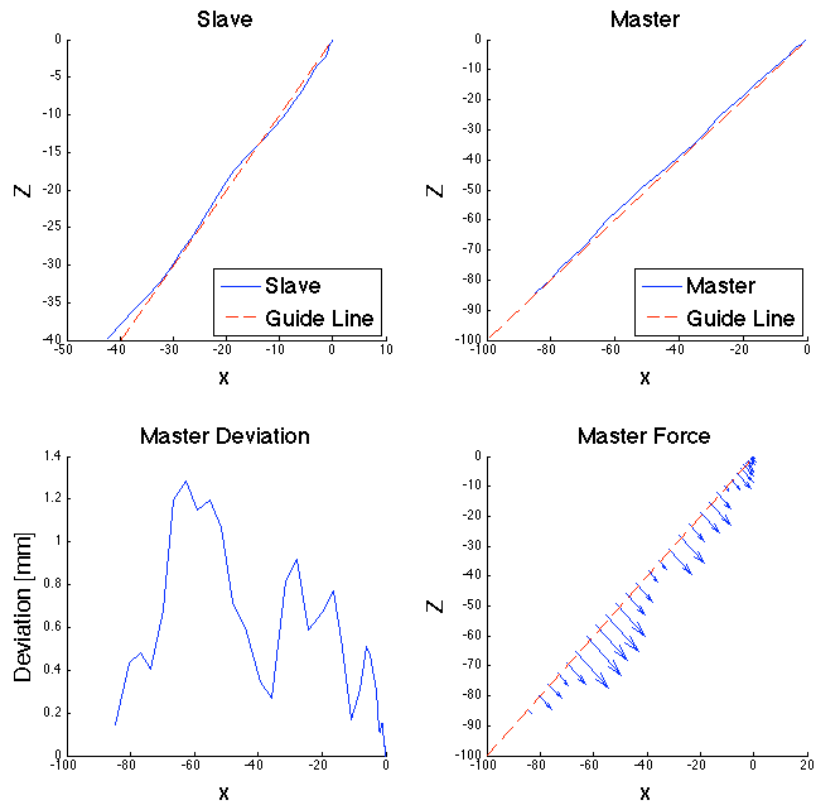
all. The position error which works as input of the guidance system is a result of the deviation between master position \mathbf{x}_m and desired position \mathbf{x}_d . This can illustrate a telemanipulated system where the position tracking is perfect $h_{21} = -1$, see Chapter 3.1, while the update frequency of the loop remains the same.

When line guidance is activated, we see that both master and slave robot follows a line very well (Figure 6.5a). The system is stable and the deviation of the master in the xz plane is $|v_{d_{xz}}| < 1.3mm$, which is very good. Figure 6.5a also shows the corresponding force $|f_{xz}| < 1.9N$. We see that the force is based on the deviation but with the opposite sign. In this test the maximum distance $v_{d_{max}} = 13mm$, which means that the forces are graded within this limit compared to the PD values, while outside the limit the force is maximum. For the other planes the deviation is $|v_{d_{xy}}| < 4.7mm$ and $|v_{d_{yz}}| < 5.8mm$ which have some higher value, but still acceptable.

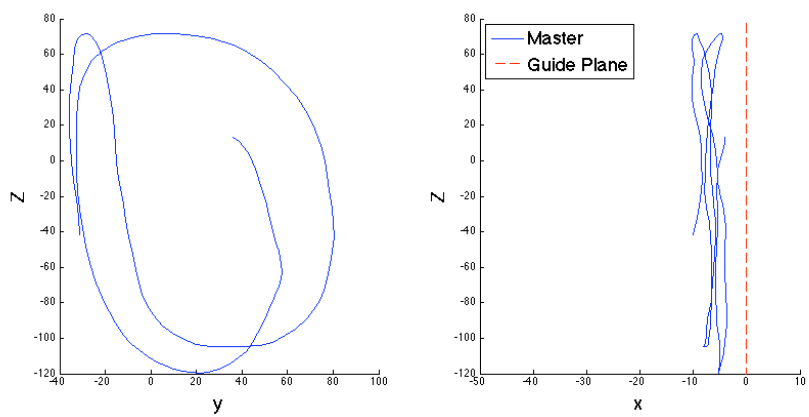
The plane guidance is also tested out, see Figure 6.5b, here with a plane parallel to the yz plane at $x=0$. We see that there are no restrictions in any axis but $x=0$. As it is implemented here the forces are created when the master crosses the plane. The plane is a very good example to show the effect of the maximum distance $v_{d_{max}}$. In this case the maximum distance is set to $v_{d_{max}} = 10mm$, and we see that the deviation compared to the plane is $v_{d_x} < 10.4mm$. The forces are directed in opposite direction orthogonal towards the plane, and the maximum force are exerted when the deviation exceeds $v_{d_{max}}$. As a matter of fact the hardness of the surface can be tuned by altering the maximum deviation parameter. When decreasing $v_{d_{max}}$ the surface will get harder, and in opposite case the surface will get softer. This can be used deliberately to grade the strictness or stiffness of the constraints. If it is very important to not cross the constraint, a hard surface will be beneficial, or else one can increase the force gradually to alert a closing constraint.

6.3.1 Comparing Master and Slave Guidance

A clear improvement appears when comparing the master based (MG) and slave based (SG) guidance. In the case of line guidance we see that



(a)



(b)

Figure 6.5: (a) Line and (b) plane guidance based on master guidance

the MG deviations in general are smaller than for the SG and the MG follows the line better. There are some oscillations about the line for the SG while for the MG the same phenomenon are absent. With the MG line guidance the accuracy is high enough to give the operator a clear cognition of the line to follow, and the guidance clearly improves the operators ability to move along a straight line.

As for the plane guidance, the SG guidance is good enough to generate forbidden areas giving the operator a clear sensation of being in the wrong place. The accuracy though is not high, so the requirements of the strictness can not be too high, but to restrict areas where there are enough space the no-go-zone mode does make a benefit. Where it is limited operating space, and close between operating space and forbidden areas, the constraints needs to be strict, the maximum distance must be small, and in this case the MG shows to be efficient thus the stiffness can be very high without making the system unstable.

Chapter 7

Discussion and Conclusions

A control system for a teleoperated system with haptic guidance has been proposed in this report, and steps have been followed to put together, analyse and implement the system with the highest possible performance. Weaknesses and strengths have been revealed in control schemes, algorithms and hardware through the work, and some conclusions are drawn on the way, but this chapter will summarise and discuss the results further and draw some conclusions.

7.1 Hardware

The Phantom Omni and AESOP are two very different manipulators, both in kinematic and dynamic configuration, and their intention of use separate significantly. This is a relatively new basis for research within teleoperated systems and not common to be found in the literature, and thus, there are some new challenges to meet within performance and stability.

7.1.1 Phantom Omni

The Phantom Omni is a light weight manipulator created for the purpose of controlling systems, and is optimised for low friction, high degree of freedom and spatial manipulations. Forces are available mainly

to create feedback forces to the operator and not to control the manipulator like a robot. This has made the system identification more difficult, and thus, the characteristics have only been found for one axis, assuming this to be representative for the others. Some advantages are easy and intuitive manipulation of real and virtual systems in 6 degrees of freedom, low frictional forces to avoid user fatigue and it is simple to use in embedded systems due to it is open source. But some drawbacks are the limited work space, requiring the use of clutch-like features to increase the work space, and the stiffness of the forces are limited when simulating a hard touch.

