CHALMERS
UNIVERSITY OF TECHNOLOGY

Robot-Assisted System for Orthopaedic Surgery

Master's thesis in Biomedical Engineering & Systems, Control and Mechatronics

Johanna Gullman & Robert Slipac

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:EENX30

Robot-Assisted System for Orthopaedic Surgery

JOHANNA GULLMAN
ROBERT SLIPAC

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2020

Robot-Assisted System for Orthopaedic Surgery
JOHANNA GULLMAN
ROBERT SLIPAC

Supervisor and Examiner: Yiannis Karayiannidis, Department of Electrical Engineering
Supervisor: Katharina Hausmair, Ortoma

Master's Thesis 2020:EENX30
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Robot-Assisted System for Orthopaedic Surgery
JOHANNA GULLMAN
ROBERT SLIPAC
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Among other factors, the success rate of orthopaedic surgery is linked to the skill and experience of the individual surgeon, and surgical error can lead to severe patient injuries. The use of surgical navigation or robotic guidance systems during orthopaedic surgery, such as arthroplasty and spine surgery, has the potential to increase surgical accuracy and precision, and to ensure a consistent outcome. This can lead to a decrease in complication rates and length of stay in the hospital. Existing surgical robotic guidance systems, which usually consist of highly specialised robot manipulators and software, are expensive to purchase, maintain, and use. The objective of this thesis is to develop a robotic guidance system demonstrator in collaboration with Ortoma AB. The system shall guide a robotic arm equipped with a surgical tool to a preoperatively planned pose obtained from Ortoma's surgical planning software. Contrary to existing robotic systems, this thesis aims to show that it is possible to create a robotic guidance system with an off-the-shelf collaborative robot. The proposed system uses a Universal Robots 10 and a commercially available visual tracking device. The system receives a pre-planned target position from Ortoma's planning software, together with the patient calibration as input. Tracking device-robot calibration is performed by using 3D point correspondences between the two coordinate frames to compute the 3D rigid body transformation aligning them. Differential kinematics are used to calculate joint velocities, which are then used to control the robot in Cartesian space. The presented results show that, after successful calibration, the system can consistently position the surgical tool in the pre-planned target with high accuracy and precision. In conclusion, the proposed system demonstrates the possibility to achieve highly accurate results using standard system components in combination with a surgical planning software.

Keywords: robotics, surgical navigation system, robotic guidance, orthopaedics

Acknowledgements

We would like to thank the company Ortoma for offering us the opportunity to conduct this master's thesis project, and for providing us with essential equipment. We are also eternally grateful for our supervisors, Katharina Hausmair (Ortoma) and Yiannis Karayiannidis (Chalmers), who have given us invaluable help and support along the way. Special thanks to Dr. Gunnar Németh, Professor of Orthopaedic Surgery and chairman of Ortoma's Board of Directors, who helped us gain better insight of the medical procedure and essential considerations, regarding a robot-assisted surgical system, from a surgeon's point of view. Finally, we would like to extend our gratitude towards Rikard Karlsson who helped us with scanning and 3D printing a model of the vertebrae.

Johanna Gullman & Robert Slipac, Gothenburg, August 2020

Contents

List of Figures

List of Tables

Acronyms

3D three-dimensional
API Application Programming Interface
CT computed tomography
DH Dennavit-Hartenberg
DLS damped least-squares
DLT direct linear transformation
DOF degrees of freedom
GUI graphical user interface
LAN local area network
OR operating room
ROS Robot Operating System
SVD singular value decomposition
UR10 Universal Robots 10

1
Introduction

This chapter presents the objective, as well as medical background, of the thesis. Already existing systems and medical limitations are also discussed.

1.1 Using Robots in Medical Applications

While contemplating using robots in medical applications, and more specifically in an operating room (OR), there is a vast number of advantages and disadvantages to take into consideration. Many stakeholders are involved, such as the patient, the healthcare professionals, the hospital, insurance companies, and the healthcare system as a whole. Furthermore, ethical aspects also have to be considered and how the introduction of robots in the OR can affect the individual as well as society.

An essential aspect to take into consideration is the cost associated with robot-assisted surgery. Factors increasing the overall cost are for example the purchasing the system, the maintenance of it, as well as additional training for the surgeons. Furthermore, an increased time consumption can be expected due to setup and calibration procedures [1]. Consequently, there is a possibility that the number of performed surgeries per day decrease which results in less income for the hospital. On the other hand, there could also be cost benefits in the form of reduced complication rates, and thus less revision surgeries needed, as well as shorter stay in the hospital post-operation [1]. Consequently, there will always be a trade-off between the time consumption of the surgery and the probability that complications will require even further treatment or surgery. Thus, for more high risk associated with surgeries it could be argued that a larger time duration is a small price to pay to ensure the safety of the patient.

1.2 Objective

The objective of this thesis is to develop a surgical robotic guidance system demonstrator in collaboration with Ortoma AB. The system shall guide a robotic arm equipped with a surgical tool to a preoperatively planned pose obtained from Ortoma's surgical planning software. Existing surgical robotic guidance systems, which usually consist of highly specialised robot manipulators and software, are expensive to purchase, maintain, and use. Contrary to such systems, this thesis aims to show that it is possible to create a robotic guidance system with an off-the-shelf collaborative robot and a commercially available visual tracking device.

The main approach of the system development is to use inverse differential kinematics, by using input obtained from the visual tracking system. The visual tracking system consists of a stereo camera and fiducial markers, recognisable by the camera. Firstly, camera-robot calibration will have to be performed by using 3D point correspondences between the two coordinate frames to compute the 3D rigid body transformation aligning them. Using differential kinematics, the controller for the manipulator can thereafter be described as a P-regulator with an constant coordination system update. A graphical user interface (GUI) will also be developed in order to facilitate both calibration of the system as well as running the robot manipulator.

1.3 Scope

This project mainly focuses on creating a system where the robot behaves as desired, i.e. is able to move a medical tool to a pre-planned position and orientation, with high precision. A UR10 will be used due to its availability at Chalmers, even though a smaller robot could be more suitable for the application in mind. Tracking the involved components will be done with a stereo camera and fiducial markers, borrowed from Ortoma. Furthermore, although the medical aspects and requirements will be taken into consideration and discussed throughout the whole project, the final system will be far from surgery ready, due to time limitations as well as the fact that it would require knowledge beyond engineering.

1.4 Background: Pedicle Screw Placement

There are several orthopaedic surgery procedures in which a robot-assisted system could be applied. For this thesis the target application is pedicle screw placement in spine surgery.

Transpedicular screws are used in lumbosacral-, lumbar-, thoracolumbar-, and thoracic spinal fusion. During the surgery, several vertebrae are fixed together by inserting screws through the pedicles [2]. Figure 1.1 illustrates the fixation of pedicle screws from an axial and sagittal view respectively.

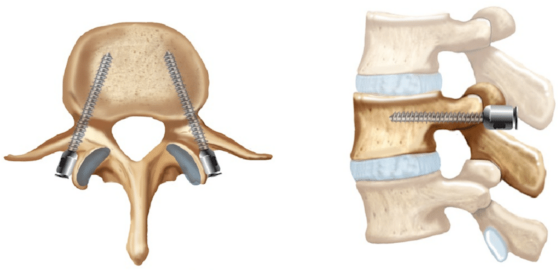

Figure 1.1: Fixation of pedicle screws, axial respectively sagittal view. From [3]. CC-BY.

The pedicle screws are thereafter connected by rods, stabilising and fixating the spine [2]. Figure 1.2 illustrates the resulting lumbar fixation.

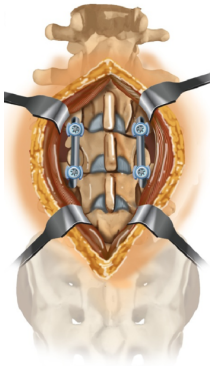

Figure 1.2: Lumbar fixation with pedicle screws and connecting rods. From [3]. CC-BY

In figure 1.3 the second lumbar vertebra, L2, is illustrated. The vertebral body is connected to the laminae by the two pedicles [4].

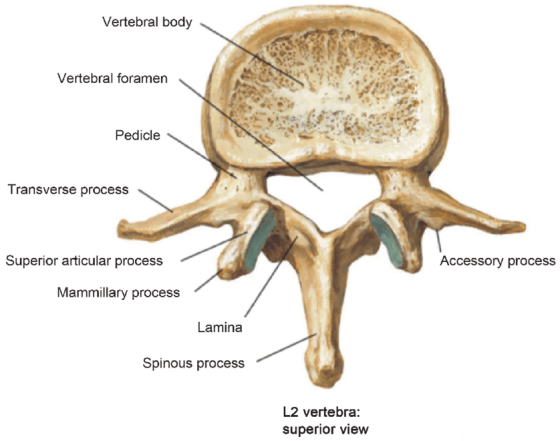

Figure 1.3: Superior view of the L2 vertebra. From [5]. CC-BY.

Normally, the pedicle screws are placed by hand while using fluoroscopy to facilitate finding the right location. Both the size of the pedicle and the size of the screw affects the margin of error during the surgery. However, a pedicle violation below 2 mm is generally considered a safe zone of pedicle perforation by most surgeons [6]. For a robot-assisted surgery system, this means that the accuracy of the whole system chain has to be within 2 mm, including every possible aspect that could potentially cause errors affecting the accuracy. These aspects include, but are not limited to, computed tomography (CT) scans, data processing, planning, patient alignment, calibration, measurement systems, manufacturer tolerances, robot positioning, and human error. The success rate of the manual placement is highly linked to the skill and experience of the individual surgeon [7]. Screw malpositioning has been found to be relatively common, with misplacement rates up to 40-50 % [8], [9], [10]. Even though serious complications as a consequence of screw misplacement are

rare they include neurological, visceral, or vascular injuries [9]. Such injuries could have severe long-term consequences for the patient and could definitely decrease their quality of life. Therefore, there is potential to reduce the risk of complications with the use of a robot-assisted surgery system.

During pedicle screw placement the patient is fixed to the table but the respiratory motion could still impair correct placement [11]. The use of a robot therefore has potential to adjust accordingly and move along with the movement, with a higher accuracy than a surgeon. Where to place the screws will not be decided by the robot, but rather by the surgeon before or during the operation. Thus, the robot should only move a medical instrument into the pre-planned pose and perform high accuracy drilling and placement.

1.5 Existing Systems

Laying the foundation for the thesis is Ortoma's current surgical navigation system, Ortoma Treatment Solution™[12]. The first step when using the system consists of preoperative planning of the procedure, by visualising it on a digital three-dimensional (3D) model of the patient, obtained from a CT scan [13]. In the OR the stereo camera and screen are positioned and the hardware is calibrated. By equipping the medical instruments with fiducial markers, recognisable by the camera, the patient's anatomy can be aligned with the computer software. A reference point is also established by fixating a marker on the patient, which is necessary in the case of unforeseen movements. The procedure can then be visualised on the screen and the medical instruments can be tracked, in relation to the patient's anatomy, in real time [13]. Thereby, the surgeon can easily confirm correct positioning of the instruments during the surgery, which is further facilitated by the instrument turning green on the screen when correctly positioned.

When using a robotic guidance systems, the initial steps are usually similar to a navigational system. Based on a CT scan of the patient, a plan is made preoperatively. Using fiducial markers, or other detectable landmarks, and a stereo camera, the robot- and camera coordinate systems can be aligned [14]. Thereafter, the system can align the robot arm in the pre-planned position and orientation of the pedicle screw. An example of a robotic guidance systems for spine surgery is Mazor X™(Medtronic PLC). The system consists of a workstation, as well as a robotic arm with a built in camera, and a screen, both mounted to the operating table [15]. The system can plan the surgery either preoperatively or intraoperatively with their so called "Scan-and-Plan" mode. Studies have concluded the system to be safe and reliable, although not significantly superior to a surgical navigation system [15], [16]. Other robotic guidance systems are the ExcelsiusGPS (Globus Medical) and the ROSA ONE Spine System (Zimmer Biomet) [17]. These two systems both consist of a robotic arm and a stereo camera for navigation, and planning of the surgery is carried out by either CT scan or intraoperative fluoroscopy. All three aforementioned robotic guidance systems have in common that the robotic arm is equipped with some form of guidance tool. The system then positions the robot such that the

surgeon can insert the medical instrument through the guidance tool, and thereby perform the procedure in the desired position and with the correct orientation. In other words, the robot does not cut or drill in the patient, but enables the surgeon to do it with high accuracy and precision.

1.6 Medical Limitations

To implement a robotic arm in an OR requires knowledge about the environment and the rest of the medical equipment. There are some medical devices that are essential to keep close to the patient during an operation. For instance, an oxygen system and computer screens showing necessary information about blood pressure, heart rate, etc [7]. When implementing a robotic manipulator in such an environment it is important to feed information about the surroundings to the manipulator, in order to avoid unnecessary collisions. The robot that will be used in this project does not have any sensors to detect surrounding objects. Consequently, some assumptions and restrictions have to be made, such as assuming that the medical devices are always placed approximately at the same position each surgery. One part of this project will be to add the position of the medical equipment to the mathematical model for the controller, in order to give the manipulator some input of the working environment. The working area restricts the possible movements of the manipulator and is considered a limitation in this project.

During an actual surgery, the preoperative planning of the goal pose is done by a surgeon and patient calibration, i.e. the alignment between patient and the coordinate system used during the planning, is performed using an optimisation algorithm. However, in this project a simplified patient alignment will be carried out when creating a demonstration of the system.

Furthermore, since the surgery is intended to be carried out on a human, with a very small margin of error, accuracy and precision are both highly important for this project. The patient is expected to be under general anesthesia during this process, but still move slightly due to respiration with help of a ventilator. Muscle contractions, causing for example leg- or arm twitching are not expected since the patient only receives enough air to breathe during a surgery. The only expected movement is therefore the thorax, where the breathing takes place. This movement can be highly restricted since it depends on how much air is be pumped into the patient. It is possible to optimise the airflow entering the patient, in order to restrict respiratory movements, however this topic is also considered outside the scope for this project. The robotic manipulator is able to work with high precision, which means that optimisation of the breathing could be taken into consideration when implementing such a solution in a real operating room, in order to achieve optimal results. The robot has to reach the target with respect to the patient, and consequently has to be able to dynamically adapt to changes of the target position, based on the movements of the patient.

1.7 Relevant Research Articles and Literature Review Directions

An essential first step, to be able to turn the existing navigation system into a robotic guidance system, is to determine the relationship between the coordinate systems of the robot respectively the camera. This can be done by using 3D point correspondences between two sets, to compute the 3D rigid body transformation aligning them. In [18], Eggert et al. compares four algorithms for solving this problem. All algorithms compute the transformation, i.e. the rotation and translation, between two point sets by formulating and solving the problem as a least-squares problem. The difference between the four algorithms lies in how the transformation is represented, and how a criterion function is minimised. The different solutions are based on the following: 1) computing the singular value decomposition (SVD) of a matrix, 2) using orthonormal matrices, 3) using unit quaternions, and 4) using dual quaternions. In the paper, it was concluded that all algorithms result in a similar accuracy and robustness, with negligible differences even in applications with low noise levels.

Due to the limited space in the OR, and the fact that the visual tracking requires the markers to be in view for the camera, it is reasonable to restrict the movement of the robot manipulator. This can be done by implementing joint constraints, also known as "joint limit avoidance", for example by implementing bounds on their speed or angles. In [19], Atawnih et al. proposes a kinematic control signal that guarantees joint limit avoidance. The control signal is based on the prescribed performance control method, and it can be applied to either planned trajectories or sensor driven tasks with trajectories generated online. Tests, carried out with a 7 degrees of freedom (DOF) robot manipulator, also verified the possibility to obtain smooth joint trajectories and to accurately reach the target. In [20], the closed-loop inverse kinematics algorithm laid the foundation for Wang et al. when deriving the inverse kinematics and control of a 7 DOF redundant robot manipulator. Here, joint limit avoidance is used as a performance criterion, which is locally optimised in order to find the redundancy resolution. This is done with the Gradient Projection Method.

Inverse differential kinematics requires a Jacobian of full rank, and therefore problems arise when the robot manipulator comes close to kinematic singularities [21]. Instead of inverting the differential kinematics, the problem can then be reformulated as a damped least-squares (DLS) problem. In [22], several DLS methods are implemented and tested. The different versions are a basic DLS scheme, a weighted DLS solution, and the addition of a feedback correction term. It was concluded that the proposed refinements could improve the basic method.

2
Background

This chapter introduces the hardware and software that will be used to carry out the thesis.

2.1 Camera and Markers

The camera being used is the Atracsys fusionTrack 250. This is a stereo camera, tracking so called "markers" in real-time, with a sample frequency of 120 Hz [23]. The markers are tracked in 6D, meaning that both position and rotation of the markers are tracked, in the camera's reference frame. From the camera software, each marker's homogeneous transformation matrix, describing the transformation from marker to camera frame, is extracted. This data is used to obtain relative poses of the markers with respect to each other. The camera software was already implemented and we made no changes to it during this project. The camera has a C++ Application Programming Interface (API) that can be used via a Python wrapper provided by the manufacturer. The Python interface is used to obtain measurement data from the camera.

There are a number of different markers that Ortoma uses, each with their own unique geometry. The ones used for this thesis are called *marker 2*, *marker 4*, and *marker 9*. During this project they are also denominated depending on their use, such as "goal/reference marker", "tool marker" and "calibration unit marker". Each marker is equipped with four reflective discs, also known as fiducials, recognisable by the camera.

Figure 2.1: A marker, equipped with four reflectors which the camera detects.

A marker is attached to each medical instrument, as well as one reference marker

attached to the patient, and thus the pose of the instruments and the patient, relative to the camera, is known throughout the surgery.

2.2 Medical Instrument and Calibration Unit

Various medical instruments can be equipped with a marker and then calibrated with a calibration unit acquired from Ortoma. The calibration unit, shown in Figure 2.2, has three available calibration points, each purposed for a different tool, as well as an attachment site for calibration unit marker (marker 9) which is what the camera recognises. Additionally, Ortoma has also provided the different transformation matrices between calibration unit marker and the three calibration points.

