Robot-held camera platform for medical applications

Christopher Tvede-Möller

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6070
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

In this thesis a robot-held platform is developed. The final goal of the platform is to follow a potential surgeon’s head during operation while cameras mounted on the platform film the ongoing operation from the perspective of the surgeon. In this thesis the original concept developed by Emma Andersson and Anna Wikström is developed further by extending the design to allow placement of sensors and by implementing the control algorithms necessary for the platform to follow the potential surgeon’s movements. The implementation uses inverse velocity kinematics, PD control and mid-ranging control to achieve a proof of concept. The final implementation is able to follow movements, but some work is still necessary to ensure better following of orientation and to ensure that the implementation is safe to introduce to a medical environment.

Acknowledgements

This master's thesis was conducted at the faculty of automatic control at Lunds University in cooperation with Lunds university hospital and Cognibotics. I would like to thank my supervisor Anders Robertsson, cofounder of Cognibotics and professor at automatic control, and my examiner Charlotta Johnsson, professor at automatic control, for help with acquiring resources needed as well as valuable discussions and guidance for solving problems that arose. I would also like to thank Kiet Tran, Physician at the Paediatrics department at Lund university hospital, for valuable information about a surgeon's work procedure. Lastly I would like to thank Ph.D. student Martin Karlson for counsel about control and robotics.

Contents

1
Introduction

Surgeons today mainly communicate the happenings transpired during a completed surgery, to other surgeons and surgeons in training, verbally; possibly with some illustrative help from a whiteboard or blackboard. In order to better document the happenings during surgery, it would be desirable to record image information. This is done to some degree with overhead cameras, but the surgeons tend to be in the way during large parts of the operation.

Conceptually, cameras could be placed on top of a surgeon's head in order to remedy this, but the added weight might interfere with the surgeon's movements, and cause an undesirable load on the neck. This could be alleviated by placing the camera on a movable platform around the head held up by some mechanical stand, but having the surgeon move it manually or having someone else move it would interfere with the surgery.

In recent years robotics has developed far enough that robots can work in the same environment as humans on the same tasks or related tasks with little risk of injury to human or robot. With this in mind a robot arm carrying the proposed platform might be the solution.

1.1 Previous work

The concept of this robot-held camera platform was developed by Emma Andersson and Anna Wikström.[Andersson and Wikström, 2017] The concept focuses on the design of the platform and the placement of the cameras, but has no implementation of robotics; it only specifies that a robot arm should be used. This thesis seeks to implement the robotics necessary for a satisfactory integration with surgeons. The robot used in the thesis is the IRB 14000, also known as YuMi, developed by ABB Ltd, but any robot arm which can carry a large enough payload can be used.

Controlling the robot requires kinematics, control theory and a way of sending references to the robot. Robot kinematics and control theory are both preexisting and the necessary background will be summarised in chapter 2.4 and 2.3 respectively. Sending references to the robot can be done with the Externally Guided Motion Research Interface summarised in chapter 2.5.

Detecting the motion of the surgeon requires sensors, which in turn need to communicate with the controller of the robot. VL6180 time of flight laser sensors are used; the placement of the sensors required to get the necessary information is discussed in chapter 2.7, while the protocol used for communication with these sensors is discussed in chapter 2.6.

1.2 End goal

Implementing the control algorithms, and integrating the necessary supporting infrastructure (such as sensors, wiring and computational chips), which results in a satisfactory tracking of the surgeons movements is the ultimate goal of this thesis. This is divided into subgoals: setting up communication between the robot, the sensors and a Raspberry Pi which handles calculations and communication; designing a control algorithm which results in satisfactory tracking of translational movements; and lastly designing a control algorithm which results in satisfactory tracking of both translational movements and rotational movements.

1.3 Outline

Chapter 2: This chapter introduces the relevant theory and technical specifications necessary to implement the control and communication.
Chapter 3: This chapter walks through the different steps carried out during implementation.
Chapter 4: This chapter presents and discusses the results gained from testing.
Chapter 5: This chapter reviews how far the implementation has succeeded, and outlines a few possible areas of future work.

2

Background

This chapter summarises the theory and technical specifications that are necessary for the implementations presented in chapter 3.

2.1 Matrix Theory

Below a computationally efficient method to solve systems of equations is introduced. This method is then expanded upon so it can be used to solve matrix equations without the need of the matrix inverse. It is assumed that the reader is familiar with the concepts of systems of equations, column and row vectors, matrices, the relations between these and how the basic operations work on these constructs.

LU factorisation

If $\boldsymbol{A}$ is an arbitrary quadratic $n \times n$ matrix with upper left $k \times k$ submatrices $\boldsymbol{A}_k$ then matrix theory [Böiers, 2010] tells us that if $\det \boldsymbol{A}_k \neq 0$ for all $k \leq n$ there is a unique factorisation $\boldsymbol{A} = \boldsymbol{LU}$ with $\boldsymbol{L}$ a lower triangular matrix and $\boldsymbol{U}$ an upper triangular matrix.

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} d_1 & u_{12} & \cdots & u_{1n} \\ 0 & d_2 & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix} \tag{2.1}$$

A system of linear equations can be rewritten with the matrices in 2.1 as

$$\boldsymbol{Ax} = \boldsymbol{b} \Leftrightarrow \boldsymbol{LUx} = \boldsymbol{b} \Leftrightarrow \begin{cases} \boldsymbol{Ly} = \boldsymbol{b} \\ \boldsymbol{Ux} = \boldsymbol{y} \end{cases} \tag{2.2}$$

with $\boldsymbol{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$, $\boldsymbol{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]^T$ and $\boldsymbol{b} = [b_1 \quad b_2 \quad \cdots \quad b_n]^T$ as column vectors. This system of equations has the solution

$$\begin{cases} y_1 & = b_1 \\ y_2 & = b_2 - l_{21}y_1 \\ y_3 & = b_3 - l_{31}y_1 - l_{32}y_2 \\ & \vdots \\ y_n & = b_n - l_{n1}y_1 - l_{n2}y_2 - l_{n3}y_3 - \cdots - l_{n(n-1)}y_{(n-1)} \\ x_n & = \frac{y_n}{d_n} \\ x_{(n-1)} & = \frac{y_{(n-1)} - u_{(n-1)n}x_n}{d_{(n-1)}} \\ x_{(n-2)} & = \frac{y_{(n-2)} - u_{(n-2)n}x_n - u_{(n-2)(n-1)}x_{(n-1)}}{d_{(n-2)}} \\ & \vdots \\ x_1 & = \frac{y_1 - u_{1n}x_n - u_{1(n-1)}x_{(n-1)} - u_{1(n-2)}x_{(n-2)} - \cdots - u_{12}x_2}{d_1} \end{cases} \tag{2.3}$$

.

Introducing the $n \times m$ matrices

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_m\end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_m\end{bmatrix}$$

, where $\boldsymbol{x_i}$ and $\boldsymbol{b_i}$ are column vectors, the matrix equation $\boldsymbol{AX} = \boldsymbol{B}$ can be divided into m new matrix equations

$$\begin{cases} \boldsymbol{Ax}_1 = \boldsymbol{b}_1 \\ \boldsymbol{Ax}_2 = \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{Ax}_m = \boldsymbol{b}_m \end{cases} \tag{2.4}$$

, which all can be rewritten to the form in 2.2 and thus has a solution given by 2.3.

Introducing a permutation matrix P which reorders the rows of matrix A ensures that it is always possible to find a factorisation $\boldsymbol{PA} = \boldsymbol{LU}$. A system of linear equations can now be rewritten similarly to 2.2 as

$$\boldsymbol{Ax} = \boldsymbol{b} \Leftrightarrow \boldsymbol{PAx} = \boldsymbol{Pb} \Leftrightarrow \boldsymbol{LUx} = \boldsymbol{Pb} \Leftrightarrow \begin{cases} \boldsymbol{Ly} = \boldsymbol{Pb} \\ \boldsymbol{Ux} = \boldsymbol{y} \end{cases} \tag{2.5}$$

and the matrix equation $\boldsymbol{PAX} = \boldsymbol{PB}$ is divided into

$$\begin{cases} \boldsymbol{PAx}_1 = \boldsymbol{Pb}_1 \\ \boldsymbol{PAx}_2 = \boldsymbol{Pb}_2 \\ \vdots \\ \boldsymbol{PAx}_m = \boldsymbol{Pb}_m \end{cases} \tag{2.6}$$

which all can be rewritten to the form in 2.5 and thus have the solution given by 2.3, except the subscripts of the elements in the $\boldsymbol{b}_i$ vectors are permuted as decided by $\boldsymbol{P}$.

2.2 Quaternions

A short summary of quaternions is given as these can be used as a convenient representation of rotations.

Quaternions are extensions of the complex space defined as

$$\boldsymbol{q} = q_r + \boldsymbol{v} = q_r + q_i\boldsymbol{i} + q_j\boldsymbol{j} + q_i\boldsymbol{k}$$

where $q_r \in \mathbb{R}$ with basis $\{\mathbf{1}\}$ and $\boldsymbol{v} \in \mathbb{R}^3$ with basis $\{\boldsymbol{i} \quad \boldsymbol{j} \quad \boldsymbol{k}\}$. The basis elements have the following relation: $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. A quaternion $\boldsymbol{q}$ is called a unit quaternion if the norm of $\boldsymbol{q}$, $||\boldsymbol{q}|| = \sqrt{q_r^2 + q_i^2 + q_j^2 + q_i^2}$, is equal to 1; such a quaternion belongs to the unit sphere in $\mathbb{R}^4$, $\boldsymbol{q} \in S^3$. Unit quaternions can be used to represent rotation, and the unit quaternion rotation $\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\bar{\boldsymbol{q}}$ is equivalent to the matrix rotation $\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}$ with $\boldsymbol{R}$ given by

$$\boldsymbol{R} = \begin{bmatrix} 1-2(q_j^2+q_k^2) & 2(q_iq_j - q_kq_r) & 2(q_iq_k + q_jq_r) \\ 2(q_iq_j + q_kq_r) & 1-2(q_i^2+q_k^2) & 2(q_jq_k - q_iq_r) \\ 2(q_iq_k - q_jq_r) & 2(q_jq_k + q_iq_r) & 1-2(q_i^2+q_j^2) \end{bmatrix} \tag{2.7}$$

. Here $\bar{\boldsymbol{q}} = q_r - \boldsymbol{v}$ is the conjugate of the quaternion. The unit quaternion can be recovered from a rotation matrix as

$$\begin{aligned} q_r &= \pm\frac{\sqrt{1+R_{11}+R_{22}+R_{33}}}{2} \\ q_i &= \pm\frac{R_{32}-R_{23}}{|R_{32}-R_{23}|}\frac{\sqrt{1+R_{11}-R_{22}-R_{33}}}{2} \\ q_j &= \pm\frac{R_{13}-R_{31}}{|R_{13}-R_{31}|}\frac{\sqrt{1-R_{11}+R_{22}-R_{33}}}{2} \\ q_k &= \pm\frac{R_{21}-R_{12}}{|R_{21}-R_{12}|}\frac{\sqrt{1-R_{11}-R_{22}+R_{33}}}{2} \end{aligned} \tag{2.8}$$

; note that the representation on quaternion form is not unique as $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation.