7.1.2 AESOP

ASEOP, on the other hand, is a very rigid and heavy robot built to perform high precision tasks. The work space is somewhat larger than for the Omni, but still limited due to its intentional purpose of holding a camera or performing laparoscopic procedures. The most important challenges with this robot is the kinematic configuration, the limited velocity and the communication to control it. The z-axis joint is translational, while the two others are rotational, due to gravity and a notable device mass, the translational joint most probably has a larger motor and a different control system to generate enough torque. Thus, it moves in a different way which needs to be taken account for. The control of the robot is done in a closed system, with unknown characteristics. But one can send commands to the control system to control the movements. The drawback with this communication is the time delay of $25ms$ to read and write, and that no direct control of the robot is possible. The time delay becomes a critical point for the entire telemanipulated system.

The different kinematic configuration, compared to the Phantom Omni, leads to the need of transformations between joint and Cartesian space, which is the common spatial frame, but this is no problem as long as the kinematics are derived.

7.2 Control Scheme

The most commonly used control schemes for telemanipulated systems are presented and discussed, before the position position (PP) scheme are chosen for implementation. Mainly because of hardware limitations. If the hardware permits, the 4 channel control scheme is preferred due to better performance concerning force feedback, but as a basis for master-slave control the PP scheme is sufficient due to its potential stability.

7.2.1 Force Feedback

Concerning force feedback, the PP scheme has a very high unconstrained impedance. In addition the velocity difference between master and slave manipulator makes the position error large and gives an additional contribution to the unconstrained impedance. In order to keep the forces low in free movement the gains must be low. Usually this is a problem when recreating the hard contact forces because it requires higher gains, but apparently AESOP is not suitable for this purposes. When simulating hard contact force with the robot, it shuts down when too large resistant forces are introduced. Additionally, the rigid links and joints make it impossible to force the links into new positions by using external force, and the position readings provided by the system do not sense any change. A way to use the PP scheme is to externally measure the position of the robot using some tracking system, and then generate forces on this basis. But most likely it would be preferable to use another control scheme, as the force position scheme or the 4 channel. With the equipment used in this report, the shortest way is to implement the force position scheme, because all the required equipment are available. To implement the 4 channel, force measurements needs to be taken in both sides of the telemanipulated system, which will be a challenge with the Omni.

The force feedback generated by the PP scheme in this configuration can be utilised to limit the velocity of the operators movements, because when moving too fast compared to the slave, the operator will feel frictional forces limiting the velocity so that the slave manages to follow

the master. For small movements at low frequencies forces will not be generated.

7.2.2 Stability and Performance

When analysed theoretically the PP scheme shows to be potential stable. This means that the right conditions needs to be present for the system to be stable. These conditions are not only internal tuning of gains, also external factors like the human operator and the environment affects the stability. In this report only the human operator, since the force feedback are generated by virtual constraints and not by real contact. The theoretical analysis have been performed excluding the effect of the human operator, thus, the stability margins can be increased to a certain degree due to the high damping effect of the human operator.

In general, we see from the analysis that the system is stable for low frequencies, but as the stability bandwidth is not very large, an increase in frequency leads to a potentially unstable state. Introducing a time delay in the analysis, shows that a small time delay is acceptable, but as soon as the delay increases (above $5ms$ for the values in Chapter 5.1.1) the system gets potentially not passive. The time delay also affects the position tracking performance, by introducing a phase delay between master and slave which increases the position error, and thus, forms a basis for increased free movement force feedback.

When comparing the frequency response of the chosen transfer functions in Chapter 5.1.1 with the ideal values of the hybrid matrices, we see that the performance is not very good. This comparison provides a basis for tuning the system parameters, and in some cases we can increase the performance quite significantly by using this measure. Though on the real system with three degrees of freedom it is difficult to acquire the hybrid matrix, and the trial and error procedure is the best way to tune the system to optimise for stability and performance. But some general relations from the theoretical analysis can still be employed when tuning the real system parameters, like the relation between the various controller gains and the impedance of the manipulators.

7.2.3 Improving Position Tracking

The position tracking between master and slave appears to be one of the main challenges for the system performance, and to increase the performance this problem needs to be addressed. By introducing an external tracking system to measure the slave position, the time delay could be decreased to the half of the today's value, because the only need is to write velocities to the slave control system, while the readings are provided by the tracking system. With a smaller time delay, the position tracking also will improve.

Another solution is to introduce a predictive control scheme to control the slave robot. This requires the future trajectory to be known, which is the case for the line guidance but not for the plane guidance. Though, for the line guidance, it is still one degree of freedom which is not constrained, the movement along the trajectory. In this case it is not possible not to predict the movement of the operator, and thus, not possible to use a predictable scheme. But since no forces are introduced in this direction it will not lead to any problem.

Regardless of the attempts done to improve the position tracking, it will only work within the capabilities of the manipulators. The dynamic response of the AESOP is limited and differs in the various axes, and to a large extent this is the reason for the poor performance of the teleoperated system.

7.3 Haptic Guidance

The haptic guidance system is incorporated in the conventional PP scheme in a way that adds the generated forces to the forces already present due to the PP controller. This makes the system completely separated from the PP scheme, and is simple to incorporate with other control schemes. But still the guidance is highly dependent on the performance of the control scheme.

7.3.1 Limitations

The guidance system is implemented as a control loop outside the tele-manipulated system, and thus the limitations of this system also applies for the guidance system. The time delay and position tracking are the most significant challenges with the hardware available. The guidance system is dependent on the position tracking, and a poor position tracking introduces stability problems to the guidance system. A reason for this is that the force response of the master manipulator is fast, and compensate a position error fast. When a phase delay is introduced between master and slave, the master can compensate a deviation faster than the slave, and as the tracking error increases for higher frequencies the error increases, and the system gets unstable. Thus, the gains needs to be limited to avoid introducing too high forces at the master side.

7.3.2 Slave Based Guidance

An objective of this report is to develop a guidance system which can improve the performance when performing some tasks. Line and plane guidance are two modes introduced to do this.

With the limitations of the system in mind, we can see how the performance of these guidance types are. In the case of line guidance, the objective is to guide the operator along a given trajectory to increase the precision and accuracy. If this is supposed to give any benefit to the operator the forces must be quite stiff, in order to emulate a distinct trajectory. The stiffness decides the strictness of the trajectory, small forces gives a larger deviation from the trajectory and the constraints are sensed as soft, while high forces give smaller deviation (assuming a stable system) and hard constraints. Accordingly we need high forces to provide high precision.

The slave based guidance does not give the ability to increase gains much before the system gets unstable. Assuming a high damping effect from the operator, or in other words a firm grip, the best possible performance within the stability limits gives a result which is not satisfactory. Deviations from the trajectory of up to $15mm$ are not

sufficient to give high accuracy and precision, and the sensation of a line is vague. Though, if the working space is large enough, this feature can be utilised to guide the operator along a preferred trajectory, but without making harm if deviating from the trajectory. In general we see that if movements are slow and limited the gains can be increased noteworthy without destabilising the system, and thus, increasing the stiffness and accuracy.

The plane guidance represent the forbidden area or no-go-zone guidance. In this mode we see that the guidance works better and the system is stable for higher gains. Still the forces are not sensed as if hitting a wall, but they are introduced gradually compared to the degree of violation of the constraints. The objective is to inform the operator that a forbidden area is approaching, and this works very well with forces driving the operator away. Though the same applies here as for the line guidance, small working spaces requires more strict constraints, and thus, this mode is not sufficient in those cases.