Figure 2.2: Pointer tool and calibration unit.

In this case the tool to be integrated with the surgery system is a pointer tool, to which tool marker (marker 4) can be attached.

2.3 Robot Manipulator

A robot manipulator is made up of a number of links which are connected by joints, enabling mobility [21]. When a single series of links connects the first- and last part of the manipulator, this is called an open kinematic chain. In this case the DOF is also equal to the number of joints. Alternatively, when the links form a loop, this is known as a closed kinematic chain [21].

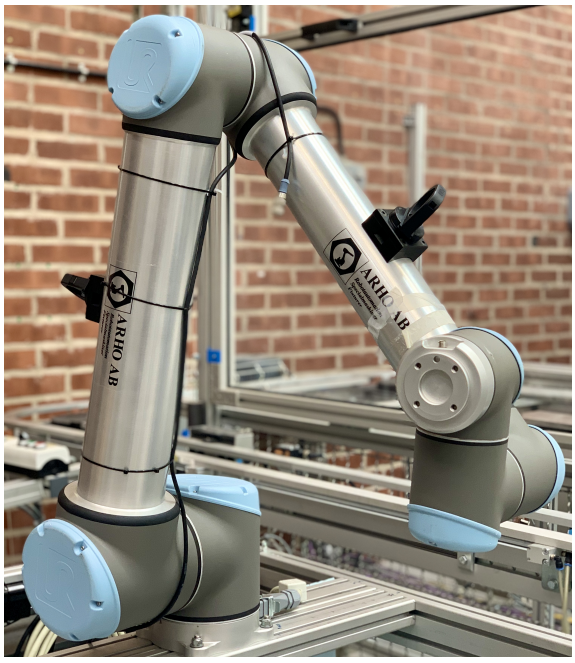

Figure 2.3: The UR10.

The robot used in this project is the UR10, illustrated in Figure 2.3. This is a collaborative robot arm with six DOF, enabling positioning and orientation in three-dimensional space. The six revolute joints are called base, shoulder, elbow, wrist 1, wrist 2, and wrist 3 [24]. The last link of a robot manipulator is also known as the end-effector [21].

The UR10 also has a touch screen, called "teach pendant", from which it is possible to adjust joint angles. The teach pendant is also equipped with an emergency stop button.

2.4 Robot Operating System (ROS)

The UR10 has a driver enabling it to be used with the Robot Operating System (ROS). ROS is an open source framework extensively used in robotics and computations in ROS are done by a network designed in a node graph architecture, meaning that it consists of several nodes which when connected together form a graph. A robotic system can have communication setup with several components through ROS using nodes, where each node can be seen as a program, performing a specific computation or task [25].

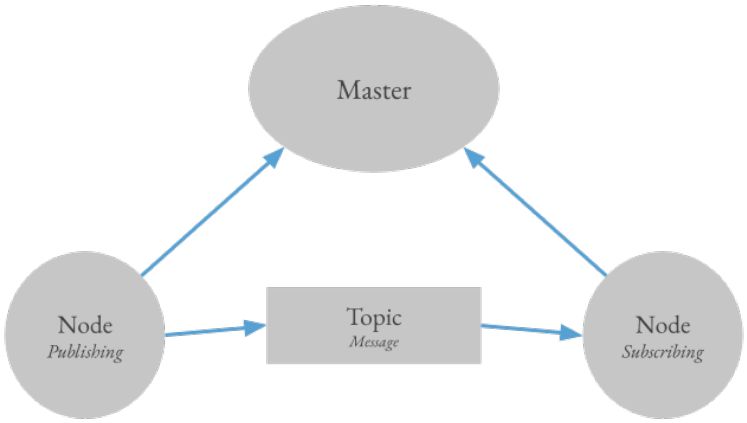

Figure 2.4: The ROS node graph architecture.

In the computation graph, illustrated in Figure 2.4, the ROS master enables communication and date exchange between nodes [25]. The nodes are the programs performing the task at hand, communicating through messages. A node is said to publish to a topic when sending a message through a data bus, where the data bus is known as a topic, and to subscribe to a topic when receiving a message [25]. The nodes are also possible to program in various programming languages, out of which Python was the choice for this thesis.

3
Theory

This chapter covers the mathematical concepts used and implemented in the system.

3.1 System Behaviour and P-regulator

The robot manipulator is implemented with a controller to predetermine how the system is expected to behave. A robot manipulator can be controlled in the task space or in the joint space. In task space, or Cartesian space, the end-effector pose is expressed in terms of position and orientation. In joint space, the robot pose is expressed in terms of angular displacement of each joint [21]. In this project, Cartesian space control is used, with joint velocities as a control input. The joint velocity is calculated, using inverse differential kinematics that maps a Cartesian velocity defined as a P-regulator, and then published with a known frequency in order to get the desired behaviour. The inverse differential kinematics use the end-effector orientation and position to calculate joint velocities, while the P-regulator is used to predetermine the Cartesian velocity of the end-effector. The end-effector is the end of the last link of the robotic arm, which is used to interact with the environment.

The control law for the P-regulator, which calculates the desired end-effector velocity, is modelled as a spring that pulls the end-effector back towards a desired pose. This is formulated as in equation (3.1).

$$\boldsymbol{v} = -k(\boldsymbol{p} - \boldsymbol{p}_d) \tag{3.1}$$

where $(\boldsymbol{p} - \boldsymbol{p}_d)$ is the error between the current pose $\boldsymbol{p}$ and the desired pose $\boldsymbol{p}_d$. This is multiplied with a constant gain k. The error is expressed as a combination of a positional- and a rotational part, $\boldsymbol{p} - \boldsymbol{p}_d = \begin{bmatrix} x & y & z & q_1 & q_2 & q_3 \end{bmatrix}$, where x, y, z are the positions along respective axis and q_1, q_2, q_3 are the quaternions describing rotation. The aim is to minimise the error, and therefore minimise the distance between the end-effector and a goal position, in x-, y-, z-directions and quaternions. The minimisation converges the positional- and rotational part towards the goal at the same time if k is scalar. If the gain of the controller is implemented as a matrix instead, it is possible to decide which convergence needs a higher priority.

3.2 Differential Kinematics

This section describes the finalisation of the Cartesian space controller, where the P-regulator has been implemented, describing a predetermined velocity for the end-

effector in the Cartesian coordinates system. The desired joint velocity is then calculated using the inverse differential kinematics and the P-controller. With differential kinematics, the relationship between a robotic manipulator's joint velocities and the corresponding end-effector's linear- and angular velocities can be derived. In order to obtain the differential kinematics, one must start with forward kinematics and then derive the differential kinematic relations [21].

With forward kinematics, the end-effector pose is calculated as a function of the joint variables. Using the Dennavit-Hartenberg (DH) convention, the homogeneous transformation matrix from base to the end-effector can be derived [21]. The homogeneous transformation shows the rotation and translation from the base to the end-effector. To get this transformation matrix, the transformation matrices describing the kinematic relationship between two consecutive links have to be derived first. This can be done with the DH parameters, assuming the geometry of the robotic manipulator is known. The measurements of the robot can be obtained through the official Universal Robots website [26]. By multiplying the transformation matrices, expressing relative position and orientation of two consecutive links, the homogeneous transformation matrix from base- to end-effector frame can be calculated [21].

The end-effector's linear- and angular velocities are related to the joint velocities by the Jacobian matrix according to the differential kinematics equation:

$$\boldsymbol{v} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{3.2}$$

where $\boldsymbol{v}$ is the end-effector velocities and $\dot{\boldsymbol{q}}$ the joint velocities [21]. In turn, to derive the joint velocities from the linear velocities, the Jacobian needs to be inverted as follows:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\boldsymbol{v} \tag{3.3}$$

However, to use equation (3.3) the Jacobian has to be square and of full rank [21]. In case of kinematic redundancy, i.e. when the DOFs of the manipulator is larger than the required number of variables for a task, the Jacobian is no longer square. Therefore, the Moore-Penrose pseudo inverse is introduced.

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{J}^{\dagger}\boldsymbol{v} \\ \boldsymbol{J}^{\dagger} &= \boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{J}^{T})^{-1} \end{aligned} \tag{3.4}$$

where $\boldsymbol{J}^{\dagger}$ is the right pseudo-inverse of $\boldsymbol{J}$ [21]. Equation (3.4) describes the joint velocities based on the joint angles, in relation to the 6 DOF velocity given [21]. The 6 DOF velocity is a pre-planned velocity vector based on the controller.

A known problem in task space robot control is the singularity problem. The aforementioned six revolute joints of the UR10 gives, for this particular robot, a 6×6 Jacobian matrix which, in non-singular cases, is an invertible square matrix. Some singular poses, such as close to fully extended joints, causes problems for the robot.

3.3 Damped Least-Squares (DLS) Method

When a pose comes close to a singularity, the Jacobian is at some point an non-invertible square matrix which can lead to a large effect being the result of a small change [27]. In a practical example, the robotic arm obtains maximum values in the Jacobian matrix, resulting in maximum joint velocities. Singularities could be avoided by using a pre-planned trajectory, but since the objective of this project is to follow the reference marker (marker 2) to reach a target position with respect to the marker, assuming that the reference marker can be moved, this creates countless possible trajectories. Thus, a better and flexible solution should be implemented. One potential solution is the DLS method, where the Jacobian in (3.4) is slightly used for the inverse mapping [27].

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^* \boldsymbol{v} \tag{3.5}$$

where $\boldsymbol{J}^*$ is,

$$\boldsymbol{J}^* = \boldsymbol{J}^\top (\boldsymbol{J}\boldsymbol{J}^\top + \lambda^2 \boldsymbol{I})^{-1} \tag{3.6}$$

$\boldsymbol{J}$ is the Jacobian for this particular robot. The damping factor λ is an input variable which can be adjusted, a perfectly balanced λ results in the system achieving singularity avoidance while at the same time still converging as intended. If the parameter λ is too large, the convergence might not be achieved, that is the end-effector can position itself differently than the goal target. However, if the λ is too small, the system is not affected at all and singularities are thus not avoided. Therefore, it is important to find a balanced value when choosing λ. Verification of this claim can be found in Section 6 of this report.

3.4 Camera-Robot Calibration

The goal is to create a controller which uses measurement data from the stereo camera as an input. To be able to use the controller for the manipulator, using the stereo camera, the different coordinate systems need to be aligned. The main problem is that the goal pose, $\boldsymbol{p}_d$, is given in the goal marker's coordinate system. Furthermore, measurement data is registered in the camera coordinate system, while the the manipulator controller needs this data in the robot coordinate system. Therefore, the unknown relationship between camera- and robot coordinate frames has to be identified. The different coordinate systems, i.e. that of the robot base, the camera, and a marker, are illustrated in Figure 3.1.

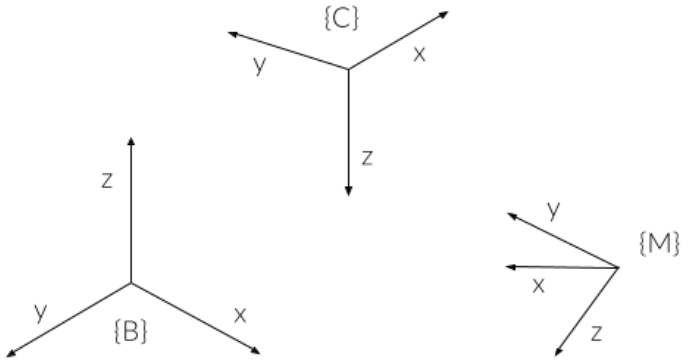

Figure 3.1: The coordinate systems for the robot base, the camera, and a marker ($\{B\}$, $\{C\}$, and $\{M\}$ respectively).

From the camera software it is possible to extract the markers' homogeneous transformation matrix, indicating position and orientation in the camera coordinate system. The rotational part of a transformation matrix can be denoted as $\boldsymbol{R}_i^j$, meaning a rotation matrix of frame i with respect to frame j.

As a first attempt to a solution it is possible to place a marker at the end-effector of the robot manipulator, in order to interpret the current end-effector pose $\boldsymbol{p}$ in the same coordinate system as the camera ($\boldsymbol{p}^c$). The goal, $\boldsymbol{p}_d$, is the current target for the error minimisation process. This gives a difference between target and current pose, which results in a velocity vector. With these parameters the controller can now be described as below.

$$\boldsymbol{v}^c = -k(\boldsymbol{p}^c - \boldsymbol{p}_d^c) \tag{3.7}$$

where $\boldsymbol{v}^c$ is the end-effector velocities expressed in the camera coordinate frame and k is the gain of the P-controller, i.e. a variable to adjust the speed. Thereafter it is simple to use the same velocity control but with the correct rotation matrix.

$$\boldsymbol{v}^b = \boldsymbol{R}_C^B \boldsymbol{v}^c \tag{3.8}$$

where $\boldsymbol{v}^b$ is the end-effector velocities expressed in the robot base coordinate frame. In order to fully control the manipulator with the impedance controller, one needs an estimate of the rotation matrix $\boldsymbol{R}_B^C$, which describes the relationship between the camera coordinate system and the coordinate system of the robot manipulator base.

Knowing the coordinate system of the robot base, one way to estimate the rotation matrix is to position the camera in a certain pose, followed by hard coding its orientation. Positioning the camera facing the manipulator in a straight line, without any tilt around the camera coordinate system's x- or y-axis, makes it possible to align the z-axes of the camera- and robot base coordinate systems. By moving a marker along the x- and y-axis of the robot frame, while observing the translation vector of the transformation matrix, describing the relationship between marker and camera frame, the corresponding axis and direction in the camera frame can be determined. With this information, the base coordinate system, expressed in the camera coordinate system, can be identified.

Hard coding the fixed rotational values works as long as the camera remains in exactly the same pose. However, since it is possible that the camera or robot are moved during the surgery, the goal is to develop a solution more general for any camera pose. A more general solution to estimate the rotation matrix, between the camera frame and robot base frame, for any camera pose, is to formulate the problem into a homogeneous least squares problem, and solving this using the SVD algorithm.

3.4.1 System Coordinate Frames

The system's different coordinate frames are illustrated in Figure 3.2, in which the notations B, C, E, M, and T stands for the robot base, the camera, the end-effector, the marker attached to the medical tool, and the tooltip respectively.

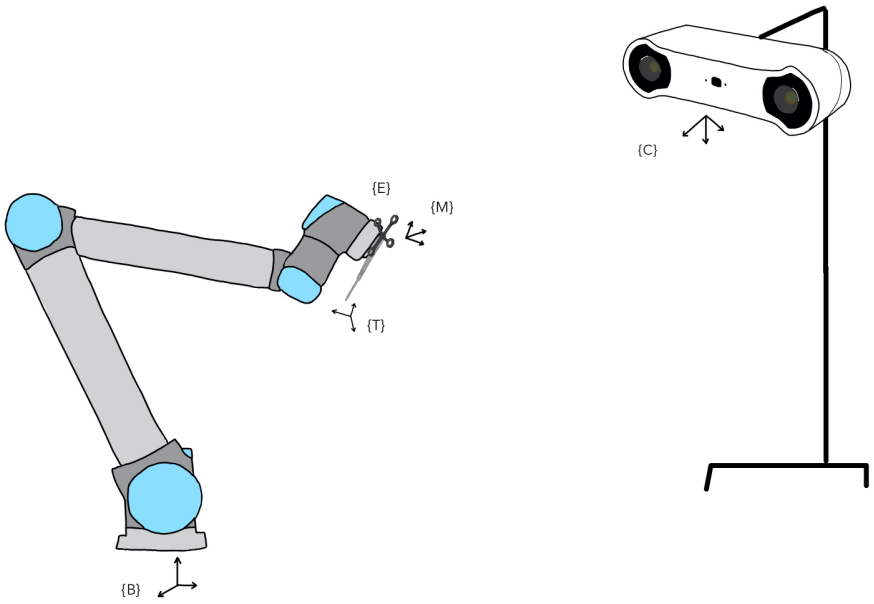

Figure 3.2: Illustration of the different coordinate frames.

The relationship between the homogeneous transformation matrices, and similarly the relationship between the rotation matrices, can be derived as follows.

$$\begin{aligned} \boldsymbol{A}_B^C &= \boldsymbol{A}_M^C \boldsymbol{A}_E^M \boldsymbol{A}_B^E \\ \boldsymbol{R}_B^C &= \boldsymbol{R}_M^C \boldsymbol{R}_E^M \boldsymbol{R}_B^E \end{aligned} \tag{3.9}$$

The transformation matrices, and the corresponding rotation matrices, $\boldsymbol{A}_B^C$ and $\boldsymbol{A}_E^M$ are unknown and have to be derived, whereas $\boldsymbol{A}_M^C$ and $\boldsymbol{A}_B^E$ are known and can be derived online if they change.

3.4.2 Homogeneous Least-Squares Problem

In order to estimate the rotation matrix $\boldsymbol{R}_B^C$, expressing the transformation between the robot- and camera frame, the first step is to collect 3D points in the camera

coordinate system $\{\boldsymbol{c}_i\}$ and the corresponding 3D points in the robot's coordinate system $\{\boldsymbol{b}_i\}$. Using the pinhole camera model equation a relationship can be built between these coordinates.

$$\boldsymbol{c}_i = \boldsymbol{P}\boldsymbol{b}_i \tag{3.10}$$

A depth is not given for this particular problem since the image points from the camera is described directly in 3D. The objective is to find the relationship of the axis coordinate frames between robot and camera. One can see this as a mathematical expression of the parameter, $\boldsymbol{P} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}$, where in this subject the desired element is $\boldsymbol{R}$. The variable $\boldsymbol{R}$ is the 3×3 rotation matrix, and $\boldsymbol{t}$ the 3×1 translation vector, of the homogeneous transformation matrix which describes the relationship between the two coordinate systems.

The parameter $\boldsymbol{R}$ corresponds to $\boldsymbol{R}_B^C$ in equation (3.8). In order to find the general $\boldsymbol{R}$ between two coordinate frames, equation (3.10) can be reformulated into a homogeneous least squares problem as follows [18].

$$\sum_{i=0}^{N} \left\| \boldsymbol{c}_i - \boldsymbol{R}_B^C \boldsymbol{b}_i - \boldsymbol{t}_b^c \right\|^2 \tag{3.11}$$

By minimising the sum of squares in 3.11, the optimal transformation between the point sets $\{\boldsymbol{c}_i\}$ and $\{\boldsymbol{b}_i\}$ can be found.