For two quaternions $\boldsymbol{q}_1, \boldsymbol{q}_2 \in S^3$ the rotation $\Delta\boldsymbol{q} = \boldsymbol{q}_1\bar{\boldsymbol{q}}_2$ represents the rotation from $\boldsymbol{q}_2$ to $\boldsymbol{q}_1$, where the formula for multiplication of two quaternions is given by

$$\boldsymbol{q}_1\boldsymbol{q}_2 = (q_r^1 + \boldsymbol{v}^1)(q_r^2 + \boldsymbol{v}^2) = (q_r^1q_r^2 - \boldsymbol{v}^1\cdot\boldsymbol{v}^2, q_r^1\boldsymbol{v}^2 + q_r^2\boldsymbol{v}^1 + \boldsymbol{v}^1\times\boldsymbol{v}^2) \tag{2.9}$$

. Mapping the rotation in quaternion representation, $\Delta\boldsymbol{q}$, to an angle representation, $\Delta\boldsymbol{\theta}$, is done with

$$\Delta\boldsymbol{\theta} = 2\log(\Delta\boldsymbol{q}) \tag{2.10}$$

, where $\log(\boldsymbol{q})$ is the natural logarithm of a unit quaternion given by

$$\log(\boldsymbol{q}) = \begin{cases} \frac{\boldsymbol{v}}{||\boldsymbol{v}||} \arccos q_r & , ||v|| \neq 0 \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T & , ||v|| = 0 \end{cases} \tag{2.11}$$

, note that the real part of this logarithm is always zero and $\log(\boldsymbol{q})$ can be interpreted as part of $\mathbb{R}^3$. [Ude et al., 2014]

2.3 Control Theory

How a PD controller is implemented in a discrete real-time system, how quaternions can be used to construct a PD controller for rotations, what mid-ranging control is and how to implement a Kalman filter in a discrete real-time system is presented below. Knowledge of the PD controller and its constituents and knowledge of the Kalman filter is assumed.

PD Controller

As a controller implemented on a computer necessarily reads new measurements and sends new signals discretely a PD-controller must be discretised in order to be implementable. In the following $e(t) = y_r(t) - y(t)$ is the error between a reference for the controlled variable, $y_r(t)$, and the controlled variable, $y(t)$; t is continuous time; and t_k is discrete time incremented with $h = t_k - t_{k-1}$.

The proportional part, P, is in continuous time given by

$$P(t) = Ke(t) \tag{2.12}$$

with design parameter K. Equation 2.12 is straightforwardly discretised as

$$P(t_k) = Ke(t_k) \tag{2.13}$$

.[Årzén, 2014]

The derivative part, D, is in continuous time given by the solution to

$$\frac{T_d}{N}\frac{dD(t)}{dt} + D(t) = KT_d\frac{de(t)}{dt} \tag{2.14}$$

where T_d and N are design parameters. The derivative of D acts as a low pass filter, which is added due to the tendency of a pure derivative to amplify measurement noise. With this low pass filter the gain at high frequencies is limited to N. Equation 2.14 is discretised by approximating the derivatives as backward differences

$$\frac{T_d}{N}\frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = KT_d\frac{e(t_k) - e(t_{k-1})}{h} \tag{2.15}$$

which is rewritten as

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) + \frac{KT_dN}{T_d + Nh}\big(e(t_k) - e(t_{k-1})\big) \tag{2.16}$$

. [Årzén, 2014]

Control of rotations with quaternions

A typical PD-controller for the angles of an orientation can be written as

$$\begin{array}{rcl} \ddot{\boldsymbol{\theta}}(t) & = & -K\big(\boldsymbol{\theta}(t) - \boldsymbol{\theta}_r(t)\big) - \frac{K}{T_d}\frac{d}{dt}\big(\boldsymbol{\theta}(t) - \boldsymbol{\theta}_r(t)\big) \\ & \Leftrightarrow & \\ \boldsymbol{\omega}(t) & = & -K\Delta\boldsymbol{\theta}(t) - \frac{K}{T_d}\frac{d}{dt}\Delta\boldsymbol{\theta}(t) \end{array} \tag{2.17}$$

, where K and T_d are design parameters, $\boldsymbol{\theta}_r$ is the desired orientation and $\boldsymbol{\theta}$ is the measured orientation represented with angles. Equation 2.17 can be rewritten with equation 2.10 into

$$\dot{\boldsymbol{\omega}}(t) = -K\log\big(\boldsymbol{q}(t)\bar{\boldsymbol{q}}_r(t)\big) - \frac{d}{dt}\log\big(\boldsymbol{q}(t)\bar{\boldsymbol{q}}_r(t)\big) \tag{2.18}$$

, where the 2 from 2.10 is incorporated into the design parameter K, $\boldsymbol{q}_r$ is the desired orientation and $\boldsymbol{q}$ is the measured orientation represented with quaternions.[Ude et al., 2014]

Mid-ranging control

Mid-ranging control is a control scheme, which can be used in systems where there are more possible inputs than outputs. The extra degree of freedom can be used to achieve some other control goal, which in mid-ranging control is to keep some of the control inputs close to a reference input, which in turn can prevent saturation of these control inputs. This is done by adding a second control loop to the standard loop where the control inputs from the standard loop is used as the controlled variable. The controllers in both loops can be of any type, and the process model is divided into three parts, two to model the two groups of control signals different influences on the process and one to model the shared influence. Such a control structure can be seen in figure 2.1. [Haugwitz et al., 2005]

Kalman filter

Given a system in the state space form

$$\begin{array}{rcl} \dot{\boldsymbol{x}} & = & A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} & = & C\boldsymbol{x} \end{array} \tag{2.19}$$

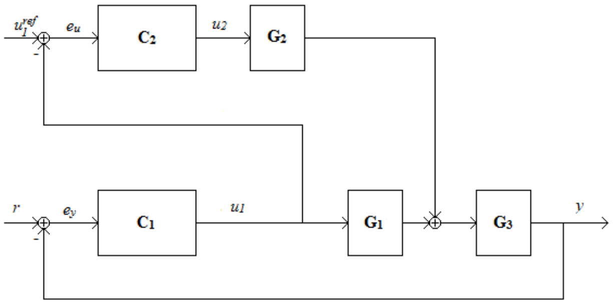

Figure 2.1 A general mid-ranging control structure. All signals can be multivariate. $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are the controllers and $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ are different parts of the process model.

, with A, B and C as matrices and the state x, the measured signal y and the control signal u as vectors, the Kalman filter is given by

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= A\hat{\boldsymbol{x}} + B\boldsymbol{u} + K(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\ \hat{\boldsymbol{y}} &= C\hat{\boldsymbol{x}} \end{aligned} \tag{2.20}$$

, where $\hat{x}$ is the state estimate and $\hat{y}$ is the filtered measurement. If the states can be measured directly 2.19 and 2.20 can be simplified since $C = I \Leftrightarrow \begin{cases} \boldsymbol{y} = \boldsymbol{x} \\ \hat{\boldsymbol{y}} = \hat{\boldsymbol{x}} \end{cases}$. Furthermore, assuming no correlation between measurements of different states the matrix K should be chosen as a diagonal matrix where each diagonal entry weights the effect of the measurements of each state on that state's estimate. Larger diagonal entries causes the estimates to converge quicker when an unmodelled disturbance affects a state, but the estimate will be more sensitive to measurement noise. Conversely, smaller diagonal entries make the estimate less noise sensitive, but cause slower convergence.

In order to implement 2.20 it needs to be discretised. Using forward difference $\dot{\hat{\boldsymbol{x}}}$ can be approximated as

$$\frac{1}{h}(\hat{\boldsymbol{x}}_{k+1} - \hat{\boldsymbol{x}}_k) = A\hat{\boldsymbol{x}}_k + B\boldsymbol{u}_k + K(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k) \tag{2.21}$$

and a new state estimate can be calculated each time a new control signal is computed as

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k + h\big(A\hat{\boldsymbol{x}}_k + B\boldsymbol{u}_k + K(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)\big) \tag{2.22}$$

.

2.4 Robotics

An introduction to conventions and kinematics of robotics is given below. Basic kinematics and the Denavit-Hartenberg Convention are first introduced. Differentiation of results from kinematics leads to velocity kinematics which is presented after.

Kinematics

In robotics each joint has its own frame with different origins and different axis directions. The relation between the representation of a point in frame i, $\boldsymbol{p}^i$, and the representation of the same point in frame j, $\boldsymbol{p}^j$, is given by

$$\boldsymbol{p}^i = \boldsymbol{R}^i_j \boldsymbol{p}^j + \boldsymbol{d}^i_j \tag{2.23}$$

, where $\boldsymbol{R}^i_j$ is a 3×3 rotation matrix representing the rotation of the axes in coordinate system j in relation to the axes in coordinate system i, and $\boldsymbol{d}^i_j$ is a 3×1 column vector representing the distance between the origins in coordinate system i. [Freidovich, 2017]

In order to represent both rotations and translations as matrix multiplications the homogeneous transformation matrix is introduced:

$$\boldsymbol{H}^i_j = \begin{bmatrix} \boldsymbol{R}^i_j & \boldsymbol{d}^i_j \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{2.24}$$

and 2.23 can now be written as

$$\boldsymbol{P}^i = \boldsymbol{H}^i_j \boldsymbol{P}^j \tag{2.25}$$

, with $\boldsymbol{P}^i = [\boldsymbol{p}^i \quad 1]^T$. Introducing $\boldsymbol{A}_i \equiv \boldsymbol{H}^{i-1}_i$ the homogeneous transformation matrix can be written as

$$\boldsymbol{H}^i_j = \begin{cases} \boldsymbol{A}_{i+1}\boldsymbol{A}_{i+2}\cdots\boldsymbol{A}_{j-1}\boldsymbol{A}_j & \text{if} \quad i<j \\ \boldsymbol{I} & \text{if} \quad i=j \\ \left(\boldsymbol{H}^j_i\right)^{-1} & \text{if} \quad i>j \end{cases} \tag{2.26}$$

.