7.3.3 Master Based Guidance

To simulate perfect position tracking conditions of the teleoperated system, an optional implementation have been done. The guidance is based directly on the master positions, assuming the slave to follow perfectly. Early in the work the update frequency of 20Hz where suspected to be the greatest challenge to the performance of the guidance system, due to recommended update frequencies of up to 1000Hz to get a optimal result when generating force feedback. In most real time robotic systems this is not possible, and the literature has argued different frequencies to be the lower limit. As the tests are done with the master based guidance, it appears to be very well sufficient with a 20 Hz update frequency, and that the real problem is the position tracking between master and slave. The gains of the master based guidance can be tuned high to generate stiff constraints without making instabilities.

This guidance scheme really reveals the possibilities when using guidance in a teleoperated system. The line guidance gives very satisfactory results with a deviation of up to $1.3mm$ in the best performing axes and tests, which is adequate to improve the accuracy and precision

of the human operator. The same result appears for the plane guidance, quite hard contact can be generated, which gives opportunities that the slave based guidance can not provide, like limiting an already small work space. The stiffness of the constraints might be increased even more if the update frequency is increased, but the performance of this setup is satisfactory.

In real life use it is not a solution to use the master based guidance due to the deviation in position between master and slave, which will introduce a deviation at the slave position even if the master position is perfect. Thus, the guidance provides only a guideline for movements, but without any guarantee of accuracy and precision compared to the environment. Another problem is when introducing constraints in the operating space, transformations needs to be done to transform the operating space into the master space, and velocities needs accurate scaling. This introduces a need for static relations between the two spaces, or some real time update procedure for the space relation. On the other hand, we see that introducing a master based guidance can improve the performance of a system with significant drawbacks, and some external tracking system might solve the problems of the separate operating spaces.

Intentionally the maximum distance parameter is made to give the operator the ability to decide the strictness of the guidance, both prior to and during procedures. Tests reveals that it also is a powerful tool when tuning the gains of the system. This must be included in the total gain of the system when optimising the performance. Ideally the tuning of the gains should be done dynamically related to the choice of maximum distance and the guidance mode.

7.4 Conclusion

A teleoperated system with haptic guidance has been developed and presented, on the basis of already known literature and knowledge of teleoperated systems. Changes and additions have been done to suite the control schemes and algorithms to the available equipment, and to optimise the performance.

Various control schemes have been introduced, before the position position (PP) scheme is chosen to be implemented on the real system. Tests show that the control scheme works well making the slave manipulator to follow the movements of the master, but with some limitations due to the slave manipulator performance. The dynamics of the slave manipulator also causes the PP scheme not to be suited for generating force feedback. As a general basis for control and communication between two manipulators the PP control scheme works well.

The guidance system algorithms and control scheme has been developed and implemented on the teleoperated system. Tests show that the algorithms and the control scheme works as it is supposed to and gives satisfactory results when the conditions are good. Though, due to poor performance of the slave manipulator the performance of the guidance system is not sufficient to help and improve the precision and accuracy of the human operator, but is adequate to guide the operator when constraints are not requested to be strict.

An alternative implementation model are used in order to improve the performance of the guidance, and it appears to give satisfactory results. As a matter of fact, the use of this model gives a performance good enough to increase the precision and accuracy of the human operator, and meets the objectives of the work to a certain degree. It works well as an example of the power of using haptic guidance in teleoperated systems, but is difficult to use in real life tasks due to challenges in the relation between master and slave operating space.

The most significant challenges in this project origins with the performance of the slave manipulator. Some actions have been suggested on how to meet these problems, but in the end, it looks like the best solution is to exchange the manipulator with another which meets the needs in a better way.

7.4.1 Future Work

To make the force feedback work, it would be interesting to try another control scheme. It would be natural to try the force position scheme, due to its similarities to the position position scheme, but also because it is simple and yet effective. If hardware give the ability, the best option

would be the 4 Channel control scheme. The position position scheme could work if some external position tracking system is incorporated.

The external position tracking system could also improve the position tracking performance of the teleoperated system, and even improve the accuracy of slave position readings. Introducing a predictive control scheme as suggested could also help to improve the position tracking.

It is quite simple to implement this control and guidance scheme on other manipulators. At the Interventional Center a high speed and precision industrial robot are prepared for research purposes, and it would be natural to implement the system on this robot.

Finally the haptic guidance system needs some extension and improvement in functionality and usability, in order to meet the requirements of non technological operators in an intuitive way. New features which is natural to implement is the use of virtual objects as forbidden areas.

Bibliography

- Aliaga, I., Rubio, Á., and Sánchez, E. (2004). Experimental quantitative comparison of different control architectures for master–slave teleoperation. *IEEE Transactions on Control Systems Technology*, 12(1).
- Anderson, R. J. and Spong, M. W. (1988). Bilateral control of teleoperators with time delay. In *Proceedings of the 27th Conference on Decision and Control*.
- Burdea, G. C. (1996). *Force and Touch Feedback for Virtual Reality*. Wiley-Interscience Publication, John Wiley and Sons, Inc.
- Cepolina, F. and Michelini, R. C. (2004). Review of Robotic Fixtures for Minimally Invasive Surgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 1(1):43–63.
- Christiansson, G. A. V. (2007). Introduction to analysis and control in haptic teleoperation. Technical report, Delft Haptics Laboratory, Delft University of Technology.
- Clark, F. J. and Burgess, P. (1975). Slowly adapting receptors in the cat knee joint: Can they signal joint angle? *Journal of Neurophysiology*, 38:1448–1463.
- Computer Motion Inc., (1999). *AESOP user’s guide for AESOP 3000 and AESOP Hermes-ready systems*. Computer Motion Inc.

- Cornella, J. (2007). Kinematics and dynamics description of AESOP 3000DS. Technical report, The Interventional Centre, Rikshospitalet University Hospital.
- Cornella, J., Elle, O., Ali, W., and Samset, E. (2008a). Improving cartesian position accuracy of a telesurgical robot. Accepted in *IEEE International Symposium on Industrial Electronics*.
- Cornella, J., Nærum, E., Samset, E., and Elle, O. J. (2008b). Tools for improving telesurgical robots performance. Accepted in *4th ARISER Summer School Textbook*.
- Delft University of Technology, (2006). Like a snail through the intestinal canal. *ScienceDaily*. <http://www.sciencedaily.com/releases/2006/09/060922093741.htm>. Last Checked: 2008-05-31.
- Elle, O. J. (2004). *Sensor Control in Robotic Surgery*. PhD thesis, Norwegian University of Science and Technology.
- Flynn, A. M., Udayakumar, K. R., Barret, D. S., McLurkin, J., Franck, D. L., and Sheckman, A. N. (1998). Tomorrow's surgery: Micromotors and microrobots for minimally invasive procedures. *Minimally Invasive Therapy and Allied Technologies*, 7(1):343–352.
- Franklin, G. F. (1993). *Feedback Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Grigg, P. (1976). Responses of joint afferent neurons in cat medical articular nerve to active and passive movements of the knee. *Brain Research*, 118:482–485.
- Howe, R. and Kontarinis, D. (1992). Task performance with a dextrous teleoperated hand system. In *Proceedings of SPIE*, volume 1833, pages 199–207.
- IRCAD, France Telecom, and Computer Motion, (2001). "Operation Lindbergh", A World First Telesurgery: The Surgical Act Crosses The Atlantic! *Press Conference in Paris 2001*.

