On task specification and inverse kinematics for a redundant surgical robot for bone removal

Talha Ali Arslan

D&C 2016.004

Thesis

submitted in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

in

Department of Mechanical Engineering
Eindhoven University of Technology

Coach:	engr. Jordan Bos
Supervisors:	dr. Alessandro Saccon prof.dr. Henk Nijmeijer
Committee:	prof.dr. Henk Nijmeijer dr. Alessandro Saccon engr. Jordan Bos prof.dr. Herman P.J. Bruyninckx

January, 2016

Preface

This thesis is the outcome of my MSc study in Mechanical Engineering within the Dynamics&Control Research Group in Eindhoven University of Technology (TU/e) in Eindhoven, The Netherlands. The project started on March, 2015 and it was carried out in the Robotics Lab.

I would like to thank my coach Jordan Bos, for all the discussions we had during our meetings and through e-mails, for his suggestions and feedbacks which made me understand and learn things faster. I would like to thank Dr. Saccon, for dealing with me since the times before my internship last year which he also supervised, for always being supportive, positive and communicative. I would like to thank prof. Nijmeijer for his highly efficient supervision since the beginning of my study, for whenever I needed an answer, a connection or a project, he provided a good one.

And finally, I would like to thank my parents, for their unconditional love and support, which made these two and a half years invaluable for me, in other ways too.

Talha Ali Arslan
January, 2016

Abstract

In this thesis, solutions to the problems of task specification and inverse kinematics are developed for the surgical robot RoBoSculpt, which is designed to assist surgeons and perform bone removal procedures around the human ear and the skull base. The robot has eight degrees of freedom in total, which makes it inherently redundant. Its non-standard kinematic model includes six revolute joints, a prismatic joint and a rotary tool for milling and drilling. The initial seven degrees of freedom, excluding the rotation of the rotary tool, are utilized and task augmentation is used to resolve the redundancy. An additional constraint task is defined specifically for this robot, along with end-effector position and orientation tasks. With the proposed task specification method, task trajectories are generated such that a bone removal procedure can be performed inside an approximation of an ear bone cavity, without any collisions between the end-effector, its supporting link and the ear cavity. The inverse kinematics problem is solved by inverting the first-order differential kinematics with augmented task space. A standard differential inverse kinematics algorithm, which is computationally efficient and free of representational singularities, is adopted and modified for augmented task space approach. It can be used for both pre-defined tasks and real-time operations. Additionally, the end-effector manipulability is analysed. With the analysis, the capacity of the end-effector motion and the performance of the kinematic tracking can be assessed in various configurations of the robot, such as throughout a given task or inside a specified region of the robot workspace. Moreover, decisions on the workspace placement of the patient with respect to the robot can be made, to ensure that the robot is operated in configurations where high end-effector manipulability is maintained. Finally, since the robot is currently at its design phase, the software tools have been extended to allow making design modifications in the kinematic model and performing forward and inverse kinematics simulations with visualization.

Contents

Preface **i**

Abstract **iii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Contributions . . . 4
1.4 Outline . . . 5

2 Literature Review **7**
2.1 Surgical Robots . . . 7
2.1.1 Bone Removal Robots . . . 8
2.2 Inverse Kinematics . . . 8
2.2.1 Overview . . . 9
2.2.2 Closed-form Solutions . . . 9
2.2.3 Numerical Solutions . . . 10
2.2.4 Differential Inverse Kinematics . . . 10
2.2.5 Redundant Serial Manipulator Kinematics . . . 12
2.3 Singularity Problem . . . 18
2.3.1 Singular Value Decomposition of a Jacobian matrix . . . 18
2.3.2 Manipulability Measures . . . 18
2.4 Discussion . . . 19

3 Robot Description **21**
3.1 Robot Description . . . 21
3.1.1 Kinematic Model . . . 21

4 Task Specification Solution and Results **25**
4.1 Task Specification . . . 25
4.1.1 Ear Volume Approximation . . . 25
4.1.2 End-effector Position Trajectory . . . 27
4.1.3 End-effector Orientation Trajectory . . . 28
4.1.4 Constrained Use of Prismatic Joint for Spherical Clearance . . . 38
4.2 Results . . . 40
4.3 Discussion . . . 43

5 Inverse Kinematics Solution and Results **45**
5.1 Differential Inverse Kinematics Algorithm with Augmented Task Space . 45
5.1.1 Overview . . . 45
5.1.2 First-order Differential Kinematics . . . 46
5.1.3 End-effector Position Error . . . 47
5.1.4 End-effector Orientation Error . . . 48
5.1.5 Desired End-effector Angular Velocity . . . 49
5.1.6 Additional Constraint Task Error . . . 49
5.1.7 Overall Task-Space Error . . . 50
5.1.8 Augmented Jacobian and the IK Algorithm . . . 50
5.2 Results . . . 52
5.3 Discussion . . . 56

6 Manipulability Analysis and Software Implementation Details **59**
6.1 Manipulability Analysis . . . 59
6.1.1 Results . . . 62
6.1.2 Discussion . . . 65
6.2 Software Implementation and Toolbox . . . 66
6.2.1 Toolbox Setup . . . 67
6.2.2 Simulation . . . 67

7 Conclusion and Recommendations **69**
7.1 Conclusion . . . 69
7.2 Recommendations . . . 70

A End-effector Orientation Error **77**

B Forward Kinematics **81**
B.1 Rigid Body Rotations & Translations . . . 81
B.1.1 Position and Orientation Representation . . . 81
B.1.2 Homogeneous Transformations . . . 83
B.2 Denavit-Hartenberg Parametrization . . . 84
B.3 Forward Kinematics . . . 84

C Additional Figures **87**
C.1 Intra-collisions . . . 87

D Homogeneous Transformation and Jacobian Matrices **89**
D.1 Homogeneous Transformation Matrix of the End-effector Frame of RoBoSculpt . . . 89
D.2 Geometric Jacobian Matrix of the End-effector Frame of RoBoSculpt . 90

List of Symbols

Symbol	Description	Unit
q	Vector of joint displacements	
x	Vector of task variables	
θ, ϕ, α	Angular parameters	[rad]
$o_0x_0y_0z_0$	Inertial reference coordinate frame	
$o_dx_dy_dz_d$	Desired end-effector coordinate frame	
$o_i^j x_i^j y_i^j z_i^j$	Coordinates of frame $o_ix_iy_iz_i$ expressed in frame $o_jx_jy_jz_j$	
R_i^j	Rotation matrix with unit vector coordinates of frame i with respect to frame j	
R_d^0	Desired orientation of the end effector frame w.r.t. reference frame	
p_d^0	Desired position of the end effector w.r.t. reference frame	
T_i^j	Homogeneous transformation matrix from j^{th} to i^{th} frame	
A_i	Homogeneous transformation matrix from $i-1^{th}$ to i^{th} frame	
R_i^{i-1}	Rotational transformation matrix from $i-1^{th}$ to i^{th} frame	
o_i^{i-1}	Origin of i^{th} frame w.r.t. $i-1^{th}$ frame	
$v_{i,j}^k$	Translational velocity of frame i with respect to frame j, represented in the coordinates of frame k	
$\omega_{i,j}^k$	Angular velocity of frame i with respect to frame j, represented in the coordinates of frame k	
J, J_A, J_{aug}	Geometric, analytical, augmented Jacobian matrices	
r_s	Radius of the ear section plane	[m]
r_b	Radius of the cone bottom (cavity opening)	[m]
d_{cl}	The clearance of the tool (final link) from the cone edge	[m]
k_{cl}	Length of the milling tool inside the ear	[m]
s_{cl}	Clearance between the origins of the ear section and the 6^{th} frame	[m]
$\bar{h}_s$	Distance between cone base and cone section	[m]
c_{cl}	Increment for the spherical clearance	[m]

Notation	Description
$\square_i$	i^{th} term of vector $\square$
$\square_i^j$	$\square$ belonging to i^{th} frame expressed with respect to j^{th} frame
$\dot{\square}, \ddot{\square}$	First and second order derivatives of $\square$ with respect to time
$\square^T, \square^+$	Transpose and Moore-Penrose pseudoinverse of $\square$

Abbreviations	Description
CT	Computed tomography

2D, 3D	Two dimensional, three dimensional
DOF(s)	Degree(s) of freedom
CAD	Computer-aided design
DH	Denavit-Hartenberg
R,P	Revolute joint, prismatic joint

List of Figures

1.1 A visual model of the surgery environment with the robot and the patient with both ear volumes depicted as irregular cone-like shapes (Source: Jordan Bos, 2015) . . . 2
1.2 Approximate dimensions (in mm) and shape of the ear volume with some of the important ear structures (black) (Source: Jordan Bos, 2015) . . . 3

3.1 Robot CAD model viewed from side (Source: Jordan Bos, 2015) 21
3.2 Robot schematic with the DH parameters and frames 22

4.1 Approximate ear cone with different orientations. In part (a), the cone section frame $o_s x_s y_s z_s$ is aligned with the reference frame $o_0 x_0 y_0 z_0$ by rotation R_{pre}. In part (b) and part (c), following the rotation of R_{pre}, the cone section frame is rotated by $\pi/6$ about its current x-axis and y-axis, respectively. . . . 26
4.2 Illustration of the surgical robot and the approximate ear cone 26
4.3 Illustration of a position trajectory spanning the cone section. On the left, a trajectory is shown with fine spacing between each point and a small number of layers, whereas on the right, one with coarse spacing and a large number of layers is shown. . . . 27
4.4 A continuously differentiable position trajectory spanning the cone section 28
4.5 Clock-wise spiral continuous trajectory 28
4.6 Representations of the cone section frame $o_s x_s y_s z_s$ and the intermediate frame $o_{d'''} x_{d'''} y_{d'''} z_{d'''}$ in (a) and the end-effector frame $o_7 x_7 y_7 z_7$ (in red, green, blue) with RoBoSculpt and a virtual cone (red-green) as reference for orientation in (b) . . . 29
4.7 Representation of rotations - Plot (a) shows the pre-rotation by R_{pre} around x_s to obtain the intermediate frame $o_{d'''} x_{d'''} y_{d'''} z_{d'''}$ while Plot (b) shows the initial rotation by the *approach angle* θ around $z_{d'''}$ to obtain $o_{d''} x_{d''} y_{d''} z_{d''}$ . . . 30
4.8 Representation of the *approach angle* θ 31
4.9 Representation of rotations - Plot (a) shows the second rotation by the *inclination angle* ϕ around $x_{d''}$ to obtain $o_{d'} x_{d'} y_{d'} z_{d'}$, while Plot (b) shows the third rotation by the *self-rotation angle* α around $z_{d'}$ to obtain the desired end-effector frame $o_d x_d y_d z_d$ . . . 31
4.10 Representation of the *inclination angle* ϕ 32
4.11 Representation of the *self-rotation angle* α, for a desired pose of the end effector frame . . . 32
4.12 Illustration of cone section and tool axis projection onto it, where σ_x and σ_y are functions of the desired end-effector position 33

4.13 Illustrations of the choice of the approach angle θ, when θ is set to be 0 (a) and when it is chosen as in (4.5) (b), viewed from the top of the cone section 34
4.14 Illustrations of the choice of the inclination angle ϕ, when ϕ is set to be zero in (a), constant in (b) and when it is chosen according to (4.7) in (c) 35
4.15 Cone edge clearance - Upper part of the conic cylinder is the cone opening, through which the tool (in blue) enters to reach cone section surface at the bottom 36
4.16 Description of the spherical clearance volume (blue dome) around the ear volume section in (a) and prismatic joint's actuated part (blue) and base (red), which is not allowed inside the spherical volume, in (b) 38
4.17 Constant spherical clearance of the 6th frame with its position trace (black) 40
4.18 Robot with ear volume approximation cone 40
4.19 Task 1 - The desired end-effector position and orientation trajectories on cone section plane viewed from front in (a), from side in (b) and from top in (c) 41
4.20 Task 2 - The desired end-effector position and orientation trajectories on cone section plane viewed from front in (a), from side in (b) and from top in (c) 41
4.21 Task 1 - The desired end-effector position trajectory plot (top), the desired end-effector orientation trajectory plot (middle) and the prismatic joint's desired constraint displacement trajectory plot (bottom) vs. time 42
4.22 Task 2 - The desired end-effector position trajectory plot (top), the desired end-effector orientation trajectory plot (middle) and the prismatic joint's desired constraint displacement trajectory plot (bottom) vs. time 43

5.1 Joint displacement, velocity and acceleration profiles from differential inverse kinematics for Task 1 52
5.2 Joint displacement, velocity and acceleration profiles from differential inverse kinematics for Task 2 53
5.3 Task 1 - The plots of tracking errors of end-effector position (top) and orientation (bottom) 54
5.4 Task 2 - The plots of tracking errors of end-effector position (top) and orientation (bottom) 54
5.5 Task 1 - End-effector position (top) and orientation (bottom) tracking errors with no gains (open-loop) 55
5.6 Task 1 - End-effector position (top) and orientation (bottom) tracking errors with increased gains 55

6.1 Manipulability analysis data with respect to μ_1 is represented with a colormap, where the robot is in the center and blue points surrounding the data represent the orientation trace of the end-effector at each sample (3D plot from side-view) 60
6.2 Plot (a) shows the manipulability analysis with respect to μ_1 while Plot (b) shows the analysis with respect to μ_2 (3D plot from side-view) . . . 61
6.3 Horizontal section of the 3D grid of the mean values of manipulability data 61
6.4 Filtered manipulability measure data with respect to end-effector position and orientation range 62

6.5 Manipulability analysis in a small volume - Sampling the Workspace (with end-effector orientation reference (green line), obtained orientation at each point (blue trace)) 63
6.6 Two ear volume placements where the lower-left cone is in the same place as Task 1. The placement on lower left is denoted Test 1 (gray) and the one on upper right is denoted Test 2 (red) (3D plot from side-view) . . . 64
6.7 Manipulability measure μ_1 throughout a task with different placements in the robot workspace 65
6.8 A brief schematic of the software implementation in Matlab 66

B.1 Representation of a rotation between frame $o_1x_1y_1$ and $o_0x_0y_0$ by θ . . . 82

C.1 RoBoSculpt with Links 2, 3 and 5 re-meshed with triangles 87
C.2 A re-meshed robot part with vertex normal vectors 88

Chapter 1

Introduction

1.1 Motivation

The first time a robot was used in a surgical operation was in 1983 in Vancouver. The robot was named Arthrobot and it helped in improving surface conformity and accuracy of orientation of a hip joint replacement operation. Since then, surgical robots have assisted more and more surgeons and helped improving the quality of healthcare. Some important achievements which result from the use and assistance of surgical robots in healthcare can be summarized as:

- Reducing the amount of invasiveness for a surgery, thus reducing the impacts on and the recovery period of the patient,
- Reducing the duration of the operation,
- Allowing the surgeons to achieve high precision for complex movements at different levels with ease, by various technologies such as tremor filtration, motion scaling and force feedback.

Robots are used in various fields of surgery and it is fair to say that more advances in the field of surgical robots are to come. One popular field of surgery for robots is bone removal and reshaping. Since operating on bones requires more power, rigidity and amount of work, specially designed or modified robots take the place of manual hand tools to assist surgeons. Ear surgeries are special cases for the use of robotics for bone removal, since to reach middle and inner ear, a portion of the ear temporal bone, which contains critical and delicate structure, must be removed. These surgeries, which are conventionally done manually by hand tools, require a clear visual and a high level of precision while a high-speed rotary tool is being used. In this respect, use of a robotic bone removal device, which is guided by 3D Computed Tomography (CT) images, has the potential to improve the quality of surgery. Furthermore, it may be possible to reduce the duration, overall costs and the necessary amount of invasiveness of bone removal surgeries, which may minimize the impact on the patient. Bone removal at the ear temporal bone and skull base are generally required for surgical procedures such as tumor removal and implantation. The structure within the temporal bone such as nerves that are linked to facial muscles, hearing, balance and taste, can be preserved if less invasiveness, need for less visual access and high precision is achievable. A surgical robot can use CT images to navigate towards the target under supervision of the surgeon, while evading important structures. Hence, the surgeon does not solely have to rely on

visual access any more, which can increase safety and reduce invasiveness of surgery. The accurate motions of the robot can increase precision of the bone milling task and reduce the operation time, for example in the case of Cochlear implantations where the install area of the implant and paths of the electrodes are carved out in the temporal bone. Such a device can also be used in other areas of the body where bone milling and restructuring are carried out, such as oral and maxillofacial surgery, orthopedics and spinal surgery.

1.2 Problem Statement

A novel lightweight modular robotic arm with eight DOFs (degrees of freedom) has been recently designed at the Department of Mechanical Engineering of the Eindhoven University of Technology (TU/e). The design, which is proposed by Jordan Bos MSc, is the first step towards the realization of a novel precision bone-machining robot to assist surgeons in ear bone removal surgery (see the patent application details [1] for future reference). It consists of an initial serial kinematic chain composed by six revolute joints connecting five modular rotating units to a fixed base, terminated with a prismatic joint, which is followed by a rotary tool at the end of the chain. Its CAD (Computer-aided Design) model in a surgery environment with a patient is shown in Figure 1.1. The rotary tool, which is the robot end-effector, is intended to perform drilling and milling tasks on human ear temporal bone. Bone drilling is a cutting process where the rotating tool tip is moved into the bone along the rotary tool axis to cut or enlarge a hole. Bone milling is a machining process where the bone is fed to the rotating tool tip at an angle with respect to the tool's rotation axis (generally at a right angle) to remove bone along the route of the tool tip.

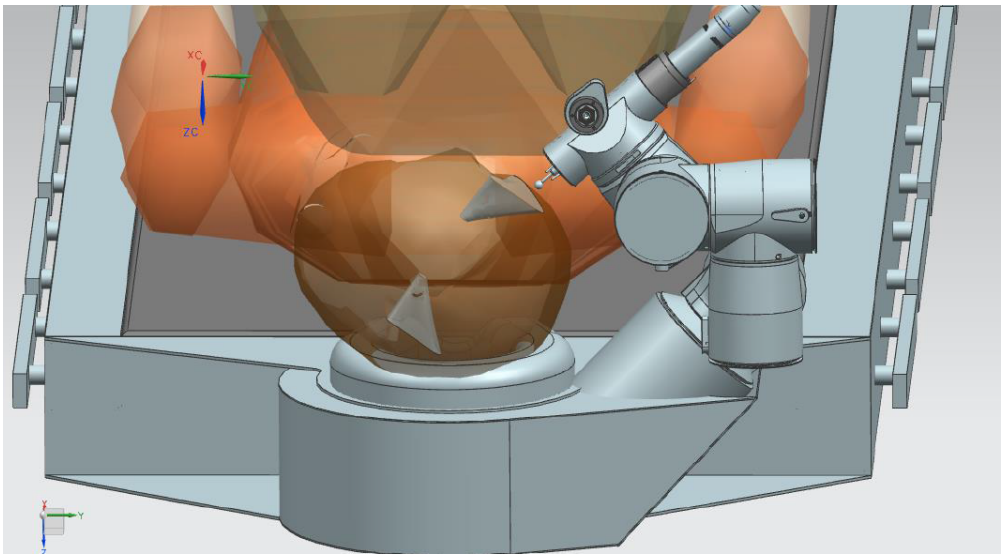

Figure 1.1: A visual model of the surgery environment with the robot and the patient with both ear volumes depicted as irregular cone-like shapes (Source: Jordan Bos, 2015)

As seen in Figure 1.2, the ear volume, where the surgical operation takes place, can be seen as a somewhat irregular cone, with some of the delicate structures in the middle ear. Depending on the type and required invasiveness of a surgery, structures such as ear ossicles (sound wave vibration transfer), ear drum, semicircular canals (sense

of balance and acceleration), cochlea (auditory sensing), cochlear nerves (auditory sense transfer nerve), vestibular nerves (balance and acceleration sense transfer nerve) and facial nerves (motor control of facial muscles) may be inside the operation volume.

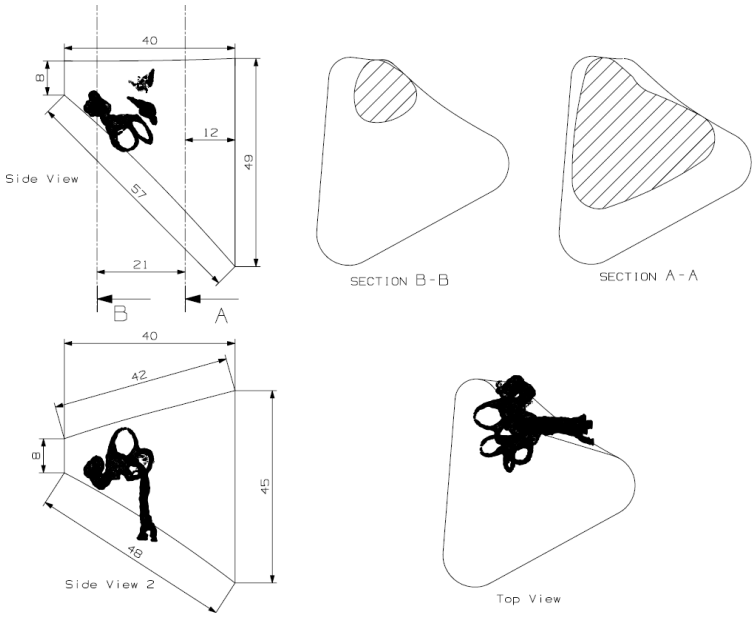

Figure 1.2: Approximate dimensions (in mm) and shape of the ear volume with some of the important ear structures (black) (Source: Jordan Bos, 2015)

With the robot design at hand [1], it is aimed to use the robot to assist the surgeon in the bone removal process. During an operation, 3D images obtained from CT scans are to be used to precisely remove and navigate through less important obstacles while avoiding the important structures along the way to the target locations. With the precision that can be achieved by using CT scans, the use of this robot can reduce the amount of invasiveness throughout the surgery. This can reduce the impact of the surgery on the patients afterwards, since there is a strong possibility of maintaining inner ear functions such as senses of hearing, balance, taste and motor control of facial muscles, which would otherwise require additional post-operations for recovery, if possible.

From a dynamics point of view, the robot has eight DOFs. However, the high-speed rotation of the rotary tool is out of scope of this thesis, since the rotating unit's rapidly changing orientation is not relevant in setting an end-effector pose for the robot. Rather than the orientation of the actual tool tip, the orientation of its axis of rotation is relevant in setting an end-effector pose, from a kinematics point of view. When the high speed motion of the rotary tool, which does not change the tool tip's position, is not taken into account, the initial seven DOFs are sufficient in finding, describing and setting an end-effector pose. Hence, from this point onwards, the robot and its kinematics will be described with seven DOFs.

To use RoBoSculpt in ear surgeries and to move its end-effector for precision milling and drilling tasks, the inverse kinematics problem must be solved. Having seven DOFs, RoBoSculpt is an inherently redundant manipulator since completely specifying the end-effector frame position and orientation is not enough to get a unique set of joint displacements. Additionally, a non-standard kinematic structure, which misses a typical *wrist* present in most industrial manipulators with six DOFs, is proposed due to the

requirements on rigidity and dynamic performance for the tasks of bone milling and drilling.

To ensure successful bone removal by RoBoSculpt, firstly, a solution to the inverse kinematics problem must be developed for this inherently redundant manipulator. As the robot is planned to also assist surgeons during surgery, the inverse kinematics solution should be suitable for real-time use, which requires it to be computationally efficient and reliable. Secondly, it is necessary to devise a strategy to define the desired end-effector motions in terms of parameters that are meaningful for milling and drilling procedures in concave surfaces and for avoiding contacts between the patient and the robot parts other than with the end-effector only. Thirdly, to maintain the manipulability of the end-effector at a high level throughout given tasks, manipulability measures and singularities must be investigated in the robot's workspace. In addition to focusing on these main topics, the subject of intra-collisions between robot parts should also be considered. Finally, since the robot is at its design phase, the software, which is to be developed for testing task specification, generation, inverse kinematics solution and manipulability analysis, should be modifiable and expandable to do design assessments on the robot and its kinematic model.