Following the Denavit-Hartenberg (DH) Convention the joint i of a robot has a homogeneous transformation matrix defined as

$$\boldsymbol{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.27}$$

, where the DH parameters are: the rotation around $\vec{\boldsymbol{z}}_{i-1}$, θ_i; the displacement along $\vec{\boldsymbol{z}}_{i-1}$, d_i; the displacement along $\vec{\boldsymbol{x}}_i$, a_i; and the rotation around $\vec{\boldsymbol{x}}_i$, α_i. For a prismatic joint all parameters except d_i are constant, and for a revolute joint all parameters except θ_i are constant; the variable parameter is called the joint variable and is denoted q_i. [Freidovich, 2017]

Given the joint variables for all joints any point in frame j can now be converted to frame i with 2.25. This can be used to solve the forward kinematics problem: Given the joint variables, what is the representation of a point given in the frame of the last joint, the end-effector frame, in the base frame?

A common problem is the inverse of this, called inverse kinematics: Given a point in the base frame, what joint variables places a point in the end-effector frame at this point? Unlike forward kinematics, where the problem always has a unique solution, inverse kinematics can have several solutions. It can be shown that a unique solution only exists if the robot has six joints; if the robot has less than six joints a solution does not always exist, and if the robot has more than six joints the solution is not always unique. The latter case might be desirable as it allows the same position to be reached in different ways, which in turn allows the robot to, for example, avoid obstructed paths. It is often not possible to find an analytic solution to the inverse kinematics problem.[Freidovich, 2017]

Velocity Kinematics

Given an n-joint robot the homogeneous transformation matrix between the end-effector frame and the base frame is given by

$$\boldsymbol{H}_n^0(\boldsymbol{q}(t)) = \begin{bmatrix} \boldsymbol{R}_n^0(\boldsymbol{q}(t)) & \boldsymbol{d}_n^0(\boldsymbol{q}(t)) \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{2.28}$$

where it is now noted that the Matrix is a function of the joint variables $\boldsymbol{q}(t) = [q_1(t) \quad q_2(t) \quad \cdots \quad q_n(t)]^T$ which in turn are functions of time when the robot is moving.

A point which has constant coordinates in the end-effector frame, $\boldsymbol{p}^n$, but is moving in the base frame, $\boldsymbol{p}^0(t)$, has a relation between representations given by 2.23 as

$$\boldsymbol{p}^0(t) = \boldsymbol{R}_n^0(\boldsymbol{q}(t))\boldsymbol{p}^n + \boldsymbol{d}_n^0(\boldsymbol{q}(t)) \tag{2.29}$$

with time derivative

$$\dot{\boldsymbol{p}}^0(t) = \boldsymbol{\omega}_n^0(t) \times \boldsymbol{R}_n^0(\boldsymbol{q}(t))\boldsymbol{p}^n + \boldsymbol{v}_n^0(t) \tag{2.30}$$

, where $\boldsymbol{\omega}_n^0(t)$ is the angular velocity of the end-effector frame in the base frame, and $\boldsymbol{v}_n^0(t)$ is the linear velocity of the origin of the end-effector frame in the base frame. [Freidovich, 2017]

The velocities in 2.30 are given by

$$\begin{cases} \boldsymbol{v}_n^0(t) & = \boldsymbol{J}_v(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) \\ \boldsymbol{\omega}_n^0(t) & = \boldsymbol{J}_\omega(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) \end{cases} \tag{2.31}$$

, where

$$\boldsymbol{J}\big(\boldsymbol{q}(t)\big) = \begin{bmatrix} \boldsymbol{J}_v\big(\boldsymbol{q}(t)\big) \\ \boldsymbol{J}_\omega\big(\boldsymbol{q}(t)\big) \end{bmatrix}$$

is a $6 \times n$ matrix called the manipulator Jacobian. With $\boldsymbol{J}\big(\boldsymbol{q}(t)\big) = [\boldsymbol{J}_1 \quad \cdots \quad \boldsymbol{J}_n]$ the values of the Jacobian can be extracted from the homogeneous transformation matrices

$$\boldsymbol{H}_{i-1}^0 = \begin{bmatrix} \boldsymbol{x}_{i-1}^0 & \boldsymbol{y}_{i-1}^0 & \boldsymbol{z}_{i-1}^0 & \boldsymbol{d}_{i-1}^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

as

$$\boldsymbol{J}_i = \begin{cases} \begin{bmatrix} \boldsymbol{z}_{i-1}^0 \times (\boldsymbol{d}_n^0 - \boldsymbol{d}_{i-1}^0) \\ \boldsymbol{z}_{i-1}^0 \end{bmatrix} & \text{if joint } i \text{ is revolute} \\ \begin{bmatrix} \boldsymbol{z}_{i-1}^0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{if joint } i \text{ is prismatic} \end{cases}$$

. [Freidovich, 2017]

Introducing

$$\boldsymbol{\xi}(t) = \begin{bmatrix} \boldsymbol{v}_n^0(t) \\ \boldsymbol{\omega}_n^0(t) \end{bmatrix}$$

equation 2.31 can be written as

$$\boldsymbol{\xi}(t) = \boldsymbol{J}\big(\boldsymbol{q}(t)\big)\dot{\boldsymbol{q}}(t) \tag{2.32}$$

. The Jacobian can be inverted when the robot has six joints, $n = 6$, and the robot is not at a singularity. The joint velocities, $\dot{\boldsymbol{q}}_*$, which result in some desired end-effector velocities $\boldsymbol{\xi}_*$, can then be calculated from 2.32 as

$$\dot{\boldsymbol{q}}_* = \boldsymbol{J}^{-1}\boldsymbol{\xi}_* \tag{2.33}$$

. When $n > 6$ all solutions can be found with

$$\dot{\boldsymbol{q}}_* = \boldsymbol{J}^+\boldsymbol{\xi}_* + (\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})\boldsymbol{b} \tag{2.34}$$

where

$$\boldsymbol{J}^+\big(\boldsymbol{q}(t)\big) = \boldsymbol{J}^T\big(\boldsymbol{q}(t)\big)\Big(\boldsymbol{J}\big(\boldsymbol{q}(t)\big)\boldsymbol{J}^T\big(\boldsymbol{q}(t)\big)\Big)^{-1}$$

is the pseudo inverse of the Jacobian and $\boldsymbol{b}$ is a vector, which can be seen as joint velocities projected onto the null space of the Jacobian by $(\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})$. They can be considered to belong to the null space as any velocity $\boldsymbol{q}'(t)$ given by

$$\boldsymbol{q}' = (\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})\boldsymbol{b} \tag{2.35}$$

will not affect the velocities of the end-effector since

$$\begin{aligned}\boldsymbol{\xi}' &= \boldsymbol{J}\boldsymbol{q}' = \boldsymbol{J}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\boldsymbol{b} \\ &= (\boldsymbol{J} - \boldsymbol{J}\boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{J}^{T})^{-1}\boldsymbol{J})\boldsymbol{b} \\ &= (\boldsymbol{J} - \boldsymbol{J})\boldsymbol{b} = \mathbf{0}\end{aligned} \tag{2.36}$$

. Note that in the equations above the dependence of t and $\boldsymbol{q}(t)$ is not always explicitly stated as this would make the equations hard to read. [Freidovich, 2017]

2.5 Externally Guided Motion Research Interface

Externally Guided Motion Research Interface, henceforth called EGMRI, is an extended version of the Externally Guided Motion module, henceforth called EGM, released by ABB Ltd. The major difference between the two is that EGMRI allows more signals to be sent and received.

EGM bypasses the path planning of ABB robots and sends references directly to the motors in the joints. These references are either joint values: motor torque, angular velocity or joint angle; or a pose defined in some reference frame. There are some drawbacks with EGM: the path planning is bypassed so the movements between two points can not be expected to be linear, and if linear movement is desirable the interpolation most be done beforehand; if the robot approaches a singularity the robot will stop execution; and EGM only works on 6-jointed robots. The latter is due to the limited amount of signals allowed to be sent and received in EGM and is not a hindrance with EGMRI. [*Chapter 9.3 Externally Guided Motion in Application manual - Controller software IRC5* 2016]

2.6 Inter-Integrated Circuit

Inter-Integrated Circuit, henceforth I^{2}C, is a protocol for a bidirectional 2-wire bus. A wire for serial data, SDA, and a wire for serial clock, SCL, is used to transmit information between devices. Devices are recognised by unique addresses, so if more devices with the same address are to be used a switch is needed. Devices can act as both receivers and transmitters, and a device initiating transmission of data is called a master and any device addressed by the master is called a slave. More than one master can use the same bus, but such a system is not considered here. [*I^{2}C-bus specification and user manual* 2014]

The logic levels of the bus is determined by the supply voltage, V_{DD}, with a LOW-threshold of $0.3V_{DD}$ and a HIGH-threshold of $0.7V_{DD}$. The bus is considered free if both SDA and SCL is HIGH. A transmission is initialised by the master device by sending a START condition, in which SDA transitions from HIGH to LOW while SCL is HIGH, and the bus is now busy. [*I^{2}C-bus specification and user manual* 2014]

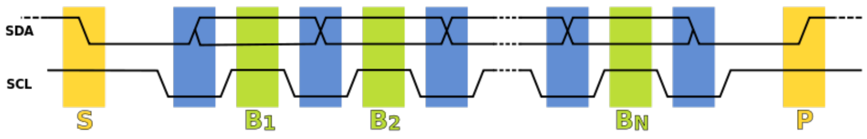

Figure 2.2 The transfer of data over the I^{2}C-bus. Note that SDA only changes between HIGH and LOW when the clock signal on SCL is low, blue areas, with the exception of start and stop conditions, yellow areas. Each clock pulse of HIGH on SCL, green areas, represents a bit.[*I2C data transfer*]

Data is sent from the transmitter on the SDA wire one byte at the time, with the most significant bit first. After a successful transfer an Acknowledge signal of one bit is sent by the receiver by pulling the SDA signal LOW. If SDA is HIGH during the Acknowledge the transfer was unable to be performed properly. [*I^{2}C-bus specification and user manual* 2014]

The value of SDA is only allowed to change when SCL is LOW as SDA must be stable when SCL is HIGH, and the value of SDA when SCL is HIGH determines a bit's value. The first byte sent after a START condition is the seven bit slave address followed by a READ/WRITE bit representing whether the master or the slave will transmit data on SDA. If the last bit is zero, WRITE, the master will be the transmitter and if the bit is one, read, the slave will be the transmitter; note that the master always generates the clock pulses on the SCL independent of what device is considered the transmitter. [*I^{2}C-bus specification and user manual* 2014]

Any number of bytes can now be sent by the transmitter, and when this is determined done by the master the master either generates a new START, in which another slave can be addressed, or a STOP, by pulling the SDA high when SCL is HIGH and the bus is now considered free again. A transfer like this is illustrated in figure 2.6. [*I^{2}C-bus specification and user manual* 2014]

2.7 Sensor placement

A model of the camera platform developed by Emma Andersson and Anna Wikström, modified to allow placement of sensors, can be seen in figure 2.3. In the plane of the semicircle of the platform translations are easily measured in two perpendicular directions; translations in the direction perpendicular to this plane are calculated as $a * \sin\alpha$ where a is the sensor measurement of the sensor placed in the extrusion and α is the angle of the sensor relative to the direction perpendicular to the semicircle plane.