- Johansson, R. S. and Westling, G. (1988). Programmed and triggered actions to rapid load changes during precision grip. *Experimental Brain Research*, 271:1–15.
- Kettenbach, J., Kronreif, G., Figl, M., Fürst, M., Birkfellner, W., Hanel, R., and Bergmann, H. (2004). Robot-assisted biopsy using ultrasound guidance: initial results from in vitro tests. *Springer-Verlag*.
- Lanfranco, A. R., Castellanos, A. E., Desai, J. P., and Meyers, W. C. (January 2004). Robotic Surgery - A Current Perspective. *Annals of Surgery*, 239(1):14–21.
- Lawrence, D. A. (1993). Stability and transparency in bilateral teleoperation. In *IEEE Transactions on Robotics and Automation*, volume 9.
- Li, M., Ishii, M., and Taylor, R. H. (2007). Spatial Motion Constraints Using Virtual Fixtures Generated by Anatomy. *IEEE Transactions on Robotics and Automation*, 23(1).
- Marayongl, P., Li, M., Okamura, A. M., and Hager, G. D. (2003). Spatial Motion Constraints: Theory and Demonstrations for Robot Guidance Using Virtual Fixtures. In *Proceedings of the 2003 IEEE International Conference Of Robotics and Automation*.
- McLaughlin, M. L., Hespanha, J. P., and Sukhatme, G. S. (2001). *Touch in Virtual Environments: Haptics and the Design of Interactive Systems*. IMSC Press Multimedia Series.
- Morvan, T., Reimers, M., and Samset, E. (2008). High performance gpu-based proximity queries using distance fields. Accepted for publication in *Computer Graphics Forum*.
- Niemeyer, G. and Slotine, J. J. E. (1991). Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1).
- Niemeyer, G. and Slotine, J. J. E. (2004). Telemanipulation with time delays. *The International Journal of Robotics Research*, 23.

- Nuño, E. and Basañez, L. (2006). Haptic guidance with force feedback to assist teleoperation systems via high speed networks. *VDI Verlag*.
- Nygaard, A. (2007). High-level control system for remotely controlled surgical robots - haptic guidance of surgical robot. Norwegian University of Science and Technology.
- Nærum, E., Cornella, J., and Elle, O. (2008). Wavelet networks for estimation of coupled friction in robotic manipulators. In *IEEE International Conference on Robotics and Automation*.
- Ottermo, M. V. (2006). *Virtual Palpation Gripper*. PhD thesis, Norwegian University of Science and Technology.
- Petermann, J., Schierl, M., Pashimeh-Azar, A., Frohlich, J. J., Heeckt, P. F., and Gotzen, L. (2000). The caspar-system (computer assisted surgery planning and robotics) in reconstruction of the acl-1 year experience. *Elsevier Science B.V.*
- Rosenbaum, D. A. (1990). *Human Motor Control*. Academic Press.
- Rosenberg, L. (1995). How to assess the quality of force-feedback systems. in Morgan K., Satava R., Sieburg H. B., Mattheus R., Christensen J. P.(Eds.), *Medicine Meets Virtual Reality - Interactive Technology and the New Paradigm for Healthcare*, IOS Press.
- Rosenberg, L. B. (1993). The use of virtual fixtures to enhance operator performance in time delayed teleoperation. *Armstrong Laboratory*.
- Sayers, C. and Paul, R. (1994). An operator interface for teleprogramming employing synthetic fixtures. *Presence*, 3(4):309–320.
- Schurr, M. O., Heyn, S. P., Menz, W., and Buess, G. (1998). Endosystems - Future Perspectives for Endoluminal Therapy. *Minimally Invasive Therapy and Allied Technologies*, 7(1):37–42.
- SensAble Technologies (2005). *OpenHaptics TM Toolkit, Version 2.0, Programmer's Guide*. SensAble-Technologies.
- SensAble Technologies (2007). Sensable technologies. Web.

- Sherrick, C. and Craig, J. (1982). *Tactual Perception - A Source Book*. Cambridge University Press, New York.
- Shew, S. B., Ostlie, D. J., and Holcomb, G. W. (2003). Robotic telescopic assistance in pediatric laparoscopic surgery. *Pediatric Endosurgery and Innovative Techniques*, 7(4).
- Shimoga, K. (1992). Finger force and touch feedback issues in dextrous telemanipulation. In *Proceedings of NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration*.
- Shimoga, K. (1993). A survey of perceptual feedback issues in dextrous telemanipulation: Part ii. finger touch feedback. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 271–279.
- Skoglund, S. (1956). Anatomical and physiological studies of knee joint innervation in the cat. *Acta physiologica Scandinavica. Supplementum*, 36(124):1–101.
- Tavakoli, M., Patel, R. V., and Moallem, M. (2005). Haptic interaction in robot-assisted endoscopic surgery: a sensorized end-effector. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 1(2):53–63.
- Taylor, R. and Stoianovici, D. (2003). Medical robotics in computer-integrated surgery. *IEEE Transactions on Robotics and Automation*, 19(5):765–781.
- Yokokohji, Y. and Yoshikawa, T. (1994). Bilateral control for master-slave manipulators for ideal kinesthetic coupling - formulation and experiment. *IEEE Transactions on Robotics and Automation*, 10(5):605–620.

Appendix A

Manipulator Details

A.1 Phantom Omni Dynamics

To recall Equation (4.17), the dynamics of the Phantom Omni can be written on the form

$$\mathbf{M}(\theta)\ddot{\theta} + \mathbf{V}(\theta, \dot{\theta})\dot{\theta} + \mathbf{N}(\theta, \dot{\theta}) = \tau \quad (\text{A.1})$$

The matrices can be defined as follows, with the inertia matrix

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \quad (\text{A.2})$$

where the elements are

$$\begin{aligned} m_{1,1} &= k_1 + k_2 \cos(2\theta_2) + k_3 \cos(2\theta_3) + k_4 \cos(\theta_2) \sin(\theta_3) \\ m_{1,2} &= k_5 \sin(\theta_2) \\ m_{1,3} &= 0 \\ m_{2,1} &= k_5 \sin(\theta_2) \\ m_{2,2} &= k_6 \\ m_{2,3} &= -0.5k_4 \sin(\theta_2 - \theta_3) \\ m_{3,1} &= 0 \\ m_{3,2} &= -0.5k_4 \sin(\theta_2 - \theta_3) \\ m_{3,3} &= k_7 \end{aligned}$$

The Coriolis and centrifugal forces

$$\mathbf{V} = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,1} & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix} \quad (\text{A.3})$$

where the elements are

$$\begin{aligned} v_{1,1} &= -k_2\dot{\theta}_2\sin(2\theta_2) - k_3\dot{\theta}_3\sin(2\theta_3) - 0.5k_4\dot{\theta}_2\sin(\theta_2)\sin(\theta_3) \\ &\quad + 0.5k_4\dot{\theta}_3\cos(\theta_2)\cos(\theta_3) \\ v_{1,2} &= -k_2\dot{\theta}_1\sin(2\theta_2) - 0.5k_4\dot{\theta}_1\sin(\theta_2)\sin(\theta_3) + k_3\dot{\theta}_2\cos(\theta_2) \\ v_{1,3} &= -k_3\dot{\theta}_1\sin(2\theta_3) + 0.5k_4\dot{\theta}_1\cos(\theta_2)\cos(\theta_3) \\ v_{2,1} &= k_2\dot{\theta}_1\sin(2\theta_2) + 0.5k_4\dot{\theta}_1\sin(\theta_2)\sin(\theta_3) \\ v_{2,2} &= 0 \\ v_{2,3} &= 0.5k_4\dot{\theta}_3\cos(\theta_2 - \theta_3) \\ v_{3,1} &= k_3\dot{\theta}_1\sin(2\theta_3) + 0.5k_4\dot{\theta}_1\cos(\theta_2)\cos(\theta_3) \\ v_{3,2} &= -0.5k_4\dot{\theta}_2\cos(\theta_2 - \theta_3) \\ v_{3,3} &= 0 \end{aligned}$$