1.3 Contributions

The contributions of this thesis to the design and realization of the surgical robot RoBoSculpt are as follows:

- A task specification strategy, which involves a parametrization to describe the desired motion of the end-effector to mimic typical hand motions performed by a surgeon during a manual bone removal procedure, is developed. The solution includes a complete definition of the end-effector position, orientation and an additional constraint task. Using geometrical approaches, firstly, the end-effector orientation is described such that tasks can be generated to orientate the end-effector in an approximate ear cone cavity without any collisions inside. Secondly, the clearance problem, which is to keep the end-effector's supporting link clear of the patient, is decoupled from the end-effector task to the rest of the robot joints by setting a constraint on the last joint which utilizes self-motions of the robot and there is no redundancy any more.

- A solution to the inverse kinematics problem, which allows to compute joint trajectories from a given task trajectory, incorporating task augmentation and redundancy resolution by the additional constraint task from the task specification part with a differential inverse kinematics algorithm, with accurate kinematic tracking and efficient computational performance to allow online use in real-time, is developed.

- A method to analyse the robot manipulability is developed, to inspect less manipulable regions of the robot workspace where instabilities and failures may occur, and to help in making design choices for patient-robot positioning, the initial configuration of joints and the robot kinematic model.

- Software tools in Matlab are developed, to evaluate the kinematic model, the design and the proposed solutions in a parametric way to be able to do redesign and

modifications in the model and workspace specifications, and test them accordingly.

1.4 Outline

The outline of this thesis is as follows. In Chapter 2, a literature review on recent surgical robots with purposes similar to RoBoSculpt and on solutions to inverse kinematics of serial manipulators is presented. In Chapter 3, the surgical robot RoBoSculpt is described along with its kinematic model. In Chapter 4, the proposed task specification method is described and corresponding task generation results are presented. In Chapter 5, the proposed solution to the inverse kinematics problem is described and corresponding inverse kinematics simulations and results are presented. In Chapter 6, the proposed method of manipulability analysis with its results and the implementation details of the software used for the simulations are presented. Finally, in Chapter 7, conclusions are made and recommendations are given.

Chapter 2

Literature Review

2.1 Surgical Robots

Since mid-1980s, robots have been taking an active role in healthcare. The first known surgical robot is the Arthrobot, which was used in 1983, for a hip arthroplasty surgery [2]. Arthroplasty is an orthopaedic surgical procedure, during which the function of a skeletal joint is restored by replacement, remodelling or realignment of bone structures. The robot was programmed with a pre-planned motion and mounted on the tip of the femur bone at the hip, where it performed a carving out task to prepare the bone cavity for a hip joint implant. Following Arthrobot was the use of the industrial robot Unimation Puma 200 in stereotactic brain surgery in 1985 [3]. The surgery involved insertion of a probe into the brain through a small hole on the skull of the patient, without any visuals from inside the brain. The success of the operation relied on the surgeon's experience on choosing an initial trajectory and straight guiding of the probe into the skull until the target location for the probe was reached. For these purposes, industrial robots on the market were screened and Unimation PUMA 200 was found to be the best suited one, due to being a programmable, computer controlled and versatile robot that was designed for accurate and delicate tasks. Other robots followed these pioneers and assisted surgeons in other procedures such as urological surgeries [4], endoscopy [5] and laparoscopic surgeries. In the year 1998, two robotic surgical systems were introduced; ZEUS by Computer Motion, Inc. and the da Vinci by Intuitive Surgical, Inc., which were both consisting of a surgical control centre and a robotic arms unit. The current state of the da Vinci Robotic Surgical System, which is one of the systems that is used in various locations world-wide, allows the surgeon the control the robotic arms through a master unit which allows motion scaling and tremor filtration, along with a magnified 3D vision. On the patient side, the robotic arms, which have seven degrees of freedom including the surgical instruments attached, are inserted to the surgery area through 1-2 cm incisions which account for being minimally invasive. Use of such a surgical system for minimally invasive robot-assisted surgeries results in decreased length of stay in hospital after surgery and fewer complications in the majority of the cases [6].

2.1.1 Bone Removal Robots

Surgical procedures involving bone removal require higher force interaction due to the structure of the human bone. A robotic system which is used for bone removal requires a structure with high stiffness and rigidity, such that deformations and deflections on the robot parts and joints due to high forces on the end-effector are kept minimal, to overcome high and variable force interaction and perform well during dynamic control. This is reflected in some of the earlier robotic systems such as PUMA 260 [7], ACROBOT [8] and ROBODOC [9] to have a large size. More recently, smaller robotic systems aimed at bone removal such as Mbars [10], MARS [11], which can be mounted on the bone itself, are developed [12]. Robots with larger sizes suffer from the fact that a movement that may occur on the patient side requires the halt of the operation until a re-calibration is done, which is why the bone that is operated on has to be fixed to the operation table. Smaller, bone-mounted robots give more freedom in the movement sense, but they require extra surgical procedures to attach the robot to the bone.

Robotic systems, which are controlled directly by movements of the surgeon as in master-slave systems, are called semi-active. Using semi-active systems, surgeons make use of the rigid structure and guiding mechanisms which eases the task of bone milling in terms of precision and speed. Robotic systems which carry out pre-planned motions autonomously are called active.

A recent study related to ear surgery, using a small sized, active and bone mounted robot proposes the use of CT scans to manually pre-plan the bone removal task on the ear temporal bone, which is then monitored by the surgeons throughout surgical procedure [13]. The small sized robot, which is fixed to the skull during surgery, has four degrees of freedom, which include three prismatic joints and a revolute joint. Trajectory planning is done after the surgeons manually choose the target regions to be removed in the 3D CT scans of the bone. The selected regions are voxellized and an algorithm is used, which results in visiting all of the target voxels and coming back to the starting point. To begin the surgical procedure, the robot is fixed to the patient and a CT scanning is performed to align the pre-plan with the current placement of the patient with respect to the robot.

Another recent study is related to a robot named RobOtol [14], which is a semi-active robot operated by surgeons using a master unit. The robot has six degrees of freedom and it is fixed in the operation environment. During operation, surgeons has complete control on the movements of the robot and visual access is made available by using an additional endoscopic unit. Although the robot is designed for removal of structures and tissues other than the bone in ear surgery, its kinematic model and tool manipulation inside the auditory canal are of interest.

2.2 Inverse Kinematics

Kinematics describes the motion of points and bodies relative to each other without any consideration of the causes of motion. The kinematic model of a robot contains information about how the motions of robot links are related to each other through the configuration of robot joints. Having a kinematic model of a robot, one can find the motion of the its end-effector resulting from the individual motions of its links by use of its joints, which is so-called the forward kinematics (see Appendix B for more details). The inverse kinematics, on the other hand, is related to the problem of finding the joint configurations of a robot that satisfies the execution of a task at the end-effector level.

A joint configuration is described in the robot's configuration space, whereas a task is described in the robot's task (or operational) space.

In this section, different methods to approach and solve the inverse kinematics problem of a serial chain manipulator are evaluated. Firstly, a brief overview is given for forward and inverse kinematics. Secondly, the availability of a closed-form solution through a geometric or an analytical approach is discussed in Section 2.2.2. Thirdly, using differential methods to solve the inverse kinematics problem is discussed in Section 2.2.4. Following that, the concept of redundancy and redundancy resolution methods are discussed in Section 2.2.5 and finally, the singularity problem, methods of doing a manipulability analysis and manipulability measures are discussed in Section 2.3. Afterwards, a discussion on the reviewed topics is given.

2.2.1 Overview

The forward kinematics of a serial-chain manipulator gives the position and orientation of the end-effector frame relative to the base frame, with given joint displacements. Denoting the end-effector task variables by the vector x, which is composed of the variables describing the position and orientation of the end-effector frame, and the joint displacements by the vector q, the forward kinematics can be expressed as

$$x = f(q). \tag{2.1}$$

The set of task variables x can include the position and orientation of the end-effector and other variables that depend on joint displacement variables. Having a kinematic model, the forward kinematics equation $f(q)$ can be obtained through rigid body transformations (see Appendix B), which contains trigonometric expressions and is therefore a non-linear vector function.

In the cases that a robot is expected to execute a task, it is necessary to obtain the set of joint variables q that results in a desired task x_d, which is the inverse kinematics problem that involves finding the inverse mapping between the joint and task variables, if possible. Assuming that a solution exists (i.e. the task x_d lies in the workspace of the robot) and the forward kinematics equation $f(\cdot)$ is invertible, the inverse kinematics solution can be expressed as

$$q_d = f^{-1}(x_d), \tag{2.2}$$

where x_d and q_d denote the desired task and joint variables which are mapped through the inverse kinematics described by $f^{-1}(\cdot)$. However, obtaining the inverse mapping $f^{-1}(\cdot)$ from the forward kinematics non-linear vector function $f(\cdot)$ is not trivial at all even if it exists. Nevertheless, it is necessary to find the joint trajectories $q_d(t)$ over time t, by any means available such as analytical, geometrical, differential or numerical methods, if a robot is expected to perform given tasks described by $x_d(t)$.

2.2.2 Closed-form Solutions

Closed-form solutions are desirable since they can provide all of the possible solutions to the inverse kinematics problem faster than numerical methods, assuming that there exists at least a solution (i.e. the desired task is inside the workspace of the robot). However, closed-form solutions are not easy to obtain and they are specific to the robot's kinematic model.

Two sets of methods can be used to find a closed-form solution to the inverse kinematics problem; algebraic and geometric. Algebraic methods involve using relations between the equations on the transformation matrix of the end-effector with respect to the base. If possible, by using these relations, the equations result in the joint variables. Geometric methods involve, if possible, decomposing the inverse kinematics problem into planar sub-problems by relating the desired position and orientation to reduced sets of joint variables. Both algebraic and geometric methods are suitable for non-redundant systems and their complexity increases with the number of degrees of freedom. For a six degrees of freedom serial chain manipulator, there exist a closed-form solution if the two sufficient conditions below are satisfied [15]:

- Three consecutive revolute joints' axes intersect at a certain point,
- Three consecutive revolute joints' axes are parallel to each other.

These conditions allow the inverse kinematics problem to be decomposed into two sub-problems of finding the joint variables for a desired orientation and position.

The mapping from joint variables to task variables becomes increasingly non-linear as the number of degrees of freedom of a considered a manipulator is high. Thus, finding the inverse mapping using algebraic or geometric methods can be too complex to handle. Also, robots, which are functionally or mechanically redundant, have more degrees of freedom than necessary to execute their tasks and therefore the inverse kinematics problem of redundant manipulators has either infinitely many solutions or no solution.

2.2.3 Numerical Solutions

There are several approaches to tackle the problem of choosing a set of joint variables when infinitely many solutions to the inverse kinematics problem exist. The iterative methods include Newton-Raphson method [15], optimization approaches [16], resolved motion rate control [17], interval analysis [18] and damped least-squares approach.

In the Newton-Raphson method, first-order approximation of the set of non-linear equations is used. Based on an initial guess, the iterative algorithm searches for a solution until it converges within a set tolerance. This method is useful to find the starting configuration of the robot for different tasks that require different initial configurations.

Although numerical solutions are not robot-specific, they are generally based on a proper initial guess, they may not provide all possible solutions for some cases and they are computationally demanding to do in real time.

2.2.4 Differential Inverse Kinematics

For redundant kinematic structures with a high number of degrees of freedom, the solution to the inverse kinematics problem is not straightforward due to the highly non-linear relationship between the joint and task space variables. Since the first-order differential kinematics is a linear mapping between the joint and task velocity spaces, it can be used to obtain the inverse kinematics solution numerically.

Let the set of joint variables which belong to the configuration space of a robot be

$$q = (q_1, \ldots, q_n)^T, \tag{2.3}$$

where n is the number of degrees of freedom of the robot and q_i with $i \in \{1, \ldots, n\}$ denote the joint variable which is the relative displacement of body i with respect to body $i-1$ according to the joint characteristics.

Let the set of task variables which belong to the task space of a robot be

$$x = (x_1, \ldots, x_m)^T, \tag{2.4}$$

where m is the number of degrees of freedom that the task is described by and each task variable x_i is one of the terms that describe the task. Generally, $m = 6$ holds where the first three components in x describe the end-effector position and the last three describe the end-effector orientation using three angular variables as a minimal representation of orientation such as Euler angles. Such a task can be described as

$$x = [o_x \; o_y \; o_z \; \alpha \; \beta \; \gamma]^T. \tag{2.5}$$

The relationship between the robot configuration (2.3) and the task configuration (2.4) in an appropriate space can be established at position, velocity and acceleration levels. Recall that forward kinematics is described as

$$x = f(q), \tag{2.6}$$

where $f(\cdot)$ is a non-linear vector function, mapping the joint configuration q to the task configuration x.

The first-order differential kinematics is obtained by differentiating the forward kinematics (2.6) with respect to time as

$$\dot{x} = J_A(q)\dot{q}, \tag{2.7}$$

where $\dot{x}$ is the task space velocity, $\dot{q}$ is the joint space velocity and $J_A(q) = \frac{\partial f}{\partial q}$ is the analytical Jacobian matrix with the dimensions $m \times n$. Note that $\dot{x}$ describes the task space velocity which represents the rate of change in position and the parameters from the minimal description of the orientation. Hence, it does not include directly the angular velocities of the end effector, in which case the geometric Jacobian would be used.

Similarly, one can also obtain the second-order differential kinematics as

$$\ddot{x} = J_A(q)\ddot{q} + \dot{J}_A(q, \dot{q})\dot{q}. \tag{2.8}$$

The end-effector spatial velocity which contains the translational and angular velocities is

$$v = \begin{pmatrix} \dot{p} \\ \omega \end{pmatrix}_{6\times 1}, \tag{2.9}$$

and the following transformation holds between the spatial velocity v and the rate of change of the task variables $\dot{x}$

$$\dot{x} = T(x)v, \tag{2.10}$$

where $T(x)$ is the transformation matrix depending on the chosen angular representation for the orientation.

The transformation matrix $T(x)$ has the form

$$T(x) = \begin{pmatrix} I & 0 \\ 0 & R_{mr} \end{pmatrix}, \tag{2.11}$$

where I is an 3×3 identity matrix since there is no transformation necessary between first-order differential of positional variables and R_{mr} is the matrix which relates the first-order differential of the angular parameters to the angular velocity of the end effector, depending on the adopted minimal representation of the orientation.

For a given manipulator, the mapping between the joint and task space velocities is described as

$$v = J(q)\dot{q}, \tag{2.12}$$

where $J(q)$ is the geometric Jacobian matrix with dimensions $6 \times n$ which relates the joint-space velocity directly to the end effector velocity. From (2.7), (2.10) and (2.12), we can relate the geometric Jacobian matrix to the analytical Jacobian matrix as

$$J_A(q) = T(x)J(q). \tag{2.13}$$

By considering (2.10) and (2.12) with $m = n = 6$ (i.e., the Jacobian is a 6×6 square matrix) and assuming that the Jacobian is invertible, the set of joint velocities which is the solution to the first-order differential kinematics in (2.12), can be obtained as

$$\dot{q} = J^{-1}(q)v, \tag{2.14}$$

where $v = T^{-1}(x)\dot{x}$ is the spatial end-effector velocity, assuming that $T(x)$ is invertible (i.e. there is no representational singularity which causes rank deficiency in $T(x)$) and $J(q)$ is invertible (i.e. no kinematically singular configuration is reached).

With a given initial joint configuration $q_0 = q(0)$, the joint positions can be computed by integrating computed velocities from (2.14) with given desired position $x_d(t)$ and velocity trajectories $v_d(t)$ as

$$q(t) = \int_0^t J^{-1}(q(\tau))v_d(\tau)\, d\tau + q_0. \tag{2.15}$$

This method of inverting the kinematics requires that the Jacobian is square and invertible. However, in the case of redundant manipulators, the Jacobian is not square. Additionally, kinematic and representational singularities should be taken into account. Lastly, in theory, integration of the first-order inverse kinematics gives the joint profile that executes the given task. However, in practice, errors result from numerical integration and a drift between the final and the desired end effector pose in task space takes place. These issues are discussed in the next sections.

2.2.5 Redundant Serial Manipulator Kinematics

A kinematically redundant manipulator has more degrees of freedom than it is strictly required to execute its task. The additional degrees of freedom provide a higher level of dexterity, which can be used to avoid singularities, obstacles in the workspace and joint limits. However, it also results in infinitely many solutions and the problem of choosing one.

The redundancy of a manipulator depends on its given task. In three-dimensional space, a manipulator task is generally described by the position and orientation of

the end-effector which requires six degrees of freedom. However, some tasks can also be described using a subset of the generally required six degrees of freedom (e.g. to position the end-effector without any constraints on its orientation or the given task). In this regard, robotic manipulators with six or less degrees of freedom may or may not be redundant, depending on the task given, whereas a manipulator with seven or more degrees of freedom can be described as inherently redundant.

Although minimal complexity in design is a desirable factor since it reduces costs and maintenance issues, redundant manipulators have greater dexterity which allow for motions which would otherwise not be possible. These may include internal motions which allow movement of the joints while the end-effector task does not undergo a change. Additionally, the infinitely many solutions in the workspace of the robot allow for obstacle, intra-collision and singularity avoidance while the same task is carried out. However, it should be noted that with the given complexity of the system that allows for high dexterity, kinematic singularities, joint limits and intra-collisions become more crucial for safe operation and therefore must be considered in the robot motion planning.

Inverse Differential Kinematics for Redundant Manipulators

For redundant manipulators, the number of joint space variables exceeds the number of the variables that describe the task ($m < n$). As a result, the Jacobian matrix of a redundant manipulator is low-rectangular having more columns than rows and infinitely many solutions exist as long as it is full rank (i.e., the manipulator is not at a singular configuration). The first-order differential kinematics (2.7), which is a linear mapping between the joint and task space velocities with the Jacobian as the coefficient matrix [17], has the general solution that can be expressed for redundant manipulators as

$$\dot{q} = J_A^+ \dot{x} + (I - J_A^+ J_A)\dot{q}_0, \tag{2.16}$$

where J_A^+ is the pseudo-inverse of the analytical Jacobian matrix and its dependency on q is omitted for clarity. Also $(I - J_A^+ J_A)$ represents the null-space projection matrix of J_A and $\dot{q}_0$ is an arbitrary vector incorporating the infinitely many solutions that result from redundancy. These are the so-called null-space velocities which result in internal motions where the end-effector stays stationary in terms of its task, while the joints move.

The pseudo-inverse of a matrix J, which is denoted by J^+, satisfies the following Moore-Penrose conditions [19]

$$\begin{array}{rcl} JJ^+J & = & J \\ J^+JJ^+ & = & J^+ \\ (JJ^+)^T & = & JJ^+ \\ (J^+J)^T & = & J^+J \end{array} . \tag{2.17}$$

When $J_A \in \mathbb{R}^{m \times n}$ is a low-rectangular matrix ($m < n$) and it has full rank (m independent row vectors), its pseudo-inverse is defined as

$$J_A^+ = J_A^T (J_A J_A^T)^{-1}. \tag{2.18}$$

When the choice of the arbitrary vector is set to $\dot{q}_0 = 0$, (2.16) becomes

$$\dot{q} = J_A^+ \dot{x}, \tag{2.19}$$

which provides the least-squares solution with minimum norm to (2.7).

Equation (2.16) is a general solution to the inverse differential kinematics of a redundant manipulator, which gives all least-squares solutions to the problem of minimizing $||\dot{x} - J_A\dot{q}||$. Using the arbitrary vector $\dot{q}_0$ and the orthogonal projection into the null-space of J_A, solutions can be obtained in many different ways that result in different joint velocities, while the task velocity is the same. Alternatively, the minimum norm solution from (2.19) can also be chosen.

Optimization for Redundancy Resolution

When the Jacobian matrix is not square, the least squares solution to $\dot{x} = J_A(q)\dot{q}$ is

$$\dot{q} = J_A^+(q)\dot{x}, \tag{2.20}$$

where $J_A^+ = J_A^T(J_A J_A^T)^{-1}$ (omitting the dependency on q for simplicity) is the pseudo-inverse of the Jacobian matrix when it is low rectangular and full rank. In addition to this solution, which gives a minimum norm result, the orthogonal projection matrix N_0, which is described as

$$N_0 = (I - J_A^+(q)J_A(q)), \tag{2.21}$$

can be used to exploit the underdetermined nature of the problem. Using (2.21), any arbitrary vector $\dot{q}_0$ can be projected into the null space of $J_A(q)$ to include null-space velocities in the solution without changing the resulting task-space velocity $\dot{x}$, as

$$\dot{q} = J_A^+(q)\dot{x} + N_0\dot{q}_0. \tag{2.22}$$

This approach can also be used for acceleration (i.e., second-order differential kinematics level) as

$$\ddot{x} = J_A(q)\ddot{q} + \dot{J}_A(q,\dot{q})\dot{q}, \tag{2.23}$$

$$\ddot{q} = J_A^+(q)(\ddot{x} - \dot{J}_A(q,\dot{q})\dot{q}) + N_0\ddot{q}_0. \tag{2.24}$$

Both (2.22) and (2.24) show that for given task velocity and acceleration profiles, infinitely many solutions to joint velocity and acceleration profiles exist. To choose a solution among the infinitely many, the arbitrary vector can be defined according to properly set criteria, which impose secondary tasks. However, it should be noted that whether a minimum norm solution as in (2.20) or a general solution as in (2.22) is used, the solution is still a local optimization and it does not ensure that a global velocity or acceleration minimization is achieved. Therefore, criteria set with secondary tasks in the general solution are not guaranteed along the whole task trajectory [20]. Below are three different examples of utilizing the redundant degrees of freedom for different purposes where the arbitrary vector $\dot{q}_0$ is defined [21] in the form

$$\dot{q}_0 = k_0\frac{\partial H}{\partial q}^T, \tag{2.25}$$

which results in the general solution as

$$\dot{q} = J_A^+(q)\dot{x} + k_0(I - J_A^+(q)J_A(q))\nabla H(q), \tag{2.26}$$

where k_0 is a positive constant and H is an objective function that is secondary with respect to the primary task of trajectory tracking. A measure, which is desired to

be maximized, is a proper choice for the objective function H, since the gradient of H defines the direction of the solution, which is locally compatible to the primary objective of trajectory tracking.

Possible choices for the objective function H in (2.25) to impose secondary tasks are [21],

- For joint limits, the secondary objective function H can be chosen as a performance criteria defining the distance from joint limits based on the current joint configuration, which must be maximized and which is defined as

$$H(q) = -\frac{1}{2n}\sum_{i=1}^{n}\left(\frac{q_i - q_{i,mid}}{q_{i,max} - q_{i,min}}\right)^2, \tag{2.27}$$

  where n denotes the number of degrees of freedom, and q_i, $q_{i,min}$, $q_{i,mid}$ and $q_{i,max}$ denote, at i^{th} among n joints, the current joint position, the lower limit, the mid-value and the upper limit for the joint displacement, respectively.

- For singularity avoidance, the secondary objective function H can be defined as the manipulability measure

$$H(q) = \sqrt{det(J(q)J^T(q))}, \tag{2.28}$$

  which vanishes as a singular configuration is reached and by maximizing it, singularities may be avoided.

- For obstacle avoidance, the secondary objective function H can be defined as the distance from an obstacle in the workspace as

$$H(q) = \min_{p,p_o} ||p(q) - p_o||, \tag{2.29}$$

  where p_o is the position of a chosen point on the obstacle and $p(q)$ is the position of a generic point on the manipulator, and by maximizing the above quantity, collisions with obstacles can be avoided.

Use of the arbitrary vector q_0 is promising for the freedom it allows in setting secondary tasks such as joint limits avoidance and singularity avoidance. However, this is only a local optimization. In fact, every form of the pseudo-inverse solution is locally optimal. Even if a solution is found at a given time step, this does not ensure optimality of the solution for the whole task trajectory. Alternatively, optimization at the second-order differential kinematics level or global optimization can be utilized. However, the selection of joint velocities in real time would not be computationally feasible, since any change in the trajectory requires repetition of the optimization procedure for the whole trajectory.

Task-Space Augmentation for Redundancy Resolution

To make use of the redundant degrees of freedom, the task at hand can be dimensionally augmented by a proper number of degrees of freedom to include objectives in addition to the end-effector task. As a result, a set of joint variables from the infinitely

many solutions can be chosen which fulfill the end-effector task along with the added objectives. Two approaches are the extended Jacobian [22, 23] and the augmented Jacobian [24, 25].

