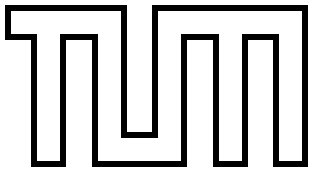


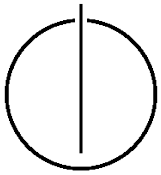
FAKULTÄT FÜR INFORMATIK

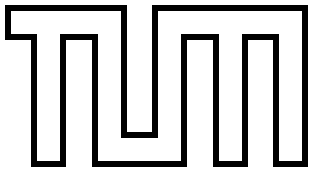
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Engineering Science

Path Planning for an Ophthalmic Surgical Robot using V-REP

Anh Tuan Pham





FAKULTÄT FÜR INFORMATIK

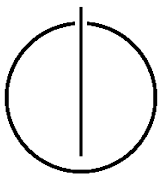
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Engineering Science

TEST TEST TESt

Pfadplanung für einen Assistenzroboter in der Augenchirurgie mit V-REP

Author:	Anna Baumeister
Supervisor:	Prof. Dr.-Ing. habil. Alois Knoll
Advisor:	Mingchuan Zhou, M.Sc.
Date:	26. Juni 2018



I assure the single handed composition of this bachelors's thesis only supported by declared resources.

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Munich, September 14, 2017

München, den 14. September 2017 Anna Baumeister

Acknowledgements

First and foremost, I would like to thank my advisor Mingchuan Zhou for his enthusiasm towards this project, endless patience and willingness to discuss about a problem for hours to figure out the solution together. I also would like to say special thanks to Manuel Viermetz for some excellent advice and last-minute proofreading. Lastly, I would like to thank my parents for their continuous support not only during my academic career but throughout my entire life.

Abstract

Robotic assisted minimally invasive surgery has gained increasing acceptance in recent years due to providing superior accuracy and instrument manoeuvrability. This is of interest especially for ophthalmic surgery, which requires high precision due to the delicate structure of the eye. Some severe diseases, such as retinal vein occlusion, cannot be treated appropriately today because of the physiological tremor in the surgeons hand. The surgical assistant developed in the iRAM!S project offers the capability to declare an arbitrary point in the workspace of the robot as a programmable Remote Center of Motion (RCM) and suppress motion of the tool at this point while maintaining high precision and rigidity through its novel Parallel Coupled Joint Mechanisms (PCJM). This thesis contrasts two methods for implementing the RCM constraint for the surgical robot: The augmented Jacobian method and the Levenberg-Marquardt algorithm. In the first part, the forwards and inverse kinematics as well as the Jacobian matrix for the end effector are reviewed. A virtual fixture is implemented by defining an extended robot task and the corresponding augmented Jacobian matrix. In the second part, the precision of the augmented Jacobian method is evaluated using the V-REP simulation environment and compared to the built-in inverse kinematic solver, which utilizes the Levenberg-Marquardt algorithm. The augmented Jacobian method is shown to maintain the RCM point with high accuracy, even for a suboptimal choice of control parameters.

Contents

1 Introduction

Introduction:
Drug injections in eye surgeries could be a very challenging task for human surgeons. Tremors and erroneous posing of the needle could lead to damages to the patient's eyes and are not easy to prevent completely. Using an assistance robot, which provides higher precision and eliminates human errors could reduce the risks. An important task in development of the robot is to calculate the position and rotation as accurately as possible for a real-time system. An algorithm developed by Ramona Schneider, which uses OCT images, to tackle technical difficulties such as lack of available information, image noises and deformation. This allows the system to execute the calculations with great accuracy. Still there is a problem while implementing it in a real time system: The algorithm takes too much time and resources to execute. A different approach is required to accelerate the algorithm to a real-time suitable execution time. Using the same algorithm developed by Ramona, the code first needs to be restructured to utilize the full advantages of graphic cards with OpenCL, by parallelizing the code as much as possible. Furthermore, an in-depth research to understand the structure and specification of a graphic card is required to optimize the code to reach the full potential of the processing units.
1.1 The Algorithm
1.1.1 Objectives:
The main goal of Ramona's work is to develop a reliable method to compute the position and rotation of a beveled needle. The needle can be digitally reconstructed as a point cloud by utilizing the obtained OCT images. By matching the point cloud with the CAD model, an accurate determination of rotation and position can be achieved.
Two different approaches were implemented and taken into consideration. Which are (Source):
A developed approach for object recognition and pose estimation for rigid objects using Clustered Viewpoint Feature Histograms (CVFH) and camera roll histograms described in a paper by Aldoma et al. [5].
A new approach similar to the Iterative Closest Point (ICP) algorithm which calculates and utilizes the direction of the needle, and then finds the rotation-shift combination along the direction with the best matched points.
After several trials and tests it was concluded that the second approach was the best solution from the two approaches presented. The solution was also adopted in this thesis with two main objectives: Reproduction of the same accuracy, acceleration the software to an acceptable level for real time systems. Broadly speaking, the program consists of two main parts: Processing OCT images and a shifting algorithm.

The goal of processing images is to read, analyze a set of pictures and reconstructing a needle point cloud from given input data. The important parts of this process is to remove noise, downsample OCT Images, which are black and white slice captures of the needles. For example : ¡EXAMPLE$_{IMAGE_{OCT}} > Whereasthewhiteareaistheactualsliceoftheneedle.Withinthescopeofthi$
$consumingtask, willbetranslatedintoGPUconformedcode.Therefore, thefocusofthisthesisistheshiftingo$

2 Background

In this chapter, the mechanical design and kinematic relationships of the surgical robot developed in the iRAMs! project will be reviewed. The Jacobian matrix and the Jacobian pseudoinverse approach are presented as an iterative solution to the inverse kinematics problem. Lastly, there is a short introduction to the V-REP simulation environment and the Dampened Least Squares method used by V-REPs internal inverse kinematics solver.

2.1 Mechanical Design

The surgical robot considered in this thesis consists of two subsequent *Parallel Coupled Joint Mechanisms* (PCJMs), one single prismatic joint and one optional revolute joint, providing six degrees of freedom.
Each PCJM consists of two stick-slip piezo actuators moving parallel to each other at a distance *d*. Translating both sliders the same distance results in a pure translation of the manipulator, while a differential movement results in an additional rotation. In contrast to a linear chain of prismatic and revolute joints, the PCJM approach provides higher stiffness and precision.
For a simpler kinematic analysis, the PCJM elements can be treated as a prismatic and subsequent revolute joint. The two translations L_1 and L_2 correspond to the translation of the prismatic joint q_0 and angle of the revolute joint q_1 as

$$q_0 = \frac{L_1 + L_2}{2} \tag{2.1}$$

$$q_1 = \arctan(\frac{L_2 - L_1}{d}) \ . \tag{2.2}$$

Conversely, given the two joint positions q_0 and q_1, the slider positions can be uniquely determined via:

$$L_1 = \frac{d \cdot \tan(q_1)}{2} + q_0 \tag{2.3}$$

$$L_2 = 2q_0 - L_1 \ . \tag{2.4}$$

2.2 Forward Kinematics

In order to describe the position of the end effector relative to the base coordinate frame we determine the Denavit-Hartenberg parameters [11] of the robot. First, a coordinate system is placed in each joint according to the following rules:

- The z-axis must point along the axis of rotation or translation of the joint.
- The x-axis for the base frame is a free choice. For each subsequent joint, the x-axis is co-linear with the common normal of the current and previous z-axis, with its origin at the intersection with the new z-axis. $x_n = z_n \times z_{n-1}$. The origin of the new coordinate system is not necessarily in the center of the joint.
- The y-axis is chosen so that it forms a right handed coordinate system with x and z.

Figure 2.1 illustrates the placement of the coordinate systems for each joint. Having established the coordinate systems, we can now determine the Denavit-Hartenberg parameters d_i, θ_i, a_i and α_i:

- d_i is the depth of origin i to origin $i-1$ along the previous z-axis z_{i-1}.

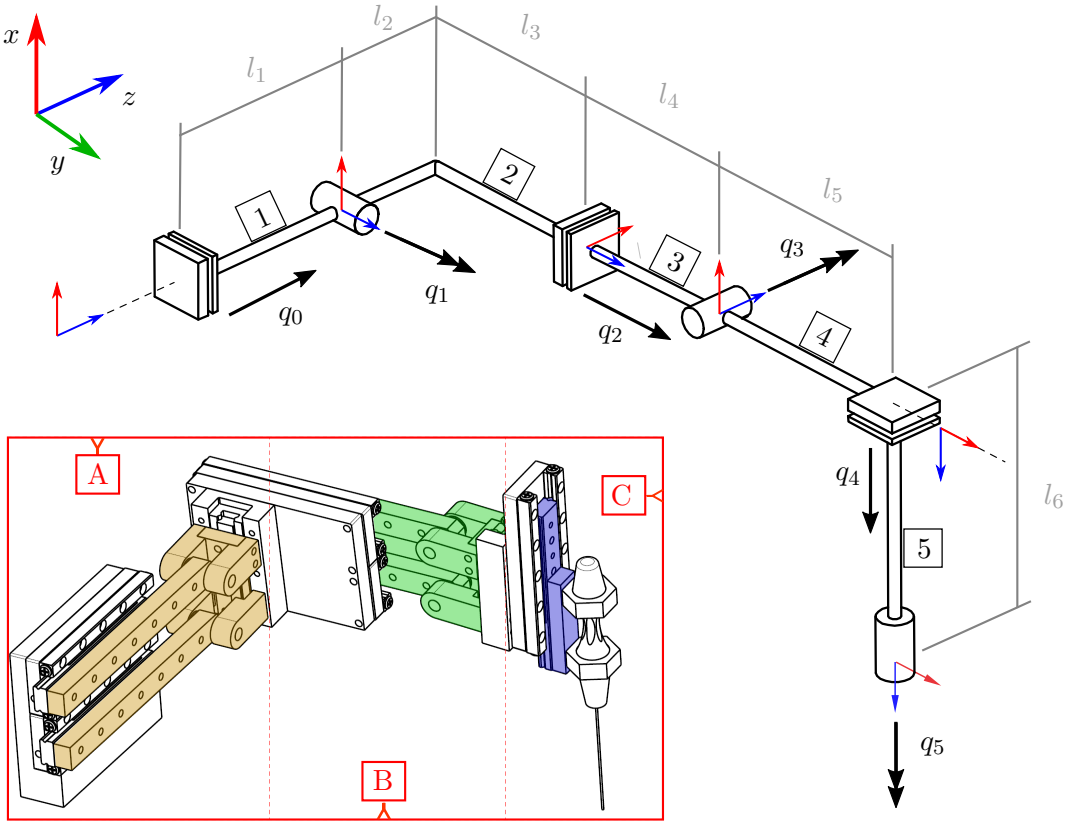


Figure 2.1: The simplified serial manipulator, consisting of two subsequent PCJM elements (A and B) and a simple prismatic stage (C). Each PCJM element can be modelled as a translation and a following rotation. Adapted from [9].