The gravity effect of the system

$$\mathbf{N} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \quad (\text{A.4})$$

where the elements are

$$\begin{aligned} n_1 &= 0 \\ n_2 &= k_8\cos(\theta_2) + k_{10}(\theta_2 - 0.5\pi) \\ n_3 &= k_9\sin(\theta_3) \end{aligned}$$

The variables k_1 to k_{10} are the inertia and gravity effect, mass and length of the links

$$\begin{aligned}
k_1 &= I_{Cyy} + 0.5I_{Ayy} + 0.5I_{Byy} + 0.5I_{Azz} + 0.5I_{Bzz} \\
&\quad + 0.125m_A l_2^2 + 0.125m_B l_1^2 + 0.5m_A l_1^2 + m_B l_3^2 \\
k_2 &= 0.5I_{Byy} - 0.5I_{Bzz} + 0.125m_B l_1^2 + 0.5m_A l_1^2 \\
k_3 &= 0.5I_{Ayy} - 0.5I_{Azz} - 0.125m_A l_2^2 \\
k_4 &= m_A l_1 l_2 \\
k_5 &= 0.5m_B l_1 l_3 \\
k_6 &= I_{Bxx} + m_A l_1^2 + 0.25m_B l_1^2 \\
k_7 &= I_{Axx} + 0.25m_A l_2^2 \\
k_8 &= 0.5m_B l_1 g + m_A l_1 g \\
k_9 &= 0.5m_A l_2 g \\
k_{10} &= K_2
\end{aligned}$$

The manipulator is divided into different rigid segments as the links and the base. I_{Mnn} is the moment of inertia of segment M in the direction n , m_M denotes the mass of segment M , while l_l represent the length of link l . K_2 is the gain of a gravity compensating spring added to joint 2. By experimental study these variables were found to be

$$\begin{aligned}
k_1 &= 0.00179819707554751 \\
k_2 &= 0.000864793119787878 \\
k_3 &= 0.000486674040957256 \\
k_4 &= 0.00276612067958414 \\
k_5 &= 0.000308649491069651 \\
k_6 &= 0.00252639617221043 \\
k_7 &= 0.000652944405770658 \\
k_8 &= 0.164158326503058 \\
k_9 &= 0.0940502380783103 \\
k_{10} &= 0.117294768011206
\end{aligned}$$

Appendix B

Haptic Guidance Source Code

B.1 Initiating the System

```
#if defined(WIN32)
# include <conio.h>
#else
# include "conio.h"
#endif

#include <string>
#include <HD/hd.h>
#include <HDU/hduError.h>
#include <HDU/hduVector.h>
#include <HDU/hduMatrix.h>
#include <process.h>
#include <windows.h>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <math.h>
```

```
#include <ctime>

#include "Aesop/a3000.h"

#define dirX 0
#define dirY 1
#define dirZ 2

//WriteSysParam alternatives
#define HD_POSITION 0
#define HD_VELOCITY 1
#define HD_FORCE 2
#define HD_BUTTON_1 3
#define R_POSITION 4
#define R_INIT_POSITION 5
#define R_VELOCITY 6

//Guidance alternatives
#define NONE_GUIDANCE 0
#define PLANE_GUIDANCE 1
#define LINE_GUIDANCE 2
#define POINT_GUIDANCE 3

// System variables
struct DeviceDisplayState
{
int writeState;
HDdouble position[3];
HDdouble HinitPos[3];
HDdouble velocity[3];
HDdouble force[3];
HDdouble Rposition[3];
HDdouble RinitPosition[3];
HDdouble Rvelocity[3];
float RjointVel[4];
float RjointAng[7];
```

```
float RposMeasure[6];
HDboolean button1State;
};

//Parameters for Guidance
struct GuidanceParameter
{
int guidanceType;
int LastGuidanceType;
hduVector3D <HDdouble> lineVector;
hduVector3D <HDdouble> vdLast;
hduVector3D <HDdouble> PerrLast;
hduVector3D <HDdouble> Rpos;
hduVector3D <HDdouble> Hpos;
hduVector3D <HDdouble> p1;
hduVector3D <HDdouble> p2;
hduVector3D <HDdouble> p3;
HDdouble MaxForce;
hduVector3D <HDdouble> Km;
hduVector3D <HDdouble> Cm;
hduVector3D <HDdouble> Ks;
hduVector3D <HDdouble> Cs;
hduVector3D <HDdouble> Kg;
hduVector3D <HDdouble> Cg;
HDdouble MaxDistance;
double TimeInit, TimeLast, TimeNow, TimeFin;
};

//*** Robot Parameters ***
int DS_PORT = 3;
const double L2= 384.4879;
const double L31= 76.7056;
const double L32= 304.8001;
double L3, qe;
```

```
/** Global variables **/

//Secure struct object, use WriteSysParam / ReadSysParam to access
DeviceDisplayState sysParam;

GuidanceParameter guideParam;
HDSchedulerHandle gCallbackHandle = 0;
double tini, tfin, tr, lastTr; //times
boolean run;
boolean first;
boolean button;
boolean Ron;
int testNum =0;
int sign=1;

/** ReadKeyHit parameters **/
int planeCount;
boolean writeFirst;
boolean lineOK;
boolean planeOK;
boolean planeNext;
boolean buttonLast;
boolean CheckGuidParam;
hduVector3D <HDdouble> point1;
hduVector3D <HDdouble> point2;

/** Functions init **/
void mainloop(void *pUserData);
void CalculateForce(void *pUserData);
void AesopCartToJointVel(void *pUserData);
void AesopForwardKinematics(void *pUserData);
void writeFile(void *pUserData);
void readKeyHit(void *pUserData);
HDCallbackCode HDCALLBACK HapticLoop(void *pUserData);
HDCallbackCode HDCALLBACK WriteSysParam(void *pUserData);
HDCallbackCode HDCALLBACK ReadSysParam(void *pUserData);
```

```
/* Main
 * - Init AESOP/Omni and global parameters
 * - Stop AESOP/Omni after mainloop is quit
 */
int main()
{
  sysParam.force[dirX]=0;
  sysParam.force[dirY]=0;
  sysParam.force[dirZ]=0;

  //Default Guidance parameters
  guideParam.guidanceType=NONE_GUIDANCE;
  guideParam.LastGuidanceType=guideParam.guidanceType;
  guideParam.MaxDistance=10;
  guideParam.MaxForce=3;
  guideParam.lineVector.set(0,0,0);
  guideParam.vdLast.set(0,0,0);
  guideParam.PerrLast.set(0,0,0);
  guideParam.p1.set(0,0,0);
  guideParam.p2.set(0,0,0);
  guideParam.p3.set(0,0,0);

  //Master spring/damper gain K/C
  guideParam.Km.set(.01, .01, .0);
  guideParam.Cm.set(.001, .001, .005);

  //Slave spring/damper gain K/C
  guideParam.Ks.set(1, 1, 1);
  guideParam.Cs.set(1, 1, .6);

  //Guidance spring/damper gain K/C
  guideParam.Kg.set(.091, .081, .2);
```

```
guideParam.Cg.set(.005, .005, .001);

run = true;
first=true;
Ron=true;
button=false;
buttonLast=button;
lineOK=false;
planeOK=false;
planeNext=true;
CheckGuidParam=false;
writeFirst=true;
point1.set(0,0,0);
point2.set(0,0,0);
planeCount = 1;

//create struct object to be used all but for Omni manipulations
DeviceDisplayState temp;
temp.button1State=0;
temp.Rvelocity[0]=0;
temp.Rvelocity[1]=0;
temp.Rvelocity[2]=0;

/** Init Robot **
if (!Initialize(DS_PORT)){
cout << "Working to open AESOP Communications.\n\n" << endl;
while(!Initialize(DS_PORT)){for(int i=0;i<1000000000;i++){}}
//exit(1);
}
JtPosReq(temp.RjointAng);
qe=temp.RjointAng[6];
L3=L31+L32*cos(qe);

/** Init Haptic **
HHD hHD = hdInitDevice(HD_DEFAULT_DEVICE);
```



```

}
```