Recall the direct forward kinematics and the first-order inverse differential kinematics

$$x = f(q), \tag{2.30}$$

$$\dot{x} = J_A(q)\dot{q}, \tag{2.31}$$

where $q \in \mathrm{R}^n$ and $x \in \mathrm{R}^m$ are the sets of joint and task space variables having the specified dimensions, respectively.

Extended Jacobian Approach

In the extended Jacobian approach, the task vector is augmented by defining additional functional constraints along with the original task of the end-effector. Constraints are obtained as follows. Let $g(q)$ be a objective function to be optimized. When $g(q)$ is at an extreme (a minimum or a maximum, depending on how $g(q)$ is defined; it can be average joint torque which is to be minimized or a manipulability measure which is to be maximized) for $q = q_0$ under the constraint $x_0 = f(q_0)$, the following holds [22]

$$\left.\frac{\partial g(q)}{\partial q}\right|_{q=q_0} (I - J_A^T(q_0)J_A(q_0)) = 0^T, \tag{2.32}$$

which yields a set of equations that are independent constraints. The set of constraint equations have the dimension of the null space of J_A, which is $n-m$, where n and m are the number of DOF of the robot manipulator and the task, respectively. The constraint equations can be written in vector form and be denoted by $h(\cdot)$ as

$$h(q_0) = \left(\left.\frac{\partial g(q)}{\partial q}\right|_{q=q_0} (I - J_A^T(q_0)J_A(q_0)) \right)^T = 0. \tag{2.33}$$

The condition in (2.33) can be incorporated with (2.30) setting that the motion starts with x_0 and q_0 and the objective function $g(q)$ is extremized (minimized or maximized, depending on the choice of $g(q)$) at all times, as

$$\begin{pmatrix} x(t) \\ 0 \end{pmatrix} = \begin{pmatrix} f(q(t)) \\ h(q(t)) \end{pmatrix}, \tag{2.34}$$

and by differentiating the above expression with respect to time

$$\begin{pmatrix} \dot{x}(t) \\ 0 \end{pmatrix} = \begin{pmatrix} J_A(q(t)) \\ \frac{\partial h(q(t))}{\partial q} \end{pmatrix} \dot{q}(t), \tag{2.35}$$

where the extended Jacobian is obtained as

$$J_{ext} = \begin{pmatrix} J_A(q) \\ \frac{\partial h(q)}{\partial q} \end{pmatrix}. \tag{2.36}$$

If the initial configuration q_0 is chosen such that it extremizes the objective function $g(q)$ and the square extended Jacobian is invertible provided that no singular configuration is attained, the solution to the inverse kinematics at the first-order differential level is obtained as

$$\dot{q} = J_{ext}^{-1} \begin{pmatrix} \dot{x} \\ 0 \end{pmatrix}, \tag{2.37}$$

which can be integrated over time to obtain joint trajectories $q(t)$ which track a task trajectory $x(t)$ while optimizing $g(q(t))$.

Augmented Jacobian Approach

In the augmented Jacobian approach, a constraint task x_c with a proper dimension is introduced to be fulfilled in addition to the original end-effector task x. The relation between the constraint task vector x_c and the joint variables q can be considered similar to the direct forward kinematics equation as

$$x_c = f_c(q), \tag{2.38}$$

where the constraint task has a dimension equal to the number of redundant degrees of freedom (i.e. $x_c \in \mathrm{R}^{n-m}$). Its first-order differential with respect to time follows as

$$\dot{x_c} = J_c(q)\dot{q}. \tag{2.39}$$

With the addition of the constraint task as defined by its kinematics equation (2.38), the task vector is augmented as

$$x_{aug} = \begin{pmatrix} x \\ x_c \end{pmatrix} = \begin{pmatrix} f(q) \\ f_c(q) \end{pmatrix}. \tag{2.40}$$

Then, the first-order differential kinematics follows as

$$\dot{x}_{aug} = \begin{pmatrix} J_A(q) \\ J_c(q) \end{pmatrix} \dot{q}, \tag{2.41}$$

where $J_c = \dfrac{\partial x_c}{\partial q}$ is the Jacobian of the constraint task. Hence, the augmented Jacobian matrix is defined as

$$J_{aug}(q) = \begin{pmatrix} J_A(q) \\ J_c(q) \end{pmatrix}, \tag{2.42}$$

which has the dimension $n \times n$. By inverting the linear mapping between the joint variables q and the augmented task variables x_{aug} in $\dot{x}_{aug} = J_{aug}(q)\dot{q}$, we obtain

$$\dot{q} = J_{aug}^{-1}(q)\dot{x}_{aug}, \tag{2.43}$$

where the solution $\dot{q}$ satisfies both the end-effector task x and the constraint task x_c, given that the augmented Jacobian matrix $J_{aug}(q)$ does not become singular. This makes the choice of the constraint task critical. The inverse differential kinematics equation in (2.43) can then be integrated over time to obtain joint trajectories $q(t)$ which satisfy an augmented task trajectory $x_{aug}(t)$.

2.3 Singularity Problem

A robot configuration q_s is singular if the Jacobian matrix $J_A(q)$ is rank deficient at $q = q_s$. From the first-order (2.7) and second-order differential kinematics (2.8), it is straightforward to see that at a singular configuration, it is impossible to generate end effector task velocities and accelerations in certain directions due to the rank deficiency. From (2.13), it can be stated that a singularity might occur due to a loss of rank of the transformation matrix $T(x)$ and/or the geometric Jacobian matrix $J(q)$. A singularity from the transformation matrix $T(x)$ is not directly related to the motion capability of the manipulator since it depends on the adopted representation of the orientation of the end-effector. This is why such a singularity is referred to as *a representation singularity*. A singularity occurring from the geometric Jacobian matrix is due to the joint configurations where the desired task velocities become infeasible for any joint velocity command due to a loss of mobility. This is why such a singularity is referred to as *a kinematic singularity*.

2.3.1 Singular Value Decomposition of a Jacobian matrix

To analyze the first-order differential kinematics (2.7), motion capabilities and possible singularities, singular value decomposition (SVD) can be used to determine the rank of the Jacobian which can be decomposed as

$$J = U\Sigma V^T = \sum_{i=1}^{m} \sigma_i u_i v_i^T, \tag{2.44}$$

where U is the orthonormal $m \times m$ matrix of output singular vectors u_i, V is the orthonormal $n \times n$ matrix of input singular vectors v_i, $\Sigma = (S\ 0)$ is a matrix of dimensions $m \times n$ containing the singular values σ_i of the Jacobian matrix J at the diagonal entries of the square $m \times m$ submatrix S.

2.3.2 Manipulability Measures

Since, at a singular configuration, one or more singular values approach to zero, a number of manipulability measures can be defined. One measure, which is the so-called Yoshikawa's manipulability index [26], is defined as

$$\mu_1(q) = \sqrt{det(J(q)J^T(q))} = \sigma_1\sigma_2...\sigma_N, \tag{2.45}$$

where N is the number of degrees of freedom. The singular values are ordered with respect to decreasing values, where σ_1 and σ_N denote the maximum and minimum singular values, respectively.

Another manipulability measure, which is the inverse condition number measurement [27], is defined as

$$\mu_2 = \frac{1}{\text{cond}(J)} = \frac{\sigma_N}{\sigma_1}, \tag{2.46}$$

which gives the ratio between the minimum and maximum singular values.

Since the minimum singular value approaching to zero is the critical condition, by itself it can be defined as a manipulability measure as

$$\mu_3 = \sigma_1. \tag{2.47}$$

2.4 Discussion

For a serial chain manipulator with seven DOFs, which is inherently redundant for any end-effector task described in Euclidean space, inverting the forward kinematics to get a closed-form solution is not possible, since infinitely many solutions to configuration of the robot exist if the given task is inside the workspace of the robot. Hence, the problem of finding all possible solutions, choosing one of them and doing this in real time can be solved at the velocity level by inverting the first-order differential kinematics.

A possible option for inverse differential kinematics is to use optimization techniques, which are based on set objectives to choose a solution among infinitely many, either locally or globally for a given task trajectory. Optimization utilizes the self-motions of the manipulator to optimize set objectives. However, optimization can be either locally, which is more practical but does not guarantee an optimal solution for a given task trajectory; or globally, which guarantees the optimal solution (if exists) throughout the task trajectory but which is also not practical since any configuration or task changes during operation would require solving the optimization problem again.

Another option for solving the inverse kinematics problem of a redundant manipulator is to use redundancy resolution methods by task augmentation, where functional constraints or additional tasks can be defined to augment the task space such that the robot is not redundant any more. When the task space has the same dimension as the configuration space of the robot, then the velocity kinematics, which is linear, can be directly inverted to track a given task. However, there is no general solution for the definition of a functional constraint or an additional task, which depend on the specifications of the given task and the robot.

It will be seen in Chapter 4 that, specific to RoBoSculpt, a one dimensional task is added to the task of end-effector pose. This is also specific to its task of bone removal around the ear inside a cavity and required to avoid collisions, as it will be explained in Section 4.1.4. Then, in Chapter 5, with the augmented task space, a standard inverse differential kinematics algorithm [21] will be adopted and modified to use with augmented task space and augmented Jacobian, as explained in Section 5.1.

Chapter 3

Robot Description

3.1 Robot Description

3.1.1 Kinematic Model

RoBoSculpt is a serial chain manipulator with six revolute joints, one prismatic joint, followed by a rotary unit for milling and drilling. After the base, five modular rotating units are present. A side view of the robot model, which is attached to a surgery table, is shown in Figure 3.1, where a preliminary location for an ear volume is also shown.

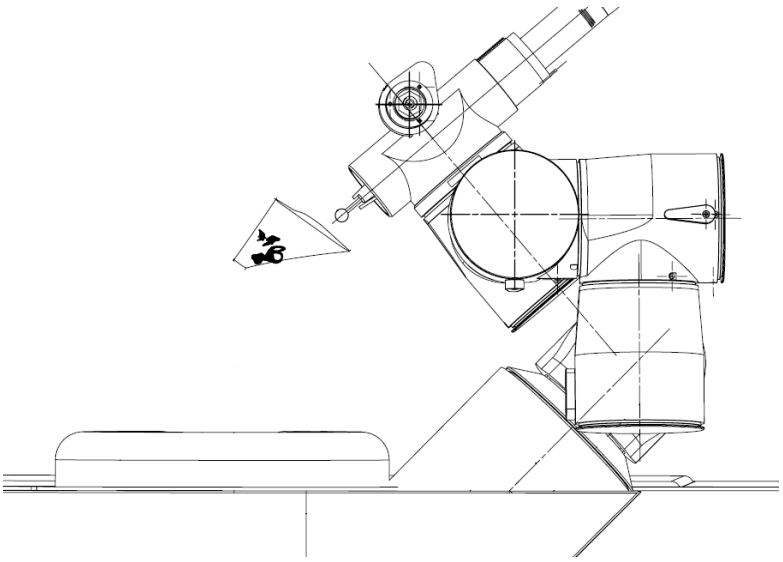

Figure 3.1: Robot CAD model viewed from side (Source: Jordan Bos, 2015)

The kinematic structure of the serial manipulator with seven DOFs can be seen in Figure 3.2 (dimensions of the entities are not to scale),

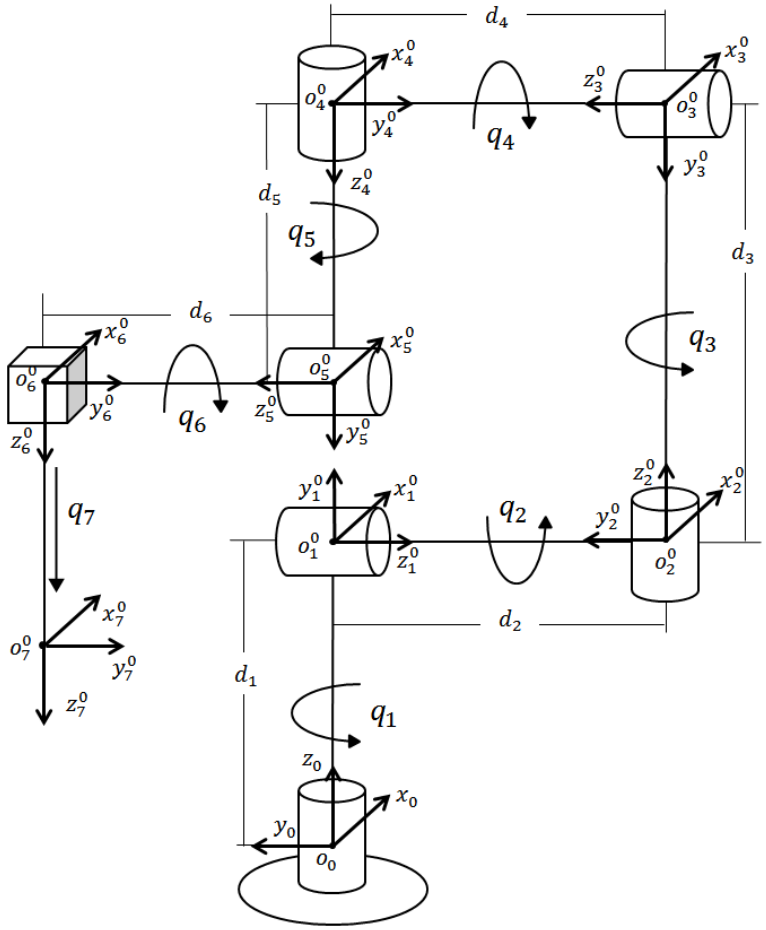

Figure 3.2: Robot schematic with the DH parameters and frames

According to the Denavit-Hartenberg (DH) convention [28], the homogeneous transformation between two consecutive frames is defined from the four parameters (θ_i, d_i, a_i, α_i) as follows

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where

$$R_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix}, \tag{3.2}$$

is the matrix defining the rotational transformation from frame $i-1$ to i and

$$p_i^{i-1} = \begin{bmatrix} a_i\cos\theta_i \\ a_i\sin\theta_i \\ d_i \end{bmatrix}, \tag{3.3}$$

is the vector defining the position of frame i with respect to frame $i-1$.

Link (i)	θ_i (rad)	d_i (m)	a_i (m)	α_i (rad)
1	q_1	0.070	0	$\pi/2$
2	q_2	0.070	0	$-\pi/2$
3	q_3	0.075	0	$-\pi/2$
4	q_4	0.070	0	$-\pi/2$
5	q_5	0.070	0	$\pi/2$
6	q_6	0.065	0	$-\pi/2$
7	0	q_7	0	0

Table 3.1: Joint displacements (q_i) and the scalar values of the constant DH parameters of RoBoSculpt

As seen in the Table 3.1, for the joint variables $q = \begin{bmatrix} q_1 & ... & q_n \end{bmatrix}^T$ where $n = 7$ is the number of degrees freedom, it holds that $q_i = \theta_i$ for revolute joints and $q_i = d_i$ for prismatic joints.

As the homogeneous transformations between each pair of frames are with respect to the axes of the local frame, the homogeneous transformation from the fixed frame to the end effector frame can be computed recursively with right multiplication as [29]

$$T_7^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6, \tag{3.4}$$

where

$$T_7^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & (p_7^0)_x \\ r_{21} & r_{22} & r_{23} & (p_7^0)_y \\ r_{31} & r_{32} & r_{33} & (p_7^0)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

is the matrix defining the homogeneous transformation from the fixed frame to the end-effector frame. It includes the matrix

$$R_7^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{3.6}$$

which defines the orientation of the end-effector frame, and the vector

$$p_7^0 = \begin{bmatrix} (p_7^0)_x \\ (p_7^0)_y \\ (p_7^0)_z \end{bmatrix}, \tag{3.7}$$

which gives the position of the end-effector frame origin. The homogeneous transformation and the geometric Jacobian matrices of the end-effector frame of RoBoSculpt are given in Appendix D.

Chapter 4

Task Specification Solution and Results

In this chapter, a solution to the problem of specifying end-effector tasks for RoBoSculpt and the results from task trajectory generation with the proposed method are presented. Firstly, the task specification solution is introduced in Section 4.1. It includes the approximation of the ear volume for bone removal task in Section 4.1.1, the definitions for desired end-effector position and orientation trajectories in Sections 4.1.2-4.1.3, and the definition for an additional constraint task to impose a clearance between the ear volume and the robot in Section 4.1.4. Secondly, results regarding the task trajectory generation are presented in Section 4.2. Finally, a discussion on the proposed task specification method and obtained results is given in Section 4.3.

4.1 Task Specification

To perform a bone removal task with RoBoSculpt, a path for the end-effector to follow must be generated, which is the input for the inverse kinematics problem to solve for the joint trajectories to enable the execution of the task by the robot. As a starting point, we propose to simplify the bone volume, which is to be milled out, to a section of a cone. The bone removal will be assumed to occur on a layer, which can be chosen at any depth inside the cone cavity. The edges of the cavity will also be considered for decisions on the end-effector orientation and robot links' (including the rotary tool and its supporting links) clearance to avoid contacts.

4.1.1 Ear Volume Approximation

To approximate the ear volume which is the main object of the bone removal procedure, a parametrically defined cone shape is used. The process of bone milling is assumed to start from the bottom of a cone (the opening) and go deeper layer by layer. The parameters we use to define the geometrical shape, position and orientation of a cone are the position of cone section frame origin o_s, the rotation matrix describing the orientation of cone section frame $x_s y_s z_s$, cone section radius r_s, cone bottom radius r_b and cone height h_c. Additionally, when the cone is generated, its bottom is directed towards the -z axis of the reference frame $(o_0 x_0 y_0 z_0)$ and an initial rotation by $R_{pre} = R_{x,\pi}$ is applied, where $(R_{x,\pi})_{3\times3}$ is a matrix of rotation about x-axis by π (see Appendix B.1), to align the cone section frame with the reference frame. Using these parameters, a de-

sired ear volume approximation can be generated as a cone by computing the vertices of the cone's bottom, top and middle sections. Then, the obtained shape can be presented using patch objects in Matlab, as seen in Figure 4.1, where the cone section frame axes x_s, y_s and z_s are represented by red, green and blue lines, respectively, on the cone section origin o_s.

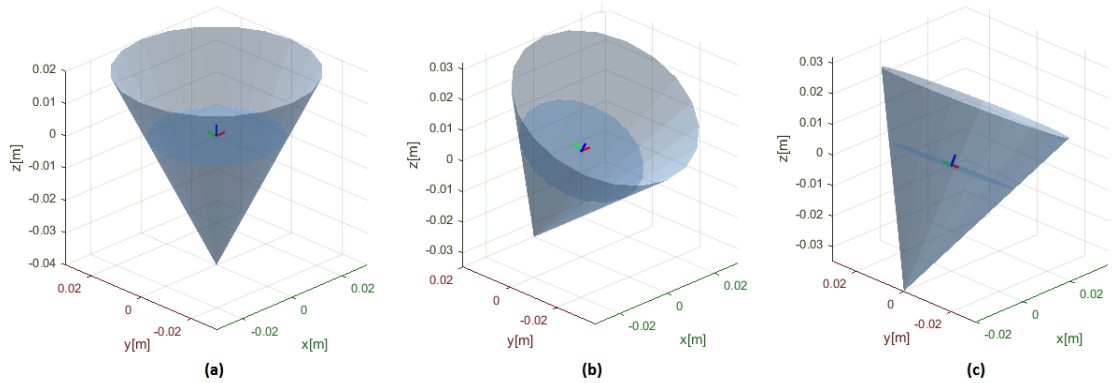

Figure 4.1: Approximate ear cone with different orientations. In part (a), the cone section frame $o_s x_s y_s z_s$ is aligned with the reference frame $o_0 x_0 y_0 z_0$ by rotation R_{pre}. In part (b) and part (c), following the rotation of R_{pre}, the cone section frame is rotated by $\pi/6$ about its current x-axis and y-axis, respectively.

The approximate ear cone can also be placed in the workspace of the robot, as seen in Figure 4.2, where the parameters describing the ear cone are chosen as, $o_s = (-0.1, 0, 0.1)$, $R_s = R_{y,\frac{\pi}{6}} R_{pre}$, $r_s = 0.02$ and $h_k = 0.06$.

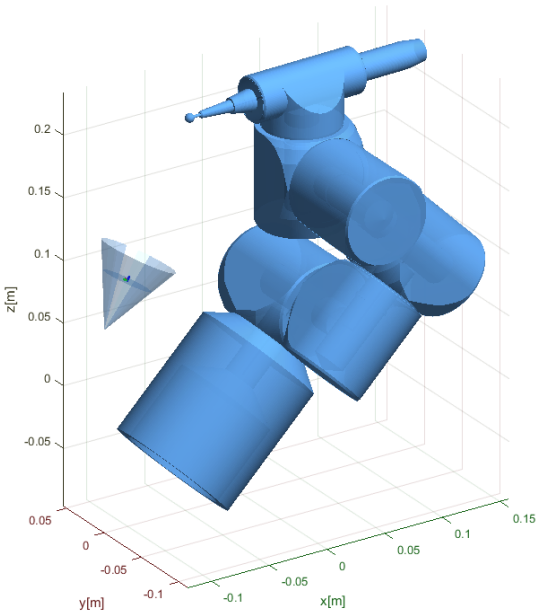

Figure 4.2: Illustration of the surgical robot and the approximate ear cone

4.1.2 End-effector Position Trajectory

We need to define a certain position trajectory, on the section of the cone, which is described by its depth from the opening of the cone. One of the many options is the side-to-side motion. Below are two generated point trajectory profiles, which are described by the number of side-to-side motions (layers), step distance (in meters) along layers and step angle (in radians) along arcs, between consecutive position trajectory points. In Figure 4.3, on the left, an example with less layers as $n_{layer} = 5$ with fine spacing of $\Delta_{step} = 0.001$m between each point can be seen and on the right, the point trajectory is generated with a higher number of layers as $n_{layer} = 20$ but with coarse spacing of $\Delta_{step} = 0.002$m between the points.

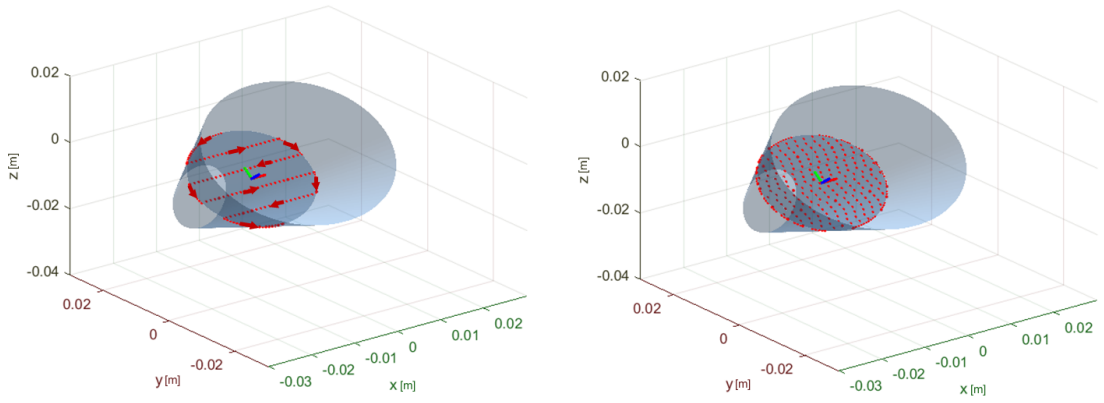

Figure 4.3: Illustration of a position trajectory spanning the cone section. On the left, a trajectory is shown with fine spacing between each point and a small number of layers, whereas on the right, one with coarse spacing and a large number of layers is shown.

It should be noted that a twice continuously differentiable end-effector position trajectory results also in continuous velocity and acceleration profiles for the end-effector and the robot joints, which is desired for dynamic control. Hence, the generated position trajectory for the end-effector must be at least twice continuously differentiable with respect to time. A sample trajectory with up-and-down motion, which spans the circular cone section, can be generated using simple trigonometric functions as

$$\begin{aligned} o_x(t) &= \frac{2r_s t}{T} - r_s \\ o_y(t) &= \sin(w_t o_x(t)) \left(\sqrt{r_s^2 - o_x(t)^2}\right) \\ o_z(t) &= 0 \end{aligned} \quad , \tag{4.1}$$

where (o_x, o_y, o_z) denotes the position of the end-effector with respect to the cone section frame $o_s x_s y_s z_s$, t denotes time variable, T is the total duration of the motion and $\omega_t = n_t \pi/(2r_s)$ sets the frequency of the side-to-side motion with n_t denoting the number of these turns. The generated trajectory is shown with the approximate ear cone in Figure 4.4, with number of turns chosen as $n_t = 10$, which is a preliminary choice for testing and can be determined in the future based on specifications such as the feeding width of the milling tool.