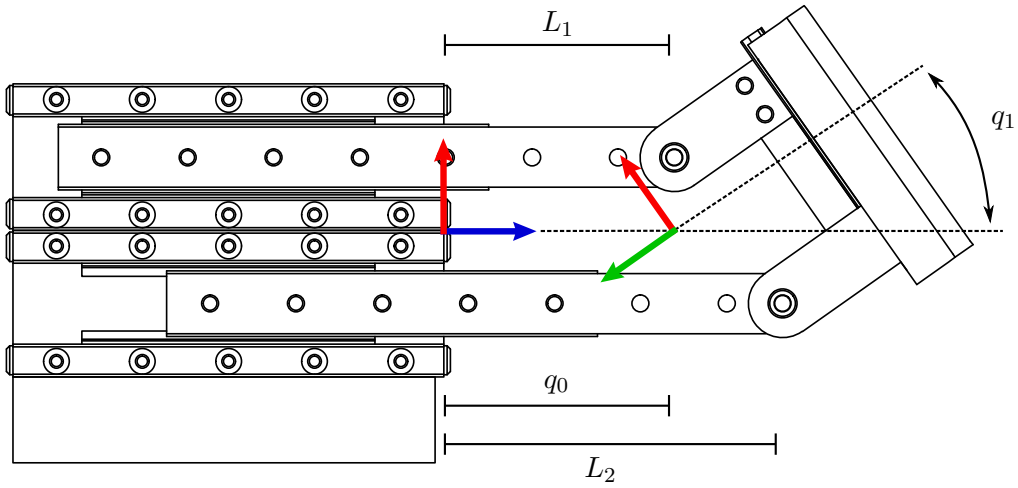


Figure 2.2: A single Parallel Coupled Joint Mechanism. Differential displacement of the two prismatic joints L_1 and L_2 results in both a translation and rotation of the subsequent link. Adapted from [16].

- θ_i is the angle about the previous z axis z_{i-1} to align x_{i-1} with the new origin i.
- a_i is the distance between the current and previous origin along the previous rotated x-axis x_{i-1}.
- α_i is the angle between z_{i-1} and z_i along the current x-axis x_i.

We can now determine the homogeneous transformation that describes the position of each coordinate frame with respect to the preceding one. For each link i, the transformation from link $i-1$ to link i is given by:

$$T_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Table 2.1 shows the values obtained for the Denavit-Hartenberg parameters. The values for the link lengths l_i are obtained from the CAD model.
The transformation from the origin to the end effector can now be obtained by subsequently multiplying the transformation matrices of each subsequent coordinate fame pair:

$$T_0^6 = \Pi_{i=1}^6 T_{i-1}^i \tag{2.6}$$

$$= \begin{bmatrix} -s1s5-c1c5s3 & c1s3s5-c5s1 & -c1c3 & l2s1-c1c3(q4+l6)-l5c1s3 \\ c3c5 & -c3s5 & -s3 & l3+l4+q2-s3(q4+l6)+l5c3 \\ c5s1s3-c1s5 & -c1c5-s1s3s5 & c3s1 & l1+q0+l2c1+c3s1(q4+l6)+l5s1s3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

with $ci, si := \cos(i), \sin(i)$, where the 3×3 matrix in the upper left describes the rotation of the tool relative to the base frame while the 3×1 vector in the upper right describes the position of the needle tip with respect to the base coordinate system and will be denoted as $O_6(q)$.

Link	θ_i	d_i	a_i	α_i	Offset	[mm]
1	0	$q_0 + l_1$	0	$-\frac{\pi}{2}$	l_1	71
2	$q_1 - \frac{\pi}{2}$	l_3	l_2	0	l_2	33.5
3	$\frac{\pi}{2}$	$l_4 + q_2$	0	$\frac{\pi}{2}$	l_3	-23
4	$\frac{\pi}{2} + q_3$	0	l_5	$-\frac{\pi}{2}$	l_4	71
5	0	$l_6 + q_4$	0	0	l_5	28
6	$q5$	0	0	0	l_6	97

Table 2.1: The Denavit-Hartenberg parameters of the serial robot. A negative value for a link length is the result of the choice for the origin of a coordinate system within a PCJM and will not lead to an error in subsequent computations.

2.3 Inverse Kinematics

The forward kinematics relation gives information about how the position and orientation of the end effector will change with a change in joint positions. Each distinct joint vector $\boldsymbol{\theta} = (q_1, q_2, ..., q_5)$ leads directly to the pose of the end effector as determined by Eqn. 2.7. It follows that in order to control the pose of the needle we need to consider the inverse problem, that is, finding a set of joint variables for a desired end effector pose. The inverse kinematics problem is, in general, more difficult to solve, may not always have a unique or best solution or be guaranteed to have a closed form equation for the solution. The Jacobian matrix provides an iterative solution to the inverse kinematics problem.

2.3.1 The Jacobian Matrix and the Jacobian Pseudoinverse

With the Jacobian matrix we want to relate the linear and angular velocity of the end effector $\dot{x}$ to the vector of joint velocities $\dot{\boldsymbol{\theta}}(t)$. It is a function of the joint angles defined by

$$J(\boldsymbol{\theta}) \quad = \quad (\frac{\partial x_i}{\partial \theta_j})_{i,j} \ , \tag{2.8}$$

with $J \in \mathbb{R}^{m \times n}$, m: degrees of freedom, n: number of links.
According to [21], each column of the Jacobian matrix can be calculated independently using the Denavit-Hartenberg parameters as follows:

$$J_0^n = [J_1, J_2, ..., J_n] \tag{2.9}$$

where n is the number of links. For this robot $n = 6$.
In case joint i is a revolute joint J_i is given by

$$J_i = \begin{bmatrix} Z_{i-1} \times (O_n - O_{i-1}) \\ Z_{i-1} \end{bmatrix} \tag{2.10}$$

and in case of a prismatic joint

$$J_i = \begin{bmatrix} Z_{i-1} \\ 0 \end{bmatrix} \tag{2.11}$$

where Z_i and O_i are composed of the first three elements of the third and fourth column in T_0^i, respectively. Using Eqns.2.5, 2.6, and Table 2.1, the Jacobian matrix for every link with respect to the base coordinate frame can be calculated. For the full transformation from the base to the end effector this results in:

$$J_0^6 = \begin{bmatrix} 0 & l2c1{+}l6c3s1{+}l5s1s3{+}q4c3s1 & 0 & c1(l6s3{-}l5c3{+}q4s3) & -c1c3 & 0 \\ 0 & 0 & 1 & -l6c3{-}l5s3{-}q4c3 & -s3 & 0 \\ 1 & l6c1c3{-}l2s1{+}l5c1s3{+}q4c1c3 & 0 & -s1(l6s3{-}l5c3{+}q4s3) & c3s1 & 0 \\ 0 & 0 & 0 & s1 & 0 & -c1c3 \\ 0 & 1 & 0 & 0 & 0 & -s3 \\ 0 & 0 & 0 & c1 & 0 & c3s1 \end{bmatrix} \tag{2.12}$$

Since the tool used by the surgical robot in this thesis is a straight needle, the rotation about the needle axis q_5 has no effect on the end effector pose. Thus for the derivation of the extended Jacobian matrix, only the transformation from the base frame to the needle tip, without the final revolute joint will be considered:

$$J_0^5 = \begin{bmatrix} 0 & l2c1{+}l6c3s1{+}l5s1s3{+}q4c3s1 & 0 & c1(l6s3{-}l5c3{+}q4s3) & -c1c3 \\ 0 & 0 & 1 & -l6c3{-}l5s3{-}q4c3 & -s3 \\ 1 & l6c1c3{-}l2s1{+}l5c1s3{+}q4c1c3 & 0 & -s1(l6s3{-}l5c3{+}q4s3) & c3s1 \\ 0 & 0 & 0 & s1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c1 & 0 \end{bmatrix} \tag{2.13}$$

Lastly, we need to check if there are any singularities of J_0^5 within the workspace of the robot. This can be done by taking a look at the determinant of the Jacobian matrix which is

$$det(J_0^5) \quad = \quad \cos(q_1)\ \cos(q_3)^2\ . \tag{2.14}$$

It can only reach zero if either q_1 or q_3 are zero, which is not a possible configuration for the robot.
Given a vector of joint velocities $\dot{\boldsymbol{\theta}}$, the vector of end effector velocities is given by

$$\dot{\boldsymbol{x}} \;=\; J(\boldsymbol{\theta})\,\dot{\boldsymbol{\theta}}\,. \tag{2.15}$$

This relationship is now used as an iterative solution to the inverse kinematics problem. For some error $\boldsymbol{e}$ between the current and target end effector pose, we seek to find an update value $\Delta\boldsymbol{\theta}$ for the joint angles such that the change in end effector positions is equal to the error $\boldsymbol{e}$.

$$\boldsymbol{e} = \Delta\boldsymbol{x} \approx J(\boldsymbol{\theta})\,(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \tag{2.16}$$

To obtain a good estimate for $\Delta\boldsymbol{\theta}$ we employ the Moore-Penrose Pseudoinverse of J, $J^\dagger$

$$\Delta\boldsymbol{\theta} = J^\dagger(\boldsymbol{\theta})\,\boldsymbol{e} \tag{2.17}$$

The resulting vector $\Delta\boldsymbol{\theta}$ is the unique vector of smallest magnitude which minimizes $||J\Delta\boldsymbol{\theta} = \boldsymbol{e}||$ [5]. The Jacobian matrix only provides an approximation of the actual change in end effector pose because it is a function of $\boldsymbol{\theta}$. For small errors, the change in joint positions is small enough for J to be roughly constant and $\Delta\boldsymbol{x}$ to be a good approximation to $\boldsymbol{e}$. If the error between the current and desired end effector pose is large, we use an iterative approach:

- Calculate the Jacobian matrix from the current joint positions $\boldsymbol{\theta}$ and compute $J^\dagger$.
- Obtain an update $\Delta\boldsymbol{\theta}$ for small step of predetermined magnitude towards the goal pose using Eqn. 2.17.
- Repeat with the new joint positions $\boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ until the pose is sufficiently close to the target pose.

Finally, it is important to note that the Jacobian pseudoinverse method is often plagued by numeric instability near singularities where J is rank deficient, resulting in large values for $\Delta\boldsymbol{\theta}$ [6]. However, since the Jacobian matrix of the surgical robot has no singularities within its workspace, this is not an issue for this thesis.