The camera points are $\boldsymbol{p}_{c,i} = (x_{c,i}, y_{c,i}, z_{c,i})$ and the robot points are $\boldsymbol{p}_{r,i} = (x_{r,i}, y_{r,i}, z_{r,i})$, for $i = 1, 2, 3, ...N$.

The averages of the camera- and robot point sets are:

$$\begin{aligned} centroid_c = \bar{\boldsymbol{p}}_c &= \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_{c,i} = \begin{bmatrix}\bar{x_c} & \bar{y_c} & \bar{z_c}\end{bmatrix}^T \\ centroid_r = \bar{\boldsymbol{p}}_r &= \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_{r,i} = \begin{bmatrix}\bar{x_r} & \bar{y_r} & \bar{z_r}\end{bmatrix}^T \end{aligned} \tag{3.12}$$

Calculating the Euclidean distance between each point and the centroid of the set which it belongs to, and appending 1 to the camera points, results in the matrices containing the new points $\mathbf{p}'_{c,i}$ and $\mathbf{p}'_{r,i}$.

$$\begin{aligned} \boldsymbol{p}'_{c,i} = \boldsymbol{c}_i - \bar{\boldsymbol{p}}_c &= \begin{bmatrix}x'_{c,i} & y'_{c,i} & z'_{c,i} & 1\end{bmatrix}^T \\ & \qquad\qquad l \\ \boldsymbol{p}'_{r,i} = \boldsymbol{b}_i - \bar{\boldsymbol{p}}_r &= \begin{bmatrix}x'_{r,i} & y'_{r,i} & z'_{r,i}\end{bmatrix}^T \end{aligned} \tag{3.13}$$

In equation (3.14) the matrix has been modelled to fit this problem specifically. Firstly, using direct linear transformation (DLT), a homogeneous linear system is formulated including all projection equations [28]. However, the points $\boldsymbol{p}'_{c,i}$ in (3.13) are expressed in 3D and have been modelled to fit all corresponding points $\boldsymbol{p}'_{r,i}$ from

the robot frame. In the $\boldsymbol{M}$ matrix below each $\mathbf{0}$ represents a vector [0 0 0 0].

$$\underbrace{\begin{bmatrix} \boldsymbol{p'_{c,1}}^T & \mathbf{0} & \mathbf{0} & -x'_{r,1} & 0 & 0 & \dots \\ \mathbf{0} & \boldsymbol{p'_{c,1}}^T & \mathbf{0} & -y'_{r,1} & 0 & 0 & \cdots \\ \mathbf{0} & \mathbf{0} & \boldsymbol{p'_{c,1}}^T & -z'_{r,1} & 0 & 0 & \cdots \\ \boldsymbol{p'_{c,2}}^T & \mathbf{0} & \mathbf{0} & 0 & -x'_{r,2} & 0 & \dots \\ \mathbf{0} & \boldsymbol{p'_{c,2}}^T & \mathbf{0} & 0 & -y'_{r,2} & 0 & \dots \\ \mathbf{0} & \mathbf{0} & \boldsymbol{p'_{c,2}}^T & 0 & -z'_{r,2} & 0 & \dots \\ \boldsymbol{p'_{c,3}}^T & \mathbf{0} & \mathbf{0} & 0 & 0 & -x'_{r,3} & \dots \\ \mathbf{0} & \boldsymbol{p'_{c,3}}^T & \mathbf{0} & 0 & 0 & -y'_{r,3} & \cdots \\ \mathbf{0} & \mathbf{0} & \boldsymbol{p'_{c,3}}^T & 0 & 0 & -z'_{r,3} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \\ t_1 \\ t_2 \\ t_3 \\ \vdots \end{pmatrix}}_{\boldsymbol{v}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.14}$$

It is then desired to find a non-zero vector in the nullspace of matrix $\boldsymbol{M}$. If there is no exact solution to $\boldsymbol{Mv} = 0$ the estimated rotation matrix is found by solving the following homogeneous least-squares problem.

$$\min_{\|v\|^2=1} \|\boldsymbol{Mv}\|^2 \tag{3.15}$$

The least-squares problem is modelled to minimise the error between the corresponding 3D points of the two coordinate systems. A proposed solution is to use the SVD method to find the transformation matrix describing the relationship between the coordinate systems.

3.4.3 Singular Value Decomposition (SVD)

An $m \times n$ matrix $\boldsymbol{M}$ can be multiplied with its transpose to form the symmetrical square matrices $\boldsymbol{MM}^T$ and $\boldsymbol{M}^T\boldsymbol{M}$, which have the same positive eigenvalues and are both positive semidefinite [29]. $\boldsymbol{u}_i$ are the eigenvectors of $\boldsymbol{MM}^T$, and $\boldsymbol{v}_i$ the eigenvectors of $\boldsymbol{M}^T\boldsymbol{M}$. Furthermore, the square root of the non-negative eigenvalues are called singular values, labeled $\sigma_\mathbf{i}$. The SVD calculation results in a square root which has two possible solutions, where one solution is negative and the other is positive. The SVD calculation is performed on the aforementioned $\boldsymbol{M}$ matrix in equation (3.14).

Using SVD, the matrix $\boldsymbol{M}$ can be factorised into $\boldsymbol{M} = \boldsymbol{USV}^T$, where $\boldsymbol{U}$ is an orthogonal $m \times m$ matrix containing the eigenvectors $\boldsymbol{u}_i$ as its columns, and $\boldsymbol{V}$ is an orthogonal $n \times n$ matrix with $\boldsymbol{v}_i$ as its columns [29]. $\boldsymbol{S}$ is a diagonal $m \times n$ matrix containing the singular values in descending order, i.e. $\sigma_1 \geq \sigma_2 \dots \geq \sigma_r \geq 0$:

$$\mathbf{S} = \begin{pmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} \tag{3.16}$$

The $\boldsymbol{V}$ and $\boldsymbol{U}$ matrices both contain the data for reconstructing the desired rotational matrix. From computations it can be derived that the solution to the optimisation problem (3.15) should be the eigenvector $\boldsymbol{v}_i$ of $\boldsymbol{M}^T\boldsymbol{M}$, corresponding to the smallest eigenvalue. This is equivalent to finding the column of $\boldsymbol{V}$ with the smallest singular value, i.e. the last column. Thus, the estimated rotation matrix between camera- and robot frame can be extracted from the last column of the $\boldsymbol{V}$ matrix. Once extracted, the 12×1 vector can be reshaped into a new $\boldsymbol{P}$ matrix, of size 3×4, which contains the parameters of $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$. Where $\boldsymbol{R}$ is the unknown 3×3 rotation matrix $\boldsymbol{R}_B^C$ required to obtain the relationship between the two coordinate systems, and $\boldsymbol{t}$ the 3×1 translation vector. With the help of SVD the transformations can be calculated for any camera pose. The homogeneous transformation matrix $\boldsymbol{A}_B^C$ can also be expressed as follows.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3.17}$$

Thus, a general transformation between the robot frame and camera frame can be extracted and used. This solution is considered an offline solution and can be used as part of a calibration process, before executing a run. The SVD calculation is only performed once, in order to obtain an initial R_B^C.

The information built together with the kinematic relationship results in moving a robotic arm according to a marker which is tracked by a 3D camera. A controller has been built as mentioned in the previous sections. However, taking into consideration that the goal is to integrate the robotic arm in a surgical environment, several disturbances are possible. For instance, disturbances of the camera position in an OR could occur due to the amount of equipment and personnel having to share a relatively small area.

3.5 Movable Camera

It is desirable to make the system insensitive to camera movements during operation, since the camera pose can change if it is blocking other equipment and therefore has to be moved. Additionally, accidental collisions between medical personnel and the camera could occur during surgery, causing the camera pose to change. The different coordinate frames are visualised in Figure 3.2. If the camera were to move, the matrices $\boldsymbol{R}_B^C$ and $\boldsymbol{R}_M^C$ would change. However, the matrix $\boldsymbol{R}_E^M$ would stay the same

due to the static relationship between the end-effector and the marker attached to it.

With the rotation matrix $\boldsymbol{R}_B^C$, obtained by aligning the camera- and robot coordinate systems using SVD, and the readily available values of $\boldsymbol{R}_B^E$ and $\boldsymbol{R}_M^C$, equation (3.9) can then be solved for the static rotation matrix $\boldsymbol{R}_E^M$.

$$\boldsymbol{R}_E^M = \boldsymbol{R}_M^{C\,-1}\boldsymbol{R}_B^C\boldsymbol{R}_B^{E-1} \tag{3.18}$$

The SVD calculation is performed once, offline, and then the static $\boldsymbol{R}_E^M$ in (3.18) is derived. Thereafter, the rotation matrix between the camera- and robot frames can once again be calculated, this time by inserting the now known values of the variables in equation (3.9). In other words, $\boldsymbol{R}_B^C$ can now be calculated even if the camera has been moved, by using the known altered values of $\boldsymbol{R}_M^C$ and $\boldsymbol{R}_B^E$, as well as the calculated $\boldsymbol{R}_E^M$. As previously mentioned, the values of $\boldsymbol{R}_M^C$ and $\boldsymbol{R}_B^E$ can be derived online, i.e. when the system is running, which means that the camera could be accidentally moved repeatedly during operation and the system would still adjust accordingly. However, since the calculated $\boldsymbol{R}_E^M$ depends on the $\boldsymbol{R}_B^C$ initially obtained from the collected 3D points, it is important that the camera is not moved during the points collecting procedure.

Combining the presented theory, the system can now be considered as a kinematic controller, with a regular P-controller, that responds to a marker detectable by a stereo camera. This system can be calibrated for any camera pose and adjust itself when disturbances occur. The goal of this project is to use a medical tool, attached at the end-effector of the robotic manipulator, and successfully point at a given target. Using any tool at the end-effector means adding another transformation to the current equation (3.9).

3.6 Joint Constraints

The robotic arm can now position the end-effector, with a tool attached to it, based on the location of markers tracked by the stereo camera. Theoretically, the robot has the ability to place the tip of the tool in any pre-planned pose in a surgical environment. However, there are several spatial constraints that have to be considered when operating in a surgical environment. For example, the patient will be lying on a surgical table. Therefore, the mathematical model is adjusted to avoid collisions. The kinematic controller is, as described in equation (3.3), $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\boldsymbol{v}$. An adjustment is made to this equation by adding joint constraints, resulting in the following.

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{J}^{-1}\boldsymbol{v} + (\boldsymbol{I} - \boldsymbol{J}^{-1}\boldsymbol{J})\dot{\boldsymbol{q}}_N \\ \phi(\boldsymbol{q}) &= \sum_{i=1}^{n}(\frac{\boldsymbol{q}_i - \boldsymbol{q}_{c_i}}{\Delta \boldsymbol{q}})^2 \\ \dot{\boldsymbol{q}}_N &= \nabla_{\boldsymbol{q}}\phi(\boldsymbol{q}) \end{aligned} \tag{3.19}$$

where n is the number of joints, i denotes the current joint, Δq is the difference between maximum- and minimum joint limit, and q_{c_i} is the middle value of the joint range. However, this implementation does not guarantee full avoidance of the joint limits since the error minimisation, resulting in the variable $\boldsymbol{v}$ in the first row of (3.19), has a larger impact when sending joint velocities to the robot manipulator than the second part of the equation. The soft constraints $\boldsymbol{q}_{min_i} \leq \boldsymbol{q}_i \leq \boldsymbol{q}_{max_i}$, do not guarantee that the limits are avoided, where $\boldsymbol{q}_{min_i}$ and $\boldsymbol{q}_{max_i}$ are set to be the boundaries of the joints. This implementation constrains the joints to avoid certain angels during movement to the goal pose, resulting in the following joint velocities [19].

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^* \boldsymbol{v} + (\boldsymbol{I} - \boldsymbol{J}^* \boldsymbol{J}) \dot{\boldsymbol{q}}_N \tag{3.20}$$

The second part of equation (3.20) contains the Moore-Penrose pseudo-inverse of the Jacobian, which is the projection operation in the null space. In order for this implementation to affect the system, the robot manipulator should be redundant, i.e. having more DOFs than necessary for the task. Thus, the Jacobian requires a change where at least one DOF is removed. Without redundancy, the current DOFs can be expressed in terms of the variables $\begin{bmatrix} x & y & z & \varphi & \theta & \psi \end{bmatrix}$, where x, y and z all represent a different axis and φ, θ, ψ are the corresponding angular rotations for the axes.

Like previously mentioned, these are soft constraints, meaning that the system will try to satisfy them but, in contrast to when using hard constraints, the system will not come to a complete halt if they are not satisfied [19]. For a system which is actually used in the OR, more advanced collision detection is necessary to avoid harming the patient, medical personnel or damaging equipment.

4

Proposed System

In order to achieve the desired behaviour of the system, the solution was implemented in several steps, gradually adding complexity. Initially, the aim was to minimise the error between the end-effector marker and the goal marker. Thereafter, the medical instrument was added to the end-effector such that the aim was to minimise the error between the instrument tooltip and the goal marker. In a final step, the system was adjusted such that the target position could be set to any pose in the goal marker's coordinate system.

4.1 Setup

This section explains the setup of the hardware and the system communication in detail. The overall system is described in Figure 4.1. The only input to the system is the transformation matrices from markers to camera at a frequency of 120 Hz. The system outputs the joint velocities of the manipulator with a P-controller for the end-effector marker respectively goal marker.

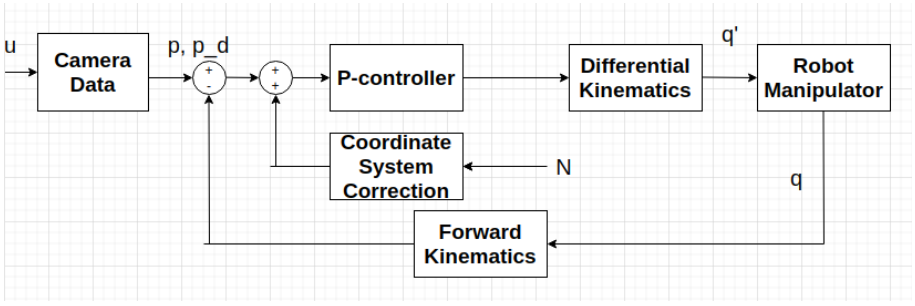

Figure 4.1: Block diagram of the system.

In the block diagram of Figure 4.1, the variable u is the input to the system which in this case is the information of the markers' position and orientation in relation to the camera. The variable N symbolises noise, meaning a disturbance for the camera position during a run. The system tracks the joint angles and joint velocities, which are updated from the error estimated end-effector velocity. The system also gives feedback in the form of system coordinate correction in the case of camera disturbance. The output from the system is the joint velocity.

4.2 Camera in Static Pose

The first step was to calibrate the system for any static camera pose, i.e. not requiring the camera to be positioned linearly on a flat surface but in any position and orientation. However, since the purpose only was to align coordinate frames, it was enough to derive the rotation matrix $\boldsymbol{R}$, from the homogeneous transformation matrix in (3.17). In order to estimate a rotation matrix between the camera- and robot frames 3D points were collected. A marker was taped to the robot end-effector, as centred as possible. The desired number of points were then collected by manually moving around the robot end-effector while also making sure that the attached marker was in view for the camera. The result was 3D points collected in the camera coordinate system, as well as corresponding points in the robot coordinate system. A homogeneous least squares problem was then formulated and solved with SVD to obtain the estimated rotation matrix, $\boldsymbol{R}_B^C$, between the two coordinate systems.

4.3 Movable Camera

Having the same setup as in Section 4.2, with a marker taped to the robot end-effector, 3D points were again collected manually. The rotation matrix $\boldsymbol{R}_B^C$ was also initially derived using SVD. Thereafter, the static rotation matrix $\boldsymbol{R}_E^M$ was calculated as in equation (3.18). Finally, a new transformation matrix between the camera- and robot frames was estimated by solving (3.9), using $\boldsymbol{R}_E^M$, and the readily available $\boldsymbol{R}_M^C$ and $\boldsymbol{R}_B^E$.

4.4 Integration of Medical Instrument

As previously mentioned in Section 2.2, a pointer tool was to be integrated with the surgery system, with the aim to make the tool tip reach the goal pose. The tool was simply attached to the centre of the end-effector as in Figure 4.2, with duct tape. A proper fixture, securely attaching the tool to the end-effector, should be used but priorities were made to derive the theory and verify hypotheses, before building and using additional resources.

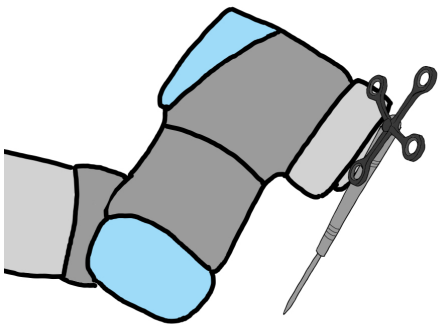

Figure 4.2: The pointer tool attached to the robot end-effector.

Because there is no static way for the tool to be attached, the tip of the tool needs to be re-calibrated each time the robot executes a new operation run. Another option for attaching the tool to the end-effector could be to use a 3D-printer to create a more form fitting structure for attachment, which would result in more possible movements as well.