Rotations are all measured by three pairs of sensors placed in the semicircle plane. Each pair is placed according to figure 2.4 which allows for an angle to be

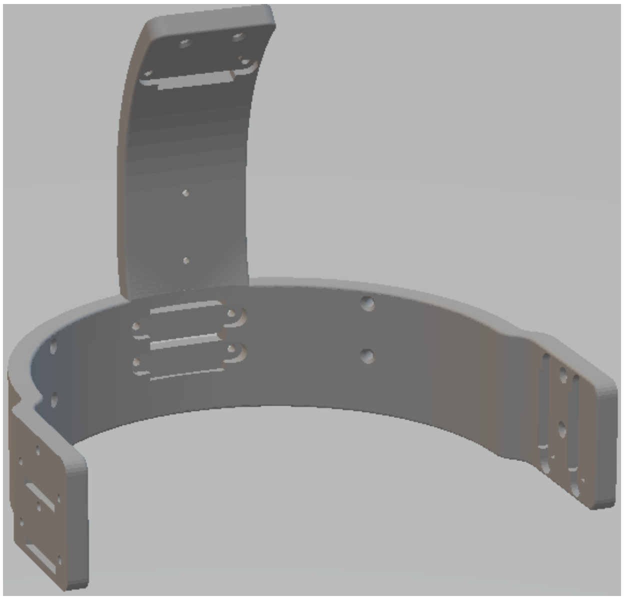

Figure 2.3 The modelled platform. Sensors are to be placed in the seven recesses, with two hidden from sight in the image.

calculated from sensor measurements as

$$\begin{cases} \tan(\theta) & = \frac{x}{y} \\ x & = a - b \end{cases} \Leftrightarrow \theta = \arctan\left(\frac{a-b}{y}\right) \tag{2.37}$$

, where the parameters are defined in the figure.

Each pair is perpendicular to the other pairs and thus each pair results in three independent angles which fully represent the rotation of the surgeon's head. Note that the head is approximated as flat between the sensors in each pair.

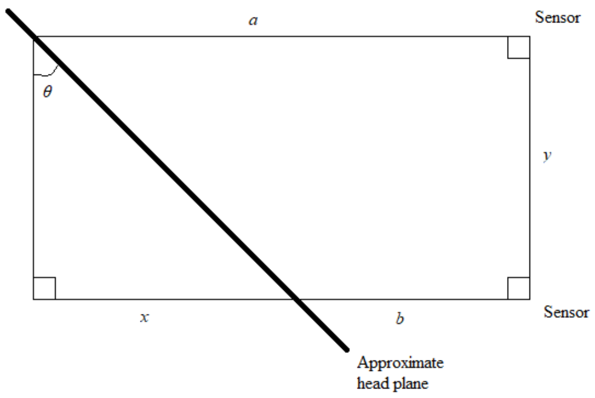

Figure 2.4 Illustration of sensor pair setup. The distance between sensors is y, the measurement from the upper sensor is a, the measurement from the lower is b and the angle that is to be calculated is θ.

3
Methods

In order to implement a robotic arm capable of following the movements of a surgeon communication between the controller, the robot and the sensors was necessary; the platform needed to be modified to allow sensor placement; and the control algorithm needed to be implemented and tuned with testing.

Before conducting any setup or implementing any algorithms a full day visit at the children's heart centre at Lund university hospital was also conducted in order to get a bit familiar with the environment the robot is to be implemented in.

3.1 Setting up communication

Communication between the control program, the robot and the sensors is handled by a Raspberry Pi 3 Model B. The control program executes on the Raspberry and also contain functions which are used to communicate with the sensors. A bridge acting as a server also executes on the Raspberry and is used to communicate between the control program and the robot.

Communication with the robot

The robot is connected to the Raspberry via an ethernet cable, and communication with the robot is carried out via a bridge between labcomm and EGMRI provided by the computer science department at LTH. Once the control program and the robot has connected to the bridge data is sent from the robot to the control program in the form of joint values, joint velocities and joint torques. The control program uses these combined with sensor data to calculate references, joint values, joint velocities and joint torques, which are sent back to the robot. This procedure is repeated at 250 Hz.

Communication with sensors

The vl6180 sensors are connected to the Raspberry, via a multiplexer of model TCA9548A, with the I^2C-bus described in section 2.6. An SCL and an SDA wire is connected between the Raspberry and the multiplexer, which in turn is connected to

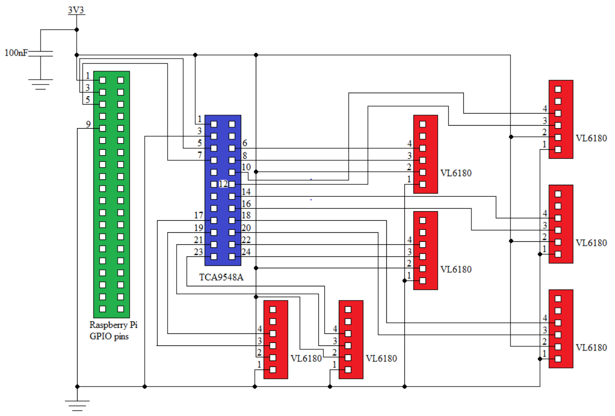

Figure 3.1 A circuit diagram illustrating how the seven sensors are connected. Pin 3 on the Raspberry is the SDA pin and pin 5 is the SCL pin. These are connected to pin 5 and 7 on the multiplexer, which are the SDA and SCL pins. The multiplexer switches the SDA-SCL connection to the SDA-SCL pairs $0-7$, counted counter-clockwise from pins 17 and 19 to pins 8 and 6. Note that the last pair, pins 8 and 6, are not used. Each pair is connected to a sensor via pin 3, SDA, and pin 4, SCL, on the sensor. Voltage is supplied via pin 1 and 9 on the Raspberry.

each of the seven sensors, labelled $0-6$, via an SCL and an SDA wire per sensor. By sending a 1 left shifted $0-6$ steps to the multiplexer it connects an SCL-SDA pair, labelled with the number of shifted steps, to the SCL-SDA pair connecting the raspberry and the multiplexer.

As the multiplexer and the sensors have different I^{2}C addresses both can be detected and communicated with at any time, and changing which sensor to communicate with is done by sending a 1 left shifted $0-6$ steps to the multiplexer as described above. The Raspberry also supplies voltage to the sensors and multiplexer; a circuit diagram illustrating this and the I^{2}C connections can be seen in figure 3.1.

The sensors update the range measurements continuously in a thread separate from the rest of the control program. The program then uses the latest measurements when calculating control action. Starting a new measurement is done by writing the command to start the measurement to the register in the sensor governing this, then polling the register which notifies if the measurement is done, then reading the measurement from the register in which it is stored and finally clearing the interrupt caused by starting the measurement by writing to the register governing this, after which a new measurement is started. The sensors have more registers than the ones mentioned which allows for changing settings and different functionality, but these are not touched upon in this thesis.

3.2 Platform design

The original platform designed by Emma Andersson and Anna Wikström had no space for the sensors necessary, and the original design had to be altered. The modified design can be seen in figure 2.3, where the height of the semicircle and the width of the extrusion have been increased from $30mm$ to $47mm$; and eight recesses have been added, with drill holes and holes for wires, where the sensors and the multiplexer can be placed. The recesses for the sensors are placed as described in section 2.7 while the recess for the multiplexer is placed on the back of the extrusion so wiring between sensors and multiplexer can be placed outside the platform and not be in the way for a surgeon wearing the platform. The position of the cameras relative to each other are still the same as in the original design.

3.3 Control design

The EGMRI allows reference torques, reference joint velocities and reference joint values to be sent to the internal controller and it allows measuring of the current torques, joint velocities and joint values and distance measuring sensors allowed measurement of the pose of the head relative to the platform as described in chapter 3.2. As control with torques requires an extensive dynamical model of the robot and its load, the ability to use reference torques was ignored in the design of the controller, and the input signals for the internal controller was thus references in joint velocities, $\dot{q}_{ref}$, and joint values, q_{ref}, with the resulting output signals the actual joint velocities, $\dot{q}$, actual joint values, q, and the pose of the head relative to the platform, y^7.

The goal of the project is to keep this pose at a, in the frame of the platform, constant reference, which is constructed by measuring the pose of the head when the program is started; several measurements are done in order to apply a median filter and get a more accurate reference. In order to determine the reference pose and the pose of the head in the base frame these poses are turned into homogeneous transformation matrices and then multiplied from the right with the homogeneous transformation matrix $\boldsymbol{H}_7^0$ given by equations 2.26 and 2.27, the measured current joint values and the Denavit-Hartenberg parameters in table 3.1. In the base frame it is now possible to find joint values which make the pose of the head, y^0, coincide with the reference pose, r^0; this is however complicated as the relation between head pose and joint values are nonlinear. Instead it would be desirable if some control signal could be generated in cartesian space and then converted to joint values.