2.4 The V-REP Simulation Environment

2.4.1 Introduction to V-REP

In order to verify and evaluate the kinematic relations established in the previous section, the movement of the robot is simulated using V-REP[19]. V-REP is a general purpose robot simulator with an integrated development environment distributed by Coppelia Robotics [1]. A V-REP simulation or *simulation scene* consists of several different *scene objects* that are assembled in a *scene hierarchy* with each element having exactly one parent element and a variable amount of child objects. The objects relevant for our purposes are:

- **Shape:** Any rigid body (e.g. a link of the surgical robot), represented by a triangle mesh. There are different types of shapes supported by V-REP: *pure* shapes (e.g. spheres, cuboids), *convex* shapes and *random* shapes, which are neither pure nor convex. All shapes used in this simulation are random shapes, since no collision detection or dynamic calculation will be performed that would benefit from modelling the robot using simple shapes with a lower triangle count.
- **Joint:** A joint object constrains how two *scene objects* move relative two each other. In the scene hierarchy, a joint will be placed as a child of the first object and parent of the second object. Relevant for this model are revolute joints (rotation about one axis) and prismatic joints (translation along one axis).
- **Dummy:** A dummy object has no physical properties or dimensions, it is used as a 'helper object'. In the surgical needle model, it is used to introduce different frames of reference on the same link, for example at the base, RCM point and tip of the needle. Dummy objects are also used as *tip* and *target* objects for kinematic and dynamic calculations.
- **Graph:** A graph object records a given data stream and may directly display it. For the surgical robot, a graph object is attached to the RCM point and tip of the needle to record its position over time.

The model can be controlled either by a remote API or an embedded child script. For this thesis, all communication with the simulation is carried out by MATLAB [14] scripts. Using the V-REP remote API functions [3], the pose of all simulation objects can be received and modified.
The V-REP inverse kinematics module handles IK resoultion via *IK groups* consisting of a number of *IK elements*, one for each constraint that needs to be fulfilled simultaneously. For the surgical robot, one IK group consiting of two IK elements is needed: the first element is responsible for placing the needle tip within the eye, the other for fixing the RCM point at the trocar entry. Each IK element is made up of a *tip-target dummy pair*: The 'tip' dummy is placed on the robot while the 'target' dummy can be placed anywhere in the workspace of the robot. Once the simulation is started, each 'tip' dummy will automatically try to converge to its corresponding 'target' dummy. The position of the two target dummies, one for the tip and one for the RCM, can then be controlled by the MATLAB script.

2.4.2 Dampened Least Squares

The V-REP inverse kinematics solver uses a Dampened Least Squares (DLS) method, also referred to as the Levenberg-Marquardt algorithm (LMA) [12, 13]. It is a method to solve nonlinear least squares problems and can be seen as a mixture of the Gauss-Newton method and standard gradient descent [4]. As explained in Section 2.3, the inverse kinematics problem is to find a value for $\Delta\boldsymbol{\theta}$ which minimizes the error in the pose $\boldsymbol{e} = J(\boldsymbol{\theta})\,\Delta\boldsymbol{\theta}$. The Jacobian pseudoinverse method minimizes this quantity by setting $\Delta\boldsymbol{\theta} = J^{\dagger}(\boldsymbol{\theta})\,\boldsymbol{e}$, which

gives an optimal solution in the sense of least squares but is numerically unstable near singularities. The DLS algorithm instead minimizes

$$||J\Delta\boldsymbol{\theta} - \boldsymbol{e}||^2 + D^2||\Delta\boldsymbol{\theta}||^2, \tag{2.18}$$

where D is the non-zero *damping constant*. It was shown in [23] that the minimum of Eqn. 2.18 can be expressed as

$$\Delta\boldsymbol{\theta} \quad = \quad J^T(J\,J^T + D^2 I)^{-1}\boldsymbol{e}\,. \tag{2.19}$$

One large advantage of Eqn. 2.19 is that the matrix inversion never has to be explicitly computed, instead a vector $\boldsymbol{f}$ can be found through row operations such that

$$(J\,J^T + D^2 I)^{-1}\boldsymbol{f} \quad = \quad \boldsymbol{e}\,. \tag{2.20}$$

Then, the DLS solution is given by $J^T\boldsymbol{f}$. One important choice is the value of the damping constant D. A larger value of D corresponds to more stable behaviour near singularites but reduces the convergence rate. Methods for choosing and adapting D based on the configuration of the robot are proposed for example in [7] and [8].

3 Method

In the first part of this chapter, the concept of the augmented Jacobian matrix is introduced. In this approach, the redundancy of the robot is used to define the additional kinematic task of maintaining a remote center of motion. The resulting composite Jacobian matrix can then be used to simultaneously control the position of the tool tip within the eye and prevent movement of the needle at the point of incision. A method for estimating the precision of the augmented Jacobian matrix is proposed. The second part of this chapter describes the transfer of the CAD model of the robot into a V-REP simulation model and explains how to adapt the model in order to be usable by the built-in inverse kinematics solver.

3.1 Remote Center of Motion Constraint

After entry into the eye via the trocar, the movement of the tool is further constrained. In order to minimize damage on the surrounding tissue, the needle may only translate along its own axis or rotate about the entry point.
When moving the needle outside of the eye, we care about the position of the tool tip, but not necessarily about its orientation. For each position of the tool tip, there are infinitely many ways in which the other links can be arranged in order to reach the desired end effector position, the manipulator is *redundant*. Disregarding the rotation about the tool axis, the robot has $n = 5$ degrees of freedom. $m = 3$ degrees of freedom are needed to control the position of the tool tip. Now let $r = n - m$ be the degree of 'redundancy'. Following the approach presented in [20], a set of r task-related kinematic functions,

$$\boldsymbol{\Phi} \quad = \quad \{\phi_1(\boldsymbol{\theta}), \dots, \phi_r(\boldsymbol{\theta})\} \,, \tag{3.1}$$

can be formed to describe the variation of these remaining degrees of freedom and define an additional task for the robot. The choice of these functions is somewhat arbitrary, yet there are some choices that produce algorithmic singularities (discussed at the end of this section).
By then specifying a set of end effector position coordinates,

$$\boldsymbol{Y} \quad = \quad \{y_1(\boldsymbol{\theta}), \dots, y_m(\boldsymbol{\theta})\} \,, \tag{3.2}$$

and a set of kinematic functions $\boldsymbol{\Phi}$, for each robot task there is a unique solution for the joint positions. Both the kinematic functions and the end effector coordinates can be combined

to provide a set of generalized *configuration variables* for the manipulator,

$$\begin{aligned} \boldsymbol{X} &= \{\boldsymbol{Y}, \boldsymbol{\Phi}\} = \{y_1(\boldsymbol{\theta}), \dots, y_m(\boldsymbol{\theta}) \vdots \phi_1(\boldsymbol{\theta}), \dots, \phi_r(\boldsymbol{\theta})\} \quad (3.3) \\ &= \{x_1(\boldsymbol{\theta}), \dots, x_n(\boldsymbol{\theta})\} \, . \quad (3.4) \end{aligned}$$

We obtain the *augmented forward kinematic model* which relates the configuration vector $\boldsymbol{X}$ to the joint angles $\boldsymbol{\theta}$ via the *augmented Jacobian matrix* $J_a(\boldsymbol{\theta})$.

$$\dot{\boldsymbol{X}}(t) = J_a(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \, , \quad (3.5)$$

where

$$J_a(\boldsymbol{\theta}) = \begin{bmatrix} J_e(\boldsymbol{\theta}) \\ J_c(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}} \end{bmatrix} \, . \quad (3.6)$$

Is the $n \times n$ augmented Jacobian matrix, consisting of $J_e \in \mathbb{R}^{m \times n}$, that is related to the motion of the end effector and $J_c \in \mathbb{R}^{r \times n}$ related to the additional kinematic task. This accomplishes *simultaneous* control of end effector motion as well as utilizing the redundancy to achieve an additional task.
The procedure for the calculation of the extended Jacobian Matrix has already been presented in [18] and will be reviewed in this thesis. In this setup, the additional kinematic task is to force the velocity of a point $\boldsymbol{p}_{rcm}$ on the needle to zero. The remote center of motion is assumed to be on the axis of the tool and can be written as:

$$\boldsymbol{\Phi} = \boldsymbol{p}_{rcm} = \boldsymbol{p}_{base} + \lambda(\boldsymbol{p}_{tip} - \boldsymbol{p}_{base}) \, , \quad (3.7)$$

with $\boldsymbol{p}_{rcm}$, $\boldsymbol{p}_{base}$, and $\boldsymbol{p}_{tip} \in \mathbb{R}^{3\times 1}$ and $\lambda \in \mathbb{R}^{1\times 1}$.
Differentiating with respect to time:

$$\dot{\boldsymbol{p}}_{rcm} = \dot{\boldsymbol{p}}_{base} + \lambda(\dot{\boldsymbol{p}}_{tip} - \dot{\boldsymbol{p}}_{base}) + \dot{\lambda}(\boldsymbol{p}_{tip} - \boldsymbol{p}_{base}) \quad (3.8)$$

As previously established, the change in position of any link i can be described as $p_i = J_i q_i$ and thus:

$$\dot{\boldsymbol{p}}_{rcm} = J_{base}\dot{\boldsymbol{\theta}} + \lambda(J_{tip}\dot{\boldsymbol{\theta}} - J_{base}\dot{\boldsymbol{\theta}}) + \dot{\lambda}(\boldsymbol{p}_{tip} - \boldsymbol{p}_{base}) \, , \quad (3.9)$$

where J_{base} and J_{tip} are the Jacobian matrices corresponding to point $\boldsymbol{p}_{base}$ and $\boldsymbol{p}_{tip}$ respectively. J_{tip} describes the linear velocity of the tool tip, disregarding the rotation around its own axis (q_5). Hence, J_{tip} is equal to the top three rows of J_0^5 calculated in the previous section. The points $\boldsymbol{p}_{tip}$ and $\boldsymbol{p}_{base}$ are on the same rigid link, only at a different distances to he previous joint. Thus, J_{tip} can be modified to describe the motion of point $\boldsymbol{p}_{base}$ by substituting l_6 with 0, yielding J_{base}. $\dot{\boldsymbol{p}}_{rcm}$ can be expressed in matrix form as:

$$\dot{\boldsymbol{p}}_{rcm} = \begin{bmatrix} J_{base} + \lambda(J_{tip} - J_{base}) \\ \boldsymbol{p}_{tip} - \boldsymbol{p}_{base} \end{bmatrix}^T \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\lambda} \end{bmatrix} = J_c \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\lambda} \end{bmatrix} = J_c \, \dot{\boldsymbol{\theta}}_{ext} \, , \quad (3.10)$$