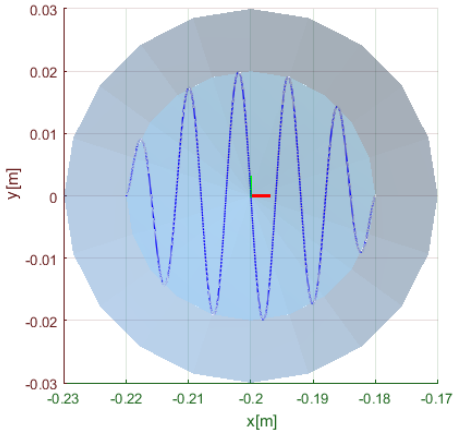

Figure 4.4: A continuously differentiable position trajectory spanning the cone section

Other approaches to generate a position trajectory that spans the planar surface of the cone section can also be formulated, such as a spiral shaped trajectory, as seen in Figure 4.5, where radial and angular step distances and direction are chosen to be $\Delta_{radial} \approx 4 \cdot 10^{-6}$m, $\Delta_{angular} = 0.01$ and clock-wise, respectively.

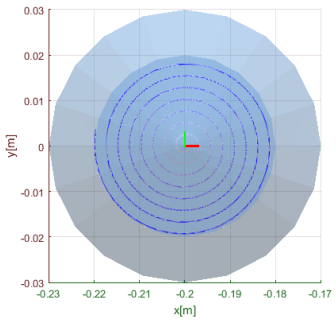

Figure 4.5: Clock-wise spiral continuous trajectory

At this stage, a twice (at least) continuously differentiable position trajectory would be acceptable if it spans the section plane with set parameters, which can later be modified for the specifications of the milling task, such as the diameter of the tool tip, feed rate and width. In the next chapter, where the inverse kinematics solution will be described and tested, the end-effector position trajectory as defined in (4.1) (also shown in Figure 4.4) will be adopted, with an additional version, where the resulting position trajectory will be rotated by $\pi/2$ about the cone section vertical axis. This way, the desired position trajectory will require the end-effector of the robot to move continuously, in an up-and-down motion in one case and a side-to-side motion in the other.

4.1.3 End-effector Orientation Trajectory

Defining the end-effector orientation trajectory is more critical than the position trajectory, since the task involves the milling of the inner section of the cone volume while the execution of the task with the end-effector must be maintained collision-free with respect to the surrounding surfaces.

To describe the orientation of a frame in Euclidean space completely, one needs at least three parameters. In our case, we want to manipulate the robotic arm, such that we set a desired position and orientation profile for the end-effector frame completely, which requires at least six parameters; three for the position and three for the orientation.

The end-effector orientation is chosen as the orientation of the rotary tool's axis instead of the tool tip, which rotates rapidly during milling and drilling, making its angular displacement irrelevant for kinematic task specification. Normally, two parameters are sufficient to define an orientation of the end-effector (rotary tool's axis), which would result in a collision free path inside the cavity. However, a third parameter, which describes a rotation about the tool axis, is necessary for the accessibility of an end-effector pose and for the amount of displacement required by the rest of the robot joints. Hence, in the following section, we define a set of three parameters that are directly related to the pose of the end-effector frame according to the task of bone removal in a manner similar to orientation of a human wrist. Then, a geometric solution is formulated to compute the desired angular parameters.

We also define a constraint on the actuation of the last joint (prismatic) based on the pose of the end-effector, to ensure that the only part of the robot that comes close in contact with the ear volume is the rotary tool tip, which is at the end-effector frame. In other words, it is desired to keep the supporting link of the end-effector away from the ear volume at a specified spherical clearance during the actuation of the prismatic joint. The constrained motion of the prismatic joint utilizes the self-motions of the seven DOFs robot and decouples the clearance problem from the end-effector to the rest of the robot and augments the task space to the dimension of the robot DOF.

Parametrization

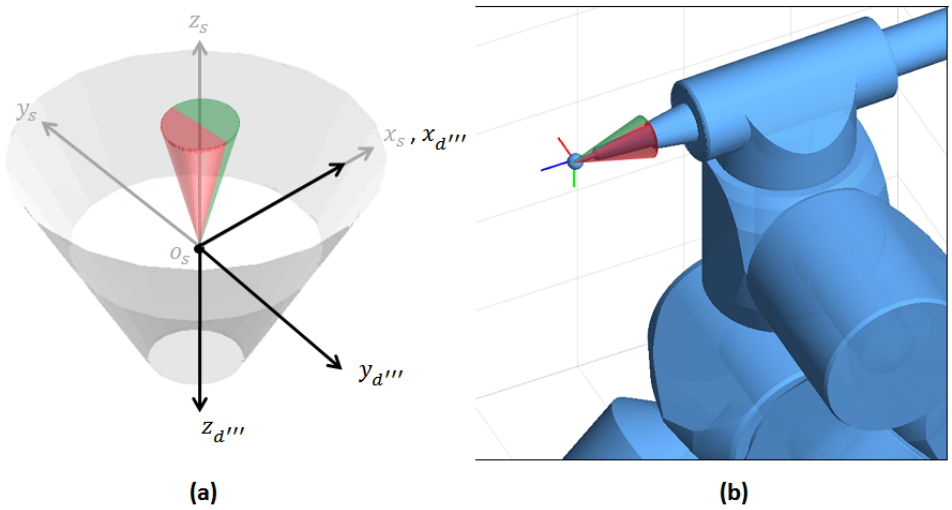

Figure 4.6: Representations of the cone section frame $o_s x_s y_s z_s$ and the intermediate frame $o_{d'''} x_{d'''} y_{d'''} z_{d'''}$ in (a) and the end-effector frame $o_7 x_7 y_7 z_7$ (in red, green, blue) with RoBoSculpt and a virtual cone (red-green) as reference for orientation in (b)

We propose to parametrize the orientation of the end-effector based on an *approach angle* θ, an *inclination angle* ϕ and a *self-rotation angle* α about the prismatic actuation axis. To combine these angular parameters and obtain a desired orientation for the end-effector frame, a constant pre-rotation and three consecutive rotations according to the

angular parameters will be used. Initially, we start with the orientation of the cone section, which is seen in Figure 4.7 (a). If we denote the rotation matrix describing the orientation of the cone section as R_s^0, then we can describe the orientation of the end-effector resulting from the pre-rotation with R_{pre} and the rotations by θ,ϕ and α on each current axes Z,X and Z respectively, as

$$(R_7^0)_d = R_s^0 R_{pre} R_{z,\theta} R_{x,\phi} R_{z,\alpha}, \tag{4.2}$$

which is similar to a ZXZ Euler angles parametrization with the addition of pre multiplications with R_s^0 which is the constant rotation matrix describing the orientation of the cone section frame and with R_{pre}, which is the constant rotation matrix corresponding to the pre-rotation of π about x_s axis for convention of the tool's desired direction, as

$$R_{pre} = R_{x,\pi}, \tag{4.3}$$

which is shown in Figure 4.7(a). With the obtained intermediate frame $o_{d'''}x_{d'''}y_{d'''}z_{d'''}$ adopted, the end-effector would be pointing vertically downwards on the section with its x-axis aligned with x_s. The rotations by the angular parameters of θ,ϕ and α will be applied on this frame. To relate these rotations to the robot, a virtual cone is depicted in Figure 4.6(b) which will also be present in the next figures depicting the rotations.

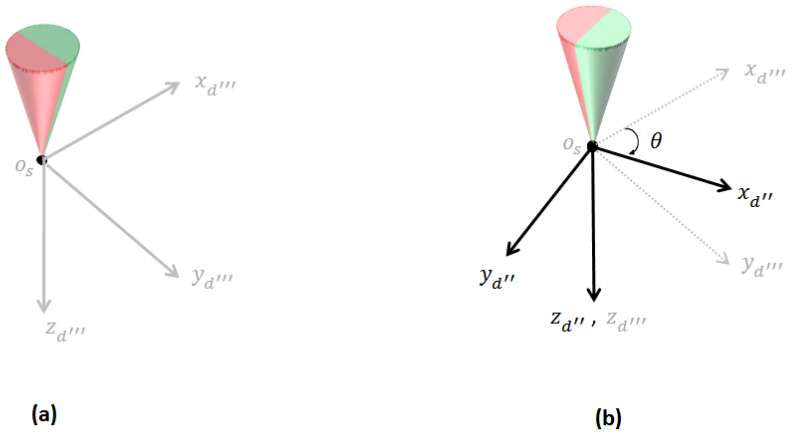

Figure 4.7: Representation of rotations - Plot (a) shows the pre-rotation by R_{pre} around x_s to obtain the intermediate frame $o_{d'''}x_{d'''}y_{d'''}z_{d'''}$ while Plot (b) shows the initial rotation by the *approach angle* θ around $z_{d'''}$ to obtain $o_{d''}x_{d''}y_{d''}z_{d''}$

After the pre-rotation $R_{pre} = R_{x,\pi}$, the frame is rotated by the *approach angle* θ, with respect to its z-axis $z_{d'''}$ as seen in Figure 4.7(b). The *approach angle* θ, can be seen as the angle from the positive y axis of the cone section frame to the projection of the end effector frame's z-axis (in the direction outwards from the tool tip) on the cone section plane. In other words, from a top view (with respect to cone section), it is the angle the milling tool makes relative to an imaginary line aligned with y-z axes of the section to be milled, as depicted in Figure 4.8.

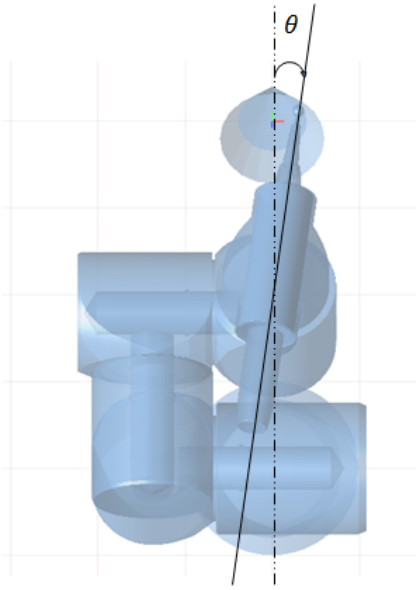

Figure 4.8: Representation of the *approach angle* θ

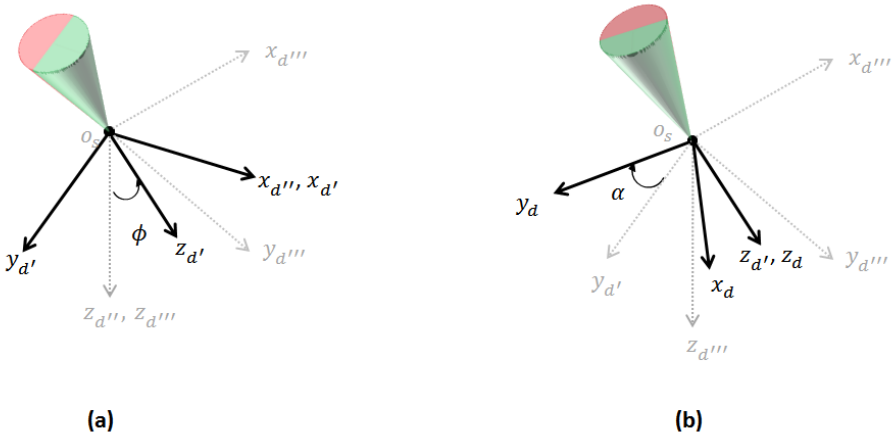

Figure 4.9: Representation of rotations - Plot (a) shows the second rotation by the *inclination angle* ϕ around $x_{d''}$ to obtain $o_{d'}x_{d'}y_{d'}z_{d'}$, while Plot (b) shows the third rotation by the *self-rotation angle* α around $z_{d'}$ to obtain the desired end-effector frame $o_d x_d y_d z_d$

Following the rotation by θ, the obtained frame is rotated by the *inclination angle* ϕ, with respect to its current x-axis $x_{d''}$ as seen in Figure 4.9(a). The *inclination angle* ϕ, is the angle that the end-effector frame makes with normal of the cone section plane. This relates to how the tool tip interacts with the surface that is milled, as depicted in Figure 4.10.

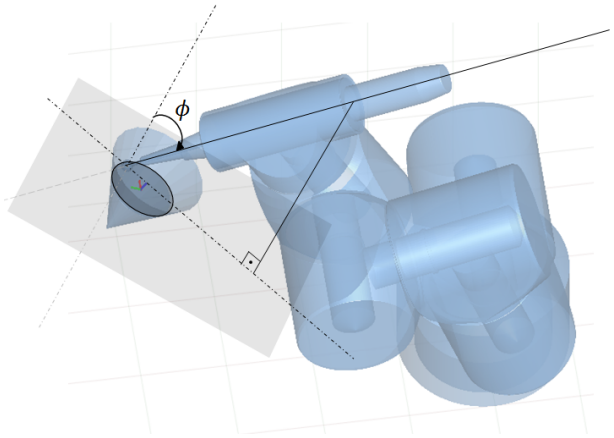

Figure 4.10: Representation of the *inclination angle* ϕ

Finally, the obtained frame is rotated about its current z-axis $z_{d'}$ by the *self-rotation angle* α, as seen in Figure 4.9(b). The *self-rotation angle* α, which is represented in Figure 4.11, is the rotation of the end-effector frame about its z-axis. It has no effect in task specification directly since its motion is equivalent to the motion of the drill, on the task side. It also has no effect in collision avoidance for the last link with the surrounding cone. However, the *self-rotation angle* α is important in setting an achievable task configuration for the robot. By incorporating the approach angle and the self-rotation angle together, the end-effector can direct to a point from a wider approach angle, with less displacement for the overall robot joints.

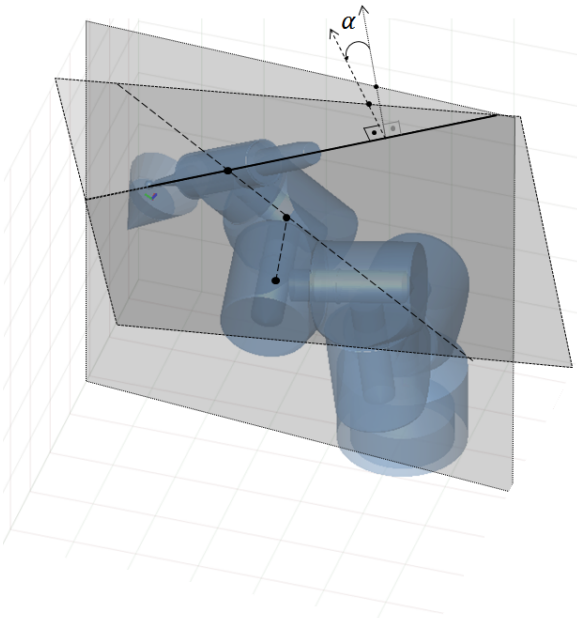

Figure 4.11: Representation of the *self-rotation angle* α, for a desired pose of the end effector frame

Choice of the parameters θ, ϕ and α

All three angular parameters are of equal importance to keep the robot clear of the cone edges and to be able to execute a task at desired locations. The *approach angle* θ is the angle formed between the y-axis of the section and the line obtained by projecting the tool axis onto the cone section plane. A small θ would mean that the tool reaches the desired area almost directly (seen from a top view), whereas a large θ would mean that the tool would reach the section from its far sides, which would also create extra kinematic challenges. Hence, it is desirable to keep θ within the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$.

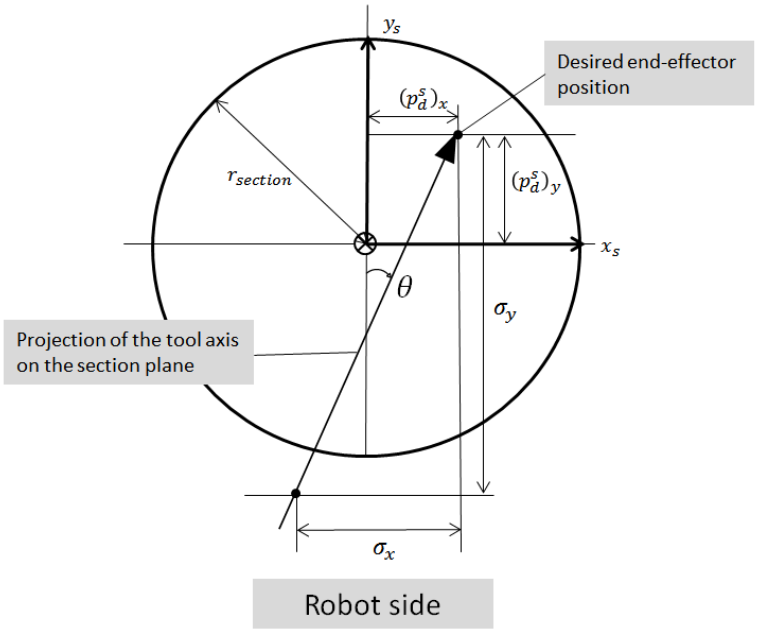

Figure 4.12: Illustration of cone section and tool axis projection onto it, where σ_x and σ_y are functions of the desired end-effector position

Since our aim is to avoid collision between the tool and side surfaces, we may plan the tool's approach angle based on the side it is closest to and its distance from the robot side. In other words, as the tool operates on a position closer to one side, its approach angle should keep the tool towards the other side such that the tool does not come close to the edges as the tool tip approaches there. This can be solved geometrically based on the position of the end-effector and the section parameters. A particular solution, with its visualization given in Figure 4.12, for determining a desired *approach angle* θ is to define a reference point, where the projection of the tool onto the section should intersect. The geometric solution for the approach angle θ is defined as

$$\theta = -\left(\text{atan2}\,(\ \sigma_y\ ,\ \sigma_x\ ) - \frac{\pi}{2}\right), \tag{4.4}$$

where the expression for the variables σ_x and σ_y is defined as

$$\begin{aligned} \sigma_x &= 2(p_d^s)_x + (p_d^s)_x \left(\frac{(p_d^s)_x}{r_s}\right)^2 + (p_d^s)_x \left(\frac{(p_d^s)_y - r_s}{2r_s}\right)^2 , \\ \sigma_y &= (p_d^s)_y + 2r_s \end{aligned} \tag{4.5}$$

and atan2 is the four quadrant arctangent function defined as

$$\text{atan2}\left(cos(\beta), sin(\beta)\right) = \beta \quad \text{if} \quad \beta \in [-\pi, \pi]. \tag{4.6}$$

Note that in (4.4), there is a minus sign, so that θ in the Figure 4.12 is computed, where its direction of rotation is opposite to the z_s axis (i.e. into the section plane).

In Figure 4.12, the projection line is drawn between the position of the end-effector with respect to the cone section frame, described by $((p_d^s)_x, (p_d^s)_y)$, and a virtual location, set by the variables σ_x and σ_y, on the *robot side* of the section. If that virtual point on the *robot side* is kept stationary, the tool's projection would always cross there while doing the side-to-side motion. However, there may be requirements for the inclination angle, which may relate to the milling task, and setting an inclination angle ϕ while setting a stationary point for the tool's axis to cross may overdefine the task and cause sharp movements. Hence, instead of keeping that point stationary, we choose to change it as a reference for the approach angle based on the quadratic relations described in (4.5). In Figure 4.13(a), the computed orientation trajectory trace of a task where the approach angle θ is kept constant at 0, is given, whereas in Figure 4.13(b), θ is computed according to (4.5). Note that the inclination angle ϕ and self-rotation angle α are kept constant as $\phi = \pi/2$ and $\alpha = 0$.

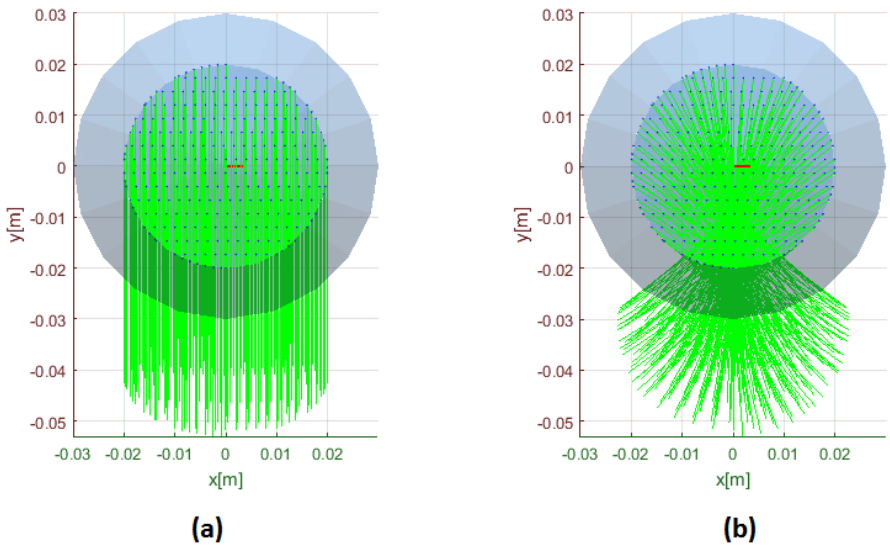

Figure 4.13: Illustrations of the choice of the approach angle θ, when θ is set to be 0 (a) and when it is chosen as in (4.5) (b), viewed from the top of the cone section

For the milling task, using the rotary tool with a large inclination angle ϕ, described in Figure 4.10, is more efficient when spherical tips are used, which is related to the feeding rate of the material (bone) to the rotating spherical tool tip. However, the inclination angle should have a limit since the milling may also take place inside a cavity and a large angle would result in collision of the rotary tool with the cavity edges. In future applications, specifications of a milling task may require certain orientation profiles, which is also discussed in Recommendations in Section 7.2. For now, there are no strict requirements on this. Hence we focus on keeping the rotary tool clear of the cone edges. To check if the choice of ϕ is collision free regarding the tool and the cone edges, a method to check cone edge clearance is presented later in this section.

Choices for the inclination angle include setting it to zero, keeping it constant, or changing it as a function of how close the end-effector gets to the part of the cone wall in proximity of the robot. In Figure 4.14, the inclination angle is presented from a side view to the cone section (robot assumed to be on left) and in (a), the case with $\phi = 0$ is presented, where the orientation trace is seen to be perpendicular to the section. In (b), ϕ is chosen to be constant as $\phi = \pi/8$, taking into account that the chosen value should be smaller than the angle between the section and cone wall. In (c), ϕ is chosen according to a particular solution as given in (4.7) with $\bar{\phi} = \pi/4$

$$\phi = \bar{\phi} \tanh(\frac{(p_d^s)_y + r_s}{0.5 r_s}), \tag{4.7}$$

where $\bar{\phi}$ is the nominal inclination angle. This definition is an example form for ϕ, which can be modified according to how the ear approximation is placed in the robot's workspace. This form basically results in an inclination angle around $\bar{\phi}$ at the side of the section away from the robot and continuously gets smaller (i.e., the tool axis tends to become orthogonal to the cone section) as the end-effector moves closer to the side of the robot. These are modifiable options for the inclination angle ϕ, which should be set according the specifications of a given task.