The relation between the error in pose, e, and how the platform should move is straight forward. For position the error is simply the difference in position and the platform should move to diminish this error, which can be done by letting the error in each axis be the input to a PD controller with acceleration along that axis as output. For orientation the error can be mapped to quaternion space and the angular

Link	d_i/m	a_i/m	α_i/rad
1	0.110	-0.030	$-\frac{\pi}{2}$
2	0	0.030	$\frac{\pi}{2}$
3	0.2465	0.0405	$-\frac{\pi}{2}$
4	0	-0.0405	$\frac{\pi}{2}$
5	0.265	0.0135	$-\frac{\pi}{2}$
6	0	-0.027	$\frac{\pi}{2}$
7	0.032	0	0

Table 3.1 DH-parameters of the IRB 14000 "YuMi" robot. No θ_i:s are given as all joints are revolute and all θ_i:s therefore variable.

Joint	min/deg	max/deg
1	-168.5	168.5
2	-143.5	43.5
3	-123.5	80.0
4	-290.0	290.0
5	$-$ 88.0	138.0
6	-229.0	229.0
7	-168.5	168.5

Table 3.2 The allowed range for each joint. Note that this table uses the labelling of joints as preferred by ABB. The physical order of the joints, starting at the base, is 1, 2, 7, 3, 4, 5 and 6.

acceleration can be computed from a PD controller on the form given by equation 2.18. Acceleration is chosen as the output to get smother movement than if velocity or position was the output; even smoother movement could be achieved if jerk was instead the control output, but this is deemed unnecessary. The PD controller is easily implementable given the discretisation from equations 2.13 and 2.16.

In order to get signals that can be sent to the internal controller the control signal, u, is integrated to velocity, ξ, which in turn is converted to desired joint velocities via the inverse velocity kinematics described by equation 2.34 where the pseudoinverse of the Jacobian, $\boldsymbol{J}^+$, is calculated using the LU-factorisation method from chapter 2.1.

The joint velocities from the inverse velocity kinematics are integrated to get the desired joint values and both are sent as references to the internal controller. As the internal controller shuts down if a joint value reference outside of the allowed range, given in table 3.2, is sent, the joint values are saturated to be within this range before being sent.

Converting the control output to joint space at the velocity stage instead of the other two possible stages is preferable, because converting directly from acceleration would require the derivative of the Jacobian and be a more complicated conversion; because converting from position requires inverse kinematics which has no general analytic solution; and because it improves control to have the joint velocities as an input to the internal controller.

Conversion from cartesian velocity to joint velocities does not have a unique solution and any joint velocity, b, projected onto the the null space of the Jacobian as in equation 2.35 does not affect the cartesian velocity. It is thus possible to use mid-ranging control to compute these joint velocities and project them onto the null space in order to also control the joint values. A simple way to control the joint values in this way is to choose a joint which is then given reference value and then use a simple P controller to attempt to keep the joint value at the reference.

It was discovered during testing that the measurements from the sensors were too noisy to use directly, and that the integrated control signal was indistinguishable from the noise in the measured velocity. Both therefore needed to be filtered. With the cartesian pose, x_1, and cartesian velocity, x_2, as states $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ the state space system for the position along each axis and the rotation around each axis can be modelled as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{I}\boldsymbol{x} \end{aligned}$$

where the -1 is due to the velocity measured being the velocity of the platform and not the head.

The resulting Kalman filter is given by equation 2.20 as

$$\dot{\hat{\boldsymbol{x}}} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \hat{\boldsymbol{x}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} k_p & 0 \\ 0 & k_v \end{bmatrix} (\boldsymbol{x} - \hat{\boldsymbol{x}})$$

which is discretised with 2.21.

As the measured velocities are the joint velocities and not the cartesian, it is necessary to apply the Jacobian, $\boldsymbol{J}$, as in equation 2.32 before sending the velocities to the Kalman filter and as the estimated velocities from the Kalman filter then are in the base frame they are rotated to the end frame with a rotation matrix, $\boldsymbol{R}_0^7$, found by taking the transpose of the rotation matrix in $\boldsymbol{H}_7^0$, before being used to calculate the pose estimate. As there is a risk of the pose measurement to result in outliers a moving median filter was used to reject these. A schematic for the discussed controller can be found in figure 3.2.

For the sake of pole placement the process in figure 3.2 is modelled as a double integrator, ignoring the nonlinearities in the Jacobian and the saturation and assuming that the ABB controller is able to follow references perfectly. This results in the process transfer function

$$P(s) = \frac{-1}{s^2} \tag{3.1}$$

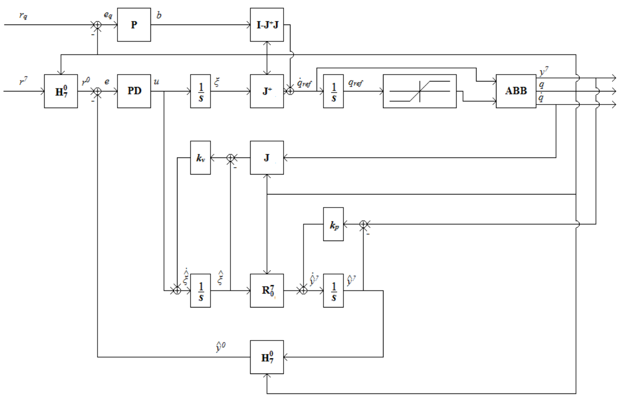

Figure 3.2 A schematic of the control system.

where the minus sign is due to the velocity measured being the velocity of the platform and not the head.

The transfer function of the of the PD-controller, not considering the low pass filtering of the derivative, is given by

$$C(s) = k(1 + sT_d) \tag{3.2}$$

and the closed loop transfer function is given by

$$G(s) = \frac{PC}{1 + PC} = \frac{-k(1 + sT_d)}{s^2 - skT_d - k} \tag{3.3}$$

. The poles of the system 3.3 are given by

$$s^2 - skT_d - k = 0$$

$$\Leftrightarrow$$

$$s = \frac{kT_d}{2} \pm \sqrt{\left(\frac{kT_d}{2}\right)^2 + k}$$

. As the slowest pole, the one closest to zero, determines the speed of the system it is redundant to have two different poles and we place both poles at the same location $s = -c$, where c is a positive real number. These design choices result in the following equations determining the parameter values given the location of desired double pole.

$$\begin{cases} \sqrt{\left(\frac{kT_d}{2}\right)^2 + k} & = 0 \\ s & = \frac{kT_d}{2} \\ s & = -c \end{cases} \Leftrightarrow \begin{cases} k & = -c^2 \\ T_d & = \frac{2}{c} \end{cases}$$

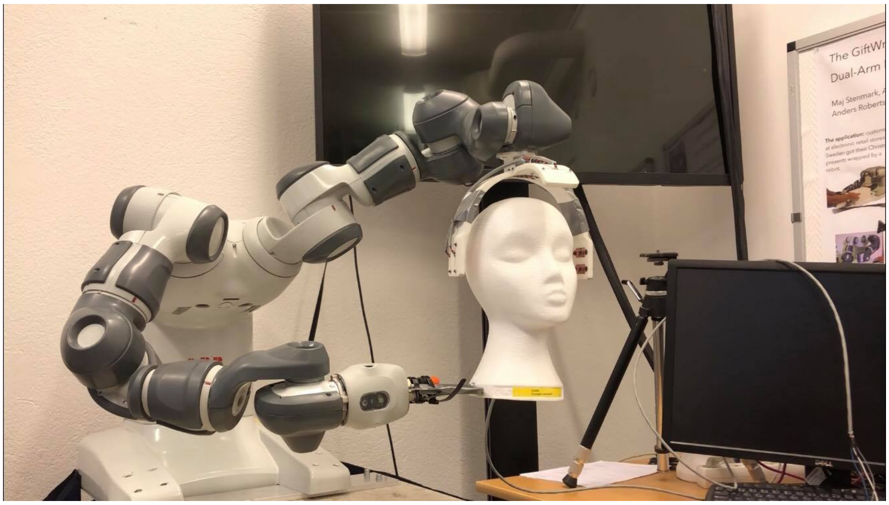

Figure 3.3 The head and platform mounted on the robot.

Note that when controlling orientation $k = -2c^2$ due to the 2 from equation 2.10 incorporated into the control parameter.

3.4 Testing

Testing was carried out in order to determine what parameters in the Kalman filter gave satisfactory convergence and noise reduction and to determine what pole placement gave satisfactory control. First measurements with different Kalman parameters were made while the platform was stationary, in order to test the filtering of the pose only. Then measurements with different Kalman parameters was made while the platform was accelerating along and then around each axis, in order to test the filtering for both the pose and the velocity. Once the Kalman parameters had been tuned testing of different pole placements was made in order to get satisfactory control. Testing was made using the styrofoam head seen in figure 3.3. During the testing of pole placement the head was moved by the other arm of the robot along a predetermined path.

4 Results and discussion

Below follow the results and discussion obtained while testing. In the Kalman filter plots the actual measurements are plotted with a solid red line while the estimates are plotted with a dashed cyan line. The pose is measured in the coordinate frame of the platform, with θ, ψ and ϕ as the rotation around the x, y and z axis respectively, while the velocity is measured in the base frame of the robot, with v_k denoting linear velocity along the k axis while ω_k is denoting rotational velocity of rotation k. In the error plots e_k is the error along k or the rotational error of rotation k and e_{mid} is the error in the joint value controlled by mid-ranging.

The pose measurements are prefiltered by a median filter which discards a measurement if it is more than a number of standard deviations away from the median. The standard deviations used was calculated by making a lot of measurements while stationary and the result for each type of measurement can be seen in table 4.1.

Measurement	Standard deviation
x	$7.8675 \cdot 10^{-4} m$
y	$4.9263 \cdot 10^{-4} m$
z	$9.8633 \cdot 10^{-4} m$
θ	$5.8086 \cdot 10^{-2} rad$
ψ	$1.0418 \cdot 10^{-1} rad$
ϕ	$6.14483 \cdot 10^{-2} rad$

Table 4.1 The standard deviations of each measurement. Used together with a moving median filter to determine whether a measurement is an outlier or not.

4.1 Stationary filtering

During the first tests the head and the platform were kept stationary in order to tune the Kalman filters for the pose. Keeping the platform stationary here means having a constant control signal of $0ms^{-2}$, but some small movement can still be seen in some of the results, possibly due to a non-zero velocity estimate.

As a starting point the parameters of the Kalman filters was set to $k_p = k_v = 2$, and the convergence and noise reduction of the pose filters with these default parameters can be seen without median filtering in figure 4.1 and with median filtering in figure 4.2. Note that an outlier, seen in the x-measurement in both figures, influences the estimate relatively much when no median filter is applied, due to the sensitivity of linear filters from outliers.