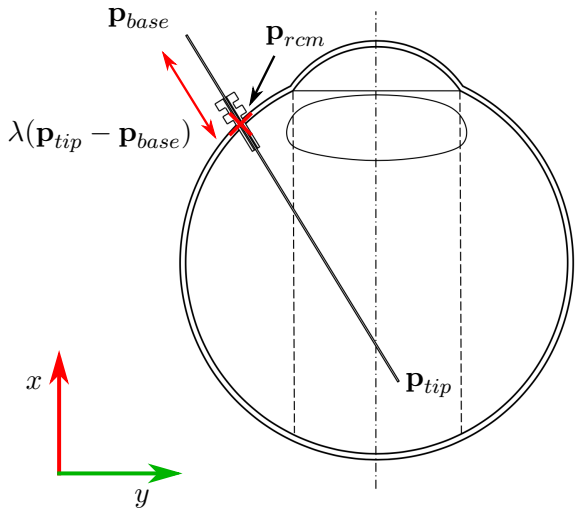


Figure 3.1: Location of $\boldsymbol{p}_{rcm}$

with $\dot{\boldsymbol{p}}_{rcm} \in \mathbb{R}^{3\times1}$, J_{tip}, $J_{base} \in \mathbb{R}^{3\times5}$ and $J_c \in \mathbb{R}^{3\times6}$. The set of extended joint angles $\boldsymbol{\theta}_{ext}$ are given as

$$\dot{\boldsymbol{\theta}} \quad = \quad \{q_1, \ldots, q_5, \lambda\} \; . \tag{3.11}$$

Now we can enforce the remote center of motion constraint by setting the velocity of the RCM point to zero. That is, $\boldsymbol{p}_{rcm}$ should not change its position.

$$\dot{\boldsymbol{p}}_{rcm} = J_c \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\lambda} \end{bmatrix} = 0^{3\times1} \tag{3.12}$$

Hence we have defined the additional kinematic task $\boldsymbol{\Phi} = \{x_{rcm}, y_{rcm}, z_{rcm}\}$ and its associated Jacobian J_c in Cartesian coordinates with respect to the robot base frame. Although forcing $\dot{\boldsymbol{p}}_{rcm}$ to be zero is a constraint of dimension three, modelling the variation of λ increases the degrees of freedom by one. Hence, $r = 2$ degrees of freedom are lost due to the RCM constraint and $m = 3$ degrees of freedom remain to place the tool tip within the eye. We choose a set of variables $\boldsymbol{Y}$ to represent the position of the end effector that doesn't introduce algorithmic singularities.

$$\boldsymbol{Y} \quad = \quad \{x, \phi, \psi\} \; , \tag{3.13}$$

where x is the global x-position of the tool tip and ϕ and ψ describe the angle of the tool. $\boldsymbol{Y}$ and $\boldsymbol{\Phi}$ can now be combined to form a set of generalized kinematic functions for the end effector performing the extended task,

$$\boldsymbol{X}_{ext} \quad = \quad \{x, \phi, \psi, x_{rcm}, y_{rcm}, z_{rcm}\} \; , \tag{3.14}$$

and the associated augmented Jacobian

$$J_a = \begin{bmatrix} J_e & 0_{3\times 1} \\ & J_c \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}} \end{bmatrix} \tag{3.15}$$

where J_e is the Jacobian matrix relating the end effector x-position and tool angles ϕ and ψ to the joint positions. Thus, J_e is composed of the first, fifth, and sixth row of $J_{base} \in \mathbb{R}^{3\times 5}$. Lastly, we need to check if there are any singularities within J_a, i.e. if the kinematic functions we chose have any dependencies. One has to differentiate between kinematic and algorithmic singularities. Kinematic singularities refer to any configurations in which J_e is rank deficient, e.g. when two links joined by a revolute joint become co-linear. It was previously established that this is not possible due to the mechanical design of the robot. Algorithmic singularities are introduces by the choice of the kinematic functions $\boldsymbol{\Phi}$, if J_c in itself is rank-dependent or if two rows of J_e and J_c are linearly dependent. For the presented case the determinant

$$det(J_a) \quad = \quad -l_6 \cos(q_1)^2 \cos(q_3)^2 \sin(q_1) \ , \tag{3.16}$$

is only zero for $q_3 = \pm\frac{\pi}{2}$, which is not a reachable configuration. J_a is listed as Eqn. 6.1 in the Appendix.

3.2 Controller Design

Having established the augmented Jacobian matrix we can now compute the change in joint position necessary to achieve a certain change in end effector position. However, iteratively using $\Delta\boldsymbol{\theta} = J^{\dagger}\Delta\boldsymbol{x}$ often does not result in a smooth trajectory from the current to the target pose. If for example the target pose differs from the current pose in only one angle, the Jacobian matrix will return a vector Q correcting the error in angle by rotating the needle. in the next iteration the angle will be right but the RCM point will have shifted and the Jacobian matrix will return a vector Q that translates the needle back. This causes the needle to oscillate until all entries in the current pose are sufficiently close to the target pose. There are two possible solutions to this problen: Either reduce the step size so that the oscillation is very small compared to the overall error in the pose, or introduce a set of weights to determine how fast a parameter should converge. This is done by using the matrix K that multiplies the values in $\Delta\boldsymbol{x}$ such that:

$$\Delta\boldsymbol{\theta} \quad = \quad J^{\dagger}\boldsymbol{K}\Delta\boldsymbol{x} \tag{3.17}$$

where $K \in \mathbb{R}^{6\times 6}$ is a diagonal, positive matrix that carries the weights for each entry in $\Delta\boldsymbol{x}$. Choosing good parameters for K is challenging because of the interdependence of the entries in $\Delta\boldsymbol{x}$. A possible choice for K is presented in Section 4.1.1.

3.3 Control Loop

Algorithm 1 shows the control loop that implements Eqn. 3.17 in order to perform a small step from the current position to a target position. It is implemented in a MATLAB script that can receive the positions of simulation objects and returns the updated values for the joint angles and λ calculated by the Jaobian matrix.

Algorithm 1 Control_Loop($\boldsymbol{K}$, target_pose, J_a, λ)

```
current_pose ← get_task_pose()
while max(abs(target_pose − current_pose)) > 0.0002 do
    error ← target_pose − current_pose
    joint_positions = get_joint_positions()
    Q ← L_to_Q(joint_positions)
    [ΔQ, Δλ] ← J_a(Q, λ)^-1 K error
    ΔL ← Q_to_L(ΔQ)
    set_joint_positions(joint_positions + ΔL)
    λ ← λ + Δλ
    move_RCM(λ)
    current_pose ← get_task_pose()
end while
```

In each iteration of the main body, the error between current and target pose is computed. The current slider positions L are obtained from the simulation and converted to the set of simplified joint variables q_i. The augmented Jacobian matrix is computed using λ and the current joint positions, multiplied with the control matrix K and the error to obtain update values both for the simplified joint variables q_i as well as λ. The joint variables are converted back to a set of slider displacements and returned to the simulation. the RCM point in the simulation model is moved according to $\Delta\lambda$ and the new current task pose $\boldsymbol{X}_{ext} = \{x, \phi, \psi, x_{rcm}, y_{rcm}, z_{rcm}\}$ is obtained. The main body of the control loop is executed until the current pose is sufficiently close to the target pose (ensuring the position of the RCM is within a bounded accuracy).

3.4 Measuring the Accuracy of the RCM position

In order to compare the precision of the augmented Jacobian matrix, a series of constrained movements of the needle within the eye is performed. Figure 3.2 illustrates the sequence of points the needle tip should follow.
Injection begins by moving the tool tip from an arbitrary point outside the eye to position **2** via unconstrained movement. Once point **2** is reached, the RCM point will be positioned at the tip of the needle ($\lambda = 1$) and control via the augmented Jacobian begins. The tool is pivoted around its tip until point **1** lies on the needle, and thus the needle is aligned with the trocar. The needle is then moved forwards in a straight line until its tip reaches point

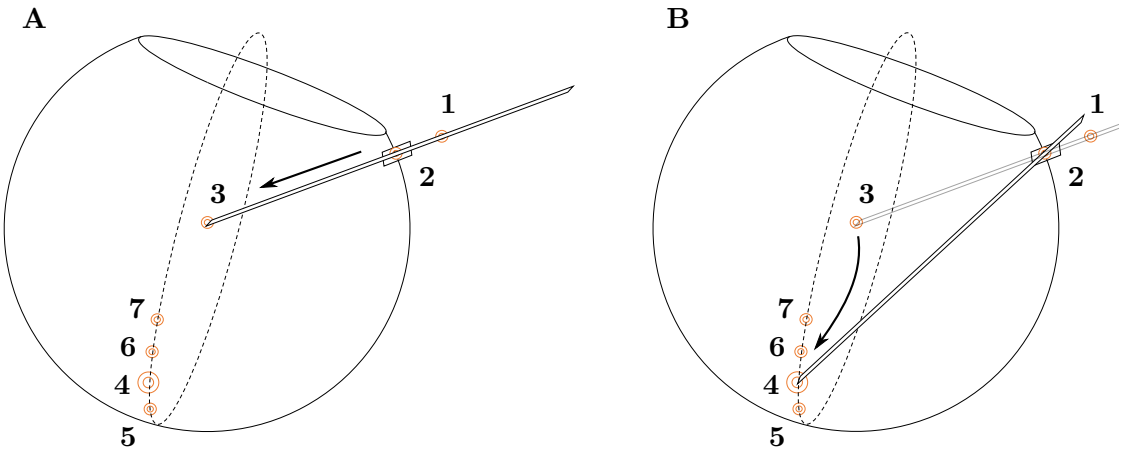


Figure 3.2: Setup for measuring the precision of the RCM point. **A**: The needle enters the eye through the trocar until it reaches point **3**. **B**: The needle tip moves from point **3** to points **4**-**7** while keeping the RCM point, located at **2**, steady.

3. From there, points **4**-**7** located on the inner surface of the eye will be visited, returning to position **3** each time.

3.5 Modelling the Surgical Robot in V-REP

This section describes the process of generating two Simulation models that can be controlled via the MATLAB remote API from the CAD data of the robot: One model for use in forwards kinematics mode, where the slider positions L_i are controlled via a MATLAB script according to the values computed by the Jacobian matrix, and one model for use in inverse kinematics mode, where only the target positions for the needle tip and the RCM are controlled by the MATLAB script and the joints are controlled by the internal IK solver. The CAD model of the robot is first imported into V-REP as a single triangle mesh. Using the automatic mesh subdivision, a new shape is generated for all elements that are not connected via a common edge and thus, all components of the robot are extracted as separate rigid links. Now, joint objects are added to specify the movement of the links relative to each other. Figure 3.3 shows the finished model of the robot.
Next, we need to specify how the different links and joints are connected with each other. In the real robot, a PCJM consists of two sliders contribute that to the translation and rotation of a subsequent stage. This cannot be directly modelled in V-REP because each element (e.g. the middle stage) can only have one parent object (i.e. one of the sliders in the PCJM). To assemble the robot model into a single kinematic chain in which every object has only one parent element, we first connect the robot base plate with the needle

via one slider of each PCJM (corresponding to L_2 and L_4). From the needle, we make our way back down towards the middle plate using slider L_3 and from the middle pate to the base plate via slider L_1. Since this second pair of sliders L_3 and L_1 is now "dangling" from their respective target plate (indicated by a green overlay in Figure 3.4) we need to inform the simulation that the two sliders should be connected to their respective base plates. Therefore, two sets of tip-target pairs are introduced. For each pair, one dummy is attached to the slider and one to the corresponding base plate and both are constrained to overlap in position and orientation (indicated by blue dotted arrows in Figure 3.4).
Until now, the modelling process has been the same for the forwards and inverse kinematics model. If we want to control the robot in forwards kinematics mode, the five prismatic joints corresponding to the stick-slip piezo actuators are set to "Force/Torque" mode and all other joints are set to "passive mode", meaning that they will not be actuated but can be passively rotated or displaced by connected moving links. For the inverse kinematics model, **all** joints are put into inverse kinematics mode so they can be controlled by the inverse kinematics solver. Additionally, we need to define the inverse kinematic task for the robot, described in the next section.