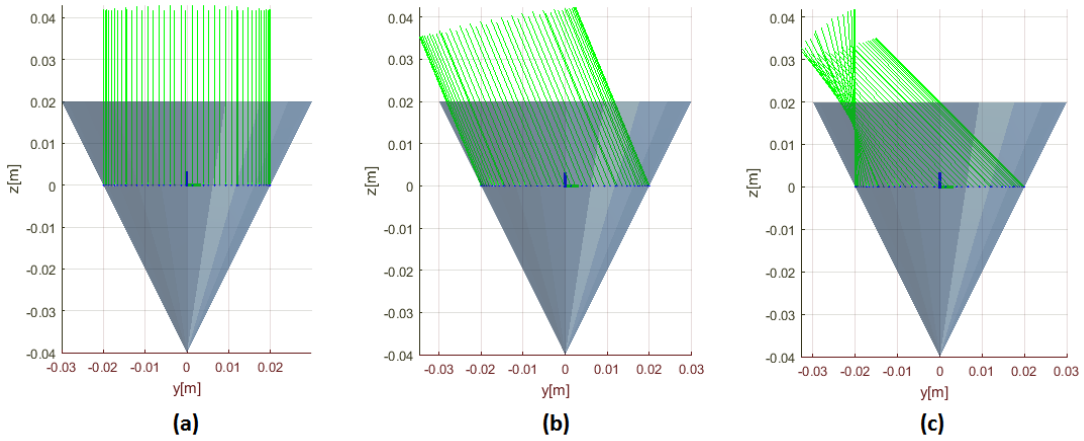

Figure 4.14: Illustrations of the choice of the inclination angle ϕ, when ϕ is set to be zero in (a), constant in (b) and when it is chosen according to (4.7) in (c)

Finally, the self-rotation angle α, which is visualized in Figure 4.11, is to be defined. Its importance is related to the robot parts before the prismatic joint. When α is zero, the position of the frames before the end-effector and the prismatic joint changes a lot to approach from different angles, whereas when it is self-rotated, this position change is reduced and there is less kinematic challenge for the robot and less amount of displacement necessary. The combination of the uses of θ and α should result in a *leaning* motion, as when a human wrist would do rather than moving the elbow all the way to achieve an approach angle. Hence, we want the self-rotation angle to rotate the end-effector frame, such that its supporting links are kept towards the direction of the

robot base. For these purposes, a suitable and simple choice for the self-rotation angle α is

$$\alpha = -\theta, \tag{4.8}$$

which reduces the positional change in the previous frames and counteracts what the approach angle θ imposes on the robot.

Cone Edge Clearance Check

A geometrical description of the cone edge clearance denoted as d_{cl} , which is the shortest distance between the cone edge (sides of the cavity opening) and the location o_c^0 where the tool axis crosses the opening surface, is shown in Figure 4.15.

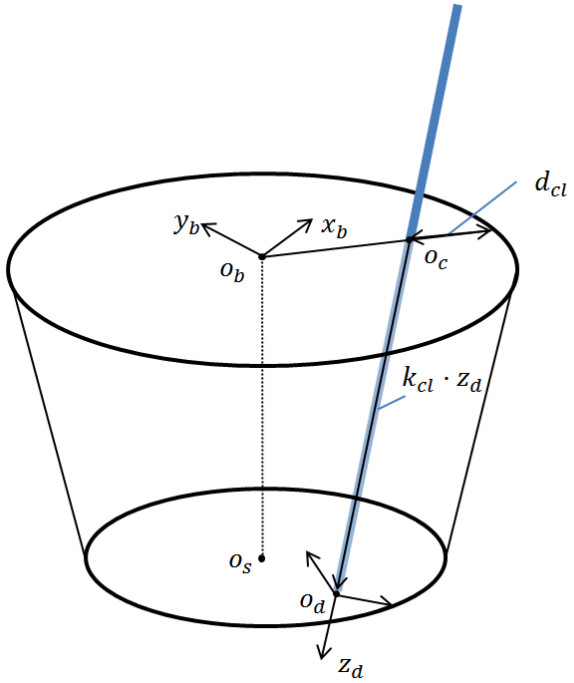

Figure 4.15: Cone edge clearance - Upper part of the conic cylinder is the cone opening, through which the tool (in blue) enters to reach cone section surface at the bottom

We know the end-effector frame's vertical coordinate vector z_d^0 and origin o_d^0 from the desired end-effector orientation and position from the previous section, the cone section frame origin o_s^0, the cone bottom frame (cavity opening) origin o_b^0, and its coordinate frame vectors x_b^0 and y_b^0 from the definition of the cone. To compute the cone edge clearance d_{cl}, first we need to find the location o_c^0 where the tool crosses the cavity opening surface. Since o_c is the intersection between the cavity opening surface which is spanned by the vectors x_b and y_b, and the tool axis which is spanned by z_d, the geometric problem of finding o_c^0 can be formulated as

$$o_d^0 = o_b^0 + mx_b^0 + ny_b^0 + k_{cl}z_d^0, \tag{4.9}$$

where the multipliers m and n form the pair (m, n), which is the (x, y) position of where the tool crosses the opening with respect to the frame $o_b x_b y_b z_b$ (i.e. $((o_c^b)_x, (o_c^b)_y) = (m, n)$). By regrouping the terms, the linear relation in (4.9) can be expressed as

$$o_d^0 - o_b^0 = \begin{bmatrix} x_b^0 & y_b^0 & z_d^0 \end{bmatrix} \begin{bmatrix} m \\ n \\ k_{cl} \end{bmatrix} \tag{4.10}$$

Hence, we can calculate m, n and k_{cl} from

$$\begin{bmatrix} m \\ n \\ k_{cl} \end{bmatrix} = \begin{bmatrix} x_b^0 & y_b^0 & z_d^0 \end{bmatrix}^{-1} (o_d^0 - o_b^0). \tag{4.11}$$

where it should be noted that the matrix $\begin{bmatrix} x_b^0 & y_b^0 & z_d^0 \end{bmatrix}$ is not a rotation matrix; it is obtained from the linear relation in (4.9). The pair (m, n) is actually sufficient to find where the tool axis crosses the cavity opening, but the sign of k_{cl} indicates the direction of the crossing. Hence o_c^0 can be expressed in two ways as

$$o_c^0 = m x_b^0 + n y_b^0, \tag{4.12}$$

or

$$o_c^0 = o_d^0 - k_{cl} z_d^0. \tag{4.13}$$

When k_{cl} is positive, the tool exits the cavity opening in the direction $-z_d$ as it is the case in Figure 4.15. When $\begin{bmatrix} x_b^0 & y_b^0 & z_d^0 \end{bmatrix}$ is invertible, which means that the axis z_d crosses the plane that the cone opening is on, the multiplier k_{cl} can be computed as in (4.11). When it is singular or when k_{cl} is found to be negative, it means that the tool never crosses that plane (either parallel to the plane or in an opposite direction) and the desired orientation must be changed.

After computing m,n and k_{cl} with (4.11), we are left with the problem of finding d_{cl}, the cone edge clearance of the tool, simply as

$$d_{cl} = r_s - |o_b^0, o_c^0|, \tag{4.14}$$

which is equivalent to

$$d_{cl} = r_s - \sqrt{m^2 + n^2}, \tag{4.15}$$

where r_s denotes the radius of the cone section and $|o_b^0, o_c^0|$ denote the distance between positions o_b^0 and $o_c^0 = o_d^0 - k_{cl} z_d^0$.

With the computed cone edge clearance d_{cl}, the generated end-effector position and orientation trajectory can be inspected to verify that the tool axis intersects the plane where the cone opening lies ($\begin{bmatrix} x_b^0 & y_b^0 & z_d^0 \end{bmatrix}$ is invertible), exits in the correct direction ($k_{cl} > 0$) and from inside the opening surface ($d_{cl} > 0$).

4.1.4 Constrained Use of Prismatic Joint for Spherical Clearance

After generating the desired position and orientation trajectories for the end-effector frame, we need to consider the frame of the prismatic joint's base; because its distance from the working area of the end-effector is critical to keep the supporting robot link clear from the cone volume, as described in Figure 4.16. Since this clearance is directly defined by the prismatic joint's actuation, we can generate a constraint for it based on the previously generated desired position and orientation profiles. We are free to define the prismatic joint displacement without changing the desired end-effector position and orientation, since there is an extra redundant degree of freedom which can be utilized. By setting this constraint, the clearance problem is decoupled from the end-effector task to the rest of the robot joints, making use of self motions of the robot, which does not change the end-effector position and orientation. As a result of this constraint, the overall task dimension will be augmented with an extra degree of freedom, which is decoupled from the end-effector task, while the redundancy is resolved since the dimensions of the overall task and robot DOF will be equal. This is also favourable for the inverse kinematics solution specific for this robot, which will be explained in Section 5.1.

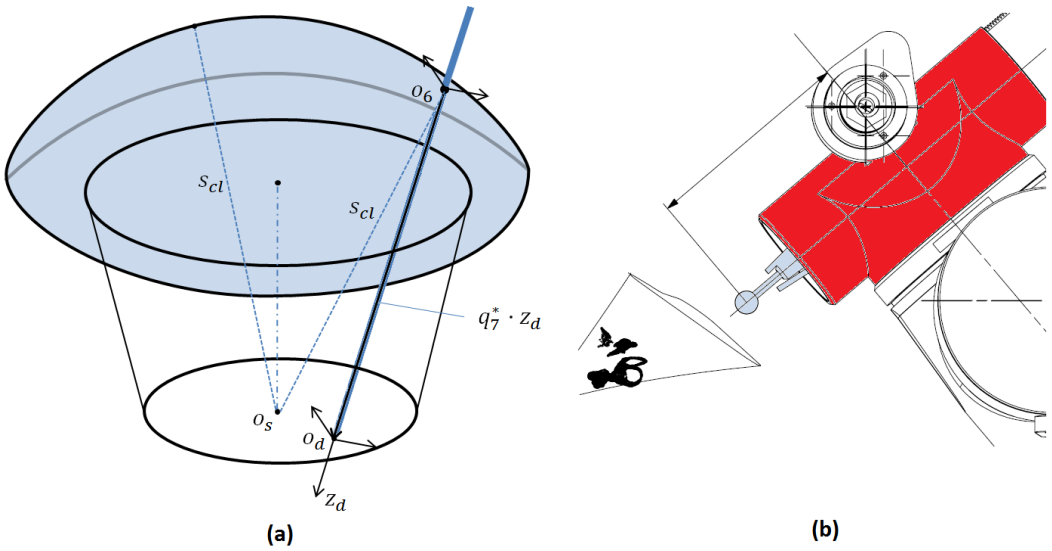

Figure 4.16: Description of the spherical clearance volume (blue dome) around the ear volume section in (a) and prismatic joint's actuated part (blue) and base (red), which is not allowed inside the spherical volume, in (b)

The approach is to set a spherical clearance surface around the cone. The sphere's center is the cone section's origin o_s^0. Hence, in any position or orientation along the section, the length of the vector from the end-effector position o_d^0 along its axis z_d^0 in the negative direction is calculated such that the resultant position of the prismatic joint's base frame o_6^0 has a distance of a certain clearance length s_{cl} to the origin of the cone section frame o_s^0.

Let o_d^0, z_d^0, q_7 and o_s^0 denote the desired position of the end effector, the vertical coordinate vector of the end-effector frame (o_d, x_d, y_d, z_d), the prismatic joint displacement and the origin of the cone section frame, respectively. We define the clearance constraint as

$$s_{cl} = |o_s^0, o_6^0|, \tag{4.16}$$

where $o_6^0 = o_d^0 - q_7 z_d^0$ denote the position of the prismatic joint's supporting link's frame origin. The right hand side of (4.16) is the norm of the distance between two points which is

$$s_{cl} = \sqrt{(o_d^0 - q_7 z_d^0 - o_s^0)^T (o_d^0 - q_7 z_d^0 - o_s^0)}, \tag{4.17}$$

$$s_{cl} = \sqrt{(o_d^0 - o_s^0)^T (o_d^0 - o_s^0) - 2q_7 (o_d^0 - o_s^0)^T z_d^0 + (q_7)^2 (z_d^0)^T z_d^0}, \tag{4.18}$$

$$(z_d^0)^T z_d^0 (q_7)^2 - 2(o_d^0 - o_s^0)^T z_d^0 q_7 + (o_d^0 - o_s^0)^T (o_d^0 - o_s^0) - s_{cl}^2 = 0. \tag{4.19}$$

Clearly, there are two solutions for q_7 in (4.19) since o_d will be inside the spherical clearance surface defined by the distance s_{cl}, which can be set in terms of the cone's top radius, section depth and section radius. Based on the direction of the desired end-effector frame's z-axis z_d, the desired displacement of the prismatic joint, $(q_7)_d$, that satisfies the spherical clearance constraint can be written as

$$(q_7)_d = \begin{array}{l} \dfrac{\sqrt{4(z_d^0)^T (o_d^0 - o_s^0)(o_d^0 - o_s^0)^T z_d^0 - 4(z_d^0)^T z_d^0 \left((o_d^0 - o_s^0)^T (o_d^0 - o_s^0) - s_{cl}^2\right)}}{2(z_d^0)^T z_d^0} \\ + \dfrac{2(o_d^0 - o_s^0)^T z_d^0}{2(z_d^0)^T z_d^0} \end{array}. \tag{4.20}$$

Also, s_{cl} is set to be

$$s_{cl} = \sqrt{r_b^2 + \bar{h}_s^2} + c_{cl}, \tag{4.21}$$

where r_b is the radius of the cone bottom (cavity opening), $\bar{h}_s$ is the vertical distance between the cone section and the cone opening and c_{cl} is the additional clearance since $\sqrt{r_b^2 + \bar{h}_s^2}$ provides the distance from the origin of the cone section to any point on the circle of the edge of the cone opening. Hence, c_{cl} should be set to be a positive scalar.

This constraint generates a spherical surface around the cone section's origin, not to be penetrated by the prismatic joint's base frame, as seen in Figure 4.17; where it is shown that with different orientations and positions, the prismatic joint's base, the trace of which is depicted in black, is kept at a spherical surface clear from the cone section origin. Note that only the distance between prismatic joint's base frame and cone section's frame is kept constant; the prismatic joint's actuation is still variable.

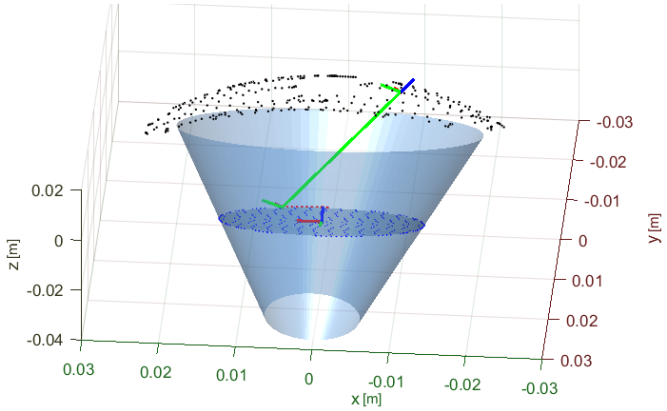

Figure 4.17: Constant spherical clearance of the 6th frame with its position trace (black)

4.2 Results

To test the proposed method of task specification, Task 1 (see Figure 4.19) and Task 2 (see Figure 4.20) are generated with different end-effector position and orientation trajectories, which are plotted in Figure 4.21 and in Figure 4.22, respectively. For both tasks, the position of the section of the approximation cone is $p_s = (-0.15, 0, 0.15)$ and the orientation of the cone section is described by the rotation matrix $R_s = R_{y,7\pi/18}R_{z,\pi/2}$, as seen in Figure 4.18 with the robot.

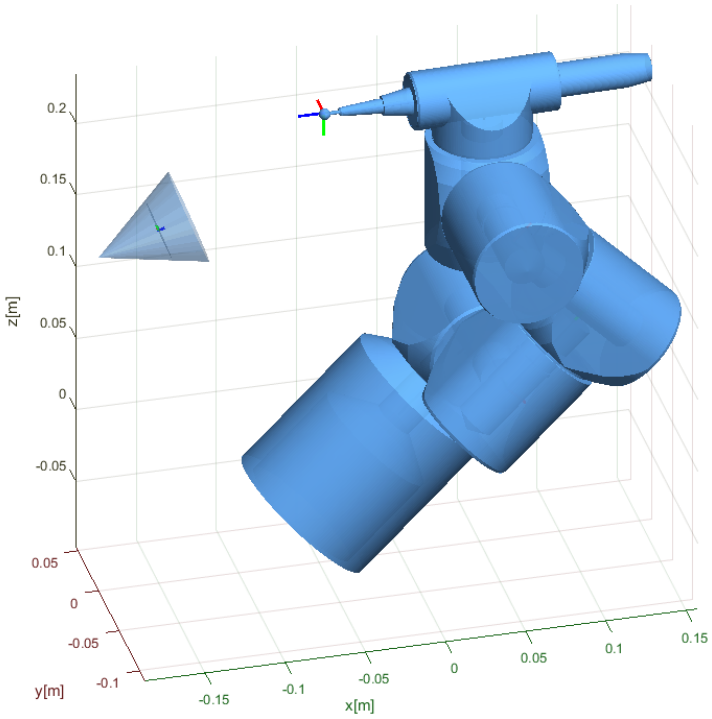

Figure 4.18: Robot with ear volume approximation cone

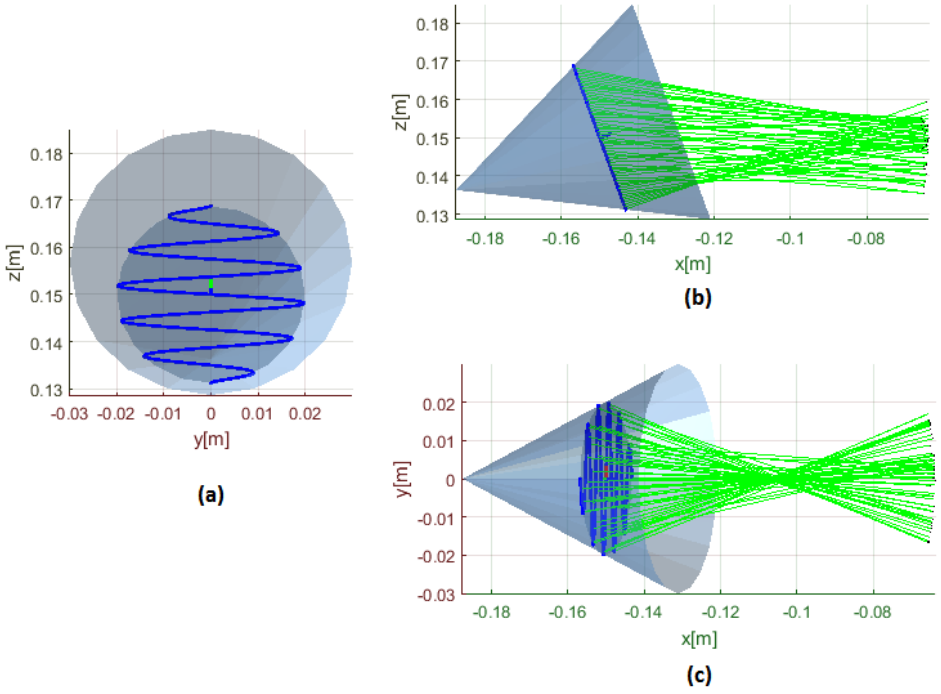

Figure 4.19: Task 1 - The desired end-effector position and orientation trajectories on cone section plane viewed from front in (a), from side in (b) and from top in (c)

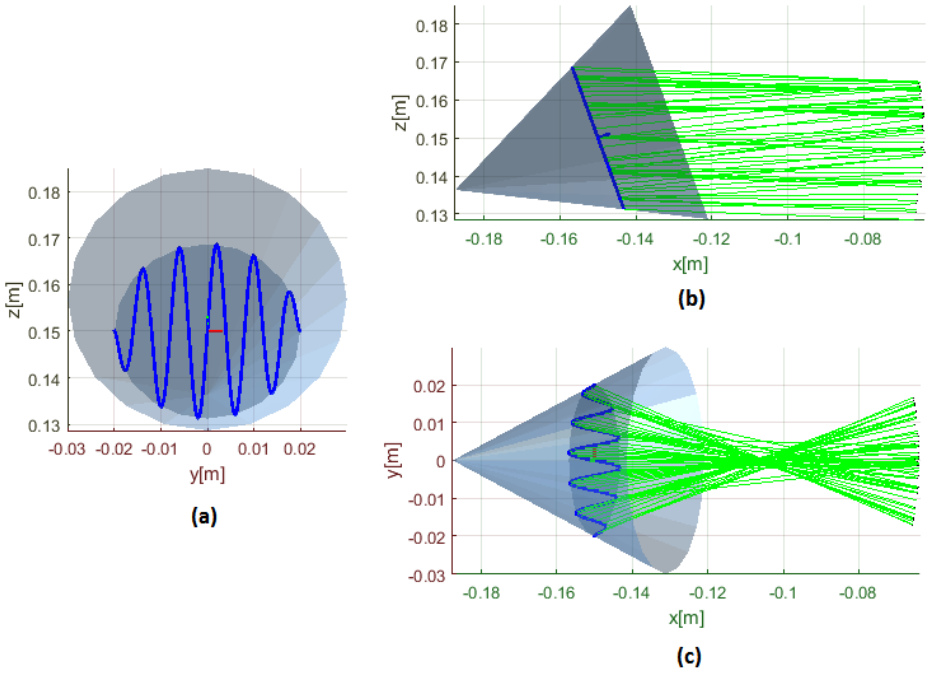

Figure 4.20: Task 2 - The desired end-effector position and orientation trajectories on cone section plane viewed from front in (a), from side in (b) and from top in (c)

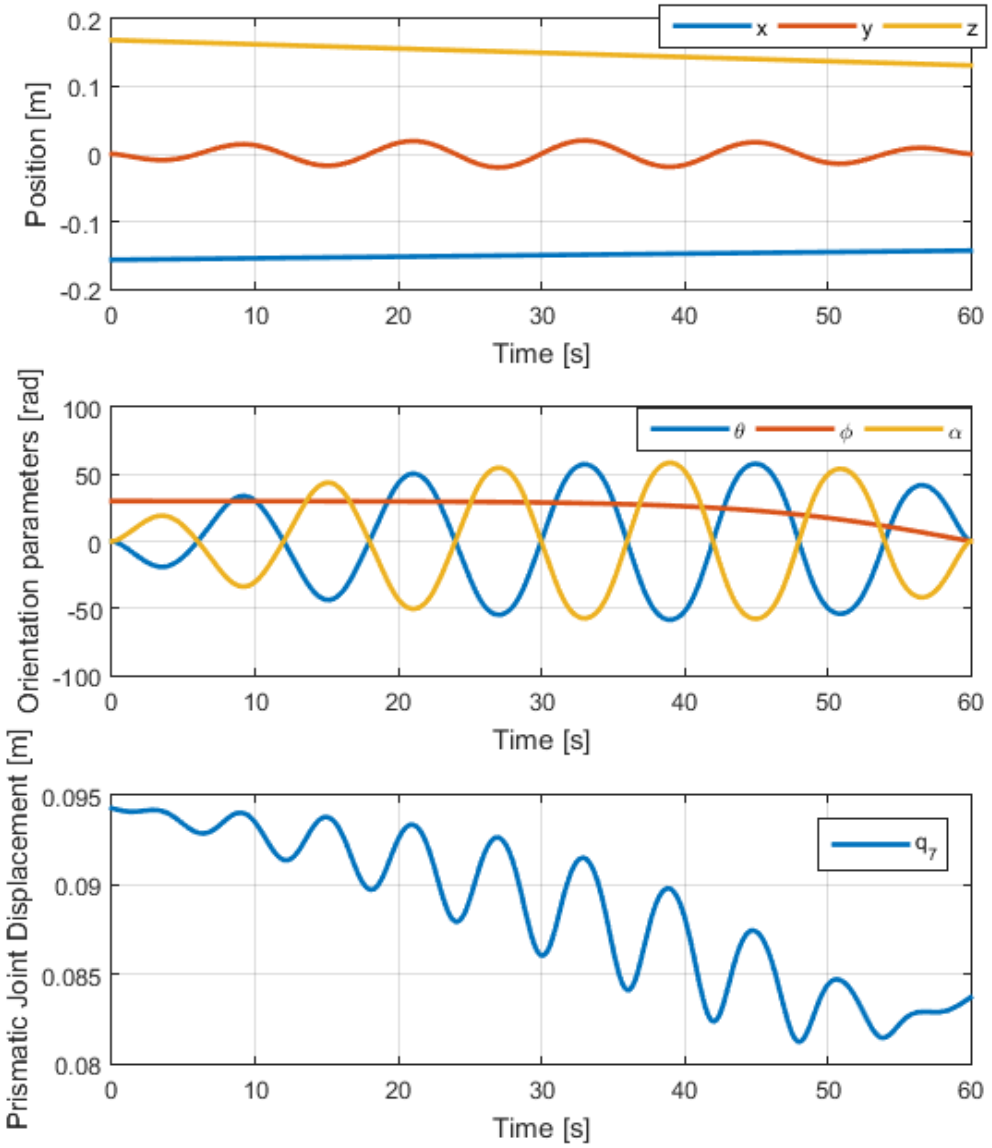

Figure 4.21: Task 1 - The desired end-effector position trajectory plot (top), the desired end-effector orientation trajectory plot (middle) and the prismatic joint's desired constraint displacement trajectory plot (bottom) vs. time

For Task 1, a continuous desired end-effector position trajectory, which is plotted in Figure 4.21 (top), is chosen. The desired side-to-side motion of the end-effector, as seen in Figure 4.19, has the orientation trajectory defined by the angular parameters θ (approach), ϕ (inclination) and α (self-rotation) which are plotted in Figure 4.21 (middle). Note that the inclination angle ϕ is defined as given in (4.7) with the nominal inclination angle $\bar{\phi} = \pi/6$. Finally, the constraint task on the motion of the prismatic joint, which ensures a spherical clearance around the cone, is plotted in Figure 4.21(bottom).