4.4.1 Tool Calibration

Firstly, the tool itself needs to be calibrated using the stereo camera, markers, and the calibration unit, see Figure 2.2. In opposite to Section 4.2, the whole $\boldsymbol{A}$ matrix in (3.17) is now important since the translation $\boldsymbol{t}$ of the tool tip is highly relevant. On the calibration unit, the distance between the pointer calibration point and the attachment site for the calibration marker is known and has been determined with very high precision. Knowing the transformation matrix between the calibration unit marker and the calibration point on the unit, denoted $\boldsymbol{A}_T^{M_9}$, the pointer tool, equipped with tool marker, is placed with its tip positioned at the calibration point. The 3D coordinate of the tool tip is then the same as the calibration point on the unit and the position of the tooltip, in relation to the attached tool marker, can be calculated as follows:

$$\boldsymbol{A}_T^{M_4} = \boldsymbol{A}_{M_4}^{C\ -1} \boldsymbol{A}_{M_9}^C \boldsymbol{A}_T^{M_9} \tag{4.1}$$

where M_4 is the marker attached at the top of the tool, T is the tooltip, M_9 is the marker attached to the calibration unit, and C is the camera. During calibration, the tip of the tool must be placed exactly in the dedicated calibration point on the calibration unit. Both the marker on the calibration unit and the marker on the tool must be visible to the camera. Calibration is performed only once, for example at the beginning of a surgery, and is valid as long as the marker stays in the exact same position on the tool, i.e. as long as the marker's position with respect to the tool tip is unchanged. Once the relationship has been established using the calibration unit, the new transformation can be added to the system.

Previously derived equations, such as equation (3.9), need to be modified when adding the tool.

$$\boldsymbol{A}_B^C = \boldsymbol{A}_{M_4}^C \boldsymbol{A}_T^{M_4} \boldsymbol{A}_E^T \boldsymbol{A}_B^E \tag{4.2}$$

where the static relationship from end-effector to marker is now replaced with:

$$\boldsymbol{A}_E^T = \boldsymbol{A}_T^{M_4\ -1} \boldsymbol{A}_B^C \boldsymbol{A}_B^{E\ -1} \tag{4.3}$$

The information given so far results in a kinematic controller with a tool at the end-effector which can successfully point at the centre of the goal markers.

4.5 Goal Calibration

The goal of this project is not to, with a 6 DOF robot manipulator, reach the so called "goal marker". Instead, that marker is attached to the patient as a reference,

and the main goal is to reach a certain pose which is known in relation to the goal marker. This point should be a position along the spine, at the vertebrae in which pedicle screws will be inserted. The position and orientation of the screw is planned by surgeons beforehand. A relationship is then established from the goal marker (marker 2) to the real targeted pose.

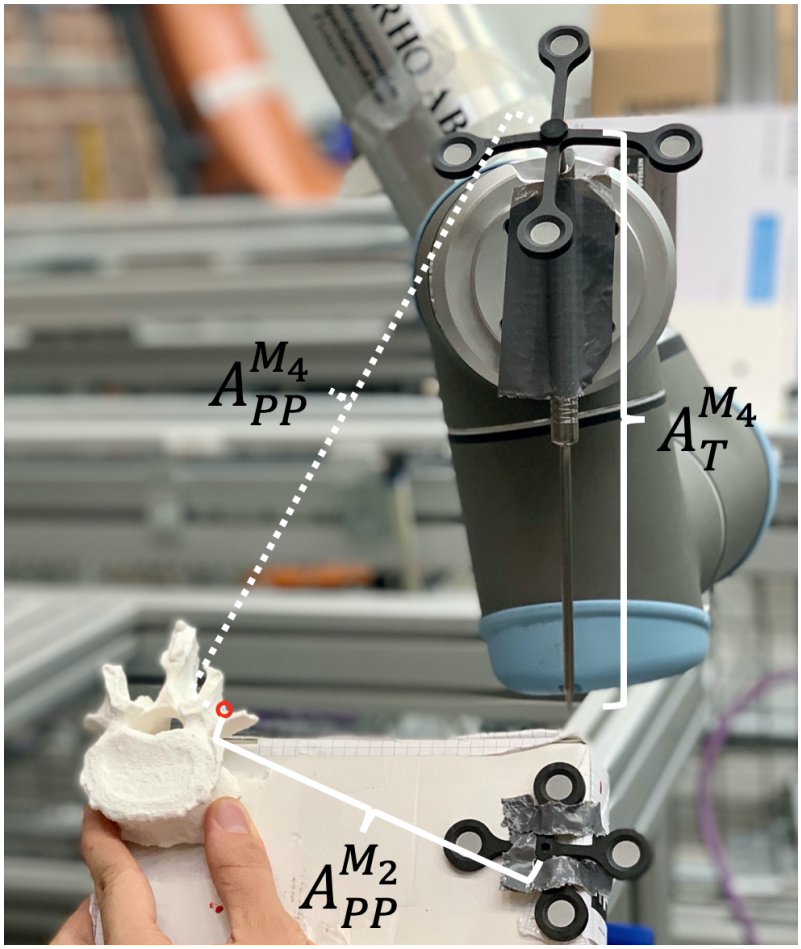

Figure 4.3: The relationships between pre-planned pose (encircled in red) and goal marker, pre-planned pose and tool marker, and between tool tip and tool marker, described by transformation matrices $\boldsymbol{A}_{PP}^{M_2}$, $\boldsymbol{A}_{PP}^{M4}$, and $\boldsymbol{A}_{T}^{M_4}$ respectively.

Considering the control input (3.7), with the target parameter, $\boldsymbol{p}_d$, needs modification by taking the known transformation, from the goal marker to the pre-planned pose, into consideration. As mentioned before, the translation vector is extracted from the transformation matrix $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}$. Where the parameter $\boldsymbol{p}_d$ is the translation vector $\boldsymbol{t}$ of the transformation matrix.

In Chapter 7, a demonstrator with a real pre-planned pose is presented. However, for testing and evaluation, a pre-planned pose will be emulated using the pointer tool, the marker attached to it, and the goal marker. The tool tip will be pointed at the desired position, with the desired orientation, and its pose with the respect to

the goal marker will be saved. The relationship between the goal marker and the tip of the tool, illustrated in Figure 4.3, is described by the transformation matrix $\boldsymbol{A}_{PP}^{M_2}$.

Firstly, the tool has to be calibrated as described in Section 4.4.1, establishing the relationship between tool tip and the marker attached to the tool (marker 4), described by the transformation matrix $\boldsymbol{A}_{T}^{M_4}$ in Figure 4.3. Thereafter, when pointing the tooltip at the desired position and orientation, the planned pose, expressed in the camera frame, can be described with the following transformation matrix.

$$\boldsymbol{A}_{PP}^{C} = \boldsymbol{A}_{T}^{C} = \boldsymbol{A}_{M_4}^{C}\boldsymbol{A}_{T}^{M_4} \tag{4.4}$$

where PP is the pre-planned pose frame , C is the camera, T is the tooltip, and M_4 is the marker attached to the tool. Since the tool tip is pointed at the desired position, with the desired orientation, $\boldsymbol{A}_{PP}^{C} = \boldsymbol{A}_{T}^{C}$ in this instance. $\boldsymbol{A}_{M_4}^{C}$ describes the relationship between the camera and the marker attached to the tool, and $\boldsymbol{A}_{T}^{M_4}$ describes the relationship between the tool marker and tool tip. Thereafter, the relationship between the pre-planned pose and the goal marker can be calculated as follows:

$$\boldsymbol{A}_{PP}^{M_2} = {\boldsymbol{A}_{M_2}^{C}}^{-1}\boldsymbol{A}_{PP}^{C} \tag{4.5}$$

where ${\boldsymbol{A}_{M_2}^{C}}^{-1}$ describes the relationship between the camera and the goal marker.

In order to ensure safety for the patient, as well as for the surrounding medical personnel, the implementation was made to halt the robot if either the tool- or goal marker goes out of view for the camera. Thus, there is no risk of the robot operating blindly.

4.6 Graphical User Interface

A GUI was created in order to facilitate calibration of the system and give a better overview of the procedure. The system is expected to always be supervised, in case there is a unexpected behaviour the user is accessed to emergency stop actions. The emergency stop does not work electrically and will not guarantee a breaking within the physical hardware. Instead the emergency button on the interface will publish a new joint velocity command to override the previous one. This new command sets the joint velocity to 0 for all joints, creating a software break. The robot is still active and still receives messages from the software through ROS nodes. This is because the user can still freely drive the robot to a new position and still continue an executing run with the same calibration setup. For example in case the calibration works well enough, but the robot is placed in a bad position before executing the run, the user is able to pause the run while re-positioning the robot.

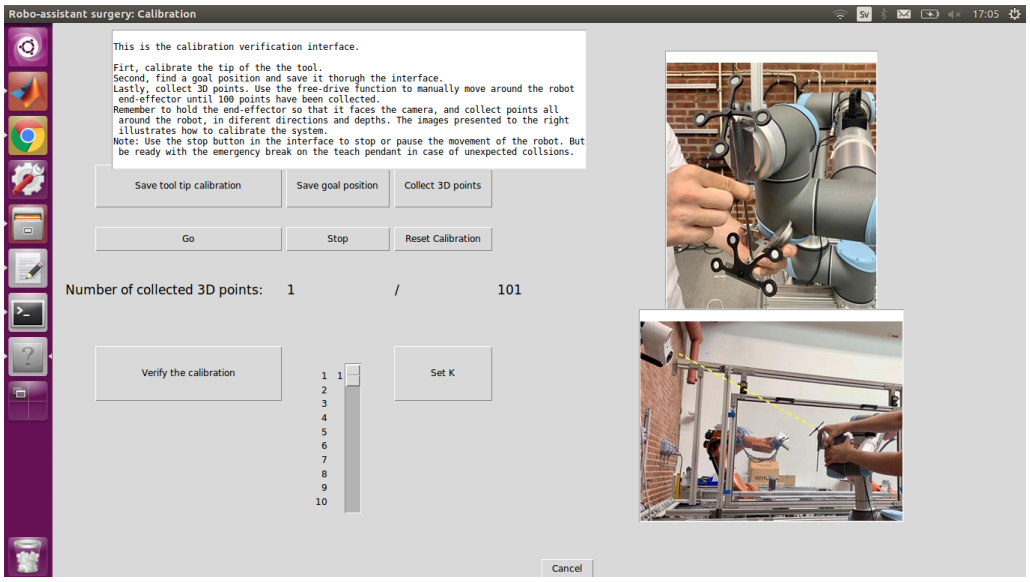

Figure 4.4: The Graphical User Interface.

The interface is using the python library *Tkinter* which creates all the functionality and graphics displayed in Figure 4.4. Calibration instructions were implemented in the GUI, as well as requirements for the operator to confirm that the system behaves as desired. Visual representations of how the calibration should be performed were also added to the GUI. A display of the current number of 3D points collected was added in order to give a better overview during the calibration. In case of incorrect calibration, the GUI should display message boxes regarding the problem. Requirements for confirmation of enough points collected, and tool tip calibration being carried out, were also implemented. Another addition to the GUI was that, after the SVD algorithm has returned a result, the operator has to verify that the robot follows a marker as it should. If the SVD calculations fail, and the end-effector does not follow the marker properly, this is recognised by the user and the GUI then restarts the calibration and increases the amount of points to be collected by 20 every time. Finally, a slide bar was implemented to enable adjustment of the speed of the robot. The speed is set on a scale from 1 to 10, where 1 is equivalent to no gain and 10 is equivalent to double the speed of the system without any gain. In other words, when set to 1 the gain for the system is also 1, and a 10 on the scale results in a system gain of 2. The scale adjusts to all intermediate steps in between 1 and 2 for the gain k. The scale was implemented to verify calculations faster, although the recommendation was still made to perform surgery at a low speed. The user can easily verify the velocity and possible estimate the collision before it occurs. If the speed is set to a higher level, it could be more difficult to estimate a collision or unexpected behaviour. Besides increasing functionality, by improving user friendliness, the implementation of the GUI also provides the system with an extra layer of safety due to the required confirmations throughout the calibration procedure.

5
Implementation

5.1 Hardware and Software Setup

The hardware used in this project was a UR10 and an Atracsys fusionTrack 250 stereo camera. They both carry their own internal IP addresses, which can be used to set up a local area network (LAN). The robotic operating system was used in order to establish the communication network between these components. In order to get the best result, it was important to consider the positioning of all components, see Chapter 6 for further evaluation. In a surgical environment it is important that the robotic arm has a sufficient amount of space, since an obstacle could restrict the robot's ability to move. Therefore, the software solutions were created while keeping the operating table in mind. A restriction was implemented, such that no full rotation of the robot base was done, in order to ensure only operating in the viewing field of the camera. The optimal distance between robot and camera was determined to be 1.5 - 2.5 meters.

5.2 Simulation Setup

Universal Robots provides a realistic simulation environment for the robotic arm which was used during this project to verify the mathematical implementations. A communication node was used to publish the command for joint velocity, which was derived in Chapter 3. The publishing was switched to another channel for the simulation environment. For the simulation, the joint velocity command was published to the channel *"/ur_hardware/script_command"*. However, the same calculated commands works for both the simulation environment and the physical system. The simulation environment is recommended to be used before the surgery, in order to observe and review the trajectory of the robotic manipulator, as well as to ensure that the position of the robot and other medical equipment in the OR is suitable. Additionally, numerical values are also expressed in the simulation which can be used to estimate the accuracy and precision of the system. This knowledge could be used to identify potential issues.

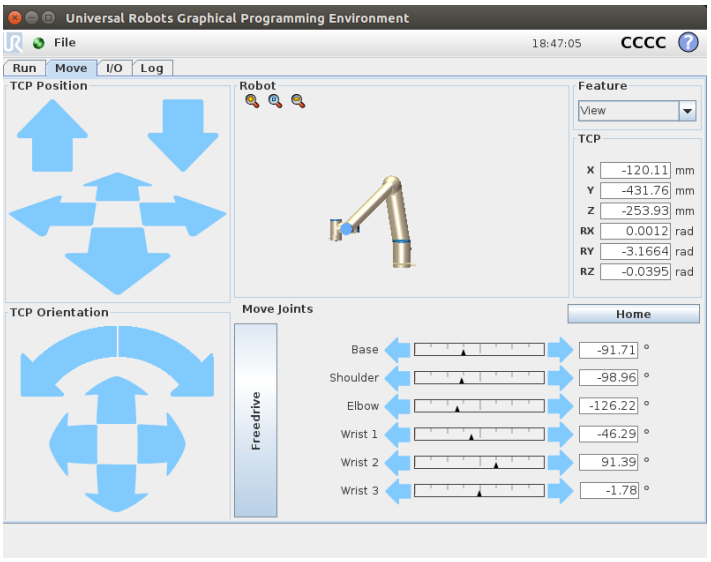

Figure 5.1: Universal Robots simulation environment.

5.3 Communication

The communication was carried out through *ROS*, in which the entire system consisted of two ROS nodes communicating with each other. The first node was used to read data from the camera, by publishing information about the markers through a channel called *"/cameraData/geomX"*, where X is the number of the marker in view for the camera. The information published to this channel was a transformation matrix containing a rotation matrix R and a translation vector t. The information was published at a frequency of 20 Hz, although the maximum publishing rate was 120 Hz for the camera. The second node in this setup subscribed to previously mentioned channel and was used for kinematic control. This node was also used to publish the velocity control command and adjusted calculations to the robot. The joint velocity command was publishing the controller to the channel *"/ur_driver/URScript"*. The communication was performed using *python-3.5*, since the camera communication required a python version of 3.5 or higher. When running the program, a GUI was displayed in order to grant the user more flexibility and control over the robotic manipulator. Considering the mathematical model is correct, the code uses the ROS node to publish the joint velocity from the inverse kinematics, using *"SpeedJ"* command which can be interpreted by the robotic hardware used in this project.

5.4 Angle of Attack

This project aims to create a mathematical model of a surgical procedure. The kinematic controller and camera combinations provide the ability to reach a certain goal pose with a tool attached to the end-effector of the robotic manipulator. However, this mathematical model does not result in a trajectory similar to a human surgeon. The convergence of the tooltip should also take the so called “angle of attack” into consideration. Meaning that the tool should reach a correct rotational convergence

some time before the positional convergence. This creates a more human-like trajectory, while also avoiding collision with the patient and other obstacles. The angle of attack is also important since the angle of the medical instrument makes a difference when drilling holes in the vertebra or inserting pedicle screws.

Since the angle of attack is important for this project, the scalar k is modified to prioritise the rotational part before the positional part. The gain is now modelled as a matrix, $\boldsymbol{k} = diag\,(k_p, k_p, k_p, k_r, k_r, k_r)$, where k_p is the gain for the positional part of the system and k_r is the gain for the rotational part of the system. If the rotational part should converge before the positional part, then one needs to choose $k_p < k_r$.

5.5 Joint Constraints

Before the implementation of joint constraints, the system only tried to minimise the shortest path in all directions, i.e. x-, y-, z-directions and quaternions. Since the system does not get any input from the surroundings, taking the shortest path could lead to collisions with obstacles, or with the robot itself. Therefore, the joints of the robot were constrained in order to avoid collisions, and to reduce the risk of the marker attached to the pointer tool going out of view for the camera.

Another consideration to this solution is that the camera and robot is static during the operation. The camera and screen are commonly positioned directly above the head of the patient[7]. The robotic arm is expected to be in front of the surgeon, on the opposite side of the operating table, in other words to the side of the patient. Consequently, the robot can be constrained to stay within specified minimum- and maximum joint values, based on this information. Without more detailed knowledge regarding the environment, other possible obstacles could not be taken into consideration, although there were some that could always be expected, like the presence of the operating table. It was also desired to implement joint limitations so that the robot stays in the field of view of the camera. This is illustrated in Figure 5.2.

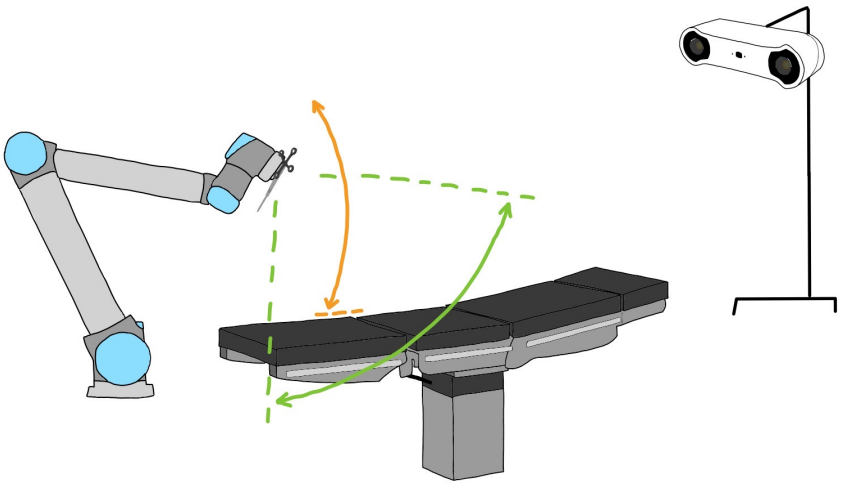

Figure 5.2: Implemented joint constraints.