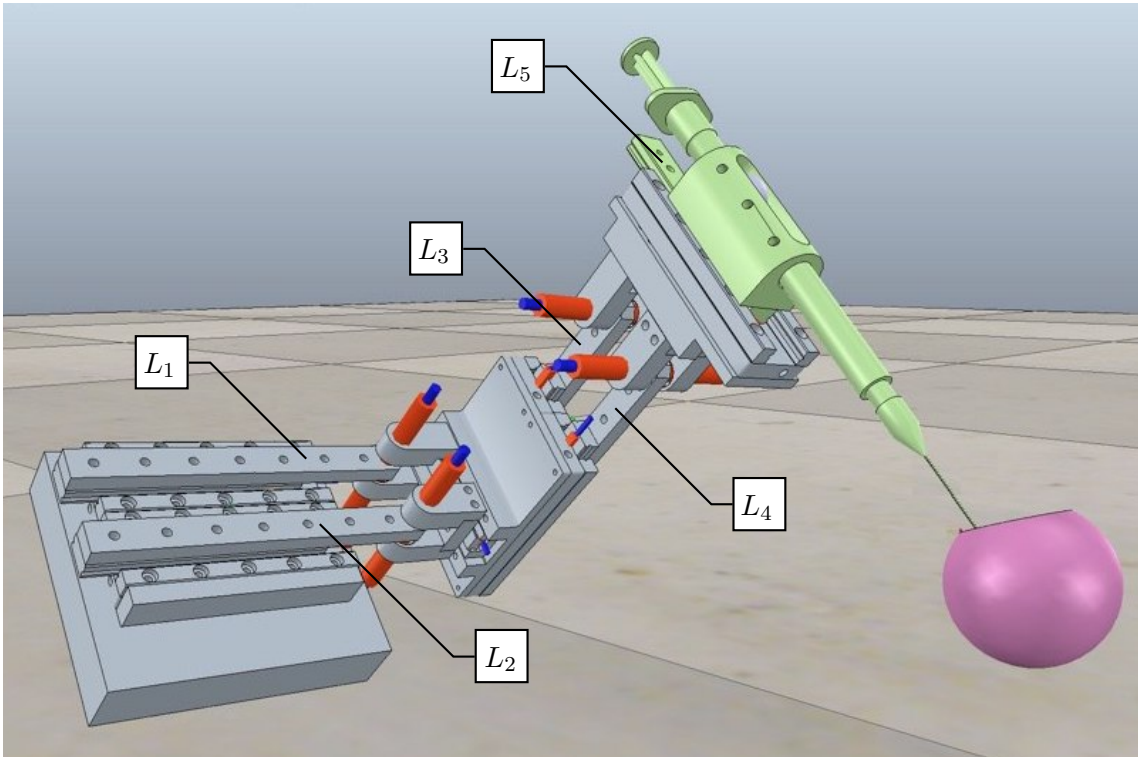


Figure 3.3: The surgical robot modelled in V-Rep

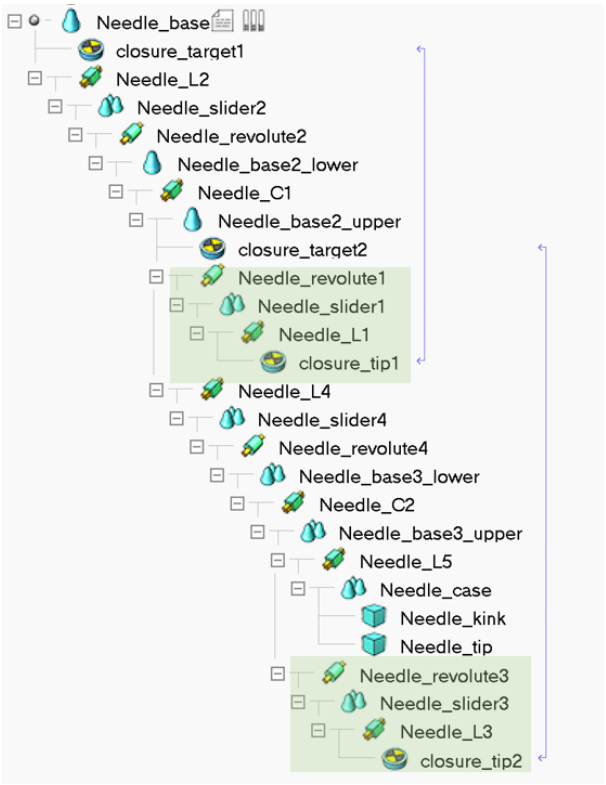


Figure 3.4: The scene hierarchy for the finished robot model. The green boxes indicate the sidearms of the kinematic chain that need to be constrained to be connected to their respective base plate via a tip-target pair (blue arrows).

3.5.1 Defining the Inverse Kinematic Task

Figure 3.5 shows the kinematic chain used to define the inverse kinematics task. Since the robot consists of subsequent PCJMs, it cannot be modelled as a linear kinematic chain (every element can only have one parent in the scene hierarchy). Instead, three distinct IK groups that are defined. The first IK group governs the position of both the tool tip and the RCM point by finding a joint configuration that minimizes the distance to their respective targets. The tip and RCM position need to be included in the same IK group since they are on the same rigid link and in a way represent conflicting goals for the needle pose between which a compromise needs to be found. The other two IK groups are used to 'close the loop', that is to connect the side chains to their respective bottom plates. The two loop-closure IK tasks are added solely for cosmetic reasons, since even without them the position of the second slider in each PCJM could be uniquely reconstructed from the position of the first slider and the angle of the subsequent revolute joint.

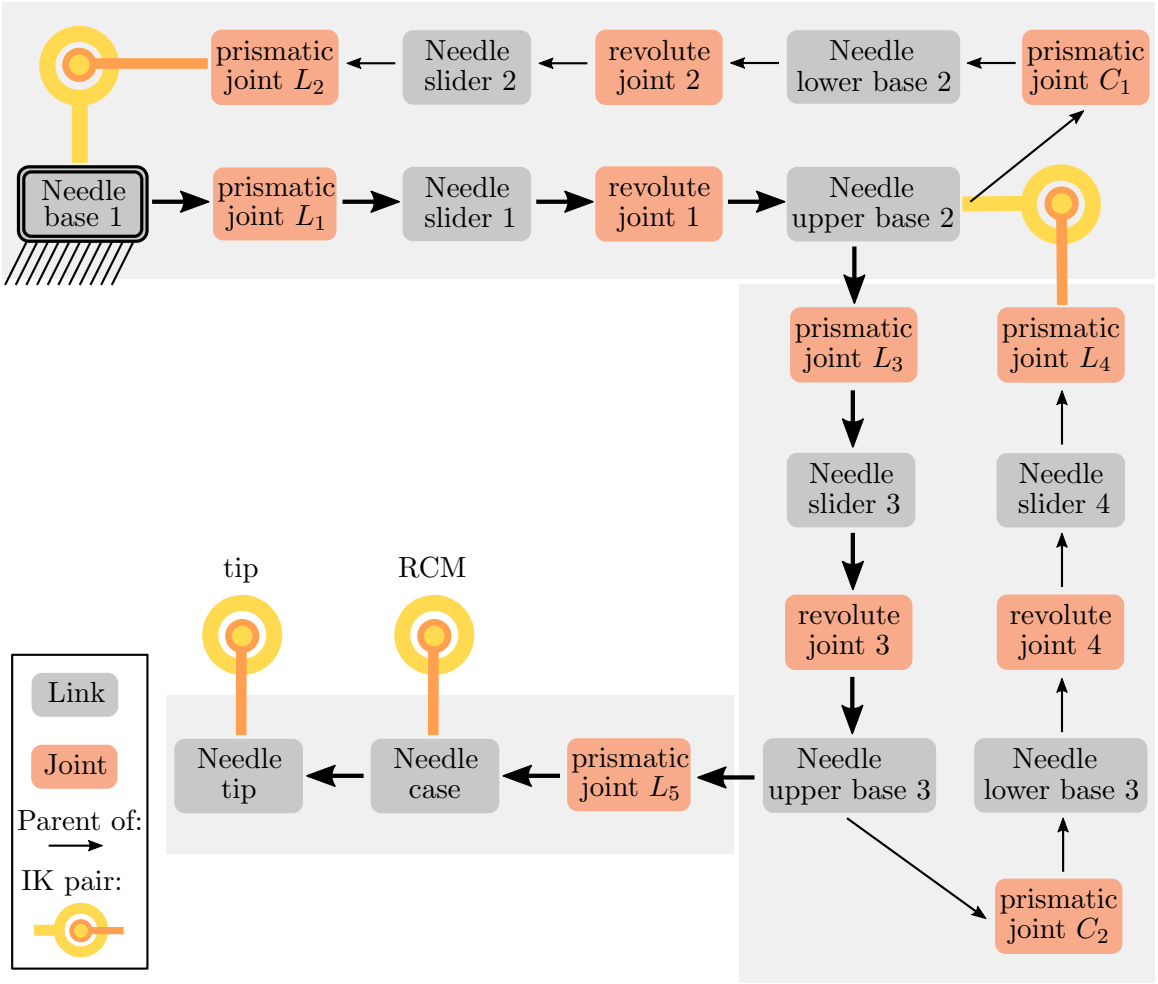


Figure 3.5: Schematic overview of the kinematic chain in V-REP. Objects connected by a bold black arrow indicate the primary kinematic chain which corresponds to the simplified serial robot in 2.1. Objects within a grey box belong to the same stage of the robot, i.e. one of the PCJMs or the final prismatic stage. A total of 4 IK pairs illustrated by yellow rings constrain (1) the position of the needle tip (2) the position of the RCM and (3,4) the position of the second slider within each PCJM

4 Results

In this chapter the accuracy of the RCM point when performing injections at different sites on the inner surface of the eye will be evaluated using the method described in 3.4. Additionally, the choice of control parameters for the augmented Jacobian method will be discussed.