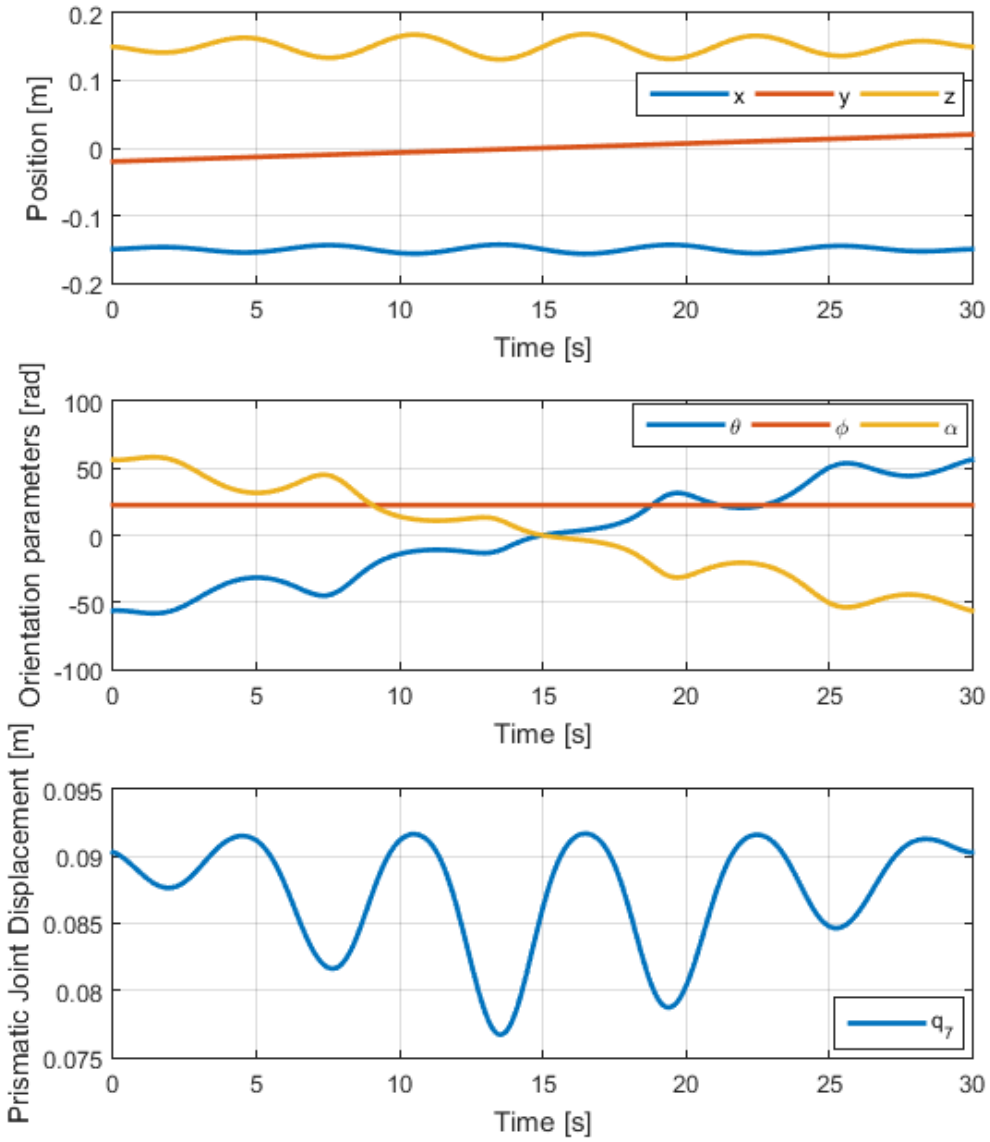

Figure 4.22: Task 2 - The desired end-effector position trajectory plot (top), the desired end-effector orientation trajectory plot (middle) and the prismatic joint's desired constraint displacement trajectory plot (bottom) vs. time

For Task 2, a continuous desired end-effector position trajectory, which is plotted in Figure 4.22 (top), is chosen, which results in an up-and-down motion as seen in Figure 4.20. The orientation trajectory profile, which is described by the parameters θ, ϕ and α, is plotted in Figure 4.22 (middle). Note that the inclination angle ϕ for Task 2 is defined to be constant as $\phi = \pi/8$. Finally, the constraint task on the prismatic joint for spherical clearance, is plotted in Figure 4.22 (bottom).

4.3 Discussion

The proposed task specification method resulted in position and orientation trajectories, which span a specified layer on an approximated ear cone, while the rotary tool is kept clear from the edges of the cavity it enters. Additionally, the constraint task,

which is specified on the last joint's motion for spherical clearance around the ear area, has been used.

With the addition of the constraint task, it is seen that while the end-effector frame is at a desired position and orientation, the prismatic joint is at a specified value, which keeps the supporting link of the end-effector away from the ear cone with a set clearance. It should be noted that both the constraint task on the prismatic joint and the orientation task variables depend on the position task. Hence, it is critical to define the desired end-effector position trajectory continuously.

Since the desired translational and angular velocities of the end-effector are to be used in the inverse kinematics part in Chapter 5, it is better to define them at task trajectory generation phase rather than using numerical differentiation during inverse kinematics simulations (see Section 5.2). In the presented results, the translational velocity profile was defined as a function of time, whereas the angular velocity profile was obtained through an orientation error description in axis-angle representation. Hence, it should be noted that the angular velocity profile depends on small simulation step time during task trajectory generation and small angular difference between each consecutive desired end-effector orientations. The rate of change in the desired orientation depends on the position of the end-effector with respect to the cone section. Therefore, the angular velocity profile is also dependent on the translational velocity profile of the end-effector. In addition to spanning the whole section during milling, the speed of this operation is also important. By defining both translational velocity and acceleration profiles, limits can be imposed on the operation of the robot regarding actuator's limits, milling tool's feed rate and other phases.

Chapter 5

Inverse Kinematics Solution and Results

In this chapter, firstly, a solution to the inverse kinematics problem is described in Section 5.1. Secondly, the joint trajectory generation results using the inverse kinematics algorithm are presented in Section 5.2. Finally, a discussion on the obtained results is given in Section 5.3.

5.1 Differential Inverse Kinematics Algorithm with Augmented Task Space

5.1.1 Overview

In this section, the inverse kinematics algorithm for a seven degrees of freedom serial manipulator is presented. Being inherently redundant as a result of having seven degrees of freedom, the surgical robot RoBoSculpt's kinematic model does not allow finding a geometric solution to the inverse kinematic problem in closed-form. Even if the prismatic joint would be fixed, the remaining 6R kinematic chain does not allow decomposition of the inverse kinematics problem into separate position and orientation parts. As a result of the indeterministic nature of the problem, the inverse kinematics of the seven DOFs kinematic chain requires a method of choosing a desired solution among other possible solutions, in addition to finding a joint configuration that is mapped to a desired task via the forward kinematics.

First, it is necessary to be able to find the set of possible solutions. Since the forward kinematics is highly non-linear, there is no general method to solve for the joint variables which is also not specific for a given kinematic model. Fortunately, the mapping between the joint and task velocity variables is linear and a standard option is to invert, if possible, the first-order differential kinematics to solve for the velocities of the joint variables, which then can be integrated over time to give the desired solution to the inverse kinematics problem. The availability and the reliability of the solutions at the velocity level can also be investigated easily, concerning the existence of multiple solutions and kinematic singularities. Secondly, since the robot is inherently redundant, a way of choosing the set of joint configurations from multiple options must be devised. For this purpose, in addition to the task of attaining a desired end-effector pose, which includes the position and orientation tasks described by a total of six DOFs, an additional constraint task of one DOF is introduced. The description of

this additional task is given in Section 4.1.4. By having an additional task, the dimension of the task space is augmented such that it is equal to the dimension of the joint space and the solution to the linear mapping of the first-order differential of the joint and task space variables is unique, unless a kinematic singularity is reached which disallows end-effector motion in certain directions. Regarding the inverse kinematics algorithm, it is assumed that no kinematic singularities will be reached. However, singularities are present in the workspace of the robot, and a method to analyse the manipulability of the end-effector throughout the given tasks will be explained in Chapter 6 (see Section 6.1), which can help in the decision of where and how, in the workspace of the robot, the tasks need to be done.

The algorithm adopts the representation of orientation error from [21], which makes the computation of the geometric Jacobian sufficient to find the joint velocity profile for a given trajectory in the task-space. Since the computation of the analytical Jacobian, which has a higher order of complexity than the geometric Jacobian, is avoided, the solution is computationally more efficient. Additionally, since the axis-angle representation is used to find the orientation error, the algorithm is free of representational singularities.

The inverse differential kinematics algorithm for a system with seven DOFs is obtained by augmenting the task-space with the addition of a constrained functional task, which is explained in Section 4.1.4. Hence, the redundant degree of freedom is utilized and the Jacobian is augmented to a square matrix which is invertible unless the manipulator takes a configuration that results in a kinematic singularity. A method to analyse the manipulability of the end-effector beforehand, for given tasks or selected regions in its workspace or configuration space will be introduced in Section 6.1.

5.1.2 First-order Differential Kinematics

Recall the forward kinematics equation, relating the joint configuration q to the task space variables x via a non-linear vector function $f(q)$ as

$$x = f(q), \tag{5.1}$$

and its first-order differential

$$\dot{x} = J(q)\dot{q}, \tag{5.2}$$

which is the linear mapping between the joint and task space velocities via the Jacobian matrix $J(q)$. The solution to the above expression, assuming that the Jacobian $J(q)$ is a square and invertible matrix, is

$$\dot{q} = J^{-1}(q)\dot{x}. \tag{5.3}$$

To find the joint velocity profile $q(t)$, the above expression can be integrated over time to give

$$q(t) = \int_0^t J^{-1}(q(\tau))\dot{x}_d(\tau)d\tau + q_0, \tag{5.4}$$

where $\dot{x}_d(t)$ is the time derivative of the desired task trajectory $x_d(t)$. If the initial joint configuration $q_0 = q(t)$ is known, the joint trajectory $q(t)$ that moves the end effector from its initial pose $x(0)$ along the desired trajectory $x_d(t)$ can be found from

(5.4), theoretically. However, in practice, to integrate (5.3), numerical methods such as first-order Euler are used in discrete-time, as

$$q(t_{k+1}) = q(t_k) + J^{-1}(q(t_k))\dot{x}_d(t_k)\Delta t, \tag{5.5}$$

where $\dot{x}_d(t_k)$ can be determined either by numerical differentiation of $x_d(t_k)$ at t_k or by defining $x_d(t_k)$ to be continuously differentiable and by obtaining a functional expression for its time derivative (see (5.34)). The solution in (5.5) suffers from numerical errors where the resulting end-effector task variables $x(t)$ drift away from the desired task trajectory $x_d(t)$. The resulting numerical errors can be avoided by defining an error in the task space variables, since the current state of the task space variable at each time step is known, and the numerical integration loop can be closed by using the task space error as feedback. The task space error, which is the error between the desired task variables $x_d(t)$ and the attained task variables $x(t) = f(q(t))$ can be expressed as (omitting the dependency on time variable t for simplicity)

$$e = x_d - x, \tag{5.6}$$

and its time derivative

$$\dot{e} = \dot{x}_d - \dot{x}, \tag{5.7}$$

where $\dot{x}$ can be substituted from the first-order differential kinematics (5.2) as

$$J(q)\dot{q} = \dot{x}_d - \dot{e}. \tag{5.8}$$

Based on this scheme, the solution to the first-order differential inverse kinematics becomes, assuming that the Jacobian is square and invertible

$$\dot{q} = J^{-1}(q)(\dot{x}_d - \dot{e}), \tag{5.9}$$

and by choosing the error dynamics as

$$\dot{e} = -Ke, \tag{5.10}$$

where K is a diagonal, positive definite matrix, it leads to the solution of both the first-order differential inverse kinematics and trajectory tracking, as

$$\dot{q} = J^{-1}(q)(\dot{x}_d + Ke), \tag{5.11}$$

where the error in the task space variables converges to zero with a rate depending on the eigenvalues of the gain matrix K. The error variables for end-effector position and orientation and the constraint task are to be defined, along with desired end-effector velocity and augmentation of the Jacobian, which are explained next.

5.1.3 End-effector Position Error

The description of the error for the position part is straightforward and is defined as

$$e_p = p_d^0 - p_{ee}^0, \tag{5.12}$$

where p_d^0 and p_{ee}^0 denote the desired position and the current position of the end effector, respectively, with respect to the reference frame. The position error's time derivative is

$$\dot{e}_p = \dot{p}_{d,0}^0 - \dot{p}_{ee,0}^0. \tag{5.13}$$

5.1.4 End-effector Orientation Error

The orientation error however, depends on the representation used to describe the orientation of the end effector frame. If the Euler angles representation is chosen, the orientation error can be defined as

$$e_o = \phi_d - \phi_{ee}, \tag{5.14}$$

where ϕ_{ee} denote the set of Euler angles, which describe the orientation of the end-effector frame. The rotation matrix, which describes the orientation, can be obtained directly from the forward kinematics. However, the inverse operation of finding the Euler angles from that orientation or the joint configuration is computationally expensive as it requires the use of inverse trigonometric formulas. It is also subject to representational singularities. Computing the orientation error in real time requires more effort, in addition to that for the computation of the analytical Jacobian matrix. Fortunately, another representation of the orientation error (see, e.g., [21]) can be set such that the error in orientation can be obtained using only the rotational matrices of the desired and current end-effector orientations. The orientation error is obtained using the axis-angle representation, which is free of representational singularities.

Let the rotation matrix describing the desired orientation be

$$R_d^0 = [x_d^0\ y_d^0\ z_d^0], \tag{5.15}$$

and the current orientation of the end effector be

$$R_{ee}^0(q) = [x_{ee}^0(q)\ y_{ee}^0(q)\ z_{ee}^0(q)]. \tag{5.16}$$

The error between the two frames' orientations which are described by the above rotational matrices, can be defined as

$$e_o = r\sin(\vartheta), \tag{5.17}$$

where r and ϑ are the axis and angle variables, respectively, that describe the rotation needed to align both frames as (omitting the dependency of R_{ee}^0 on q)

$$R(\vartheta, r) = R_d^0 (R_{ee}^0)^T, \tag{5.18}$$

which means, if R_{ee}^0 is rotated by $R(\vartheta, r)$ (i.e. by the angle of ϑ around the axis r), R_d^0 is obtained.

An approximation of e_o can be found as (see Appendix A for details)

$$e_o = -\frac{1}{2}\left(x_{ee}^0 \times x_d^0 + y_{ee}^0 \times y_d^0 + z_{ee}^0 \times z_d^0\right), \tag{5.19}$$

and its time derivative is

$$\dot{e}_o = L^T \omega_{d,0}^0 - L\omega_{ee,0}^0\ , \tag{5.20}$$

where

$$L = -\frac{1}{2}\left(S(x_d^0)S(x_{ee}^0) + S(y_d^0)S(y_{ee}^0) + S(z_d^0)S(z_{ee}^0)\right), \tag{5.21}$$

and $\omega_{d,0}^0$ and $\omega_{ee,0}^0$ denote the desired and current end-effector frames' angular velocities with respect to the reference frame, represented in the coordinates of the reference frame, respectively.

5.1.5 Desired End-effector Angular Velocity

The angular velocity $\omega^0_{ee,0}$ can be obtained by using the geometric Jacobian, but the desired angular velocity $\omega^0_{d,0}$ is not that straightforward to compute. Since the desired orientation of the end-effector is described by the set of angles θ, α and ϕ (see Section 4.1.3) which is equivalent to an Euler-angle representation, the desired angular velocity corresponding to tracking these angular parameters cannot be obtained by direct differentiation. However, since the rotation matrices describing the desired orientation at each time step are obtained using the parameters θ,α and ϕ, Rodrigues' formula (A.11) can be exploited similarly to the derivation of the orientation error (5.19) along with a small angle approximation. The desired angular velocity corresponding to the rotation between consecutive orientations described by rotational matrices in a given time step is calculated by the following approach.

Recalling the derivation in Appendix A, let the rotation matrix $R(r,\theta)$ describe the rotation which aligns the rotational matrix describing the desired orientation R_d at time t_k with the one at time t_{k+1} as

$$R(r,\theta) = R^0_d(t_{k+1})(R^0_d(t_k))^T, \tag{5.22}$$

where r is the axis of rotation and θ is the angle of rotation. The duration of this rotation is $\Delta t = t_{k+1} - t_k$. Hence the angular velocity is

$$\omega^0_{d,0} = \frac{\theta}{\Delta t} r. \tag{5.23}$$

Substituting the expression for r from (A.21) in Appendix A into (5.23), we get

$$\omega^0_{d,0} = \frac{\theta}{2\Delta t \sin\theta} \begin{bmatrix} c_d(t_{k+1}) \cdot b_d(t_k) - b_d(t_{k+1}) \cdot c_d(t_k) \\ a_d(t_{k+1}) \cdot c_d(t_k) - c_d(t_{k+1}) \cdot a_d(t_k) \\ b_d(t_{k+1}) \cdot a_d(t_k) - a_d(t_{k+1}) \cdot b_d(t_k) \end{bmatrix}, \tag{5.24}$$

where $a_d(t_i)$, $b_d(t_i)$ and $c_d(t_i)$ denote the row vectors of $R^0_d(t_i)$. By using the small angle approximation $\sin\theta \approx \theta$, since the angle of rotation between desired orientations at consecutive time steps is assumed to be small, the desired angular velocity is obtained as

$$\omega^0_{d,0} = \frac{1}{2\Delta t} \begin{bmatrix} c_d(t_{k+1}) \cdot b_d(t_k) - b_d(t_{k+1}) \cdot c_d(t_k) \\ a_d(t_{k+1}) \cdot c_d(t_k) - c_d(t_{k+1}) \cdot a_d(t_k) \\ b_d(t_{k+1}) \cdot a_d(t_k) - a_d(t_{k+1}) \cdot b_d(t_k) \end{bmatrix}, \tag{5.25}$$

or, in terms of the column vectors $x_d(t_i)$, $y_d(t_i)$ and $z_d(t_i)$ of $R^0_d(t_i)$, the above expression is equivalent to (see Appendix A, (A.23) and (A.27))

$$\omega^0_{d,0} = \frac{1}{2\Delta t} \left(x_d(t_k) \times x_d(t_{k+1}) + y_d(t_k) \times y_d(t_{k+1}) + z_d(t_k) \times z_d(t_{k+1}) \right). \tag{5.26}$$

5.1.6 Additional Constraint Task Error

The description of the error in the additional constraint task for the displacement of the prismatic joint is defined as

$$e_{pris} = (q_7)_d - q_7, \tag{5.27}$$

where $(q_7)_d$, which is defined in (4.20), denotes the desired displacement of the prismatic joint and q_7 denotes the current displacement of the prismatic joint. The time derivative of the displacement error is obtained as

$$\dot{e}_{pris} = (\dot{q}_7)_d - \dot{q}_7. \tag{5.28}$$

5.1.7 Overall Task-Space Error

The overall error in a given task is obtained by combining (5.12), (5.19) and (5.27) as

$$e = \begin{bmatrix} e_p \\ e_o \\ e_{pris} \end{bmatrix} = \begin{bmatrix} p_d^0 - p_{ee}^0 \\ -\frac{1}{2}\left(n_{ee} \times s_d + s_{ee} \times n_d + a_{ee} \times a_d\right) \\ (q_7)_d - q_7 \end{bmatrix}, \tag{5.29}$$

and its first-order differential as

$$\dot{e} = \begin{bmatrix} \dot{e}_p \\ \dot{e}_o \\ \dot{e}_{pris} \end{bmatrix} = \begin{bmatrix} \dot{p}_{d,0}^0 - \dot{p}_{ee,0}^0 \\ L^T\omega_{d,0}^0 - L\omega_{ee,0}^0 \\ (\dot{q}_7)_d - \dot{q}_7 \end{bmatrix} = \begin{bmatrix} \dot{p}_{d,0}^0 \\ L^T\omega_{d,0}^0 \\ (\dot{q}_7)_d \end{bmatrix} - \begin{bmatrix} \dot{p}_{ee,0}^0 \\ L\omega_{ee,0}^0 \\ \dot{q}_7 \end{bmatrix}. \tag{5.30}$$

5.1.8 Augmented Jacobian and the IK Algorithm

To include the additional task of the prismatic joint for the spherical clearance, as explained in Section 4.1.4 , the geometric Jacobian matrix $J(q)$ is augmented with the Jacobian of the prismatic joint task which is

$$J_{pris} = [0\ 0\ 0\ 0\ 0\ 0\ 1], \tag{5.31}$$

and the resulting augmented Jacobian is

$$J_{aug} = \begin{bmatrix} J_p \\ J_o \\ J_{pris} \end{bmatrix} \in \mathbb{R}^{7\times 7}, \tag{5.32}$$

which relates the joint space displacement and velocity to the task space velocity of the manipulator as

$$\dot{x} = v = \begin{bmatrix} v_{ee,0}^0 \\ v_{pris,6}^6 \end{bmatrix} = \begin{bmatrix} \dot{p}_{ee,0}^0 \\ \omega_{ee,0}^0 \\ \dot{q}_7 \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_o(q) \\ J_{pris} \end{bmatrix} \dot{q} = J_{aug}(q)\dot{q}, \tag{5.33}$$

where $v_{ee,0}^0$ denotes the spatial velocity vector containing the translational velocity $\dot{p}_{ee,0}^0$ and the angular velocity $\omega_{ee,0}^0$ of the end-effector frame with respect to the reference frame and $v_{pris,6}^6$ denotes the scalar velocity of the prismatic joint's displacement.

Additionally, the set of desired task variables can be brought together as

$$\dot{x}_d = \begin{bmatrix} \dot{p}_{d,0}^0 \\ \omega_{d,0}^0 \\ (\dot{q}_7)_d \end{bmatrix}, \tag{5.34}$$

where the desired end-effector translational velocity $\dot{p}_{d,0}^0$ is obtained from the functional derivative of the desired end-effector position trajectory described in Section 4.1.2, the desired end-effector angular velocity $\omega_{d,0}^0$ is obtained as described in Section 5.1.5 and the desired velocity of the displacement of the prismatic joint $(\dot{q}_7)_d$ is obtained from numerical differentiation of $(q_7)_d$ which is described in Section 4.1.4.