A limitation was implemented to the first joint, the base, constraining it to only rotate ± 45° from the camera viewing direction. Limitations were also implemented on the second and third joint of the robotic manipulator, the shoulder and elbow respectively. Since these joints determine the vertical positioning of the end-effector during surgery they were constrained to stay within the height of the operating table, or above. The remaining joints were only slightly constrained, to avoid self-collision, while at the same time not restricting the system too much, which could instead result in loss of accuracy and convergence. The limitations can be described as minimum- and maximum values for all joints.

$$\begin{aligned} \boldsymbol{q}_{max} &= \begin{bmatrix} q_{cd} + 45° & -85° & 0° & 160° & 150° & 360° \end{bmatrix} \\ \boldsymbol{q}_{min} &= \begin{bmatrix} q_{cd} - 45° & -160° & -135° & -20° & -110° & -360° \end{bmatrix} \end{aligned} \tag{5.1}$$

where q_{cd} is a variable saved after calibrating the camera using SVD. The variable uses the joint value for the base when facing the camera direction.

As previously mentioned in Section 3.6, one DOF has to be removed from the Jacobian matrix relating the end-effector's linear- and angular velocities to the joint velocities. In this case, the fourth row of the Jacobian was nullified, which expresses rotation in the "roll" angle, φ. As long as the tool is positioned in the right angle, the rotation around its own axis does not matter. Therefore, the choice was made to disregard the roll variable.

5.6 3D Printed Spine Model

In order to create a more realistic test procedure, models of three lumbar vertebrae (L3 - L5) were 3D printed, see Figure 5.3 and Figure 5.4. The 3D printing was carried out with help from Chalmers Autonomous Systems Laboratory.

Figure 5.3: Superior view of 3D printed model of L4.

Figure 5.4: Lateral view of 3D printed model of L3-L5.

Already 3D printed models of the vertebrae were provided by Ortoma, which were then scanned with an *Artec Space Spider* 3D scanner. The new, scanned, models were thereafter 3D printed in the polymeric material polyactic acid (PLA).

6
Results

In this chapter the results of the implemented solutions and the performance of the system are presented.

6.1 Goal Pose Result

The main goal of this project was to successfully converge the positional- and rotational part of the end-effector of a UR10 robot such that the tip of a tool mounted on the end-effector is placed in a pre-planned pose at a 3D printed vertebra. The result presented here contains all aforementioned implementation, in order to generate a test which gives a general overview. First, the tool attached to the end-effector was calibrated using the calibration unit, as mentioned in Section 4.4.1. Thereafter, a pre-planned pose on the 3D printed spine model was saved by manually positioning the tool on the vertebra in a desired target position, encircled in red in Figure 6.1 below. Lastly, the camera was calibrated with the SVD solution, mentioned in Section 3.4.3. Once the calibration was successfully carried out, the robot was ready for operation.

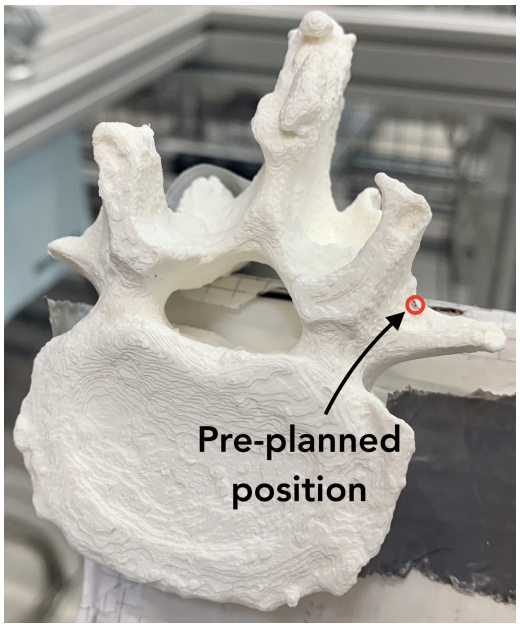

Figure 6.1: Pre-planned position on the vertebra.

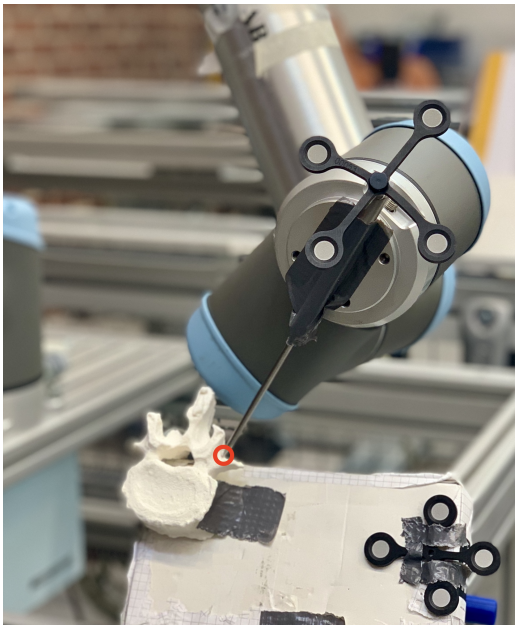

Figure 6.2: The UR10 reaching the pre-planned pose with the pointer tool.

As seen in Figure 6.1, the pre-planned pose is similar to where a pedicle screw is expected to be inserted during an actual surgery (see Figure 1.1). Figure 6.2 illustrates the resulting positioning of the pointer tool, carried out by the UR10.

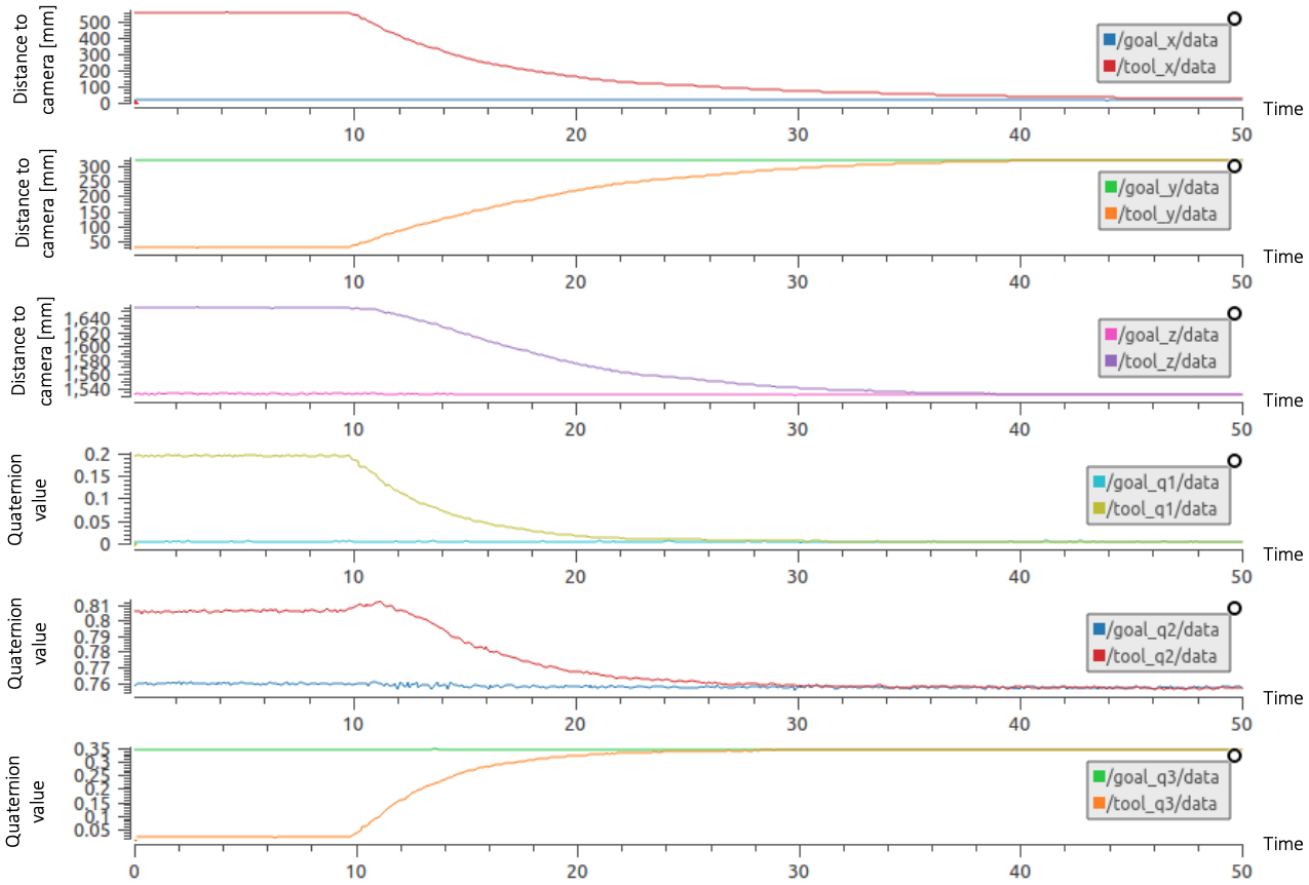

Figure 6.3: Medical tool converging towards the pre-planned goal. The first three plots illustrate the x-, y-, and z-position. The last three plots illustrate the rotational part with quaternion 1, 2 and 3. The "goal" denotes the pre-planned position and "tool" denotes the tool tip.

Figure 6.3 illustrates the results of this operation, where all positions and rotations are converging to the pre-planned position and orientation. Note that the rotational part, i.e. quaternion 1,2 and 3 (denoted $q1$, $q2$, and $q3$), is converging before the positional part. This is because the system prioritises the so called "angle of attack".

6.2 Verification of Camera-Robot Calibration

As mentioned before, the first aim of the thesis was to make a joint-controlled collaborative robot position its end-effector with respect to a stereo camera, based on fiducial markers recognisable by the camera. The provided solution was to use an algorithm with SVD in order to align the camera- and robot coordinate frames. This enabled the robotic arm to move its end-effector to a desired position, based

on measurement data received from the camera, illustrated in Figure 6.4. In the figure, the goal position and the current position are both expressed as a distance relative to the camera. The camera position is expressed as a distance in relation to the robot base.

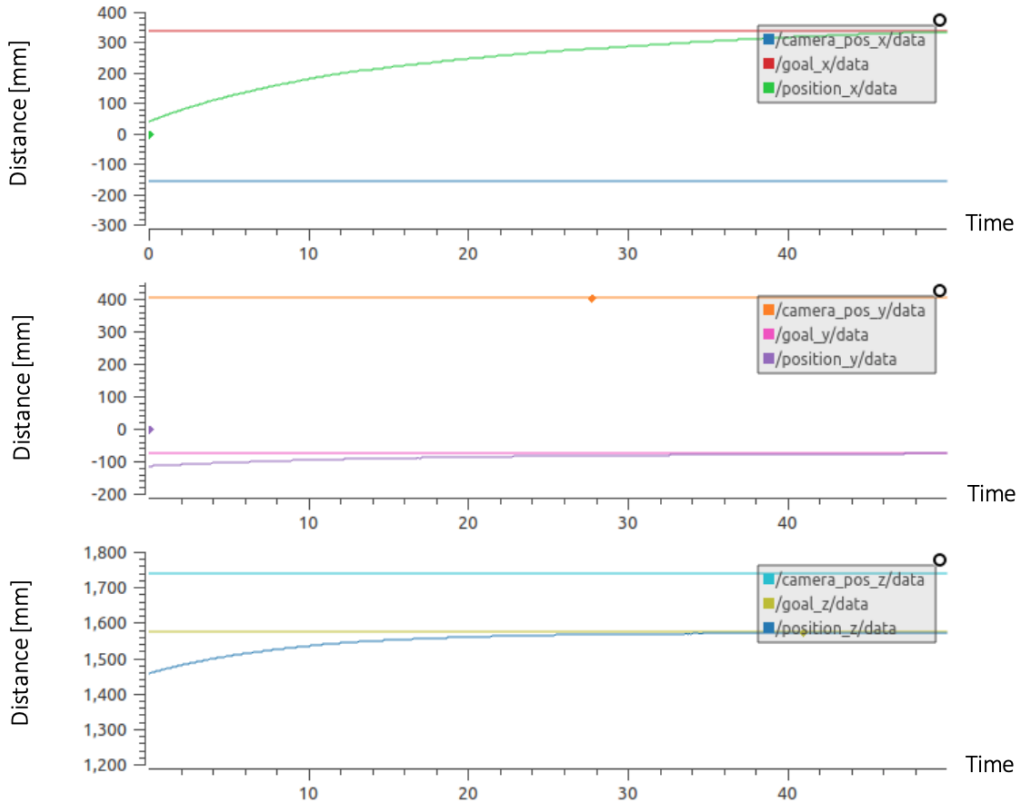

Figure 6.4: End-effector position converging towards target when using SVD.

Figure 6.4 illustrates how the position of the end-effector converges towards the desired target position, given by the transformation matrix $\boldsymbol{A}_B^C$. The variable, $\boldsymbol{A}_B^C$, describes the relationship between the two known coordinate systems. As an initial solution this did not make the system prioritise convergence in any direction, this result simply proves the success of the SVD algorithm and convergence of the end-effector. The solution was verified to work for any camera pose in the vicinity of the robotic arm, and in several positions around the robot. The result is displayed in Table 6.1, note that the quaternions are unitless but have also been multiplied by 10^3 in the table.

Table 6.1: The error decrement for camera placements at different positions around the robot.

Distance between Camera and Robot [m]			**Initial Error [mm]**						**Final Error [mm]**					
x	y	z	x	y	z	q_1	q_2	q_3	x	y	z	q_1	q_2	q_3
1.598	-1.536	0.909	42.76	33.45	-15.85	27.38	1.05	-24.63	-9.53e-03	-3.97e-02	-5.50e-02	-1.22e-01	-2.27e-02	1.14e-01
1.679	0.916	0.861	-46.77	-33.53	23.57	-21.38	25.37	12.09	1.84e-04	4.40e-03	2.18e-02	4.25e-02	-5.65e-02	-3.47e-02
-1.712	-1.244	0.895	-31.61	-47.74	21.15	6.52	-19.27	13.52	1.17e-02	-1.70e-01	-1.05e-01	-6.17e-03	-2.57e-01	1.17e-01
-1.552	1.022	0.864	13.16	72.80	-24.94	37.06	2.62	18.69	-4.68e-02	5.02e-02	-9.87e-04	-6.40e-02	-8.11e-02	-7.96e-02

Table 6.2: The error decrement for camera placements at different depths relative to the robot.

Euclidean Distance between Camera and Robot [m]	**Initial Error [mm]**						**Final Error [mm]**					
	x	y	z	q_1	q_2	q_3	x	y	z	q_1	q_2	q_3
1.096	-25.49	20.69	-25.90	-10.31	98.29	-24.80	-21.71	5.03	2.56	-12.678	81.362	-10.479
2.02	49.71	46.30	-25.77	-9.8	79.21	3.2	-1.38e-04	-5.90e-03	9.38e-04	1.29e-02	-1.94e-02	8.27e-03
3.049	37.61	59.99	-41.51	-9.01	93.15	-22.51	-4.63e-02	-2.73e-02	4.87e-02	-7.20e-01	2.09e-01	-4.81e-01
4.100	47.62	15.91	-12.15	19.44	-68.09	-14.81	-1.93e-01	9.65e-02	2.79e-02	2.04	7.92e-02	-7.66e-01

6.3 Camera Placement

The camera placement in relation to the robotic manipulator is highly important since the distance affects the final error minimisation, i.e. the difference between the current position and the goal position, at the converging point. The end-effector's ability to converge to the goal position, with a high precision, is higher for some camera placements than others. For example, if the camera is placed at a large distance from the robotic manipulator, the resulting convergence will not be as accurate as at a shorter distance.

The optimal camera placement, in terms of distance to the robot, was investigated. The camera was positioned at different depths, in relation to the robot, always facing it such that the marker attached to the robot end-effector was in view. The normal procedure of calibrating the system, for a static camera pose, was then carried out by collecting 3D points in order to estimate a rotation matrix $\boldsymbol{A}_B^C$. Thereafter the performance, in terms of how the error between the position and orientation of the end-effector marker and the goal marker decreased, was evaluated for the different camera placements. The result is displayed in Table 6.2, note that the quaternions are unitless. As can be seen in the table, the final error was significantly smaller when the camera was positioned about two meters from the robot, compared to the other distances. The test result also indicates that the SVD algorithm works for different depths between camera and robot as well, and that the kinematic controller performs with high precision.

The resulting system was able to minimise the error between the current- and goal position to a value of $1\ \mu m - 100\ nm$, when there was a two meter distance between camera and robot, in approximately 15 seconds. The robot must be positioned such that the patient always is within reach. The camera must be placed in the recommended distance of the robot and have the goal target of the patient together with the robots end-effector in the viewing ray of the camera.

6.4 Movable Camera

Once the algorithm is run, after collecting the required amount of 3D points, the system has enough information to adjust when the camera pose is subject to any disturbance, as previously mentioned in equation (3.18). Figure 6.5 illustrates the system's ability to react to movement, and the adjusted response. In the figure, the goal position and the current position are both expressed as a distance relative to the camera. The camera position is expressed as a distance in relation to the robot base. This can be compared to Figure 6.4, where the system behaves as expected without any disturbance on the camera.