4.1 Accuracy of the Jacobian Pseudoinverse Method

4.1.1 Choosing the Matrix K

When controlling the motion of the tool with the augmented Jacobian matrix, two parameters should be optimized: the control matrix $\mathbf{K}$ and the step size of the robot s which determines the magnitude of the change in joint positions for each iteration. Recalling Eqn. 3.17, we can see that each entry in $\mathbf{K}$ gives a weight to the error of one component in the pose, i.e. if one entry in $\mathbf{K}$ is large, the error in that component will converge to zero faster. On the other hand, two of the components in the pose vector are angles in radians, while the others are cartesian positions given in meters. The Jacobian matrix does not 'know' about these different length scales, thus the matrix $\mathbf{K}$ needs to correct for it. Additionally, the two angles have different levers with respect to the needle tip position, i.e. a change of 0.1 radians for both angles will result in a different displacement of the needle tip in y and z direction. As a rough correction for this, we can calculate the approximate change in the position of the tip per angle of the robot is in its initial configuration ($q_i = 0$). For a small change of angle in q_1 and q_3, the change of the z- and y-position of the needle tip is given by

$$\Delta z = \sin \Delta q_1 \sqrt{l_2^2 + l_6^2}$$
$$\Delta y = \sin \Delta q_1 \sqrt{l_5^2 + l_6^2}$$

This gives us an approximate correction factor of 10 for the entries of $\mathbf{K}$ corresponding to the angular errors. Thus, we start optimization with $\mathbf{K}$=diag([1, 10, 10, 1, 1, 1]).

4.1.2 Error in the RCM Position

In Section 3.3 a control loop was introduced which iterates until the robot's current pose matches a given target pose within a predefined magnitude. Large distances between start and target pose require a high number of iterations to converge because the Jacobian matrix

Algorithm 2 Extended_Control_Loop(**target_pose**, step_size)

```
start_pose ← get_task_pose()
total_distance ← target_pose − start_pose
step ← total_distance/abs(total_distance) · step_size
while max(abs(target_pose − current_pose)) > 0.0003 do
    intermediate_target_position ← current_pose + step
    Control_Loop(K, intermediate_target_position, J_a, λ)
    current_pose ← get_task_pose()
end while
```

attempts to correct the parameter with the largest error in one iteration, thereby displacing the other parameters heavily. These then return to their desired values in subsequent iterations as illustrated in Figure 4.1. This behaviour can be efficiently suppressed by splitting the full trajectory into several smaller parts. If the distance between these intermediate target positions is sufficiently small the control loop will converge from one position to the next within a small number of iterations without excessive overshooting.
The process of splitting the trajectory is illustrated in Algorithm 2. given a target pose that is far away from the current pose, the extended control loop splits the trajectory in many small steps of equal length, determined by the step size. For each small step, the control loop presented in Section 3.3 is executed, which brings the needle from the current position

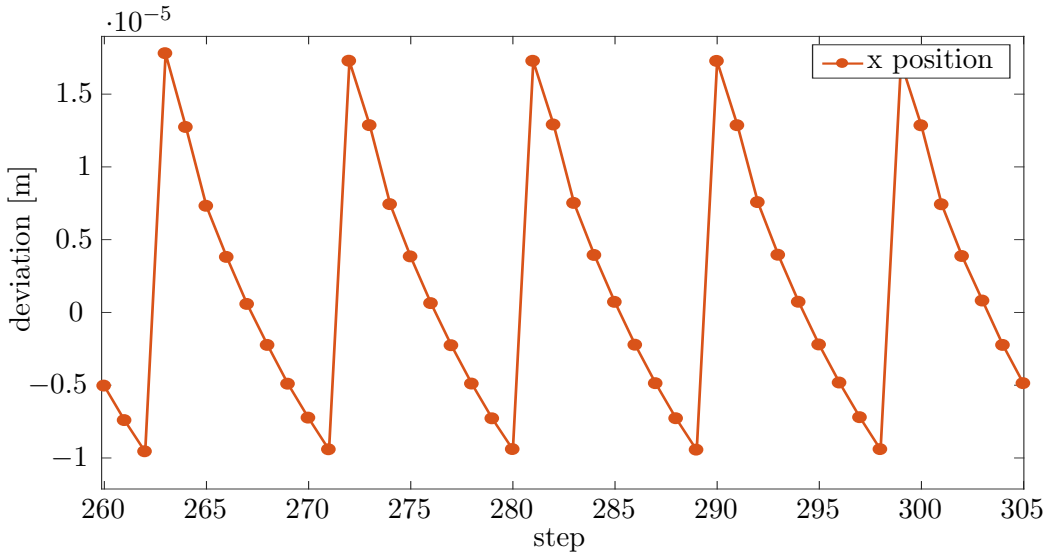


Figure 4.1: Error in the RCM position when performing a step that is too large. The step starts with a large displacement in one direction and requires many iterations of the control loop to return to the desired RCM position.

sufficiently close to the intermediate target. After the intermediate target is reached, a new small step in the direction of the target pose is calculated and again executed by the first control loop presented in Section 3.3.
As an example for a proper choice in step size, two parts of the injection process are shown: The alignment of the needle with the trocar (at point 2 in Section 3.4), in which the RCM point is positioned at the needle tip ($\lambda = 1$) and the entire robot pivots around its tip, as well as the movement from point 3 to point 7. These two were chosen because one illustrates the 'worst case' in terms of keeping the RCM point steady (as a small angle change leads to a large displacement of the RCM due to the long lever) and the other requires simultaneous adjustment of both angles of the needle while maintaining the RCM point.
Figure 4.2 shows the overall position of the RCM point during the alignment process. 500 steps according to the control loop in Section 3.3 were performed to correct the angles Φ and Ψ by 0.0048 rad ($\approx$ 0.28 deg) and 0.212 rad ($\approx$ 12.12 deg), respectively. In each of these steps, roughly 3 iterations were needed to ensure the accuracy of the RCM, resulting in a total of 1350 data points. The total computation time for this section of the injection

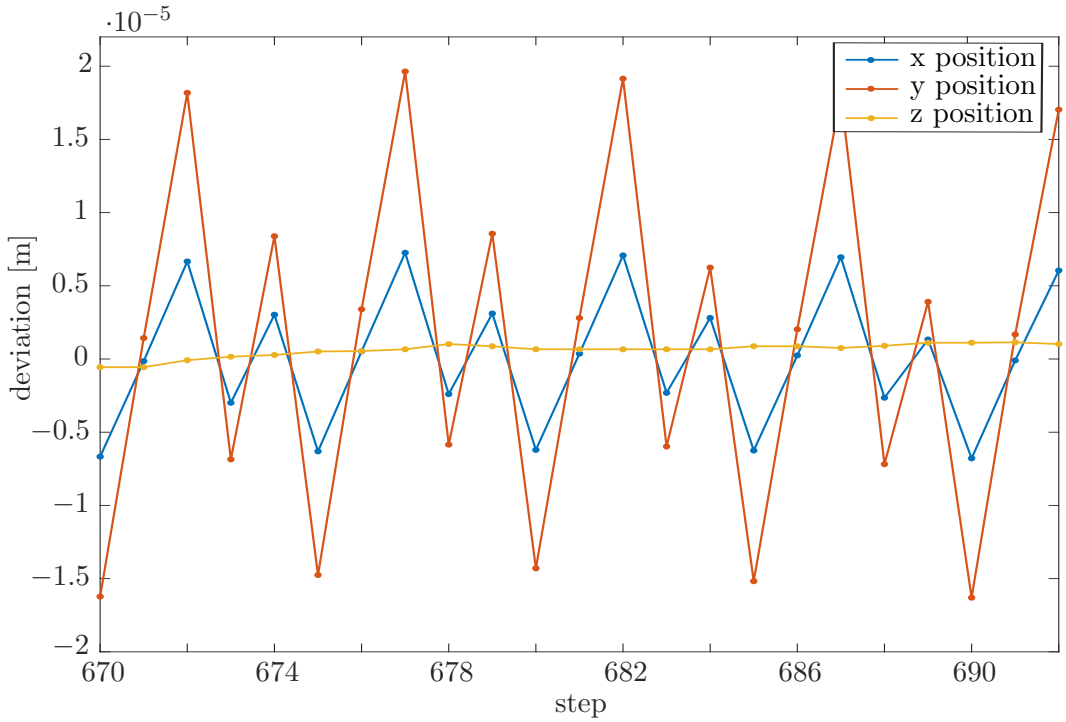


Figure 4.2: Error in the RCM position during 22 iterations of pivoting the needle around its tip during the alignment process described in Section 3.4.

process was about 3 minutes, which is quite a lot but was required to reach a high precision for the RCM point. During many iterations of this injection step, $K = diag([11012122])$ proved to be a good choice for the control matrix as a compromise between speed and accuracy.
The first thing to note about Figure 4.2 is that both y_{rcm} and z_{rcm} remain roughly constant during the entire procedure while x_{rcm} experiences some drift. This drift occurs only in x-direction because the weight in $\boldsymbol{K}$ associated with x_{rcm} is only half as large as the weights associated with y_{rcm} and z_{rcm}. When adjusting the values in $\boldsymbol{K}$, it is always necessary to find a good trade-off between accuracy of the RCM position and computation speed. By setting the weight related to x_{rcm} to 2 instead of 1 the drift is reduced but the needle pose converges to the target pose much slower because the change in x-position competes with the change in both Ψ and Φ. In this example, the drift of x_{rcm} during the alignment process is only $\pm 3 \cdot 10^{-5} m$ which is negligible compared to the radius of the trocar which is about $45 \cdot 10^{-5} m$.