B.2 Main Control Loop

```

/* Mainloop runs until Q is hit
 * - updates AESOP and Omni parameters, runs guidance as long as
 * button on Omni is pushed
 * Control buttons
 * Q - Quit
 * 0 - NONE_GUIDANCE
 * 1 - PLANE_GUIDANCE
 * 2 - LINE_GUIDANCE
 * 3 - POINT_GUIDANCE
 * +/- Increase/decrease max distance
 * o - Switches AESOP ON/OFF
 */
void mainloop(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);
guideParam.TimeInit = GetTickCount();
tini = guideParam.TimeInit;

while(run)
{
//Read values from haptic
hdScheduleSynchronous(ReadSysParam, temp,
HD_DEFAULT_SCHEDULER_PRIORITY);
button=temp->button1State;

//Read values from robot
JtPosReq(temp->RjointAng);
temp->RjointAng[0]=temp->RjointAng[0]*(float)25.4;

AesopForwardKinematics(temp); //convert joint to position space
```

```
readKeyHit(temp); // checks for keyboard hits

if(button)
{
if(first)
{
//Init pos of robot
memcpy(&temp->RinitPosition, &temp->Rposition, sizeof(temp->Rposition));

//Init pos of haptic
memcpy(&temp->HinitPos, &temp->position, sizeof(temp->position));
writeFile(temp);
first=false;
}
else if(!first)
{
CalculateForce(temp);
AesopCartToJointVel(temp);
if(Ron)MoveAllJointsContinous(temp->RjointVel);
}
writeFile(temp);
}
else if(!button && buttonLast)
{
// Give zero force to haptic
temp->force[0]=0;
temp->force[1]=0;
temp->force[2]=0;
temp->writeState=HD_FORCE;
WriteSysParam(temp);

// Stop robot
temp->RjointVel[0]=0;
temp->RjointVel[1]=0;
temp->RjointVel[2]=0;
temp->RjointVel[3]=0;
```

```

MoveAllJointsContinuous(temp->RjointVel);

first=true;
}
buttonLast=button;
guideParam.LastGuidanceType=guideParam.guidanceType;
}
}

```

B.3 Force / Velocity Generation

```

/* Calculates force to put onto the Omni
 * - calculations in AESOP coordinate space
 * - output in omni coordinate space
 */
void CalculateForce(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);

double maxDistance = guideParam.MaxDistance;//mm
double maxForce = 3.3;//Newton
int GuidanceType = guideParam.guidanceType;

HDdouble vLM, A, B, C, D, d;
hduVector3D <HDdouble> forceTemp;
hduVector3D <HDdouble> forceLast;
hduVector3D <HDdouble> VelTemp;
hduVector3D <HDdouble> Rvel;
hduVector3D <HDdouble> Rpos;
hduVector3D <HDdouble> Hpos;
hduVector3D <HDdouble> RposInit;
hduVector3D <HDdouble> HposInit;
hduVector3D <HDdouble> Perr;
hduVector3D <HDdouble> dPerr;
hduVector3D <HDdouble> vr;

```

```

hduVector3D <HDdouble> v;
hduVector3D <HDdouble> vd;
hduVector3D <HDdouble> dvd; //differentiated vd
hduVector3D <HDdouble> pd;
hduVector3D <HDdouble> pi;
hduVector3D <HDdouble> pr;
hduVector3D <HDdouble> ph;
hduVector3D <HDdouble> p1;
hduVector3D <HDdouble> p2;
hduVector3D <HDdouble> p3;
hduVector3D <HDdouble> v1;
hduVector3D <HDdouble> v2;
hduVector3D <HDdouble> n;
hduVector3D <HDdouble> nn;

//Update time
guideParam.TimeFin = GetTickCount();
guideParam.TimeNow = (guideParam.TimeFin - guideParam.TimeInit)/1000;

//Set pos vectors
Rpos.set(temp->Rposition[0], temp->Rposition[1], temp->Rposition[2]);
RposInit.set(temp->RinitPosition[0], temp->RinitPosition[1],
temp->RinitPosition[2]);
Hpos.set(temp->position[2], temp->position[0], temp->position[1]);
//convert to aesop coord.
HposInit.set(temp->HinitPos[2], temp->HinitPos[0], temp->HinitPos[1]);
//convert to aesop coord.

//Convert to relative positions
Rpos= Rpos - RposInit;
Hpos= Hpos - HposInit;

p1.set(0,0,0);
p2.set(0,0,0);
p3.set(0,0,0);
VelTemp.set(0,0,0);

```

```

forceTemp.set(0,0,0);
pi.set(0,0,0);
vd.set(0,0,0);
Rvel.set(0,0,0);
forceLast.set(temp->force[0], temp->force[1], temp->force[2]);

//**** Calculates deviation vector vd ****
switch(GuidanceType)
{
case 0: // NONE
{
break;
}
case 1: // Plane guidance
{
if(planeOK)
{
p1=guideParam.p1;
p2=guideParam.p2;
p3=guideParam.p3;
v1=p1-p2;
v2=p1-p3;
n=crossProduct(v1,v2);
nn=normalize(n);
A=n[0];
B=n[1];
C=n[2];
D= -(n[0]*v1[0]+n[1]*v1[1]+n[2]*v1[2]);
d= (A*Rpos[0]+B*Rpos[1]+C*Rpos[2]+D)/sqrt(A*A+B*B+C*C);
vd=-nn*d;
for(int i=0;i<3;i++)
{
if(vd[i]<0)vd[i]=0;
}
}
break;
}
}

```

```

}
case 2: // Line guidance
{
if(lineOK)
{
v = guideParam.lineVector;
// Calculate deviation vd from vectorline v
//v.set(1,1,0);
v.normalize();
vr = Rpos - pi;
vLM = dotProduct(v,vr) / v.magnitude();
pd = pi + vLM * v;
vd = pd - Rpos;
}
break;
}
case 3: // Point guidance
{
vd = pi-Rpos;

break;
}
default:
{
printf("No guidance type chosen!/n");
break;
}
}

//**** Calculates Pos error and diff Pos error ****
Perr = Hpos - Rpos;
dPerr = (Perr - guideParam.PerrLast) / (guideParam.TimeNow -
guideParam.TimeLast);
dvd = (vd - guideParam.vdLast) / (guideParam.TimeNow -
guideParam.TimeLast);

```

```

//**** Calculates Omni Force ****
forceTemp = -(Perr*guideParam.Km + dPerr*guideParam.Cm) +
(vd*guideParam.Kg + dvd*guideParam.Cg);
for(int i=0;i<3;i++)
{
//limits force output
if(forceTemp[i]>3)
{
forceTemp[i]=3;
}
else if(forceTemp[i]<-3)
{
forceTemp[i]=-3;
}
}
// Transfer to Omni coordinate space
temp->force[0] = forceTemp[1];
temp->force[1] = forceTemp[2];
temp->force[2] = forceTemp[0];

//**** Calculate Aesop Velocity ****
VelTemp = Perr*guideParam.Ks + dPerr*guideParam.Cs;
temp->Rvelocity[0] = VelTemp[0];
temp->Rvelocity[1] = VelTemp[1];
temp->Rvelocity[2] = VelTemp[2];

// Filter and Set force onto Omni
for(int i=1; i<3; i++)
{
double limit = .3;
if(forceTemp[i]-forceLast[i]<-limit)forceTemp[i]=forceLast[i]-limit;
else if(forceTemp[i]-forceLast[i]>limit)forceTemp[i]=forceLast[i]+limit;
}