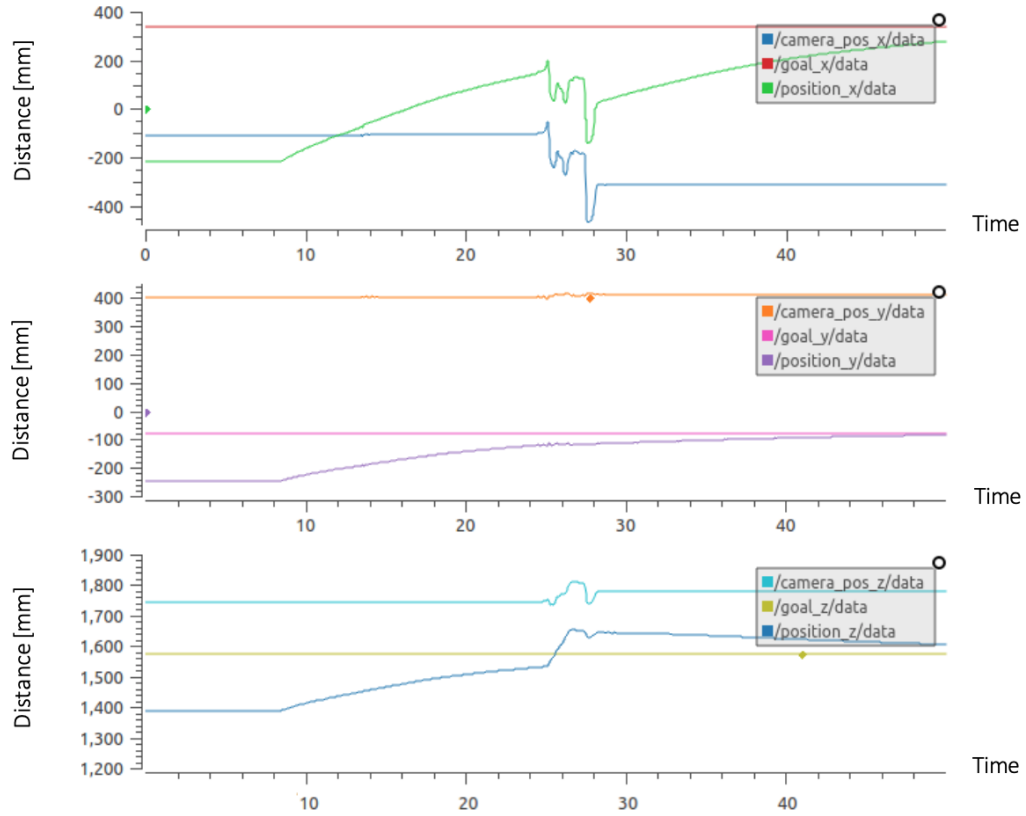

Figure 6.5: System response with implemented movable camera solution.

As illustrated in Figure 6.5, a disturbance of the camera occurs in a lateral direction, resulting in a different system behaviour compared to that illustrated in Figure 6.4. It can be observed in Figure 6.5 that the disturbance primarily affects the x- and z-direction. The y-direction is less affected but the disturbance is still caught even though it is relatively small. As a counter reaction to this disturbance, the system recovers by itself, as seen in the figure. The system also reacts, to recover from the disturbances, instantaneously, i.e. at the same time that the disturbance occurs. This indicates that the provided solution works as intended.

6.5 Angle of Attack

The main purpose of this result is to demonstrate that it is possible for the robot end-effector to converge to the goal orientation before converging to the goal position. Thus, the tool tip can be pointed at the correct position on the spine with the expected vector direction, which will mimic a real surgical trajectory.

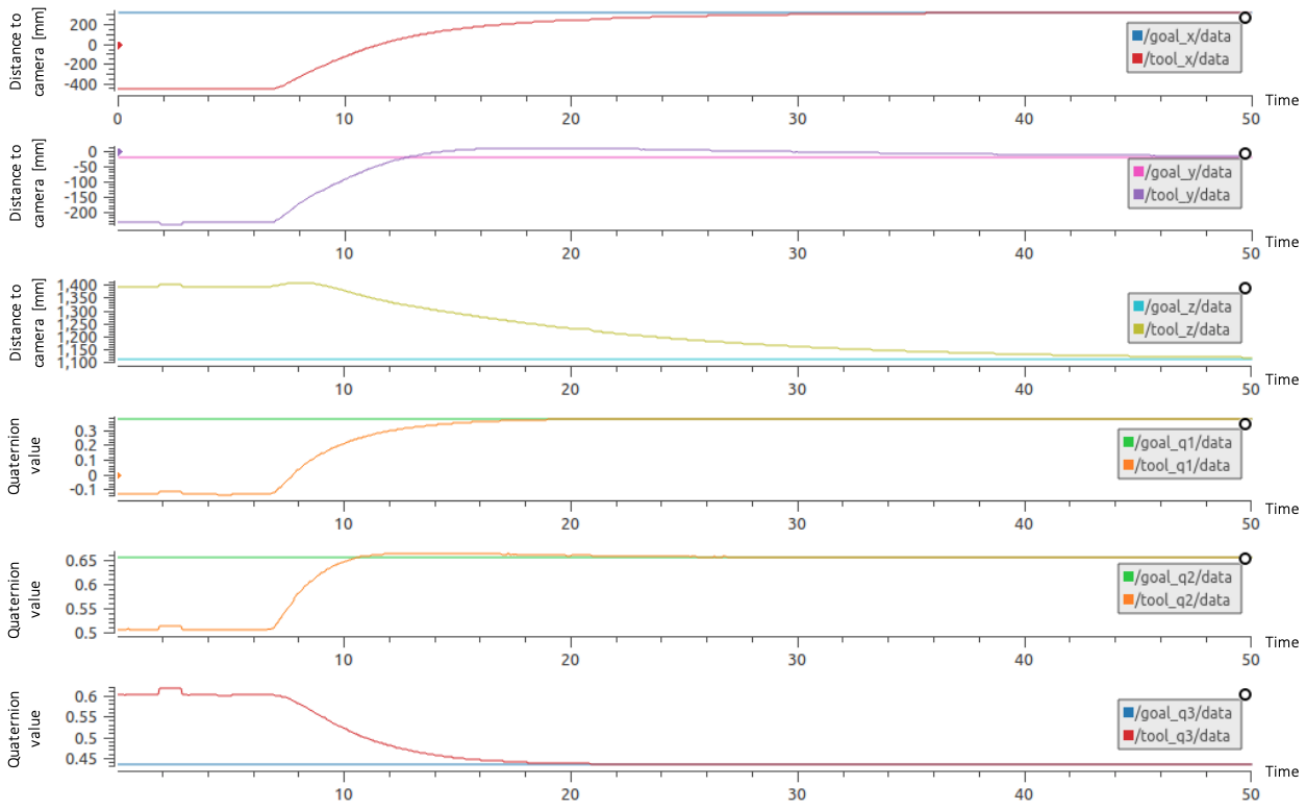

Figure 6.6: Medical tool converging towards the goal. Rotational convergence of the tool takes place before positional.

In Figure 6.6 it can be observed that the quaternions, denoted $q1$, $q2$, and $q3$, converged towards the goal about three times as fast as the positional part, denoted x, y, and z. As mentioned in Section 5.4, the constraint $k_p < k_r$ needs to be met. However, in order to create a trajectory similar to a real surgery, the relationship should be at least $k_r = 2k_p$. In Figure 6.6, $k_r = 3$, and $k_p = 1$. The optimal relationship between these two components could be further investigated, in order to achieve a behaviour as similar to a surgeon as possible.

6.6 Joint Constraints

In order to demonstrate that the implemented joint constraints affect the system as expected, the robot was run in a "bad area", i.e. where the goal pose might cause the joint angles of the robot to exceed their specified limits. This was done both with and without joint limitations, resulting in Figure 6.7 and Figure 6.8 respectively. The same start- and goal pose of the robot were used in both cases, and the results were compared. The joint constraints implemented in the system can be found in Section 5.5, equation (5.1). In Figure 6.7 and Figure 6.8, q_1 - q_6 are the joint angles, $q_{i,max}$ the maximum value of the joint, and $q_{i,min}$ the minimum value of the joint.

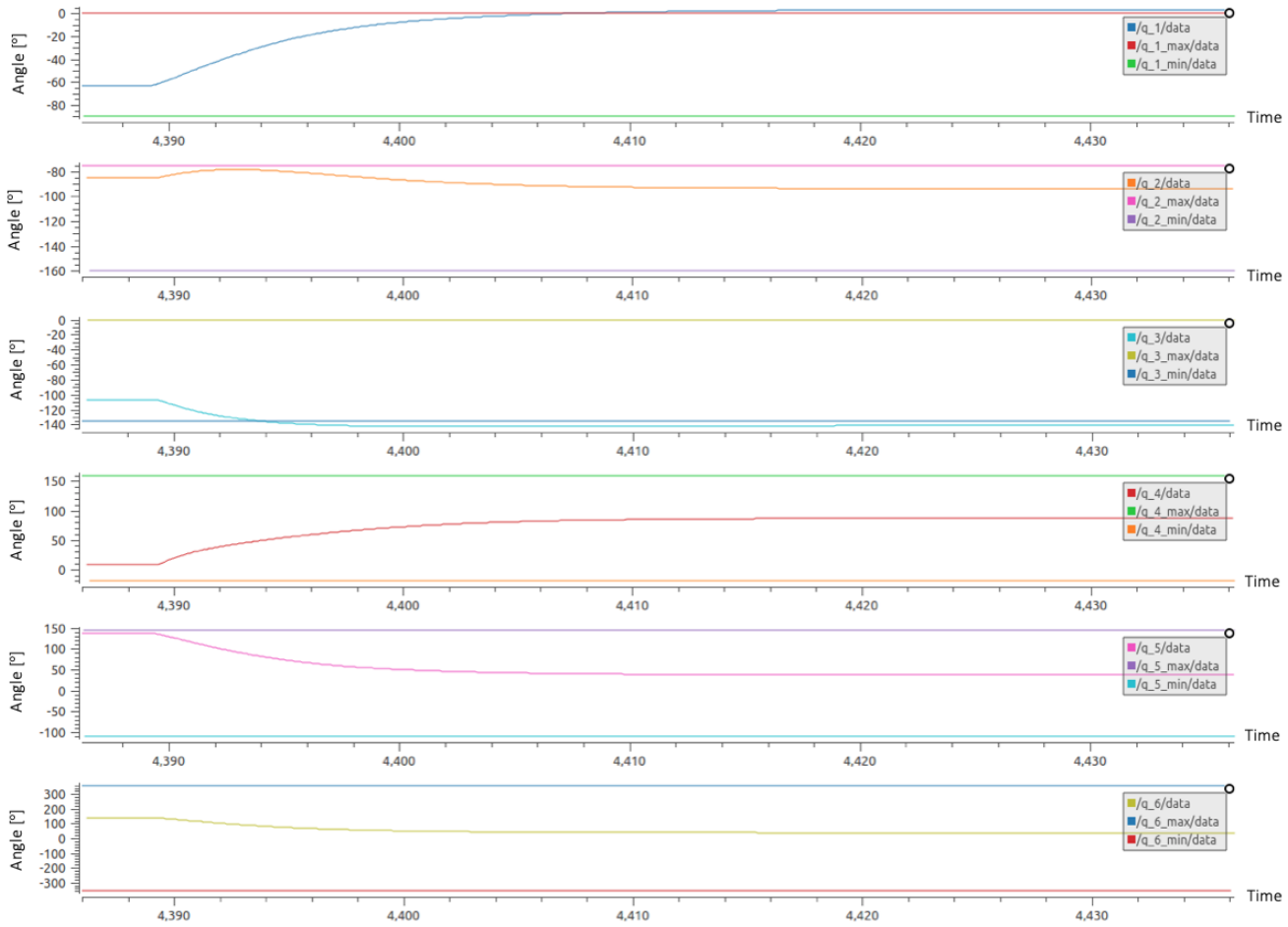

Figure 6.7: System behaviour without joint constraints.

Without the proposed joint constraints solution, the joint angles exceeded the expected limitations, as illustrated in Figure 6.7. The base and elbow, i.e. q_1 and q_3 were clearly moving outside their limitations.

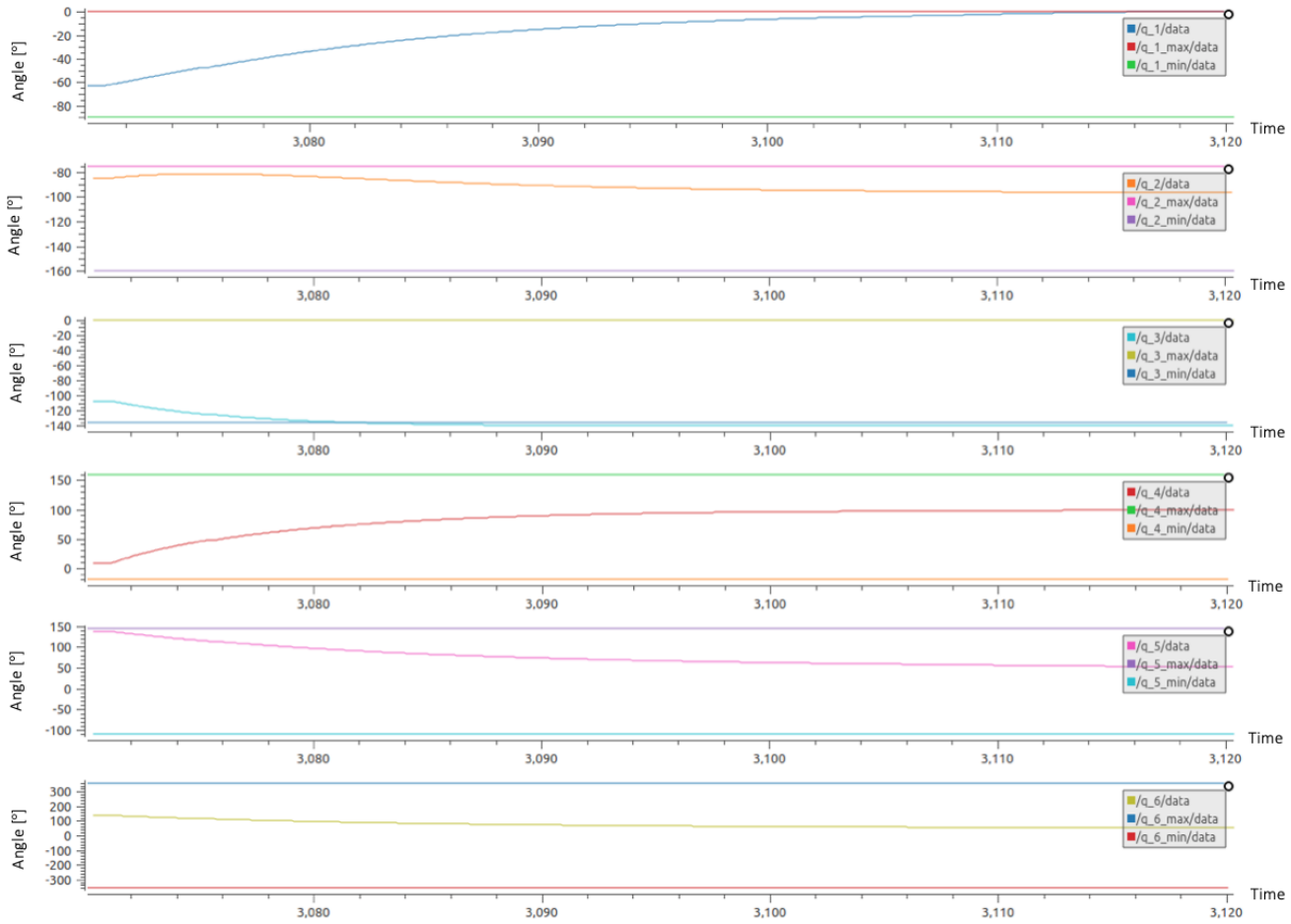

Figure 6.8: System behaviour with joint constraints.

With the mathematical implementation for the joint constraints added to the system a different behaviour was obtained, illustrated in Figure 6.8. There was a significant difference in joint angle values, compared to when no joint constraints were implemented. The base and elbow joints were now within their limited values and other joint values also improved. The values of the shoulder joint (q_2) remain further from its limit when the system runs with joint constraints, compared to when running without the constraints.

This implementation causes the joints to prioritise another trajectory, in order to stay within the limited region, if possible. However, these are soft constraints and do not necessarily guarantee full avoidance of the limits. If the goal is placed outside the limited region, the robot will ignore the limitations. Therefore, it is necessary to consider the path of the robot beforehand. In other words, the user should estimate an approximate trajectory of the end-effector. A recommendation is therefore to place the end-effector within the joint limits before the run, and also have the goal within the same region. The implementation will refrain the robot from moving in an unexpected trajectory during a run.

6.7 Accuracy and Precision

For a system meant to be used for spine surgery, accuracy and precision is of the outermost importance, since an unexpected error in movement can injure a patient significantly. The system should guarantee both accuracy and precision in order to be reliable. Accuracy indicates how well the system is able to minimise the error of position and rotation, between the tool tip and the goal position. Precision expresses the variability between measurements, but does not guarantee minimisation of the error between tool tip and goal position.

To evaluate the accuracy and precision of the system, a test was carried out. Three different goal poses were chosen, in three different planes, and the robot had to reach each goal three times. When switching goal pose, the system was also re-calibrated. For each goal, a mean error and standard deviation was then calculated from the three measurement values. The result is displayed in Table 6.3 and Table 6.4 respectively. Note that the positions x, y, and z are expressed in millimeters, but the quaternions q_1, q_2, and q_3, are unitless.

Table 6.3: The mean absolute error for the three different goal poses.

Goal no.	Mean absolute error [mm]					
	x	y	z	q_1	q_2	q_3
1	3.76610866e-01	5.21225304e-02	1.64972371e-01	6.55287397e-01	1.03102202	1.65382214e-01
2	1.41968073e-01	2.76088573e-01	3.02753556e-01	1.26173026	1.08178840	1.00460976e-01
3	8.77420659e-02	1.57040198e-01	1.00692991e-01	4.94990853e-01	2.97555726e-01	1.56611332e-01

Table 6.4: The standard deviation for the three different goal poses.