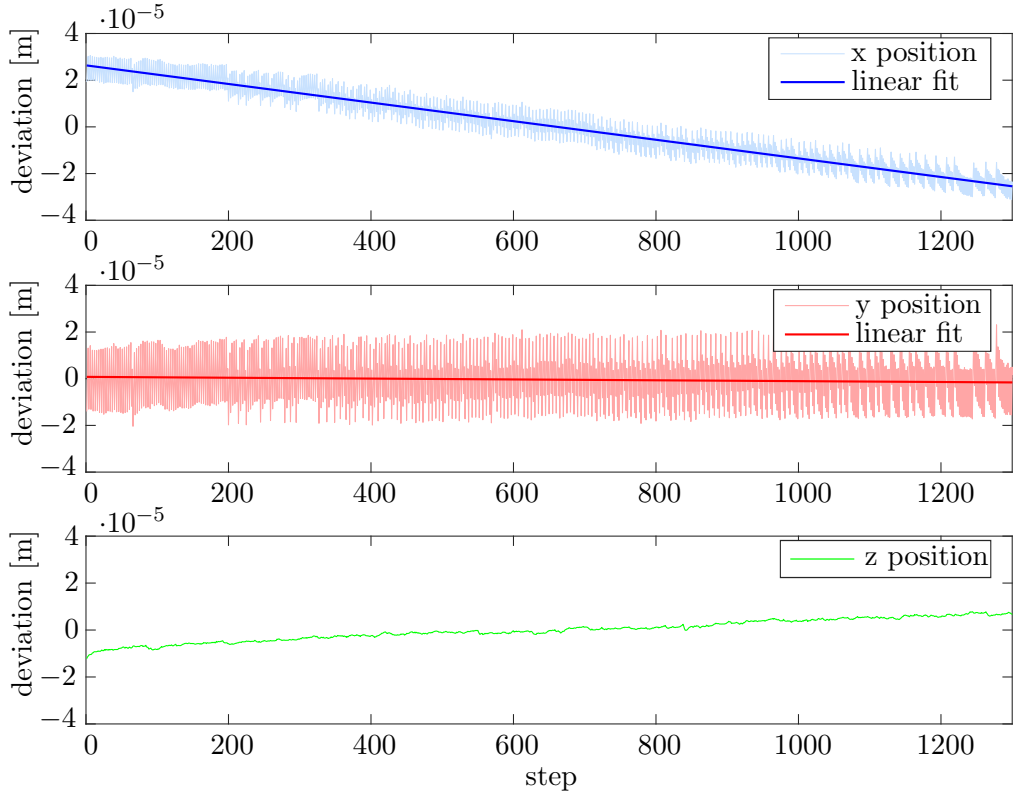


Figure 4.3: Overall error in the RCM position for the alignment of the needle with the trocar as proposed in Section 3.4.

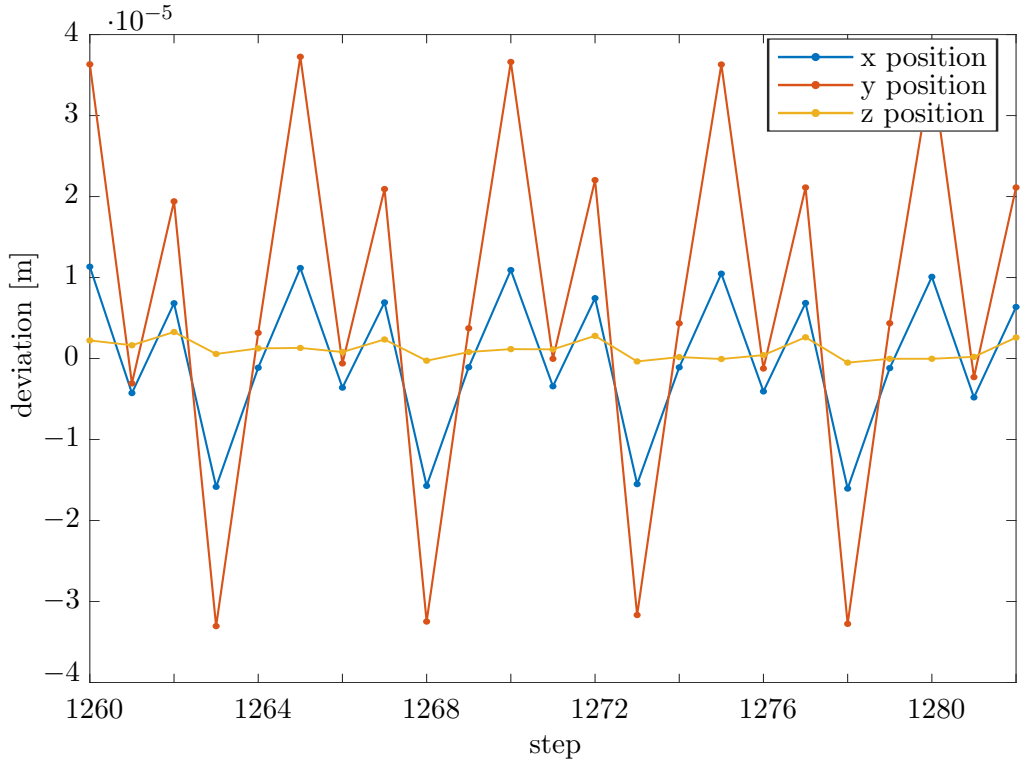


Figure 4.4: Error in the RCM position during 22 iterations of the movement from point 3 to point 7 as proposed in Section 3.4.

Both Figure 4.2 and 4.3 illustrate the influence of the step size on the accuracy of the RCM point. A total of 500 steps are performed to align the needle with the trocar, but one of the initial angles is much closer to the target angle than the other. As a result one angle will have a larger error each step and the Jacobian will calculate a larger ΔL to correct the angle, thus displacing the RCM more in one direction than the other. In this example the error in angle Ψ is large, thus each step the needle will rotate about the z-axis by a large amount and as a result displace the RCM point in y direction. the error in angle Φ is very small and thus the RCM is hardly displaced in z-position. The x-position, which roughly corresponds to the depth the needle is injected into the eye, is affected by both angle changes, therefore its error is somewhere in between.
Figures 4.4 and 4.5 show the error per step and the overall drift and for the RCM point during the movement from point 3 to point 7 proposed in Section 3.4. Both are very similar in precision to the ones for the alignment with the trocar which shows that simultaneous adjustment of two angles does not significantly decrease performance. The total computation time for this section of the injection was roughly 2 minutes.

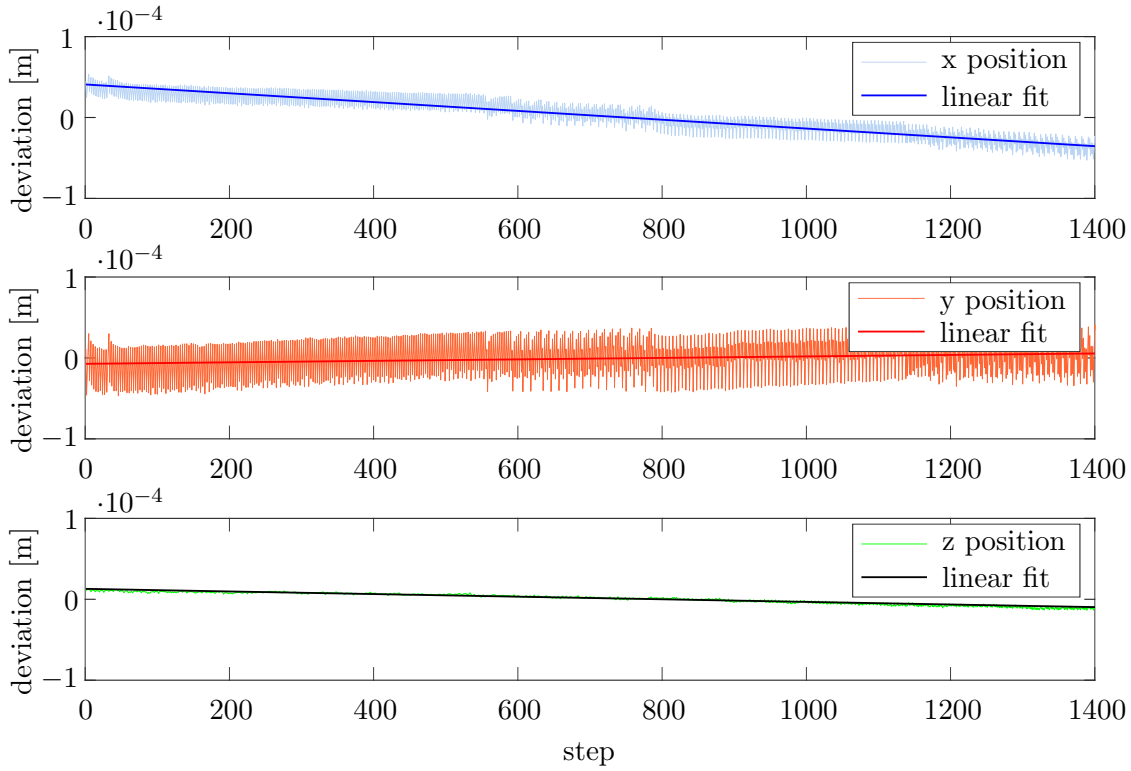


Figure 4.5: Overall error in the RCM position for the movement from point 3 to point 7 as proposed in Section 3.4.

4.2 Accuracy of the Dampened Least Squares Method

This section discusses the accuracy of V-REPs inverse kinematic solver which utilizes the Levenberg-Marquardt algorithm discussed in Section 2.4.2. The DLS algorithm functions very similarly to the Jacobian pseudoinverse method in that it linearises the IK problem near the current pose via the Jacobian matrix. There are two key points in which the DLS algorithm improves upon the Jacobian pseudoinverse approach: Firstly, the DLS algorithm introduces the damping parameter D which prevents the algorithm from running into numeric instability close to a singularity. Secondly, the DLS algorithm never performs a matrix inversion, improving its computation time especially for Manipulators with a large number of links. For the robot considered in this thesis, the DLS algorithm should not have a significantly higher performance than the pseudoinverse method, given an optimal choice for the matrix $\boldsymbol{K}$ and a sufficiently small step size, because there are no singularities in the workspace of the robot. Thus, to give an idea about the precision that is possible

to achieve with the augmented Jacobian method the precision of the built-in IK solver is evaluated. Figure 4.6 shows the setup used for showcasing the precision of the Dampened Least Squares Method. An off-axis spiral is drawn with the needle tip, keeping the RCM point steady at all times. Figure 4.7 shows the deviation of the RCM point. The damping factor D is set to 0.1.

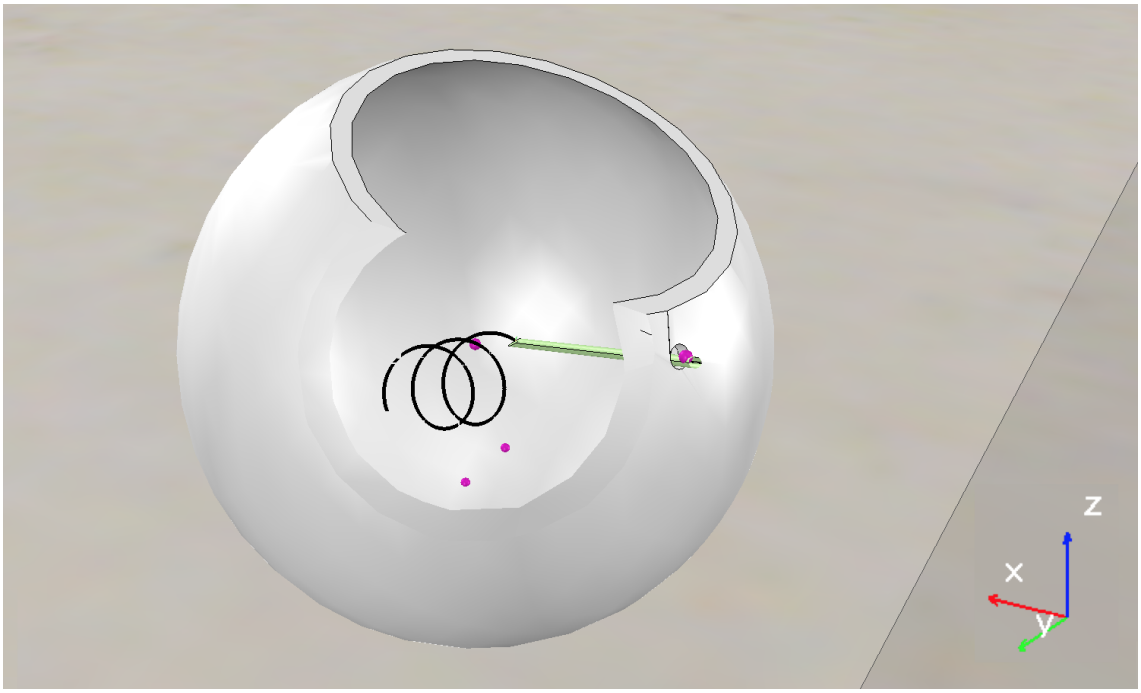


Figure 4.6: Needle tip of the Robot tracing the shape of a spiral within the eye while keeping the RCM point steady, using V-REPs inverse kinematics module. The trajectory was recorded by V-REP using a graph object positioned at the needle tip.