```



```

}
hdScheduleSynchronous(WriteSysParam, temp,
HD_DEFAULT_SCHEDULER_PRIORITY);

//For log purposes
guideParam.Rpos = Rpos;
guideParam.Hpos = Hpos;

//Update dvd parameters
memcpy(&guideParam.PerrLast, &Perr, sizeof(Perr));
memcpy(&guideParam.vdLast, &vd, sizeof(vd));
memcpy(&guideParam.TimeLast, &guideParam.TimeNow,
sizeof(guideParam.TimeNow));
}

```

B.4 Slave Control

```

/** All calculations in AESOP coordinate space,
 * Joint space: q1[mm]/q2[rad]/q3[rad], dq1[mm/s]/dq2[rad/s]/dq3[rad/s]
 * Cartesian space: dX[mm/s]
 */
void AesopCartToJointVel(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);

//Parameters of the AESOP, joint angles, arm lengths, jacobian elements
HDdouble q1, q2, q3, dq1, dq2, dq3, dx, dy, dz, invDetJ;
HDdouble j11, j12, j13, j21, j22, j23, j31, j32, j33;
HDdouble c11, c12, c13, c21, c22, c23, c31, c32, c33;

q1= (HDdouble) temp->RjointAng[0];
q2= (HDdouble) temp->RjointAng[1];
q3= (HDdouble) temp->RjointAng[2];

double g[3];

```

```

g[0]=1;
g[1]=1;
g[2]=1;
dx= temp->Rvelocity[dirX]*g[0];
dy= temp->Rvelocity[dirY]*g[1];
dz= temp->Rvelocity[dirZ]*g[2];

//Jacobian
j11=0;
j12=-L2*sin(q2)-L3*sin(q2+q3);
j13=-L3*sin(q2+q3);
j21=0;
j22=L3*cos(q2+q3)+L2*cos(q2);
j23=L3*cos(q2+q3);
j31=1;
j32=0;
j33=0;

//****Calculate Inverse of Jacobian****
//Determinant of Jacobian
invDetJ=1/((j11*j22*j33 + j12*j23*j31 + j13*j21*j32) -
(j31*j22*j13 + j32*j23*j11 + j33*j21*j12));

//Cofactor elements of Jacobian
c11= (j22*j33 - j23*j32);
c12= -(j21*j33 - j23*j31);
c13= (j21*j32 - j22*j31);
c21= -(j12*j33 - j13*j32);
c22= (j11*j33 - j13*j31);
c23= -(j11*j32 - j12*j31);
c31= (j12*j23 - j13*j22);
c32= -(j11*j23 - j13*j21);
c33= (j11*j22 - j12*j21);

//**** dTheta = invJ*dX ****

```

```

dq1= invDetJ*(c11*dx + c21*dy + c31*dz);
dq2= invDetJ*(c12*dx + c22*dy + c32*dz);
dq3= invDetJ*(c13*dx + c23*dy + c33*dz);

//Check velocity limits
if(dq1>30)dq1=30;
else if(dq1<-30)dq1=-30;

if(dq2>0.3)dq2=0.3;
else if(dq2<-0.3)dq2=-0.3;

if(dq3>1)dq3=1;
else if(dq3<-1)dq3=-1;

//convert from mm/s to inch/s
dq1=dq1/2.54;

temp->RjointVel[0]=(float)dq1;
temp->RjointVel[1]=(float)dq2;
temp->RjointVel[2]=(float)dq3;
temp->RjointVel[3]= 0;

}

/* Calculates Cartesian position of the AESOP
* based on its joint angles
* - result is given in AESOP coordinate frame
*/
void AesopForwardKinematics(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);
double q1, q2, q3, x, y, z;

//Assign values
q1=temp->RjointAng[0];

```

```

q2=temp->RjointAng[1];
q3=temp->RjointAng[2];

//Forward kinematics
x= L2*cos(q2) + L3*cos(q2+q3);
y= L2*sin(q2) + L3*sin(q2+q3);
z= L32*sin(qe) + q1;

//Put transformed values back
temp->Rposition[dirX]= x;
temp->Rposition[dirY]= y;
temp->Rposition[dirZ]= z;

}

```

B.5 Master Control

```

/* HapticLoop
 * - Runs as separate thread
 * - Read status of Omni
 * - Set forces to Omni
 */
HDCallbackCode HDCALLBACK HapticLoop(void *pUserData)
{
//Variables
DeviceDisplayState *SysParamTemp = static_cast<DeviceDisplayState
*>(pUserData);
int nButtons=0;
HDdouble jointAng[3];

HHD hHD = hdGetCurrentDevice();
hdBeginFrame(hHD);

//Read values from haptic
hdGetDoublev (HD_CURRENT_JOINT_ANGLES, (HDdouble *)&jointAng);

```

```

hdGetDoublev (HD_CURRENT_VELOCITY, SysParamTemp->velocity);
hdGetDoublev (HD_CURRENT_POSITION, SysParamTemp->position);
hdGetIntegerv (HD_CURRENT_BUTTONS, &nButtons);
SysParamTemp->button1State = (nButtons & HD_DEVICE_BUTTON_1)
? TRUE : FALSE;

//Set force to haptic
hdSetDoublev(HD_CURRENT_FORCE, SysParamTemp->force);

hdEndFrame(hHD);

return HD_CALLBACK_CONTINUE;
}

/* ReadSysParameters
 * - Synchronised access of Omni parameters
 * - Use this function to read the variables of the Omni
 */
HDCallbackCode HDCALLBACK ReadSysParam(void *pUserData)
{
DeviceDisplayState *SysParamTemp = static_cast<DeviceDisplayState
*>(pUserData);

memcpy(&SysParamTemp->force, &sysParam.force,
sizeof(sysParam.force));

memcpy(&SysParamTemp->position, &sysParam.position,
sizeof(sysParam.position));

memcpy(&SysParamTemp->velocity, &sysParam.velocity,
sizeof(sysParam.velocity));

memcpy(&SysParamTemp->button1State, &sysParam.button1State,
sizeof(sysParam.button1State));

return HD_CALLBACK_DONE;

```

```
}

/* WriteSysParameters
 * - Synchronised access of Omni parameters
 * - Use this function to set the variables of the Omni
 */
HDCallbackCode HDCALLBACK WriteSysParam(void *pUserData)
{
DeviceDisplayState* SysParamTemp= (DeviceDisplayState *) pUserData;
SysParamTemp->writeState=HD_FORCE;
switch(SysParamTemp->writeState)
{
case 0: // Writes to private HDposition variable
{
memcpy(&sysParam.position,&SysParamTemp->position,
sizeof(SysParamTemp->position));

break;
}
case 1: // Writes to private HDvelocity variable
{
memcpy(&sysParam.velocity,&SysParamTemp->velocity,
sizeof(SysParamTemp->velocity));

break;
}
case 2: // Writes to private HDforce variable
{
memcpy(&sysParam.force,&SysParamTemp->force,
sizeof(SysParamTemp->force));

break;
}
case 3: // Writes to private HDbutton1State variable
{
memcpy(&sysParam.button1State,&SysParamTemp->button1State,
```

```
sizeof(SysParamTemp->button1State));

break;
}
default:
{
break;
}
}

return HD_CALLBACK_DONE;
}
```