As the movement from the head is not modelled any movement will require the Kalman filters to reconverge, and it is therefore desirable to have as short a convergence time as possible without risking oscillatory behaviour caused by noise. As the measurements in rotation are more noisy, note the different scaling on the vertical axis in the figures, the parameter was decreased to $k_p = 1$ for orientation while it was increased to $k_p = 16$ for position. The resulting filtering can be seen in figure 4.3; note the decreased convergence time for position and the smoother estimate for orientation.

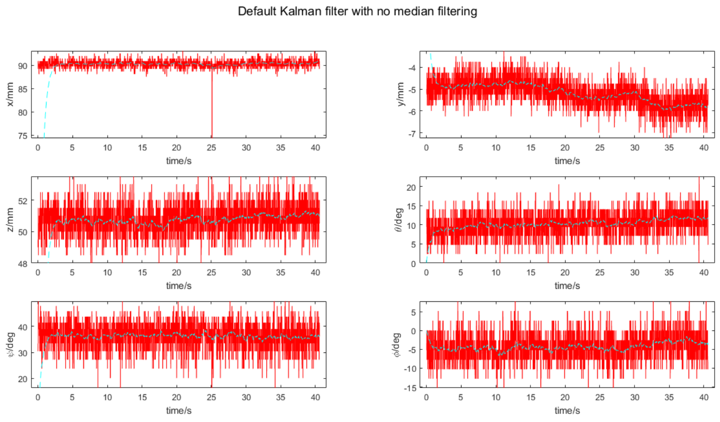

Figure 4.1 The Kalman filter with $k_p = k_v = 2$ for both position and orientation and no median filtering.

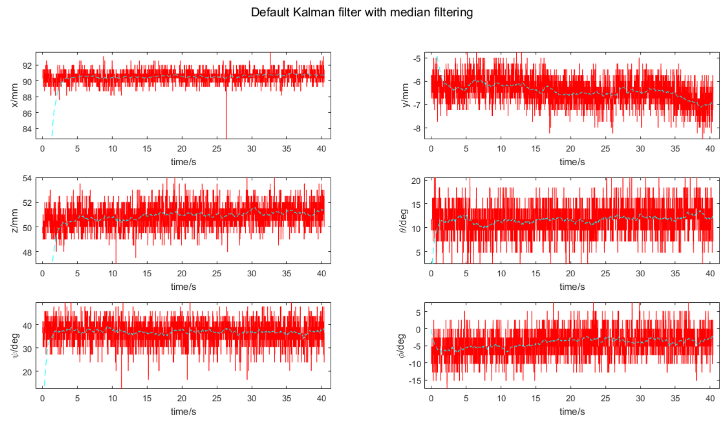

Figure 4.2 The Kalman filter with $k_p = k_v = 2$ for both position and orientation and with median filtering.

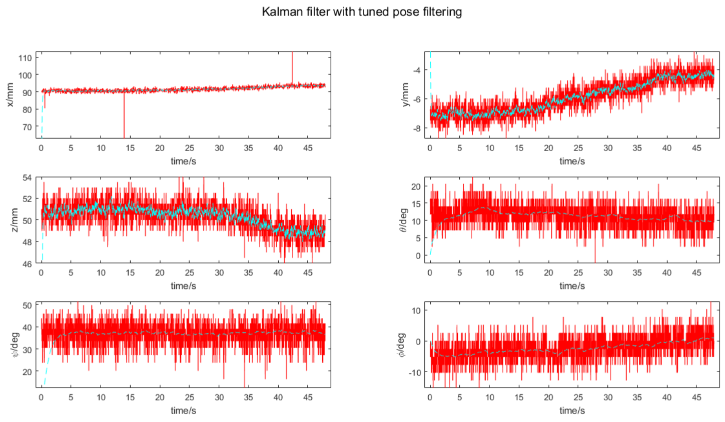

Figure 4.3 The Kalman filter with $k_p = 16$ for position and $k_p = 1$ for orientation. A median filter is used and $k_v = 2$ for both position and orientation.

4.2 Filtering in motion

In order to tune the Kalman filters for velocity the platform was accelerated along and around the different axes of the platform frame while the head was kept stationary. In the figures below the pose measurements change even if there is no acceleration in the direction measured; this is explained by the fact that the head measured on does not extend towards infinity and is not flat.

In the velocity plots in figures 4.5, 4.7, 4.9, 4.11, and 4.13 it is noted that a decreased value of k_v causes the velocity to reach larger magnitudes, but it also causes the estimate to overestimate or underestimate the velocity more while an increased value of k_v has the opposite effect. It was at first thought that the overestimation and underestimation of the velocity caused the overestimation and underestimation of the pose seen for x in figure 4.5 and somewhat for z in 4.10 as these overestimations and underestimations were remedied if the parameter k_v was increased as in figures 4.6 and 4.10. It was later realised that this is more likely caused by the median filter being too rigorous, which will be discussed more in the next section. The parameters were at last tuned to $k_v = 2$ for position and $k_v = 1$ as a smaller parameter resulted in larger speeds. Lastly it was noted that the measurements were not always able to measure changes in rotation as can be seen in 4.12. This is explained by the head being locally spherical and the angle measurements assumption of a flat surface.

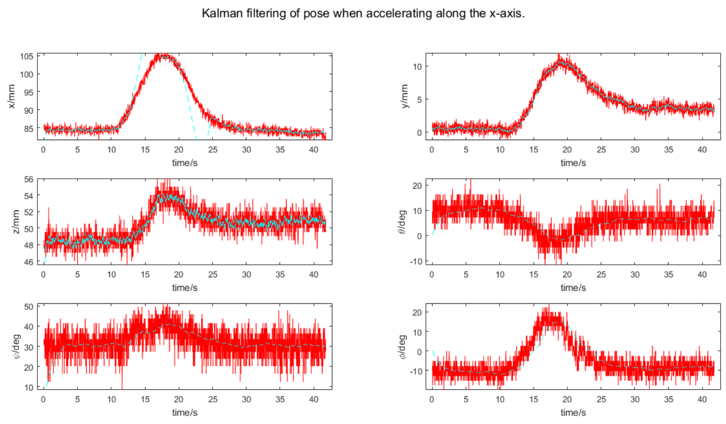

Figure 4.4 The filtered pose when accelerating along the x-axis of the platform with $k_v = 2$ for both position and orientation.

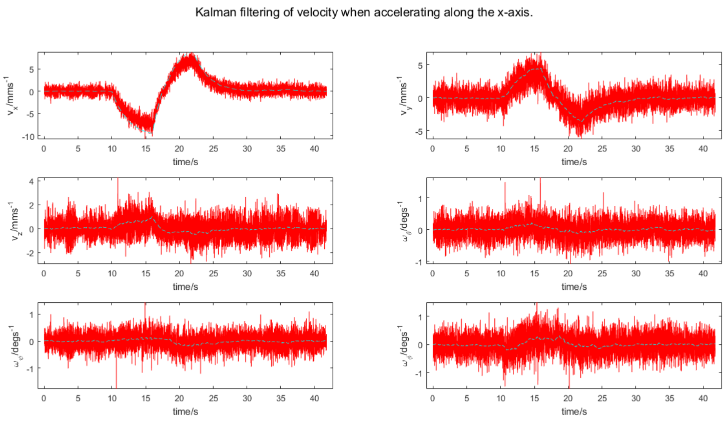

Figure 4.5 The filtered velocity when accelerating along the x-axis of the platform with $k_v = 2$ for both position and orientation.

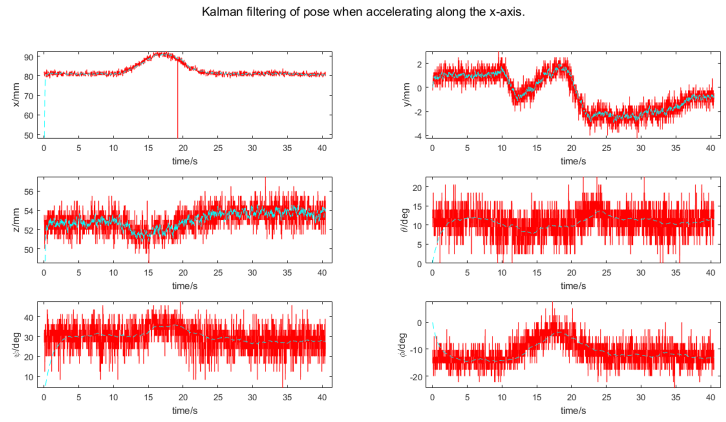

Figure 4.6 The filtered pose when accelerating along the x-axis of the platform with $k_v = 6$ for position and $k_v = 2$ for orientation.

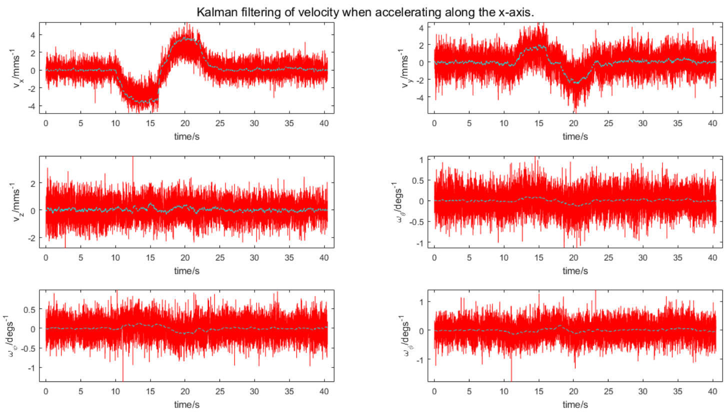

Figure 4.7 The filtered velocity when accelerating along the x-axis of the platform with $k_v = 6$ for position and $k_v = 2$ for orientation.

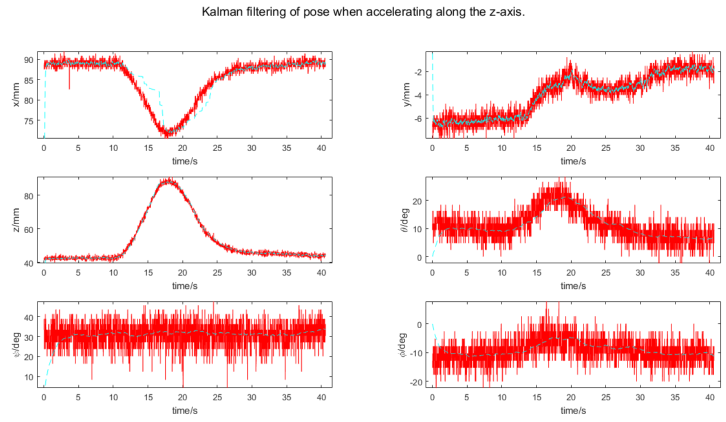

Figure 4.8 The filtered pose when accelerating along the z-axis of the platform with $k_v = 2$ for both position and orientation.