The accuracy of the DLS method proved to be extremely high compared to the Jacobian pseudoinverse method. Furthermore, the computation time to draw the shape in Figure 4.6 was only 28 seconds. although the minimum precision was set to 0.1mm, the method seemed to be much more precise, implying that a smaller damping factor could have been chosen to speed up computation even more.

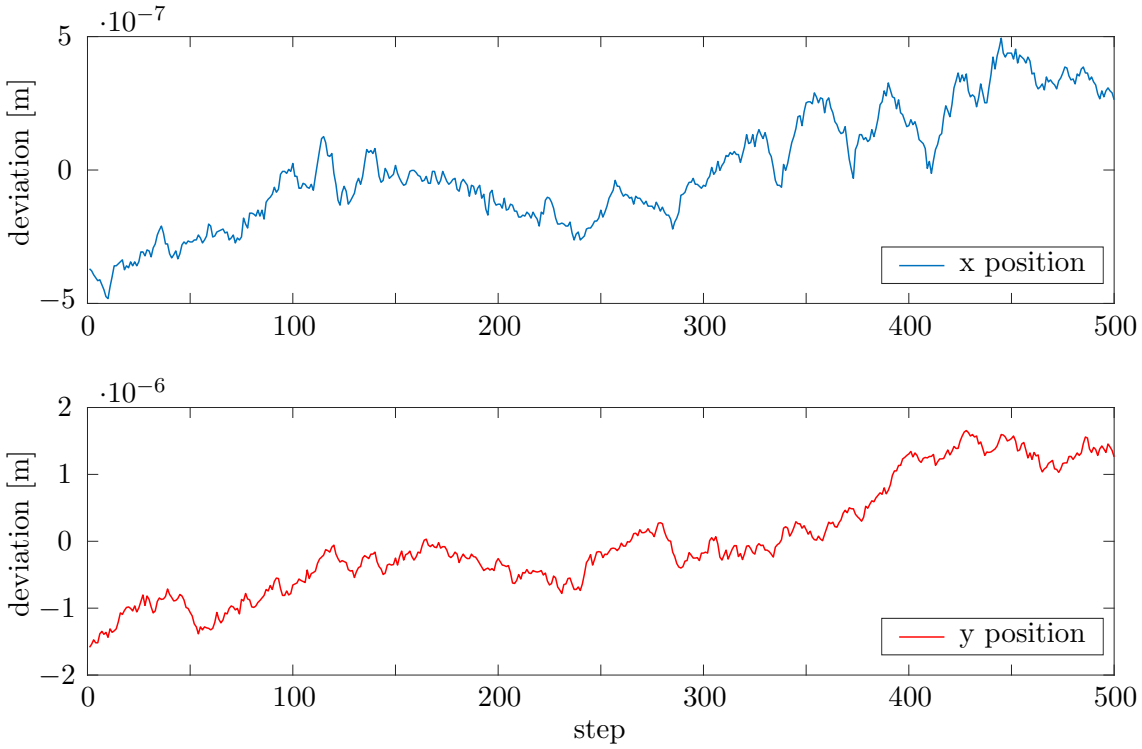


Figure 4.7: Error in the RCM position during the first 500 calculation steps of the V-REP IK solver, producing the trajectory seen in Figure 4.6.

5 Conclusion

The results in the previous section show that control of the robot via the augmented Jacobian matrix is a valid approach to solving the inverse kinematics problem. When performing sufficiently small steps the deviation of the RCM from its ideal position was small enough as to not put stress on the tissue surrounding the incision site. Even though the method proved to be precise enough in simulation, the precision for the real robot would most likely be lower since no dynamic effects were considered. To improve the overall accuracy, there are several points that can be relatively easily implemented building on the results presented but exceed the scope of this thesis. These include:

Preventing drift of the RCM position: As discussed in Section 4.1, the x-position of the RCM point experiences some drift during the course of the injection. While the drift is relatively small compared to the size of the trocar, a proper solution might significantly increase the performance. Within each iteration step, the RCM is forced to the target position $(x_{rcm}, y_{rcm}, z_{rcm})$, allowing a small deviation from the target position as a trade-off between drift rate and computation time. By adjusting the corresponding entry in the matrix K, the error of x_{rcm} is given a higher priority and will drift less, but all other position parameters will converge slower leading to a worse overall performance. This problem might be easily resolved by readjusting the position of the RCM on the needle in each iteration step before enforcing the normal control loop. This means that before each iteration the point on the needle closest to $(x_{rcm}, y_{rcm}, z_{rcm})$ is calculated and declared as the new RCM point.

Improving the control matrix K: The control matrix K used in this thesis was determined empirically and proved to be a good trade-off between accuracy and convergence speed. However, it is certainly not optimal for every kind of movement within the workspace of the robot inside the eye, as it was adjusted using only the injection procedure presented in Section 3.4. A better K can be determined by systematically computing accuracy and convergence during movements within the whole workspace inside the eye for different combinations of entries in K.

Inclusion of a third rotation into the Jacobian matrix and simulation: The model of the surgical robot considered in this thesis uses a straight needle as a tool, thus the rotation around the tool axis has no effect on the position of the needle tip and can be neglected. Having established the kinematic relationships in the robot model and a strategy to calculate

both the unconstrained and augmented Jacobian matrix, a needle with a curved tip can be implemented. A curved needle allows for injections lateral to the surface and extends the reachable surface area for injection inside the eye, but makes it more difficult to enter the eye through the trocar while maintaining the RCM point. To implement this, the Jacobian matrix needs to be extended by one row expressing the rotation of the tool around its own axis and the augmented Jacobian matrix must be updated accordingly.

Inclusion of dynamic effects into the Simulation: The V-REP simulation environment is capable of performing precise dynamic calculations. It might be interesting to estimate the loss in precision of both inverse kinematics methods when dynamic effects are considered.

6 Appendix

$$
J_a = \begin{bmatrix}
0 & l_2c1 + l_6c3s1 + l_5s1s3 + q_4c3s1 & 0 & c1(l_6s3 - l_5c3 + q_4s3) & -c1c3 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c1 & 0 & 0 \\
0 & l_2c1 + l_5s1s3 + q_4c3s1 + l_6\lambda c3s1 & 0 & c1(q_4s3 - l_5c3 + l_6\lambda s3) & -c1c3 & -l_6c1c3 \\
0 & 0 & 1 & -l_5s3 - q_4c3 - l_6\lambda c3 & -s3 & -l_6s3 \\
1 & l_5c1s3 - l_2s1 + q_4c1c3 + l_6\lambda c1c3 & 0 & -s1(q_4s3 - l_5c3 + l_6\lambda s3) & c3s1 & l_6c3s1
\end{bmatrix} \tag{6.1}
$$

with $ci, si := \cos(i), \sin(i)$.

Bibliography

[1] Coppelia robotics. `http://www.coppeliarobotics.com/`. Accessed: 2014-11-13.

[2] iRAM!S project. `http://www6.in.tum.de/Main/ResearchiRAM!S`. Accessed: 2014-11-13.

[3] V-rep remote api functions (matlab). `http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm/`. Accessed: 2017-11-14.

[4] Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[5] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

[6] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005.

[7] Stefano Chiaverini. Estimate of the two smallest singular values of the jacobian matrix: Application to damped least-squares inverse kinematics. *Journal of Field Robotics*, 10(8):991–1008, 1993.

[8] Arati S Deo and Ian D Walker. Adaptive non-linear least squares for inverse kinematics. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 186–193. IEEE, 1993.

[9] Peter Gschirr. Control and simulation of a robotic setup for assisting ophthalmic surgery. Master of science thesis, Technische Universität München, January 2014.

[10] Gary S Guthart and J Kenneth Salisbury. The intuitive/sup tm/telesurgery system: overview and application. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 618–621. IEEE, 2000.

[11] Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics*, 77(2):215–221, 1955.

[12] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[13] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[14] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.3 (R2017b)*, 2017.

[15] Ming Li and R.H. Taylor. Spatial motion constraints in medical robot using virtual fixtures generated by anatomy. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 1270–1275 Vol.2. IEEE, 2004.

[16] M Ali Nasseri, M Eder, D Eberts, S Nair, M Maier, D Zapp, CP Lohmann, and A Knoll. Kinematics and dynamics analysis of a hybrid parallel-serial micromanipulator designed for biomedical applications. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 293–299. IEEE, 2013.

[17] M Ali Nasseri, M Eder, S Nair, EC Dean, M Maier, D Zapp, CP Lohmann, and A Knoll. The introduction of a new robot for assistance in ophthalmic surgery. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 5682–5685. IEEE, 2013.

[18] MA Nasseri, P Gschirr, M Eder, S Nair, K Kobuch, M Maier, D Zapp, C Lohmann, and A Knoll. Virtual fixture control of a hybrid parallel-serial robot for assisting ophthalmic surgery: An experimental study. In *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*, pages 732–738. IEEE, 2014.

[19] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE, 2013.

[20] Homayoun Seraji. Configuration control of redundant manipulators: Theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–490, 1989.

[21] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[22] Takashi Ueta, Yoshiharu Yamaguchi, Yoshihiro Shirakawa, Taiga Nakano, Ryuichi Ideta, Yasuo Noda, Akio Morita, Ryo Mochizuki, Naohiko Sugita, Mamoru Mitsuishi, and Yasuhiro Tamaki. Robot-Assisted Vitreoretinal Surgery. *Ophthalmology*, 116(8):1538–1543.e2, nov 2017.

[23] CW Wampler and LJ Leifer. Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 110(1):31–38, 1988.