B.6 Key Hit Function

```
/* ReadKeyHit
 * - reads when keyboard buttons are pushed
 * Control buttons
 * Q - Quit
 * 0 - NONE_GUIDANCE
 * 1 - PLANE_GUIDANCE
 * 2 - LINE_GUIDANCE
 * 3 - POINT_GUIDANCE
 * +/- Increase/decrease max distance
 */
void readKeyHit(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);
button=temp->button1State;
int keypress;
int b=0; //Used for changing axis when tuning gains

// Check if buttons are pushed
if(_kbhit())
```

```
{
keypress = _getch();
if(keypress == 'Q' || keypress == 'q')
{
run = false;
}
else if(keypress == '0')
{
printf("\nGuidance type: None\n");
guideParam.guidanceType=NONE_GUIDANCE;
}
else if(keypress == '1')
{
printf("\nGuidance type: Plane\n");
guideParam.guidanceType=PLANE_GUIDANCE;
}
else if(keypress == '2')
{
printf("\nGuidance type: Line\n");
guideParam.guidanceType=LINE_GUIDANCE;
}
else if(keypress == '3')
{
printf("\nGuidance type: Point\n");
guideParam.guidanceType=POINT_GUIDANCE;
}
else if(keypress == '+')
{
guideParam.Cm[b] = guideParam.Cm[b] + .01;
printf("Cm[%i]: %f\n",b ,guideParam.Cm[b]);
}
else if(keypress == '-')
{
guideParam.Cm[b] = guideParam.Cm[b] - .01;
if(guideParam.Cm[b]<0)guideParam.Cm[b]=0;
printf("Cm[%i]: %f\n",b ,guideParam.Cm[b]);
}
```



```
}
else if(keypress == '9')
{
guideParam.Km[b] = guideParam.Km[b] + .01;
printf("Km[%i]: %f\n",b ,guideParam.Km[b]);
}
else if(keypress == '6')
{
guideParam.Km[b] = guideParam.Km[b] - .01;
if(guideParam.Km[b]<0)guideParam.Km[b]=0;
printf("Km[%i]: %f\n",b ,guideParam.Km[b]);
}
else if(keypress == 'o')
{
if(Ron)
{
Ron=false;
printf("Robot: OFF\n");
}
else
{
Ron=true;
printf("Robot: ON\n");
}
}
}
// For creating the line with which to guide along
if(guideParam.guidanceType!=guideParam.LastGuidanceType)
{
CheckGuidParam=true;
}

if(CheckGuidParam)
{
if(guideParam.guidanceType==LINE_GUIDANCE)
{
```

```
planeOK=false;
if(writeFirst)
{
printf("\nPush button and drag haptic device between \n
two end points creating a line! \n");
writeFirst=false;
guideParam.lineVector.set(0,0,0);
}
if(button && !buttonLast)
{
point1.set(temp->Rposition[0], temp->Rposition[1], temp->Rposition[2]);
printf("1. OK!\n");

}
if(!button && buttonLast)
{
point2.set(temp->Rposition[0], temp->Rposition[1], temp->Rposition[2]);
lineOK=true;
CheckGuidParam=false;
writeFirst=true;
guideParam.lineVector=point2-point1;
guideParam.lineVector.normalize();
printf("2. OK, line is set!\n");
}
}
else if(guideParam.guidanceType==PLANE_GUIDANCE)
{
lineOK=false;
if(writeFirst)
{
printf("\nWe're now going to creat a plane out of \nthree points.\n");
printf("\nPush button and drag haptic device to a \n
point and then let go of the button! \n");
writeFirst=false;
guideParam.p1.set(0,0,0);
guideParam.p2.set(0,0,0);
```

```
guideParam.p3.set(0,0,0);
}
if(button && !buttonLast)
{
planeNext=true;
}
if(!button && buttonLast && planeNext)
{
if(planeCount==1)
{
guideParam.p1.set(temp->Rposition[0], temp->Rposition[1],
temp->Rposition[2]);
printf("1. OK, next point! \n");
planeCount++;
planeNext=false;
}
else if(planeCount==2)
{
guideParam.p2.set(temp->Rposition[0], temp->Rposition[1],
temp->Rposition[2]);
printf("2. OK, and the last point! \n");
planeCount++;
planeNext=false;
}
else if(planeCount==3)
{
guideParam.p3.set(temp->Rposition[0], temp->Rposition[1],
temp->Rposition[2]);
printf("3. Thanks, the plane is set! \n");
planeCount=1;
planeOK=true;
CheckGuidParam=false;
writeFirst=true;
planeNext=true;

guideParam.p3=guideParam.p3-guideParam.p1;
```

```
guideParam.p2=guideParam.p2-guideParam.p1;
guideParam.p1=guideParam.p1-guideParam.p1;

CheckGuidParam=false;
}
}
}
else if(guideParam.guidanceType==NONE_GUIDANCE)
{
CheckGuidParam=false;
planeOK=false;
lineOK=false;
}
else if(guideParam.guidanceType==POINT_GUIDANCE)
{
CheckGuidParam=false;
planeOK=false;
lineOK=false;
}
}
}

/* For log/debug purposes
* - writes central variables to log file at Results/res
* %i.m (%i is number of test)
*/
void writeFile(void *pUserData)
{
DeviceDisplayState *temp = static_cast<DeviceDisplayState *>(pUserData);

if(first)
{
tini= GetTickCount();
testNum=testNum+1;
}
```

```

char ResFile[200];
sprintf(ResFile, "Results/res%i.m", testNum);
FILE *stream1;
stream1 = fopen(ResFile, "a");
fprintf(stream1, "Guidance type: %i \n", guideParam.guidanceType);
if(guideParam.lineVector[0] != 0) fprintf(stream1, "Line Vector:
%f %f %f \n", guideParam.lineVector[0], guideParam.lineVector[1],
guideParam.lineVector[2]);
if(guideParam.p2[0] != 0)
{
fprintf(stream1, "Point 1: %f %f %f \n", guideParam.p1[0], guideParam.p1[1],
guideParam.p1[2]);

fprintf(stream1, "Point 2: %f %f %f \n", guideParam.p2[0], guideParam.p2[1],
guideParam.p2[2]);

fprintf(stream1, "Point 3: %f %f %f \n", guideParam.p3[0], guideParam.p3[1],
guideParam.p3[2]);
}
fprintf(stream1, "\nMaster K: %f %f %f C: %f %f %f \n", guideParam.Km[0],
guideParam.Km[1], guideParam.Km[2], guideParam.Cm[0], guideParam.Cm[1],
guideParam.Cm[2]);

fprintf(stream1, "Slave K: %f %f %f C: %f %f %f \n", guideParam.Ks[0],
guideParam.Ks[1], guideParam.Ks[2], guideParam.Cs[0], guideParam.Cs[1],
guideParam.Cs[2]);

fprintf(stream1, "Guide K: %f %f %f C: %f %f %f \n", guideParam.Kg[0],
guideParam.Kg[1], guideParam.Kg[2], guideParam.Cg[0], guideParam.Cg[1],
guideParam.Cg[2]);

fprintf(stream1, "\nContent: tr, Rpos, Hpos, Hforce\n\n");
fclose(stream1);
first=false;
}

```

```
//Start time at zero
tfin = GetTickCount();
tr=(tfin-tini)/1000;
lastTr=tr;

char ResFile[200];
sprintf(ResFile, "Results/res%i.m",testNum);
FILE *stream1;
stream1 = fopen(ResFile,"a");
fprintf(stream1,"%f %f %f %f %f %f %f %f %f\n",tr, guideParam.Rpos[0],
guideParam.Rpos[1], guideParam.Rpos[2], guideParam.Hpos[0],
guideParam.Hpos[1], guideParam.Hpos[2], temp->force[2], temp->force[0],
temp->force[1]);

fclose(stream1);
}
```