Substituting (5.33) into (5.30), the time derivative of the task error can be written as

$$\dot{e} = \begin{bmatrix} \dot{e}_p \\ \dot{e}_o \\ \dot{e}_{pris} \end{bmatrix} = \begin{bmatrix} \dot{p}^0_{d,0} - J_p(q)\dot{q} \\ L^T\omega^0_{d,0} - LJ_o(q)\dot{q} \\ (\dot{q}_7)_d - \dot{q}_7 \end{bmatrix} = \begin{bmatrix} \dot{p}^0_{d,0} \\ L^T\omega^0_{d,0} \\ (\dot{q}_7)_d \end{bmatrix} - \begin{bmatrix} I_{3\times3} & 0 & 0 \\ 0 & L_{3\times3} & 0 \\ 0 & 0 & 1 \end{bmatrix} J_{aug}(q)\dot{q}, \quad (5.35)$$

and by choosing the error dynamics $\dot{e}$ as

$$\dot{e} = \begin{bmatrix} \dot{e}_p \\ \dot{e}_o \\ \dot{e}_{pris} \end{bmatrix} = \begin{bmatrix} K_p & 0 & 0 \\ 0 & K_o & 0 \\ 0 & 0 & K_{pris} \end{bmatrix} \begin{bmatrix} e_p \\ e_o \\ e_{pris} \end{bmatrix}, \quad (5.36)$$

where $K_p \in \mathrm{R}^{3\times3}$ and $K_o \in \mathrm{R}^{3\times3}$ are positive definite and diagonal gain matrices and K_{pris} is a positive scalar gain, the expression below is obtained

$$J_{aug}(q)\dot{q} = \begin{bmatrix} \dot{p}^0_{d,0} + K_p e_p \\ L^{-1}(L^T\omega^0_{d,0} + K_o e_o) \\ (\dot{q}_7)_d + K_{pris} e_{pris} \end{bmatrix}. \quad (5.37)$$

Note that the augmented Jacobian $J_{aug}(q) \in \mathrm{R}^{7\times7}$ is a square matrix and it is invertible unless a kinematically singular joint configuration is attained. Hence, the solution is obtained as

$$\dot{q} = J^{-1}_{aug}(q) \begin{bmatrix} \dot{p}^0_{d,0} + K_p e_p \\ L^{-1}(L^T\omega^0_{d,0} + K_o e_o) \\ (\dot{q}_7)_d + K_{pris} e_{pris} \end{bmatrix}. \quad (5.38)$$

By using the axis angle representation for the approximation of the orientation error, usage of the geometric Jacobian is sufficient in the inverse kinematics computations. This makes the algorithm simpler and faster, because the computation of analytical Jacobian is not required and representation singularities are not present. By augmenting the task space, the Jacobian matrix is made square, with the additional constrained motion task on the prismatic joint's displacement to make sure that the structures before the last link do not contact the ear volume and its surroundings, by defining a spherical clearance. Assuming that the Jacobian is invertible and provided that appropriate gain matrices on the errors are chosen for an appropriate sampling time, the joint trajectories can be obtained to track a desired task trajectory with minimal errors, which are mainly due to numerical integration.

To implement the differential inverse kinematics algorithm described in (5.38), numerical integration in discrete time is used to obtain the joint trajectories as

$$q(t_{k+1}) = q(t_k) + J^{-1}_{aug}(q(t_k)) \begin{bmatrix} \dot{p}^0_{d,0}(t_k) + K_p e_p(q(t_k)) \\ L^{-1}(L^T\omega^0_{d,0}(t_k) + K_o e_o(q(t_k))) \\ (\dot{q}_7)_d(t_k) + K_{pris} e_{pris}(q(t_k)) \end{bmatrix} \Delta t, \quad (5.39)$$

where Δt is the sampling time. At each simulation step, the forward kinematics and the geometric Jacobian are computed. Desired task and error variables are used. Desired task variables include the end-effector velocity $\dot{p}^0_{d,0}(t_k)$, the end-effector angular velocity $\omega^0_{d,0}(t_k)$ and the prismatic joint's velocity $\dot{q}_7$. Error variables include the end-effector position error $e_p(q(t_k))$, the end-effector orientation error $e_o(q(t_k))$ and the prismatic joint's displacement error $e_{pris}(q(t_k))$.

5.2 Results

To test the algorithm from Section 5.1.8, Task 1 (see Figure 4.19) and Task 2 (see Figure 4.20), which are generated in Section 4.2, are used. Their end-effector position and orientation trajectories are plotted in Figure 4.21 (Task 1) and in Figure 4.22 (Task 2).

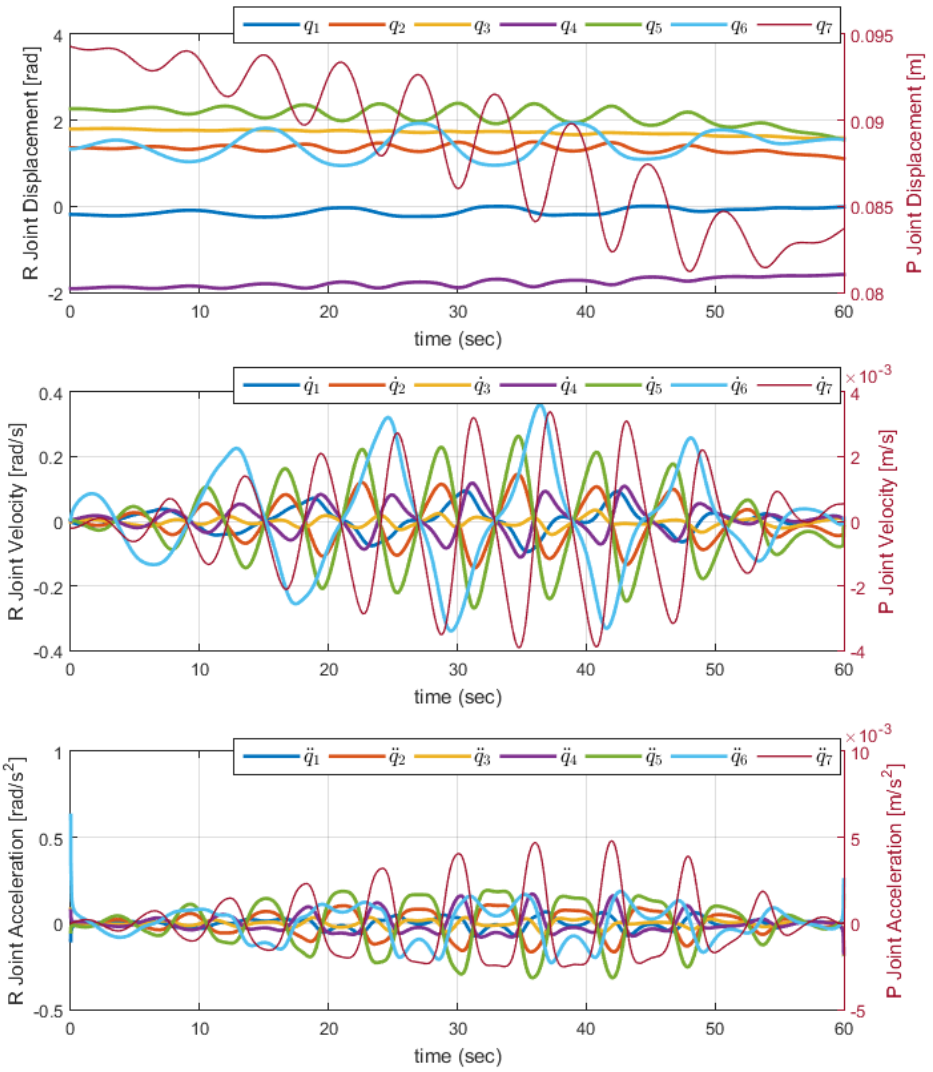

Figure 5.1: Joint displacement, velocity and acceleration profiles from differential inverse kinematics for Task 1

For Task 1 (see Figure 4.21, in Section 4.2), which is defined for a duration of 60 seconds, the differential inverse kinematics algorithm is used at 800 Hz (time step of 1.25 ms) to compute joint trajectories that correspond to the desired task trajectories. In Figure 5.1, the resulting joint displacements (top), velocities (middle) and accelerations (bottom) are plotted. The total duration of the computations is around 11.8 seconds, which correspond to a duration of approximately 0.25 ms per simulation step. The algorithm uses gain variables on the tracking errors of position, orientation and constraint tasks. These gains are set to be unity as $K_{pos} = I_{3\times3}$, $K_{ori} = I_{3\times3}$ and $K_{pris} = 1$. The plots provide information on joint displacement ranges, absolute maximum of joint

velocities and accelerations, which are required to track and complete the given task.

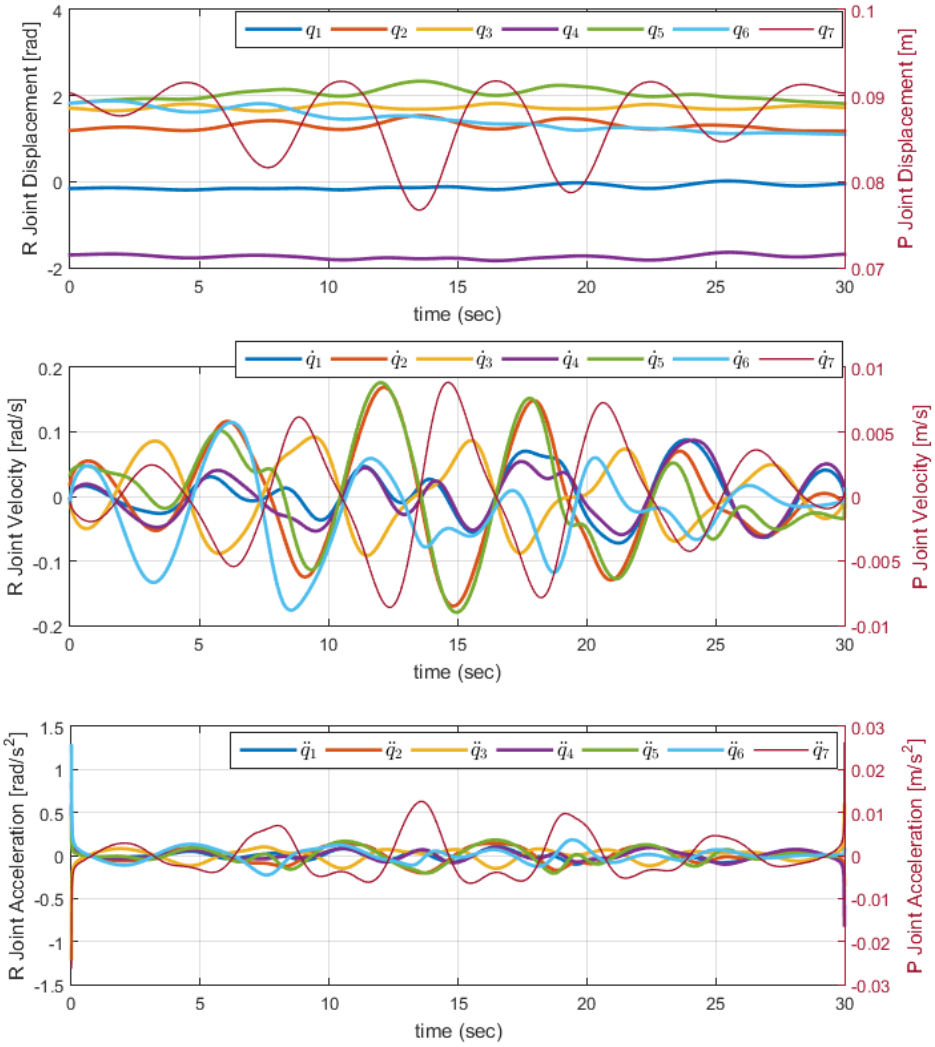

Figure 5.2: Joint displacement, velocity and acceleration profiles from differential inverse kinematics for Task 2

For Task 2 (see Figure 4.22, in Section 4.2), which is defined for a duration of 30 seconds, the differential inverse kinematics algorithm is used at 2 kHz (time step of 0.5 ms). In Figure 5.2, the resulting joint displacements (top), velocities (middle) and accelerations (bottom) are plotted, where the ranges of joint displacements, absolute maximum of joint velocities and accelerations required for the given task can be seen. The total duration of the computations is around 14.8 seconds, which correspond to a duration of approximately 0.25 ms per simulation step. The gains on errors are set to be unity as $K_{pos} = I_{3\times3}$, $K_{ori} = I_{3\times3}$ and $K_{pris} = 1$.

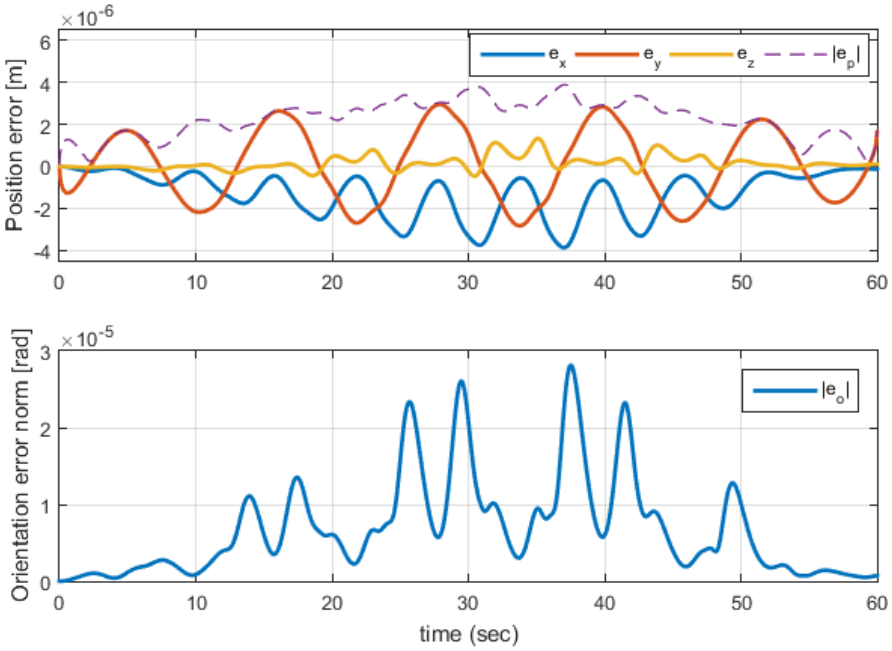

Figure 5.3: Task 1 - The plots of tracking errors of end-effector position (top) and orientation (bottom)

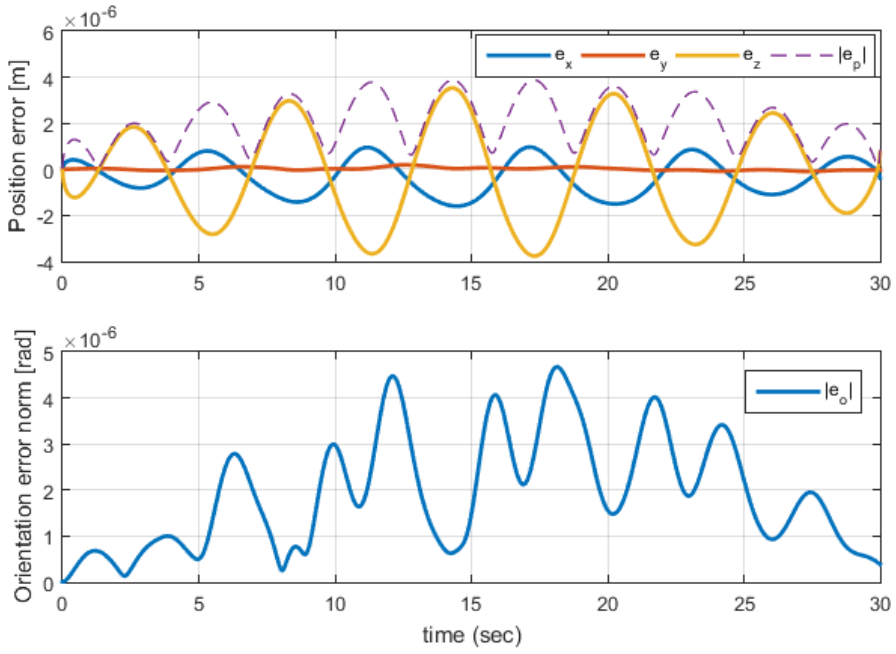

Figure 5.4: Task 2 - The plots of tracking errors of end-effector position (top) and orientation (bottom)

With unity gains on the errors in the kinematic tracking, the errors in position and orientation for Task 1 and Task 2 are given in Figure 5.3 and Figure 5.4, respectively. For the end-effector position, error norms for both tasks are below $5{\cdot}10^{-6}$ m. For end-effector orientation, error norms for both tasks are below $3{\cdot}10^{-5}$ rad. These errors are mainly due to numerical integration and approximations used in calculating the orientation error and setting the desired angular velocity of the end-effector at each time step. Note that

the tracking for the constraint task on the prismatic joint is not given since it is zero due to the definition of the augmented Jacobian. To demonstrate the effect of using different gains on kinematic tracking, results for Task 1 with $K_{pos} = K_{ori} = 0_{3\times3}$ (open-loop) and $K_{pos} = K_{ori} = 100I_{3\times3}$ are given in Figure 5.5 and Figure 5.6, respectively.

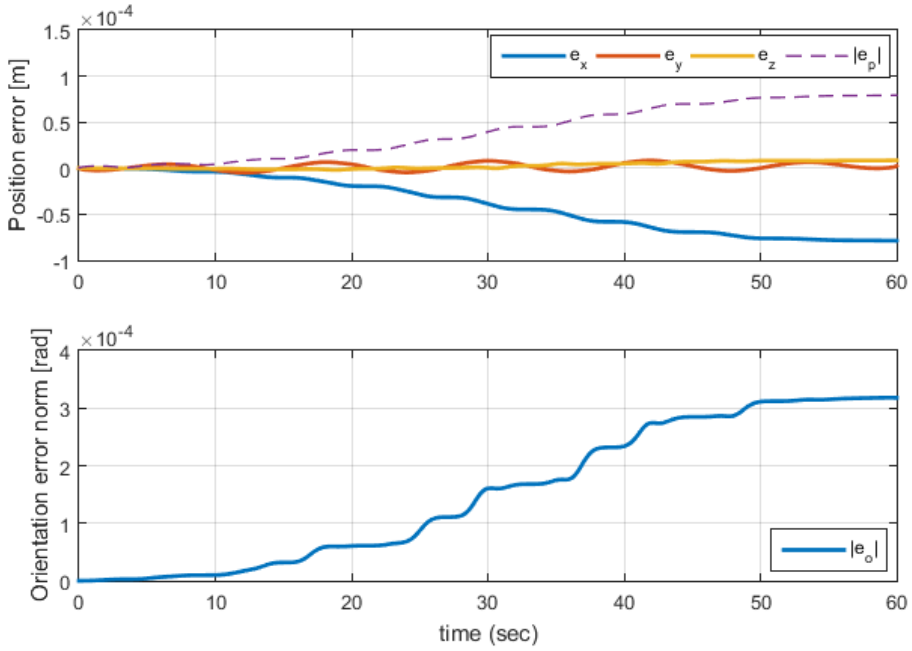

Figure 5.5: Task 1 - End-effector position (top) and orientation (bottom) tracking errors with no gains (open-loop)

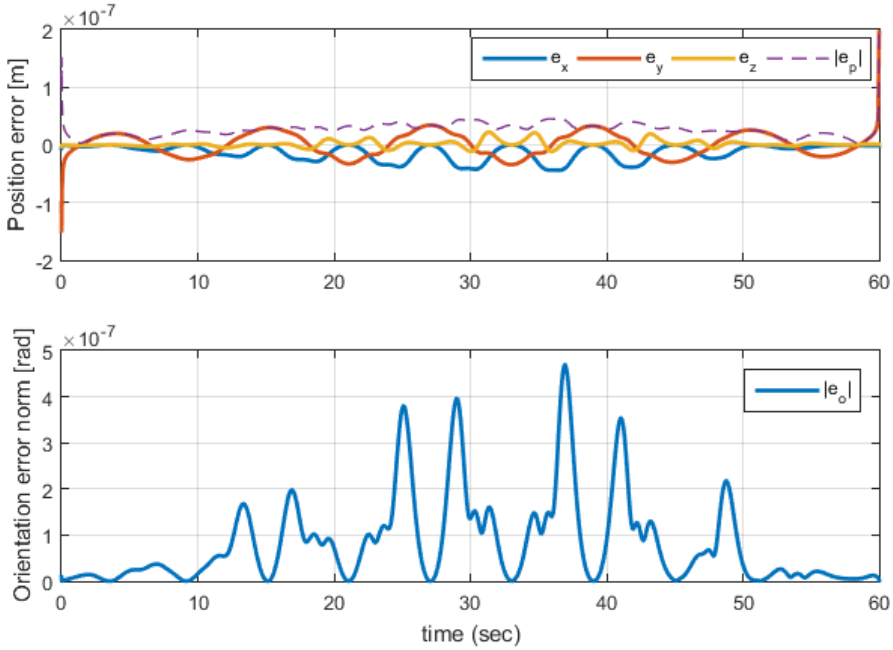

Figure 5.6: Task 1 - End-effector position (top) and orientation (bottom) tracking errors with increased gains

In Figure 5.5, it is seen that in open-loop, the algorithm tracks Task 1 with position

and orientation error norms less than 10^{-4}m and $4 \cdot 10^{-4}$rad, respectively. However, it should be noted that these errors correspond to the errors which accumulate until the end of the simulation (drift phenomenon) since there is no feedback on the errors and they would increase if the simulation duration is increased.

In Figure 5.6, the gains are increased to 100 and it is seen that there is a reduction in both position and orientation error norms, relative to the previous cases with no gains and unity gains (see Figure 5.3). Furthermore, when the relatively high errors ($2 \cdot 10^{-7}$ m) at the start and finish of the task are ignored, the position error norm throughout the task is smaller than $5 \cdot 10^{-8}$ m. Relatively high error norm values, which are still smaller than $2 \cdot 10^{-7}$ m, at the start and finish of the task are probably due to the task velocity definition and the approximations used in defining the orientation error and desired angular velocity of the end-effector. The optimal values for the gains depend on the desired minimum error threshold, numerical limits and the frequency of the simulation, which also depends on the time-resolution of the desired task. As the chosen frequency is higher, the time step is smaller and the gains can be increased. Each frequency has a limit gain value above which tracking of the desired task results in unstable mapping between the task and joint variables. If a tracking is unsatisfactory, meaning the error in position and orientation tracking are above a set threshold, then the resolution of the task and the gains can be increased.

5.3 Discussion

With two tasks that are defined using two different routing approaches, the inverse kinematics algorithm has been tested and joint trajectories corresponding to the tasks have been obtained. Numerical errors are also expected due to drift phenomena, which is a result of using numerical integration. Since the algorithm uses gains on the task variable errors as feedback, closed-loop cases with different gains and an open-loop case are tested. The results show that with a feedback on task errors (position, orientation and constraint on prismatic joint) using a unit gain, a position error profile below $5 \cdot 10^{-6}$ m (both tasks) and an orientation error profile below $3 \cdot 10^{-5}$ rad (Task 1) and $5 \cdot 10^{-6}$ rad (Task 2) are present throughout the specified task. These have been further decreased by increasing the gains to 100 which resulted in a position error norm for Task 1 below $2 \cdot 10^{-7}$m and an orientation error norm below $5 \cdot 10^{-7}$ rad. Also in the case with no gains in open-loop for Task 1, a position error norm below 10^{-4} m and an orientation error norm below $4 \cdot 10^{-4}$ rad are present throughout the task (for 60 seconds), which of course would accumulate if the simulation duration is increased. Nevertheless, the level of tracking error suggests that, even in open-loop during kinematic tracking at the velocity level, errors due to numerical integration accumulate slowly if the desired translational and angular velocity profiles are defined properly. It should also be noted that the algorithm starts from an initial joint configuration to follow a given task. Hence an initial joint configuration, which results in the initial task configuration of a given task, must be chosen. To achieve this, either the given task can be extended to start from the initial configuration of the robot or the joint configuration at the start of the given task can be computed using the same algorithm before the actual task execution.

Overall, the differential inverse kinematics algorithm with the augmented task space is capable of tracking a given task and computing the associated joint variables with acceptable errors for the bone milling task. Additionally, the algorithm takes around 0.24 ms per simulation step, where the computation of forward kinematics, geometric

Jacobian and augmented Jacobian's inverse takes place. The computational speed of the algorithm and its implementation suggests that it can used in tracking an online task trajectory input in real time at frequencies up to 4 kHz.

Chapter 6

Manipulability Analysis and Software Implementation Details

6.1 Manipulability Analysis

Manipulability is the ability of a manipulator to move its end-effector, including its position and orientation, at a given joint configuration. It is reflected on the efficiency of robot motion and its functionality, since doing the same end-effector motion at a configuration with a higher end-effector manipulability results in a smaller displacement (i.e. effort) of the joints. In other words, for the same amount of allowed change in joint displacements, the robot has a higher capacity of changing its end-effector pose when it has a higher manipulability. As given in Section 2.3.2, there are different measures related to the manipulability of a robot end-effector. In this section, we want to analyse the robot's end-effector manipulability in its workspace, to evaluate a given task in terms of the manipulability measure throughout the task and to make choices on initial robot configuration and robot-patient placement; in order to avoid less manipulable workspace regions.