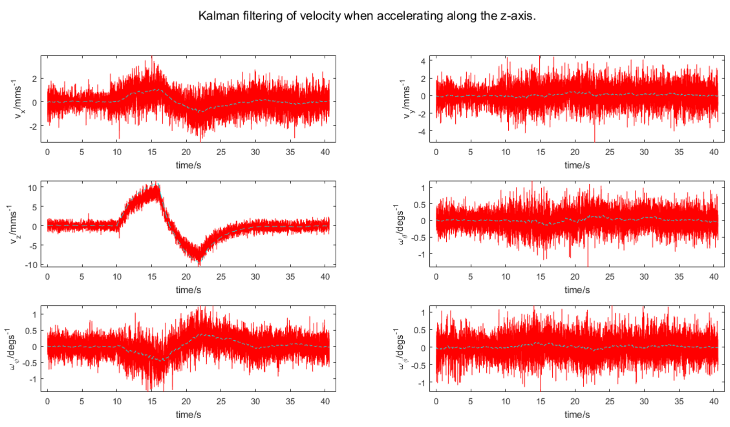

Figure 4.9 The filtered velocity when accelerating along the z-axis of the platform with $k_v = 2$ for both position and orientation.

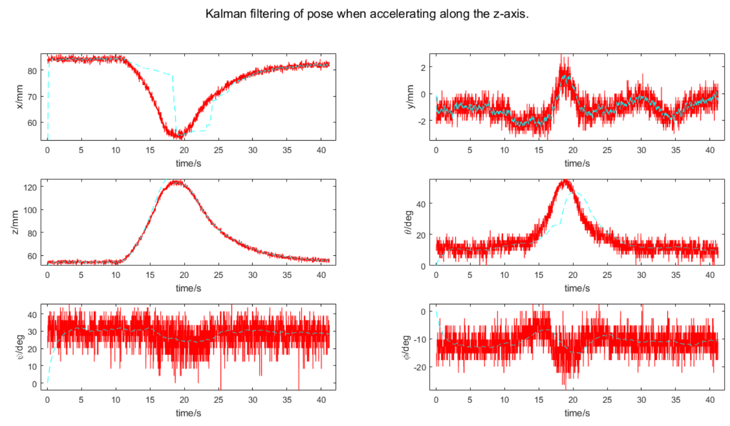

Figure 4.10 The filtered pose when accelerating along the z-axis of the platform with $k_v = 1$ for position and $k_v = 2$ for orientation.

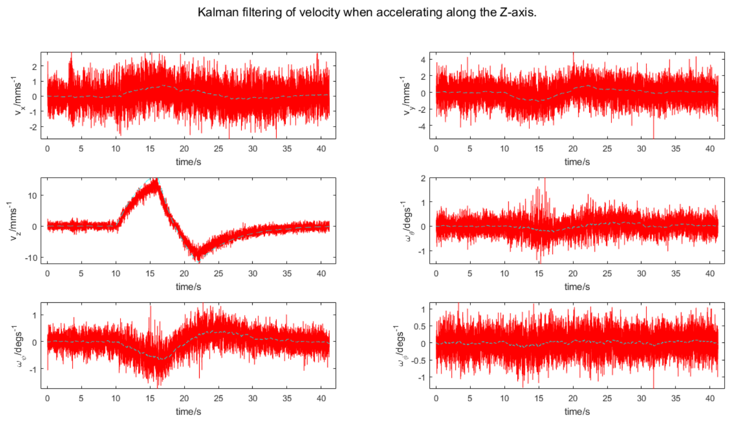

Figure 4.11 The filtered velocity when accelerating along the z-axis of the platform with $k_v = 1$ for position and $k_v = 1$ for orientation.

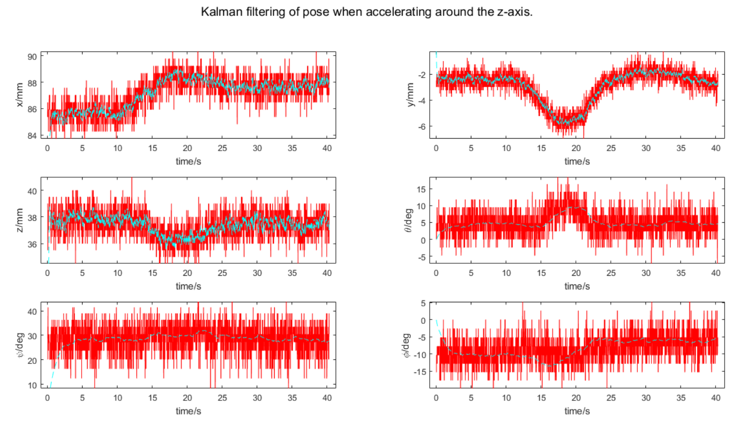

Figure 4.12 The filtered pose when accelerating around the z-axis of the platform with $k_v = 2$ for position and $k_v = 1$ for orientation.

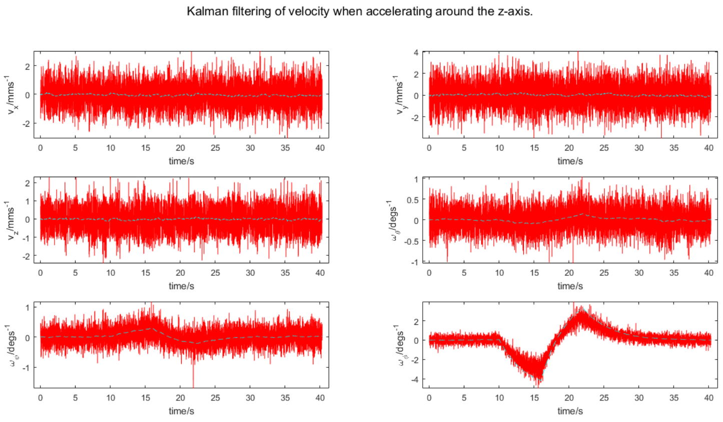

Figure 4.13 The filtered velocity when accelerating around the z-axis of the platform with $k_v = 2$ for position and $k_v = 1$ for orientation.

4.3 Control

In order to tune the controllers the head was moved by the robot's free arm along a programmed path and how well the platform followed depending on control parameters was analysed. In the first tests it was noticed that there was a substantial delay between the movement of the head and the resulting control action, as can be seen in figure 4.15, causing spikes in the error, as can be seen in figure 4.14, and slow reaction. It was determined to be caused by the median filter rejecting measurements too rigorously when measurements further away than 2 standard deviations was rejected and a test without the median filter was carried out. As can be seen in figure 4.16 this removed the delay, but instead caused the estimate to react to outliers too strongly which resulted in a sometimes jabbing movement of the platform. As no median filtering caused unwanted movements but a too rigorous filter caused delays a less rigorous filter which rejected measurements further away than 10 standard deviations was used. As can be seen in figure 4.17 this resulted in both rejection of outliers and the removal of delays. With the delays gone tuning of the controllers was possible. The placement of the poles in -4 for position and -1 for orientation was found to cause oscillations in both position and orientation. The poles were therefore moved to -2 for position and -0.5 for orientation which resulted in less oscillations, but as expected the control acted slower as can be seen by comparing figures 4.18 and 4.19. It was noticed after these tests that the wrong parameters for the filtering of velocity had been used. Applying the correct parameters resulted in faster control which can be seen by comparing figures 4.19 and 4.20, probably due to the velocities' dependence on the filter parameters.

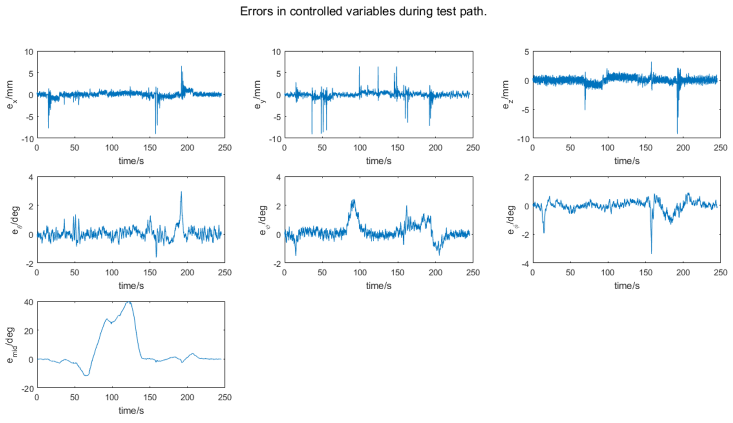

Figure 4.14 The error during movement along the path with the poles of the controllers at −4 for position and −1 for orientation and with median filter designating measurements as outliers if they are more than 2 standard deviations away.

As the performance of the mid-ranging controller is less important than the performance of the other controllers this parameter was simply put as high as it could without having a detrimental effect on the other controllers. As can be seen in the error plots the error in the mid-ranging only goes to zero when the rest of the system is stationary. This is acceptable as it is the value at stationarity that is important for mid-ranging.

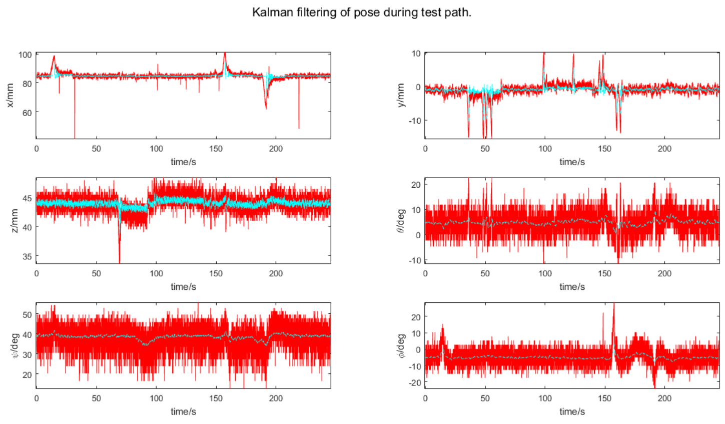

Figure 4.15 The pose measurements and the Kalman filtered estimates during movement along the path with the poles of the controllers at -4 for position and -1 for orientation and with median filter designating measurements as outliers if they are more than 2 standard deviations away.