Goal no.	Standard deviation [mm]					
	x	y	z	q_1	q_2	q_3
1	2.94434872e-01	5.25563832e-02	3.83542729e-02	6.08501510e-01	9.94498409e-01	5.23054242e-02
2	1.59388556e-01	1.79425597e-01	3.63631834e-01	4.87717649e-01	1.33136309	1.21055092e-01
3	8.54605749e-02	1.31082907e-01	1.10616391e-01	2.28283176e-01	3.18456978e-01	4.90803645e-02

The mean absolute error, displayed in Table 6.3, indicates the accuracy of the system, whereas the standard deviation, displayed in Table 6.4, indicates the precision of the system. Overall, the system achieves both a high accuracy and precision. Although, as previously mentioned in Section 1.4, the safe zone with a pedicle violation of less than 2 mm includes the whole system chain. Therefore, the parts of the system used in this thesis, which only include the robot manipulator, visual tracking device, camera-robot calibration and tool calibration, should achieve an even higher accuracy. However, due to lacking knowledge of the accuracy of all other system components, it is hard to determine the accuracy requirements for the proposed system.

6.8 Damped Least-Squares (DLS) Method

As previously mentioned, it is common for joint controlled robots to have a singularity issue. A proposed solution was therefore to use the DLS method. A test was carried out to investigate the result when using different λ values. This was done by letting the system run with different values of λ, having the same start pose of the robot and the same goal for each run. The error, i.e. the difference between the goal and where the robot positioned the tip of the pointer tool, was then measured. In Table 6.5 below, x, y, and z is the positional error expressed in millimeters. The rotational error, expressed as roll, pitch and yaw (denoted r, p, and y in Table 6.5), is unitless.

Table 6.5: The error for different λ values.

λ	**Error [mm]**					
	x	y	z	r	p	y
10	61.12	115.67	141.21	0.1527	0.0045	0.0340
1.0	3.86	14.04	5.02	0.0156	0.0014	0.0089
0.1	1.77	2.13	0.62	0.0072	0.0006	0.0116
0.01	0.32	0.58	0.34	0.0022	0.0021	0.0023
0.001	0.32	0.04	0.01	0.0021	0.0008	0.0005
0.0001	0.01	0.32	0.05	0.0004	0.0001	0.0011

As mentioned in Section 3.3, it can be verified in Table 6.5 that a large value of λ results in large errors, while a small value results in small errors. Ideally, λ should be chosen such that the error becomes sufficiently small, while still ensuring avoidance of singularities. It was also visually observed that the behaviour of the system differed depending on the value of λ. The difference in behaviour indicates different degrees of singularity avoidance.

6.9 Marker Out of View

To ensure patient safety, and prevent unpredictable system behaviour, it was desired to stop the robot from moving when either marker 4 or marker 2 goes out of view

for the camera. In other words, the robot should only move when both the medical tool and the reference marker, mounted on the patient, are visible.

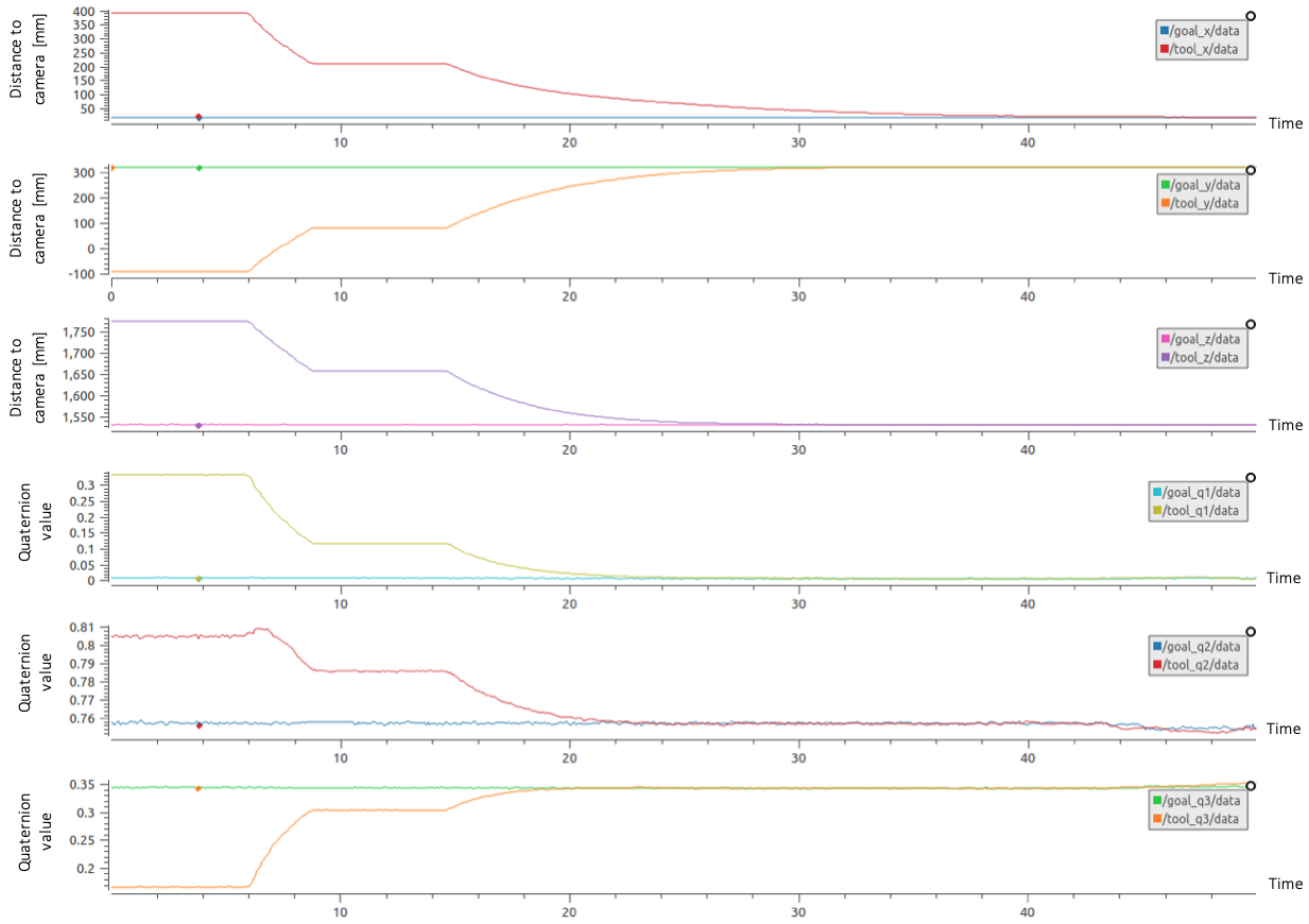

Figure 6.9: System behaviour when a marker goes out of view for the camera.

In Figure 6.9 it can be observed how the system comes to a halt when a marker goes out of view for the camera, around time index 9, and then resumes, around time index 15. For this test, the marker was simply covered and uncovered by hand, to imitate it going out of- and coming back into view. The tooltip starts converging towards the goal, stops for the duration a marker is out of view, and then continues to converge.

7
System Demonstration

It was desired to demonstrate that the proposed system could be combined with Ortoma's existing system. Therefore, a test was carried out with a goal pose which had been determined using planning software. In this case, a digital 3D model of the third to fifth lumbar vertebrae (L3 - L5) was processed using the free software MeshLab. The chosen target was a position on the L3 vertebra, similar to where a pedicle screw would be inserted during surgery, marked with a green pin in Figure 7.1. On the left, the pointer tool can also be seen positioned in the target pose.

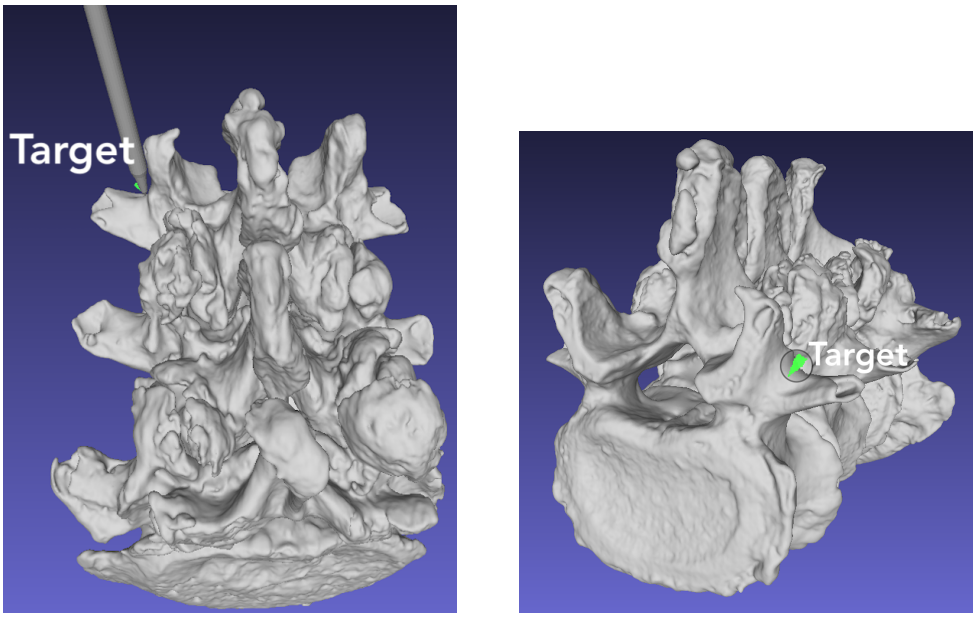

Figure 7.1: Target in the planning software, with and without the pointer tool.

Similar to the camera-robot calibration, described in Section 3.4, now patient-software calibration was necessary in order to obtain the preoperatively planned pose in the coordinate system of the reference marker. Like previously, SVD was used for aligning points on the 3D printed model of the vertebra and the digital model.

After alignment, the demonstrator was set up as in Figure 7.2. In the figure, the vertebrae are positioned in a foam mold which keeps them in place, and the reference marker is securely fixed to the L3 vertebra.

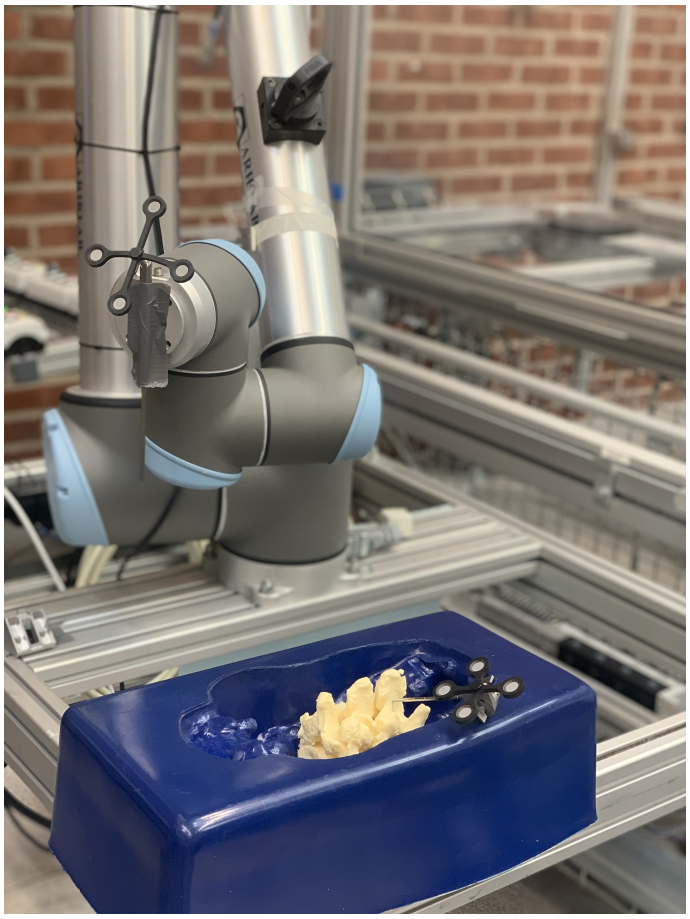

Figure 7.2: Demonstrator setup.

With the pre-planned position as input to the system, the robot successfully positioned the pointer tool in the desired pose, see Figure 7.3. A more proximate view of the pointer tool's final position is illustrated in Figure 7.4. Due to its size and depth the foam mold was turned upside down, and the vertebrae were moved, since the initial setup caused the robot to collide with the mold when trying to reach the target with the pointer tool.

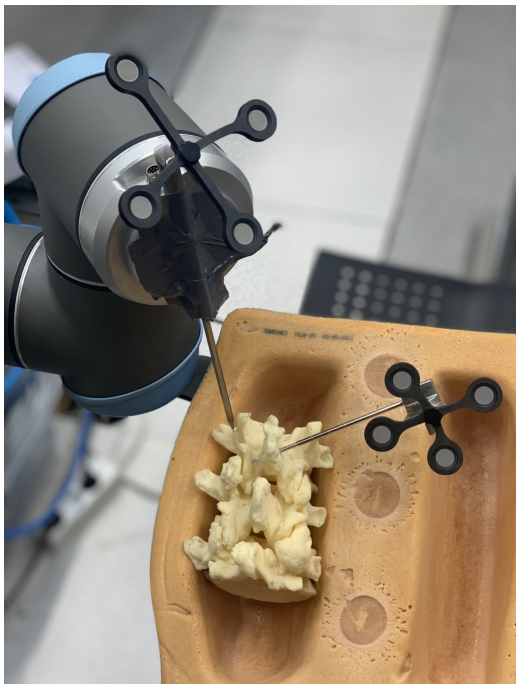

Figure 7.3: The robot positioning the medical instrument in the pre-planned pose.

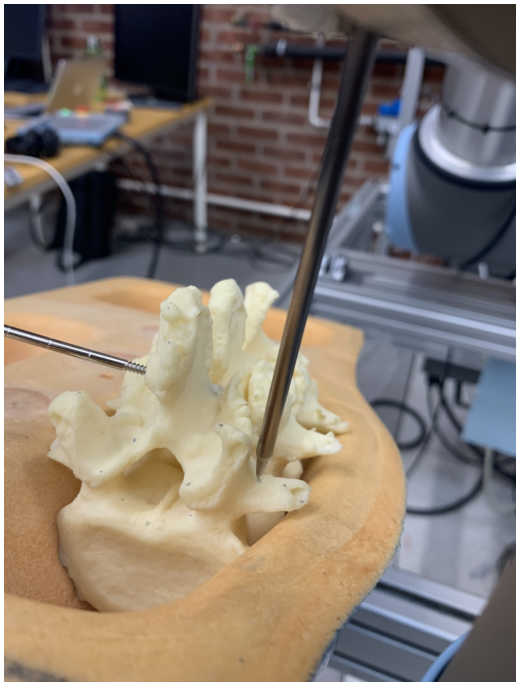

Figure 7.4: Proximate view of the robot positioning the medical instrument in the pre-planned pose.

The resulting error, between the tip of the tool and the target pose, is displayed in Table 7.1. Note that the positions x, y, and z are expressed in millimeters, but the rotation, expressed as quaternions q_1, q_2, and q_3, are unitless.

Table 7.1: The demonstrator error.

Error [mm]					
x	y	z	q_1	q_2	q_3
-7.19782311e-02	-5.40152597e-02	-6.11816314e-02	-1.26793646e-01	-1.42107727e-01	-3.70472568e-02

This demonstration has verified that the proposed system can receive a target, which has been preoperatively planned in a software, together with the calibration transformation matrix from planning software coordinates to reference marker coordinates, and thereafter the robot can move the medical instrument to that pose. The robot itself does not perform the patient-software calibration, but can use input from any planning software in which the calibration has been performed.

8

Discussion

In this chapter, some recommendations for using the system are first presented. Thereafter, the choices made when creating the system are motivated and reflected on. Methods used, and potential issues, are also brought up. Finally, the possibilities of integrating a robot-assisted surgical system in the OR, and ethical concerns, are discussed.

8.1 Recommendations for Use

When running this system it is highly recommended to use the GUI in order to calibrate and gain control of emergency shutdown. The user should have both interface and UR10 teach pendant in reach, in case of emergency stop. The GUI should be sufficient for emergency stops, although the emergency button on the touch screen is also directly linked with electric wires to the motors of the robotic arm. It is possible to adjust the speed of the robot joints while running, and it is recommended to use a low speed when the robot is close to converging to the goal. In case of an obvious collision with an obstacle, one can estimate such scenario by following the trajectory of the robot and interrupt the collision.

8.2 Collection of 3D Points

The collection of 3D points when calibrating the system could be carried out in different ways. In this project, it was decided to let a nurse or technician move the robot into different positions in order to collect these points. However, another option could be to let the robot move on its own, in a predetermined trajectory. The benefit of the latter alternative is that an engineer could implement positions already in the software of the system, thus ensuring a sufficient variety or span of 3D points. This would also decrease the workload on the person who would otherwise be tasked with moving the robot. However, it could be difficult to determine ideal positions beforehand, outside of the OR. It is also probable that the robot would collect exactly the same 3D points along the trajectory, each time carrying out the calibration. Therefore, it would be insufficient to repeatedly follow the same trajectory in order to obtain the desired number of 3D points and instead a significant number of predetermined positions would be required. Having someone to manually move the robot on the spot also makes it easier to take a varying environment into consideration, since the circumstances and different objects surrounding the robot

could vary for each individual surgery. The implemented GUI should also help minimising the risk of human error during the calibration procedure. An alternative would also be to mount a marker in a perfectly known predefined position on the robot, and to construct a tool holder that ensures that the tool is always mounted in precisely the same position on the end-effector. Then, calibration would only have to be performed one and the risk of human error would be even further decreased.

A predetermined trajectory was implemented and tested in this project. However, it was determined unstable and often resulted in the need to re-calibrate the system. The trajectory was hard coded, meaning the robot would always behave the same way, resulting in the same points being collected over and over again. Even after increasing the number of collected points, the SVD calculation would still yield unsatisfactory results. The trajectory controller was not part of the main objective, which eventually led the idea to be abandoned, instead of given more time and effort. However, if the trajectory controller was built more generic, for instance by finding the direction of camera and collecting only surrounding visible points, this could potentially be an alternative solution.

Furthermore, the robot-assisted surgery system will always be supervised by a nurse or technician, who will need further education on how the system works and how to operate it. Therefore, the decision was made in this project to let the educated operator move the robot. Even though this solution results in higher costs associated with the surgery, due to the additional training and time consumption required, there is also a higher chance of a well done calibration due to the increased adaptivity. This also makes it easier to assign responsibility for the calibration, compared to the case where an engineer designs a preplanned trajectory but the hospital personnel oversees the execution.

The optimal distance between the camera and robot was determined to be around two meters. The reason for that could be that when positioned too close to the robot, the camera's field of view becomes smaller, making it harder to collect a sufficient variety of 3D points. Conversely, when the camera is placed too far away, the depth resolution could be insufficient for distinguishing different 3D points. The recommended two meter distance also seems reasonable when considering the surroundings in the OR, where both medical personnel and other medical equipment should fit in a limited space.

8.3 Number of Points Collected

The number of 3D points collected when calibrating the system was set to be 100 since it resulted in a satisfactory accuracy and precision. A higher number of points should theoretically be able to yield a better result, moving the robot with even more precision. However, this would also result in a very high computational complexity and consequently require more time. The actual collection of points would also take longer and would thus increase the overall time consumption, and cost, for

the surgical procedure. Because of this specific application 100 points was set as a initial value and takes approximately 2-3 minutes to collect. If the calibration fails the first time, and the user selects the "re-calibrate" button on the user interface, the number of points is increased by 20 each time. Although slightly more time consuming, this implementation was made in order to increase the accuracy of the SVD algorithm.

8.4 Marker Out of View

In the current system, the robot stops moving when a marker goes out of view for the camera, in order to ensure the safety of the patient. However, this does require someone to reposition the robot or the camera before it is possible to continue, which in a real scenario could become time consuming and thereby costly.

It could be realistic to assume that the end-effector will start from a position which is in relatively close proximity to the goal position, and the marker going out of view does therefore not necessarily have to be a frequently occurring problem. Nevertheless, to decrease the potential issue of unnecessary stops, a possible solution could be to make the robot visible from more angles. This could be done by equipping the robot with more than one marker, for example a different one on each side of the end-effector. The markers have to be distinguishable from each other, and the system has to be calibrated for both of them. Alternatively, the marker attached to the medical tool could be modified and equipped with more fiducials, where half of them facing upwards and the other half facing downwards. This would enable the end-effector to rotate more while still keeping the pointer tool in the camera's field of view.

Yet another solution would be to force the end-effector to always keep the marker in view during the movement. This could be done by adding hard constraints to the joint velocities. However, this was not done during this project due to the limited time and the prioritisation of other implementations.

8.5 Pre-planning the Position of the Robot

Once all the aforementioned calibration has been carried out, and the system is ready to operate, the user should consider the placement of the robotic arm. The system will always find the shortest path, in terms of robot joint angles, between the start- and goal pose. The additions, such as the DLS method and the implementation of joint constraints, will help the system avoid self-collision. However, the system will obtain a higher joint velocity if the euclidean distance to the goal is large. Consequently, a high velocity might cause unexpected movement, which in turn could lead to collision. Although the system should be able to handle this scenario without any collisions, the operator would have to expect a larger velocity, which could appear unpredictable. Therefore, it is recommended to lower the speed through the user interface.

An alternative, when in a surgical environment, is to always place the robot's end-effector in close proximity to the pre-planned position. Even though the system is able to start with the end-effector in any pose, it could be helpful to place it relatively close to the goal. An approximate distance of 0.3-1 meter has consistently yielded satisfactory results, meaning the robot does not reach a high joint velocity while still having enough space to adjust into the expected position and orientation.

8.6 Damped Least-Squares (DLS) Method

The DLS method is a useful addition to the system since the singularity problems can be avoided. However, since this creates a trade-off between singularity avoidance and system accuracy, it can be wise to use a small λ for this particular application where high accuracy and precision is essential. Additionally, if the user positions the end-effector before the surgery so that it does not start in a position close to singularities, such as a straight arm position, it is possible that the system will never reach a singularity point. Therefore, it is recommended to choose a small λ, such as $\lambda = 0.001$, which yields high accuracy. Due to the fact that the slightest error in accuracy can result in severely injuring the patient, it is preferred to preoperatively position the end-effector so that singularities can be avoided, and a low λ can be used.

8.7 Joint Constraints

In order to control the system, the relationships between the joints and links of the manipulator are determined in order to create the 6 DOF Jacobian matrix. The Jacobian describes the relationship between the 6 DOF velocity and the angular velocity for the joints. As mentioned in Section 3.6, when implementing joint constraints, the Jacobian requires a change where at least one DOF is removed. In this application, the position and orientation of the tool tip are highly important, while the rotation of the tool around its own axis is not. Thus, one of the variables expressing orientation can be removed, which in this case was φ, denoting the roll angle.

However, it can be argued whether the implementation of joint constraints is necessary for this application. If the robot end-effector is initially positioned in close proximity to the goal, the system can already be expected to avoid all joint limitations. As mentioned in Section 5.5, the most probable collisions would be with static objects in the surrounding, such as the operating table or the surgeon. The implementation of joint constraints results in guaranteed avoidance of self-collision and collision with static objects, although a pre-planned position of the end-effector results the same outcome.

8.8 Integration of Robotics in the Operating Room

Firstly, a robot-assisted surgical system would have to go through rigorous testing to verify desired behaviour, and to follow all regulatory requirements to ensure patient safety, before being implemented in the OR. In order to guarantee a high safety of the system, the performance and efficiency of the robotic arm should be verified through simulation, discussed in Section 5.2.

In order to make robot-assisted surgery a realistic implementation it is essential that the tasks that are to be carried out by the robot can be well integrated in the already existing workflow. Changing routines and procedures within the healthcare sector can be a time consuming process, and thus it is desirable to keep as much as possible of the workflow the same. The medical personnel would probably reject the robot if they only associate it with unnecessarily complicated and time consuming additions to their work. For example, equipping the robot with a sterile cover and changing it between surgeries, has to be a quick and simple procedure. Additionally, it is important that the use of the robot does not prolong the actual surgery, since the longer the patient is cut open, the more bleeding and higher infection risk.

In Sweden, as well as in the rest of Europe to a large extent, no monetary compensation is obtained by introducing new, more advanced, technology to the OR. Therefore, the prevalence of robot-assisted surgery systems is usually lower in Europe than in, for example, the United States and Australia where the cost is a smaller issue. However, patient demands and transparency of surgery outcomes with and without robot-assistance are factors that can contribute to an increase in the use of surgical systems.

For this thesis a UR10, mounted on a relatively large metal frame, has been used since that was the best option available at the university. However, in an actual OR the ideal solution would be to use a smaller robot arm which can be mounted on the side of the operating table, with a central unit that can be placed elsewhere in the OR. This would minimise the occupied space, around an already crowded operating table.

8.9 Ethical Aspects

There are several ethical aspects that should be taken into consideration when creating and implementing a robot-assisted surgical system. For example, the high associated cost due to hardware, maintenance, and additional time requirements for each surgery, could potentially result in an increased cost per surgery. In countries where healthcare is not free, this could consequently render the robot-assisted surgery unavailable to some patients, and thus contribute to income segregation. On a more global scale, the cost of implementing a robot-assisted surgical system could be too high for some countries' healthcare systems. This could in turn result in an increased knowledge- and technology gap between developing- and developed

countries. Furthermore, once robots are introduced into the OR it is probable that potential issues associated with a medical procedure will be solved by new implementations or changes of the robot. However, in countries and hospitals without robot-assisted surgery systems it might be more helpful to change the medical procedure and the way the surgery is performed. This could consequently hinder, or slow down, the development of new or improved surgical techniques.

One of the reasons a robot-assisted surgical system holds a lot of potential is that it can facilitate surgeries where the outcome is highly related to the surgeon's knowledge, gained from a lot of experience. However, there is also a risk that surgeons cease to gain this knowledge, or forget it due to lack of practice, if the system becomes successful and widely implemented.

A common concern when it comes to introducing robots, in any field, is that they will replace humans, leaving current professionals without a job. However, a robot-assisted surgical system could on the contrary present new job opportunities, due to the requirement of a trained technician or engineer, who is familiar with the system and can have access to emergency breaks. The surgeon would still be supervising in close proximity, as well as perform other parts of the surgery. Thus, this kind of implementation has the possibility to create new jobs and is not expected to remove any of the current ones.

In case of an unforeseeable accident, it is necessary to elaborate the issue of who is to take responsibility. It could be argued that the hospital should compensate any injuries caused to the patient. On the other hand, the engineers and the company creating a surgical system should guarantee the safety, and acknowledge the limitations of the software and hardware, before marketing it. Furthermore, the risk of injuring a patient with the robot-assisted system must be lower than the current risk of the average surgeon injuring a patient when performing the surgery without assistance. If this is verified according to medical standards and regulations, it should be considered safe to integrate a robot-assisted surgical system in hospitals.

9
Conclusion

The system presented in this project is an extension of Ortoma's surgical navigation system, and uses much of the same equipment. It was concluded that it is possible to develop a robotic guidance system demonstrator which can be combined with any planning software, from which a target pose is obtained as input to the robotic guidance system. It was also concluded that the implementations made, should make it possible to integrate the system in a surgical environment. Although, a smaller collaborative robot which takes up less space in the OR should be used instead. Finally, the proposed system demonstrated the possibility to achieve highly accurate results using standard system components in combination with a surgical planning software.

The software solution was able to guarantee the convergence of a tool, attached to the robot end-effector, to a pre-planned pose within reach for a UR10. The system could guarantee a number safety constraints, although more testing and verification is required in order to cover all the possible collision or misbehaviour scenarios. Since it was desired to move the robot in a human-like trajectory, the tip of the tool is rotated as desired before convergence of the position takes place. The resulting system also provides self collision avoidance and avoidance of static objects in the room. The system can also adjust to positional disturbance of the camera. All implementations were made with patient safety in mind. However, the presented system needs further development and testing before being surgery ready.

9.1 Future Work

This project could be further extended and improved in the future, in order to eventually be ready for real surgery. Firstly, a larger number of tests for each implementation could be created to further verify the performance of the current system. In this project, the number of tests were limited in order to illustrate the possibility and performance of each implementation.

A possible improvement of the system is to expand the controller. The current controller is a P-regulator, following a simple error in positional- and rotational space. A possible extension of this project is therefore to further develop the controller into a PD- or PID-regulator, which could result in achieving a behaviour more similar to that of a surgeon. The reason for this is that the solution requires derivatives readings from the positional and rotational error extraction. This expansion would result in better control over the trajectory and precision of error estimation. Furthermore,

then the implementation could also be made to make the robot end-effector move along a certain axis when close to the target. Suggestively, first making the robot turn the tool into the desired orientation, and thereafter move along the angle of attack to the goal position, would reduce the risk of the tool tip colliding with another part of the patient on its way to the goal pose.

Another improvement is to develop a tool holder, in order to securely attach a tool to the robot end-effector. The tool holder could be 3D printed and form-fitted to both the robot end-effector and the tool. It could also be designed to enable attachment of different tools, which are expected to be used during the surgery. Having a tool holder securely fixed to the end-effector would only require a one time calibration of the tool tip, and thereafter the relationship between end-effector and tool tip could be considered static. This implementation would also give more freedom in the fifth joint angle rotation, since there is currently a possibility of self collision with the tool, as previously mentioned. This self collision occurs only because the attachment of the tool is placed poorly. However, a 3D printed tool holder can be designed such that this problem is avoided, by making the tool extend out from the end-effector instead of pointing downwards like in the current system.

The robot could also be made visible from more directions than currently, in order to reduce the risk of a marker going out of view for the camera. A suggestion for future work is therefore to design and implement a marker with fiducials facing several directions.

Finally, future work should also focus on aspects relevant in order to make the system integrable in the OR and the existing healthcare system. As previously mentioned, it would be preferred to use a smaller robot manipulator due to the limited space. Therefore, alternatives to the UR10 should be investigated and whether they can achieve the required accuracy and precision, tolerate sufficient payload, as well as the cost of purchase and maintenance. Furthermore, it should also be ensured that all parts of the system can reach a level of sterility which is sufficient for the OR. Quick and efficient ways to sterilise the robot and camera are needed to make the trouble worthwhile for the medical personnel.

Bibliography

[1] A. Hussain, A. Malik, M. U. Halim, and A. M. Ali, "The use of robotics in surgery: a review," *International Journal of Clinical Practice*, vol. 68, no. 11, pp. 1376–1382, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/ijcp.12492

[2] A. Bertelsen, J. Melo, E. Sánchez, and D. Borro, "A review of surgical robots for spinal interventions," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 9, no. 4, pp. 407–422, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1469

[3] Y.-R. Chen, S. Deb, L. Pham, and H. Singh, "Minimally invasive lumbar pedicle screw fixation using cortical bone trajectory – a prospective cohort study on postoperative pain outcomes," *Cureus*, vol. 8, 07 2016.

[4] G. Tortora, *Introduction to the Human Body 8th Edition Binder Ready Version with Lab Manual A&P 3rd Edition BRV WileyPLUS 8th Edition Prem and WileyPLUS 3rd Edition Prem Set*, ser. Wiley Plus Products Series. John Wiley & Sons, Incorporated, 2010. [Online]. Available: https://books.google.se/books?id=CdymuAAACAAJ

[5] D. Bijendra, X. Wu, Z. Jiang, L. Zhu, M. Promish, and R. Singh, "Adjacent level vertebral fractures in patients operated with percutaneous vertebroplasty," *Open Journal of Orthopedics*, vol. 08, pp. 116–126, 01 2018.

[6] J. Tang, Z. Zhu, T. Sui, D. Kong, and X. Cao, "Position and complications of pedicle screw insertion with or without image-navigation techniques in the thoracolumbar spine: a meta-analysis of comparative studies," *The Journal of Biomedical Research*, vol. 28, no. 3, pp. 228–239, 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4085560/#b24

[7] G. Németh, personal communication. July, 2020.

[8] H. Wang, Y. Zhou, J. Liu, J. Han, and L. Xiang, "Robot assisted navigated drilling for percutaneous pedicle screw placement: A preliminary animal study," *Indian journal of orthopaedics*, vol. 49, no. 4, p. 452–457, 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4510801/

[9] O. P. Gautschi, B. Schatlo, K. Schaller, and E. Tessitore, "Clinically relevant complications related to pedicle screw placement in thoracolumbar surgery and their management: a literature review of 35,630 pedicle screws," *Neurosurgical Focus FOC*, vol. 31, no. 4, p. E8, 2011. [Online]. Available: https://thejns.org/focus/view/journals/neurosurg-focus/31/4/2011.7.focus11168.xml

[10] T. Tjardes, S. Shafizadeh, D. Rixen, T. Paffrath, B. Bouillon, E. S. Steinhausen, and H. Baethis, "Image-guided spine surgery: state of the art and future

directions," *European spine journal*, vol. 19, no. 1, pp. 25–45, 2010. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2899744/

[11] Y. Liu, C. Zeng, M. Fan, L. Hu, C. Ma, and W. Tian, "Assessment of respiration-induced vertebral motion in prone-positioned patients during general anaesthesia," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 12, no. 2, pp. 214–218, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1676

[12] Ortoma, "Ortoma treatment solution™(ots)," http://ortoma.com/sv/vara-produkter/ortoma-treatment-solution/, 2020.

[13] B. Larsson. Ortoma (svenska). Youtube. [Online]. Available: https://youtu.be/Le8MXz0ToOQ

[14] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario, "Medical robotics and computer-integrated surgery," in *Springer Handbook of Robotics, 2nd Ed.*, 2016.

[15] A. Khan, J. E. Meyers, I. Siasios, and J. Pollina, "Next-Generation Robotic Spine Surgery: First Report on Feasibility, Safety, and Learning Curve," *Operative Neurosurgery*, vol. 17, no. 1, pp. 61–69, 09 2018. [Online]. Available: https://doi.org/10.1093/ons/opy280

[16] G. Mao, M. J. Gigliotti, D. Myers, A. Yu, and D. Whiting, "Single-surgeon direct comparison of o-arm neuronavigation versus mazor x robotic-guided posterior spinal instrumentation," *World Neurosurgery*, vol. 137, pp. e278 – e285, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1878875020301935

[17] J. Lonner, *Robotics in Knee and Hip Arthroplasty: Current Concepts, Techniques and Emerging Uses*. Springer International Publishing, 2019. [Online]. Available: https://books.google.se/books?id=y0eeDwAAQBAJ

[18] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: A comparison of four major algorithms," *Mach. Vision Appl.*, vol. 9, no. 5–6, p. 272–290, Mar. 1997. [Online]. Available: https://doi.org/10.1007/s001380050048

[19] A. Atawnih, D. Papageorgiou, and Z. Doulgeri, "Kinematic control of redundant robots with guaranteed joint limit avoidance," *Robotics and Autonomous Systems*, vol. 79, pp. 122 – 131, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889016000130

[20] J. Wang, Y. Li, and X. Zhao, "Inverse kinematics and control of a 7-dof redundant manipulator based on the closed-loop algorithm," *International Journal of Advanced Robotic Systems*, vol. 7, no. 4, p. 37, 2010. [Online]. Available: https://doi.org/10.5772/10495

[21] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2008. [Online]. Available: https://books.google.se/books?id=VsTOQOnQjCAC

[22] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, June 1994.

[23] Atracsys measurement solutions, "fusiontrack 250." [Online]. Available: https://www.atracsys-measurement.com/wp-content/documents/fTk250-datasheet.pdf

[24] U. R. A/S, *User Manual UR10/CB3*, 2017. [Online]. Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/27484/UR10_User_Manual_en_E67ON_Global-3.4.5.pdf

[25] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System, 2nd Edition.* Packt Publishing, 2018. [Online]. Available: https://books.google.se/books?id=MulODwAAQBAJ

[26] *Parameters for calculations of kinematics and dynamics*, https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/parameters-for-calculations-of-kinematics-and-dynamics-45257/, Universal Robots, [Accessed: 2019-11-26].

[27] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," 2004.

[28] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Springer. Springer-Verlag Berlin Heidelberg, September 2010. [Online]. Available: http://szeliski.org/Book/

[29] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2011. [Online]. Available: https://books.google.se/books?id=hdkytqtBcyQC