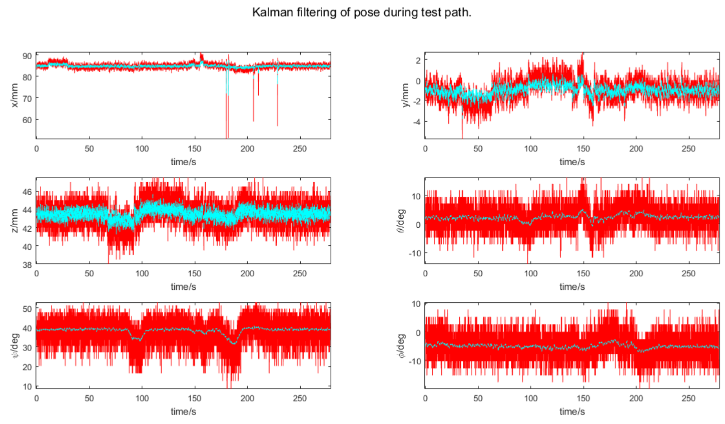

Figure 4.16 The pose measurements and the Kalman filtered estimates during movement along the path with the poles of the controllers at -4 for position and -1 for orientation and without median filter.

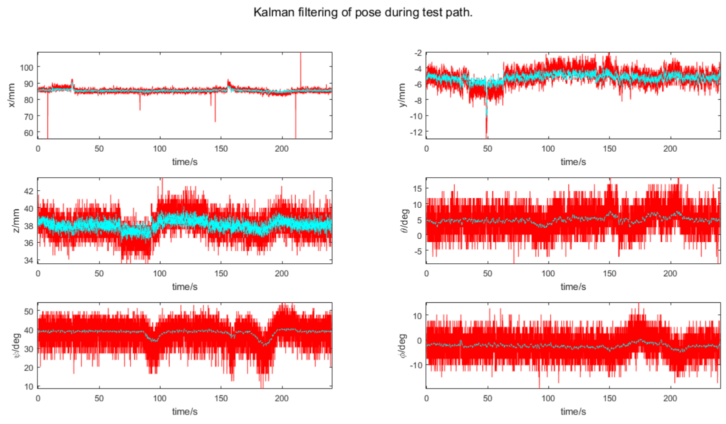

Figure 4.17 The pose measurements and the Kalman filtered estimates during movement along the path with the poles of the controllers at −4 for position and −1 for orientation and with median filter designating measurements as outliers if they are more than 10 standard deviations away.

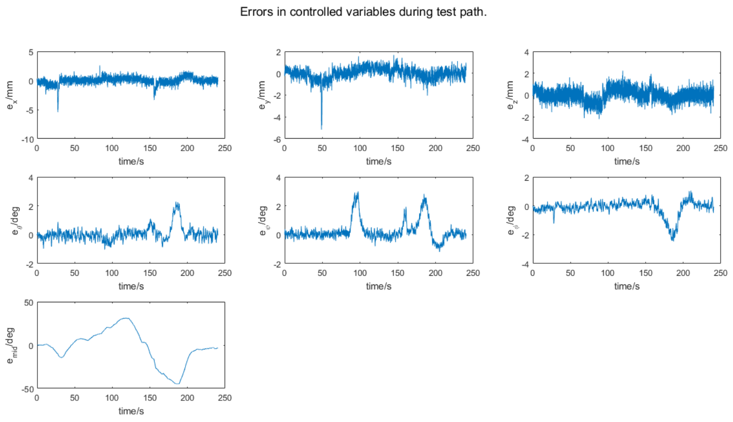

Figure 4.18 The error during movement along the path with the poles of the controllers at −4 for position and −1 for orientation and with median filter designating measurements as outliers if they are more than 10 standard deviations away.

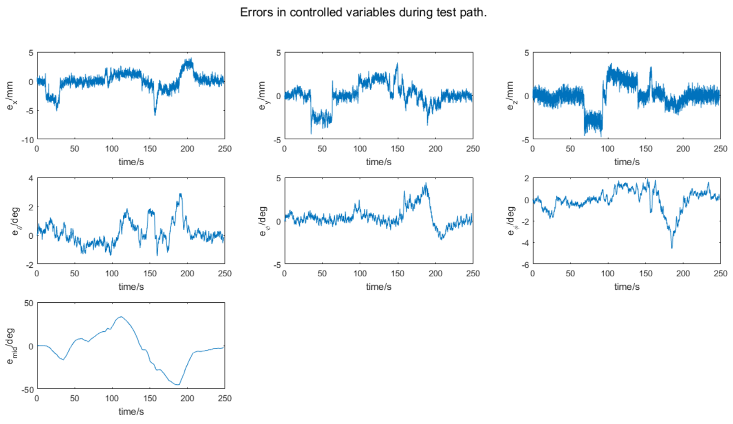

Figure 4.19 The error during movement along the path with the poles of the controllers at -2 for position and -0.5 for orientation and with median filter designating measurements as outliers if they are more than 10 standard deviations away.

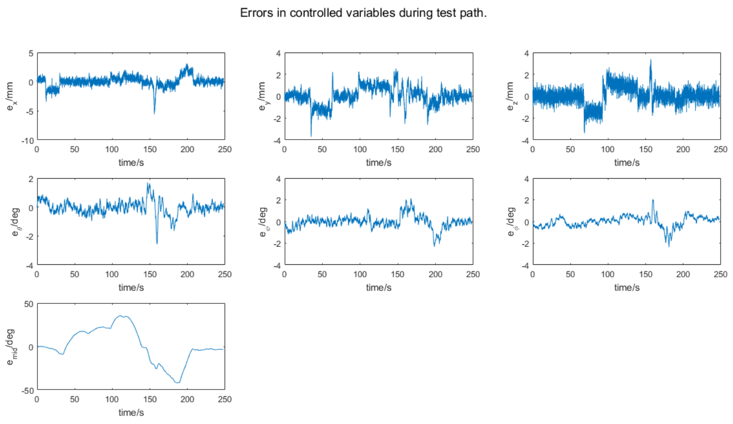

Figure 4.20 The error during movement along the path with the poles of the controllers at -2 for position and -0.5 for orientation and with median filter designating measurements as outliers if they are more than 10 standard deviations away.

5

Conclusion and future work

What conclusions can be drawn from the results are presented next. After that follow some final comments on what to do next to improve this camera platform.

5.1 Conclusion

The control algorithm implemented has given an overall satisfactory tracking of a potential surgeon's movements, although some work is still necessary before this platform can be used by a real surgeon. The biggest issue is determining the orientation from the measurements of angles; it is not always possible to distinguish two different head orientations from the measurements due to local spherical geometry. This could potentially be solved by: introducing more sensors to get more robust measurements, although this might not be feasible without an extensive redesign of the platform; measuring orientation differently, like using information from the cameras or placing a gyroscope on the head; or by fastening small flat plates on the head where the measurements are made.

Another issue is regarding safety: while testing it became apparent that the wiring to the sensors is quite sensitive and a nudge can result in the sensors losing connection requiring a restart of the whole system. The first time this happened the loss of sensor data caused the platform to slam into the table; nothing was damaged due to the safety mechanism installed on the robot causing it to shut down the motors when too large external forces act upon them, but such a slam could potentially be disastrous if occurring during operation. This specific issue can not happen again as the control will now shut down if the sensors lose connection, but testing might be required to ensure no similar safety issues exist.

Overall, the thesis has completed its goal of a satisfactory tracking of a potential surgeons movements, and while some issues require further work it is a valid proof of concept.

5.2 Future work

As mentioned above the wiring to the sensors is quite sensitive and the implementation of the platform leaves it exposed. As a loss of connection requiring a restart of the whole system is undesirable, it is sensible to make some redesign which covers the wiring and the Raspberry.

In this thesis a simple mid-range controller is implemented, but there is potential to introduce more complex mid-ranging control. Some possibilities are a controller that attempts to keep the joint values away from the saturated area or a controller which attempts to maximise the manipulability of the Jacobian, a measure of how easy it is to achieve cartesian velocities depending on joint values, which could avoid singularities and make the overall control better.

No testing with cameras mounted on the platform has been carried out, and it may be necessary to retune the control to avoid oscillations caused by a larger and differently spread weight. Developing the necessary image processing algorithms to get useful data from the cameras is also necessary before deploying the platform.

Bibliography

Andersson, E. and A. Wikström (2017). *Kamerahållare för bildåtergivning av kirurgens synfält*. Bachelor thesis. LTH.

Böiers, L. (2010). "Mathematical methods of optimization". In: Studentlitteratur. Chap. Appendix A: Some concepts from matrix theory.

Chapter 9.3 Externally Guided Motion in Application manual - Controller software IRC5 (2016). ABB.

Freidovich, L. B. (2017). "Control methods for robotic applications". In: Umeå University, pp. 23–85.

Haugwitz, S., M. Henningsson, S. Velut, and P. Hagander (2005). "Anti-windup in mid-ranging control". *Proceedings of the 44th IEEE Conference on Decision and Control and the 2005 European Control Conference*, pp. 7570–7575.

I2C data transfer. `https://commons.wikimedia.org/wiki/File:I2C_data_transfer.svg`. Wikimedia Commons Accessed: 2018-10-04.

I^2C-bus specification and user manual (2014). NXP Semiconductors.

Ude, A., B. Nemec, T. Petrič, and J. Morimoto (2014). "Orientation in cartesian space dynamic movement primitives". *IEEE International Conference on Robotics & Automation*, pp. 2997–3004.

Årzén, K. (2014). "Real-time control systems". In: Department of Automatic Control Lund University. Chap. 10. Continuous Control Loops.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* November 2018 *Document Number* TFRT- 6070
Author(s) Christopher Tvede-Möller	*Supervisor* Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Charlotta Johnsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Robot-held camera platform for medical applications

Abstract

In this thesis a robot-held platform is developed. The final goal of the platform is to follow a potential surgeon's head during operation while cameras mounted on the platform film the ongoing operation from the perspective of the surgeon. In this thesis the original concept developed by Emma Andersson and Anna Wikström is developed further by extending the design to allow placement of sensors and by implementing the control algorithms necessary for the platform to follow the potential surgeon's movements. The implementation uses inverse velocity kinematics, PD control and mid-ranging control to achieve a proof of concept. The final implementation is able to follow movements, but some work is still necessary to ensure better following of orientation and to ensure that the implementation is safe to introduce to a medical environment.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-46	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/