For the analysis of manipulability of the end-effector in the robot workspace, a sampling method is used. It involves computing forward kinematics for random joint configurations, which are chosen with a uniform probability distribution, inside a chosen range for joint displacements. The sampling is done for a certain number of trials and a manipulability measure for each sample is computed. By setting a range of motion for each joint, the manipulability analysis can be performed in the neighbourhood of a certain configuration in the robot's joint space. For the manipulability measure, below are two possible choices,

$$\mu_1 = \sqrt{\det JJ^T} = \sigma_1 \dots \sigma_n, \tag{6.1}$$

which is the so-called Yoshikawa's manipulability index [26] and

$$\mu_2 = \frac{1}{\text{cond}(J)} = \frac{\sigma_n}{\sigma_1}, \tag{6.2}$$

which is the inverse condition number measurement [27], where $\sigma_i \in \{\sigma_1 \, , \dots \, , \sigma_n\}$ denote, from minimum to maximum, the singular values of the Jacobian matrix J.

Using a large range of random displacements for the joints results in a visual seen in Figure 6.1, which is not easy to comprehend. The outer blue points represent the orientation trace of the end-effector at each sampled position. In the center, obtained

end-effector positions are represented with a colormap according to their measure of manipulability.

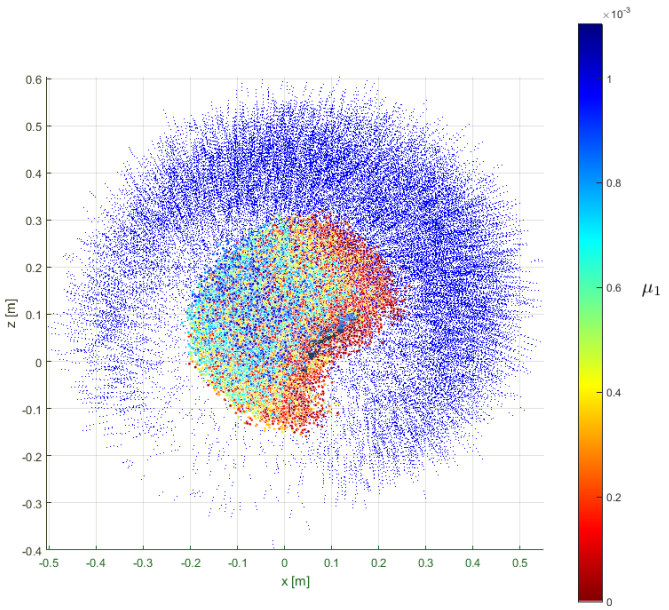

Figure 6.1: Manipulability analysis data with respect to μ_1 is represented with a colormap, where the robot is in the center and blue points surrounding the data represent the orientation trace of the end-effector at each sample (3D plot from side-view)

The complexity of this result is due to the fact that, the non-unique mapping between the joint space and the task space allows for multiple configurations with different manipulability measures at the same or close locations. Therefore, a smaller range of joint displacements for a preliminary test to compare the manipulability measures is chosen as

$$\begin{aligned} q_{mid} &= [0 \;\; \frac{\pi}{4} \;\; \frac{\pi}{2} \;\; \frac{-\pi}{2} \;\; \frac{\pi}{4} \;\; \frac{\pi}{2} \;\; 40.10^{-3}] \\ \Delta q &= [\frac{\pi}{5} \;\; \frac{\pi}{4} \;\; \frac{\pi}{10} \;\; \frac{\pi}{10} \;\; \frac{\pi}{4} \;\; \frac{\pi}{10} \;\; 40.10^{-3}] \end{aligned} . \tag{6.3}$$

As a result, the volume of the workspace spanned by the robot is reduced, but the distribution of the measurements gives an improved insight on the manipulability of the chosen range of joint displacements since the manipulability measure is more distinguishable, as seen in Figure 6.2. By its definition in (6.1), μ_1 takes into account all the singular values of the Jacobian and it is proportional to the volume of the manipulability ellipsoid, whereas the analysis with μ_2, which is given in (6.2), considers only the maximum and minimum singular values which can be deceiving since it is related to their ratio but not the real values. Hence μ_1 will be adopted for the manipulability analysis.

To evaluate the sampled portion of the workspace, two methods can be used. Firstly, a grid can be generated based on the data, to investigate a section of the sampled volume. To generate gridded data from the manipulability data, the point cloud of the manipulability analysis is filtered by setting a 3D mesh with adjustable resolution and collecting the mean value of manipulability in that volume. Alternative to obtaining the mean value in each volume, the minimum value of the manipulability measure can also be

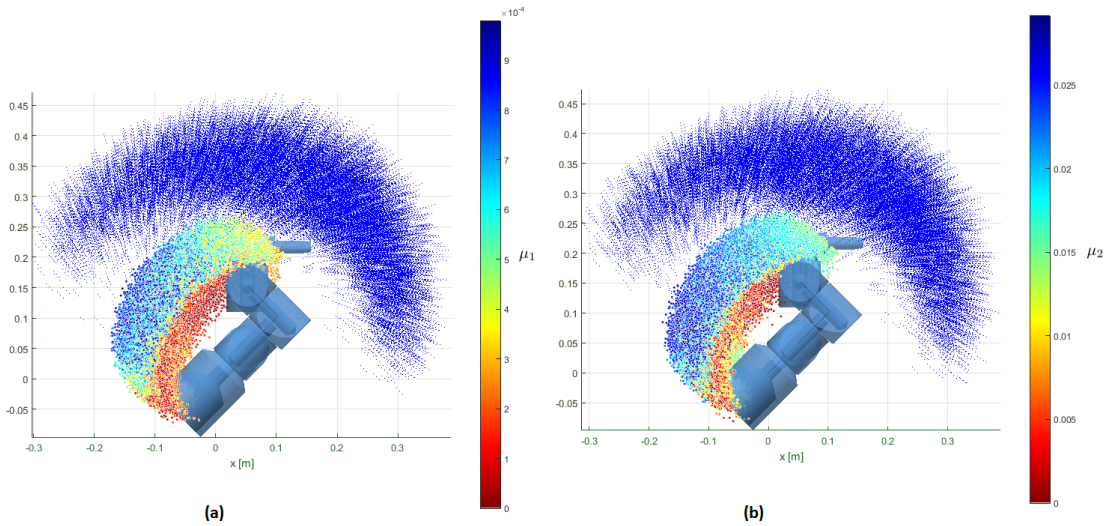

Figure 6.2: Plot (a) shows the manipulability analysis with respect to μ_1 while Plot (b) shows the analysis with respect to μ_2 (3D plot from side-view)

used for analysis. An example of this approach can be seen in Figure 6.3, where the data represented in Figure 6.2 (a) is gridded with respect to mean values of manipulability to obtain a section representation at $z = 0.1$m level, with a grid resolution of 0.005m.

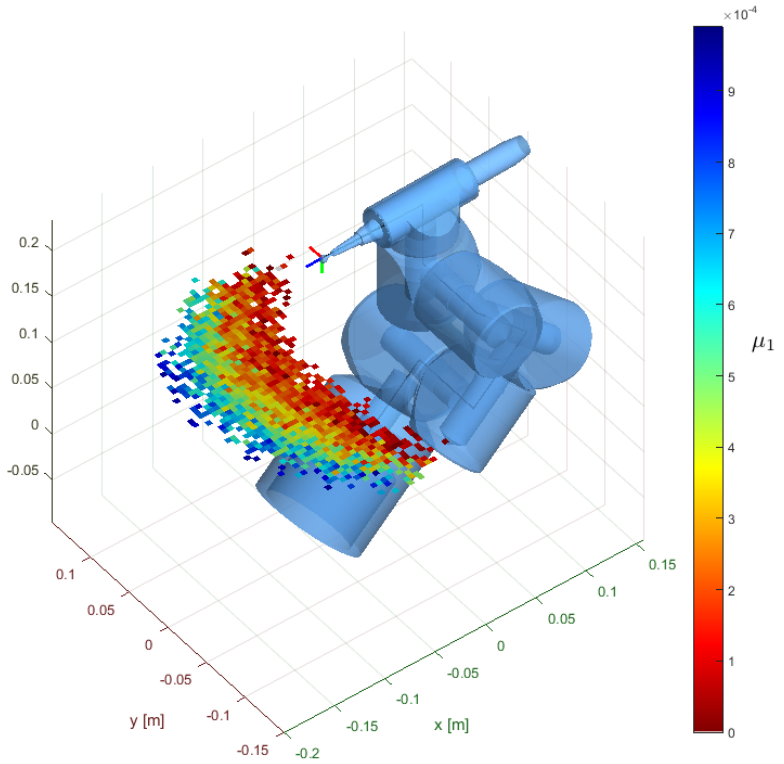

Figure 6.3: Horizontal section of the 3D grid of the mean values of manipulability data

Secondly, instead of gridding the initial data, manipulability measures can directly be filtered with respect to end-effector position and orientation at each configuration. Instead of sections, small volumes can be investigated and since orientation is also taken into ac-

count, the resulting data can give clear insights. To show this, the initial manipulability data which is plotted in Figure 6.2, is filtered with respect to conditions on end-effector position as $-0.15 < (o_{ee})_x < -0.05$, $-0.03 < (o_{ee})_y < 0.03$ and $0.1 < (o_{ee})_z < 0.25$ and an angular difference (from axis-angle representation) of 0.5rad with respect to a reference orientation $R_{ref} = R_{y,\pi/2}R_{z,\pi/2}R_{x,\pi}$ (green line), as shown in Figure 6.4.

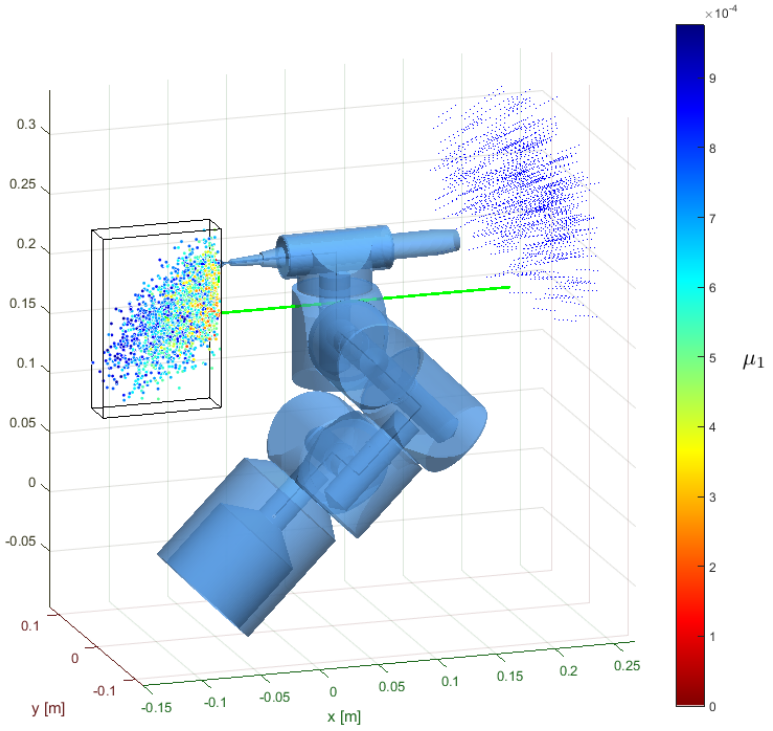

Figure 6.4: Filtered manipulability measure data with respect to end-effector position and orientation range

6.1.1 Results

In the manipulability analysis, the joint configuration of the robot is spanned with a uniform distribution on a defined range for each joint. Singular value decomposition of the augmented Jacobian is performed to compute the manipulability measure given in (6.1) in Section 6.1.

Sampling the Workspace and Filtering Manipulability Data

The middle values and the range of motion for a sample test are set as

$$\begin{aligned} q_{mid} &= [0 \;\; \frac{\pi}{4} \;\; \frac{\pi}{2} \;\; \frac{-\pi}{2} \;\; \frac{\pi}{4} \;\; \frac{\pi}{2} \;\; 40.10^{-3}] \\ \Delta q &= [\frac{3\pi}{10} \;\; \frac{4\pi}{10} \;\; \frac{3\pi}{10} \;\; \frac{3\pi}{10} \;\; \frac{4\pi}{10} \;\; \frac{3\pi}{10} \;\; 40.10^{-3}] \end{aligned} \quad . \tag{6.4}$$

Without any filtering for the position and orientation of the end-effector, which is obtained during the distribution according to the joint ranges in (6.4), the manipulability measure data are obtained as 10^6 data points within a computation duration around 140 seconds. Selectively choosing the manipulability measure data from 10^6 samples, according to the end-effector's position and orientation at each particular configuration results in a volume as shown in Figure 6.5.

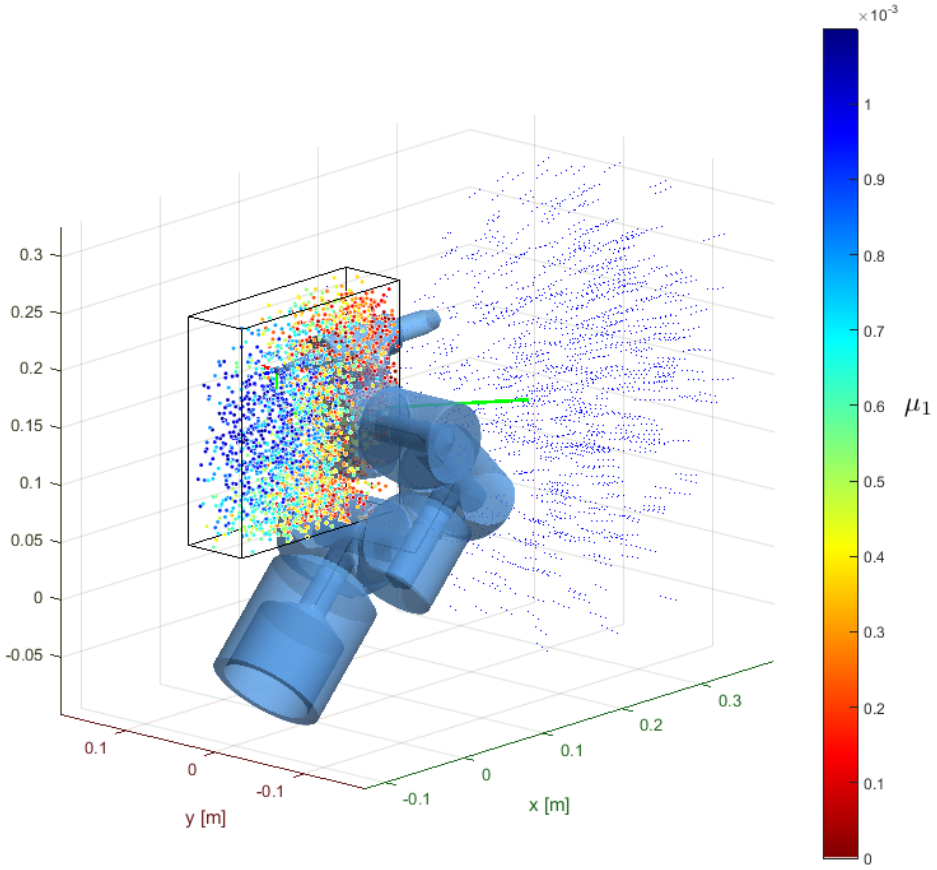

Figure 6.5: Manipulability analysis in a small volume - Sampling the Workspace (with end-effector orientation reference (green line), obtained orientation at each point (blue trace))

As seen in Figure 6.5, by selecting end-effector configurations which are close to a certain reference orientation (green line)

$$R_{ref} = R_{y,1.745} R_{z,\pi/2} R_{x,\pi} \ , \tag{6.5}$$

by 0.5 rad and in a range of position as

$$\begin{array}{rcl} -0.125 & < (o_{ee})_x < & 0.075 \\ -0.03 & < (o_{ee})_y < & 0.03 \\ 0.075 & < (o_{ee})_z < & 0.275 \end{array} \ , \tag{6.6}$$

the resulting end-effector manipulability data have a recognizable distribution for that selection.

Comparison of Task Placements

The measurements on end-effector manipulability can be used as follows. Consider the two different placements for the ear volume approximation depicted in Figure 6.6,

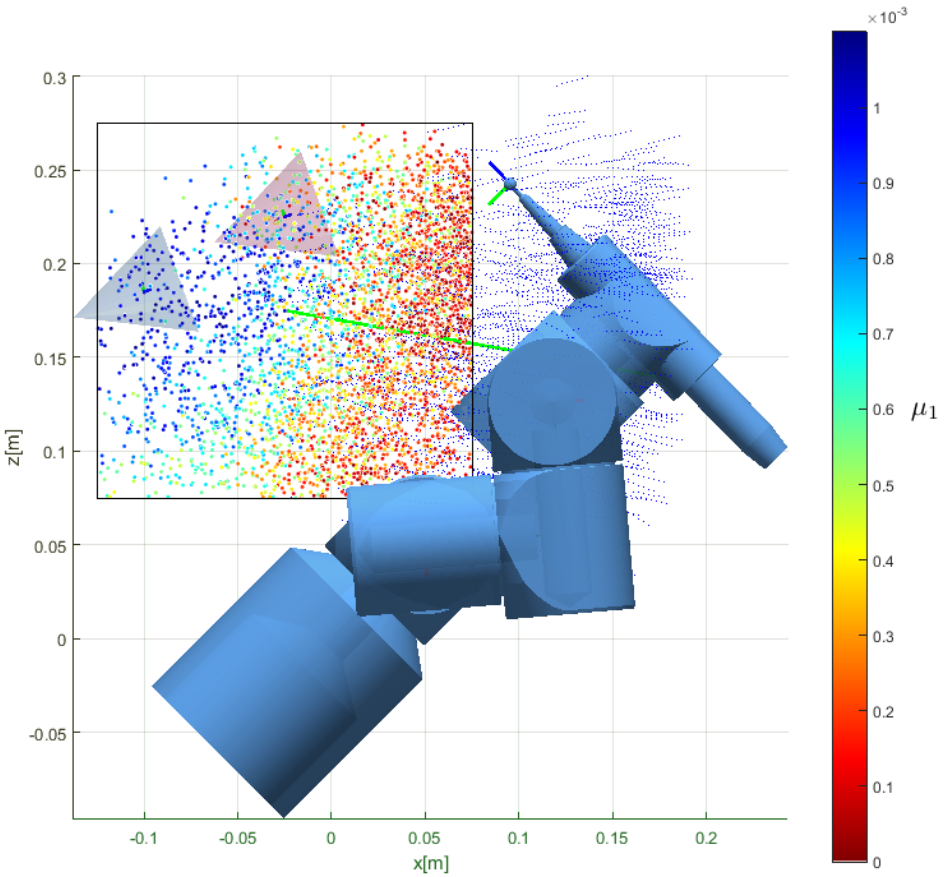

Figure 6.6: Two ear volume placements where the lower-left cone is in the same place as Task 1. The placement on lower left is denoted Test 1 (gray) and the one on upper right is denoted Test 2 (red) (3D plot from side-view)

where left (gray) and right (red) cones represent the two different locations for Task 1, which is defined in the task trajectory generation part in Section 4.2 and used in the inverse kinematics simulations in Section 5.2. The two different placements in Figure 6.6 will be referred to as Test 1 (gray) and Test 2 (red). The measured data in Figure 6.6 show a decrease in the manipulability measure for Test 2 compared to Test 1. To validate this throughout the task, the singular value data and the manipulability measures from the inverse kinematics simulation during execution of these similar tasks at two locations are recorded. The end-effector manipulability throughout the task for both placements is shown in the upper plot in Figure 6.7. It is seen that throughout the task execution, the manipulability of the end-effector is greater in Test 1 compared Test 2, as expected. Lower manipulability measure implies that greater joint displacements are necessary to manipulate the end-effector. In fact, it is seen in the lower plot of Figure 6.7 that in the case of lower manipulability, the norm of joint velocities increased throughout the task. This implies also an increase in accelerations of the joints which is not desired for dynamic control.

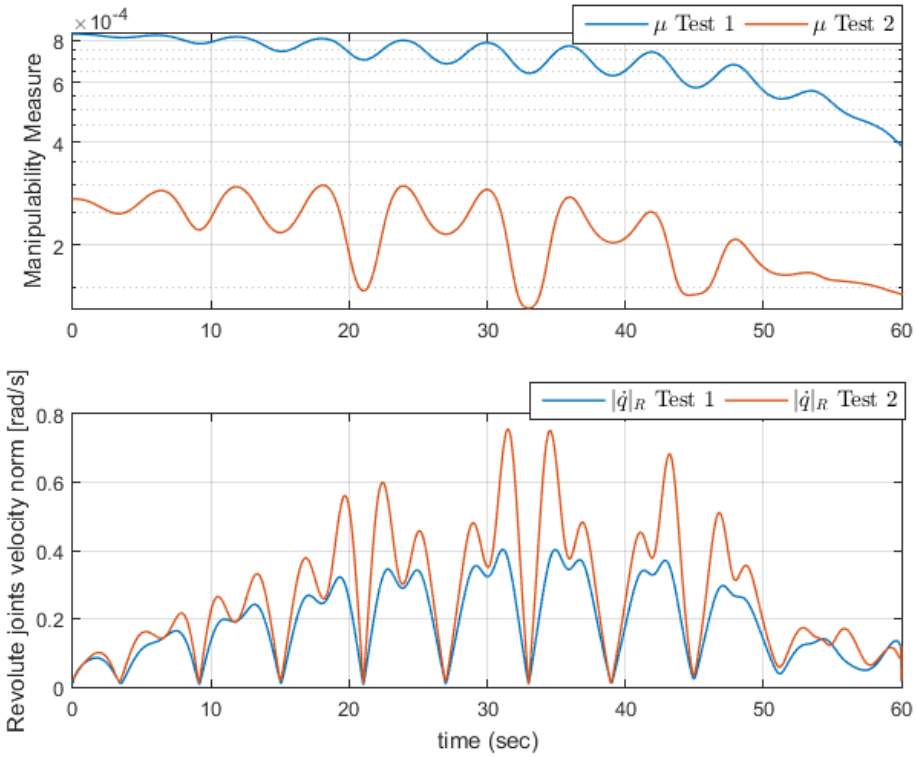

Figure 6.7: Manipulability measure μ_1 throughout a task with different placements in the robot workspace

Manipulability analysis helps in determining how the target object to be milled must be placed in the workspace of the robot. When a task is defined, range of the desired motion of the end-effector for that task is obtained. The same task can be tested for end-effector manipulability and kinematic tracking performance at various poses in the robot's workspace by changing the position and orientation of the target object. Since end-effector task includes the position and orientation completely, the range of desired motion in terms of position and orientation can be used in the manipulability analysis to determine the workspace regions, which have a higher end-effector manipulability throughout the task. As shown by the previous case, higher manipulability results in less effort for the actuators in terms of total displacement, velocity and acceleration of the joints.

6.1.2 Discussion

The manipulability analysis is done with an example case of choosing different ear cone locations in the robot's workspace. The analysis allows to investigate regions in the workspace of the robot for better manipulability in comparison to other regions. This can help in the decision on how to place an object to be milled, in terms of its position and orientation, such that the robot can execute tasks on the object with higher manipulability. In the given example, it is shown that the performance of the kinematic tracking through inverse differential kinematics can be improved in terms of the overall displacement, velocity and accelerations of the joints, by choosing a region with higher end-effector manipulability.

6.2 Software Implementation and Toolbox

A brief software structure schematic is given in Figure 6.8. The setup phase is to be performed initially and once, unless the kinematic model of the robot is changed. In this phase, symbolic expressions of the forward kinematics and the geometric Jacobian are calculated. From these symbolic expressions, C files, which will be compiled later, are generated automatically. By using Mex files in the simulation part to compute the forward kinematics and the Jacobian, one step of the algorithm reduced from 0.135 to 0.00035 seconds, which is around 385 times faster. In the simulation phase, the task generation part computes the desired task trajectories. Then, in the inverse kinematics part, the joint variables, which result in the end-effector tracking of the desired task trajectories, are computed. Finally, the results are shown and the motion of the robot is visualized.

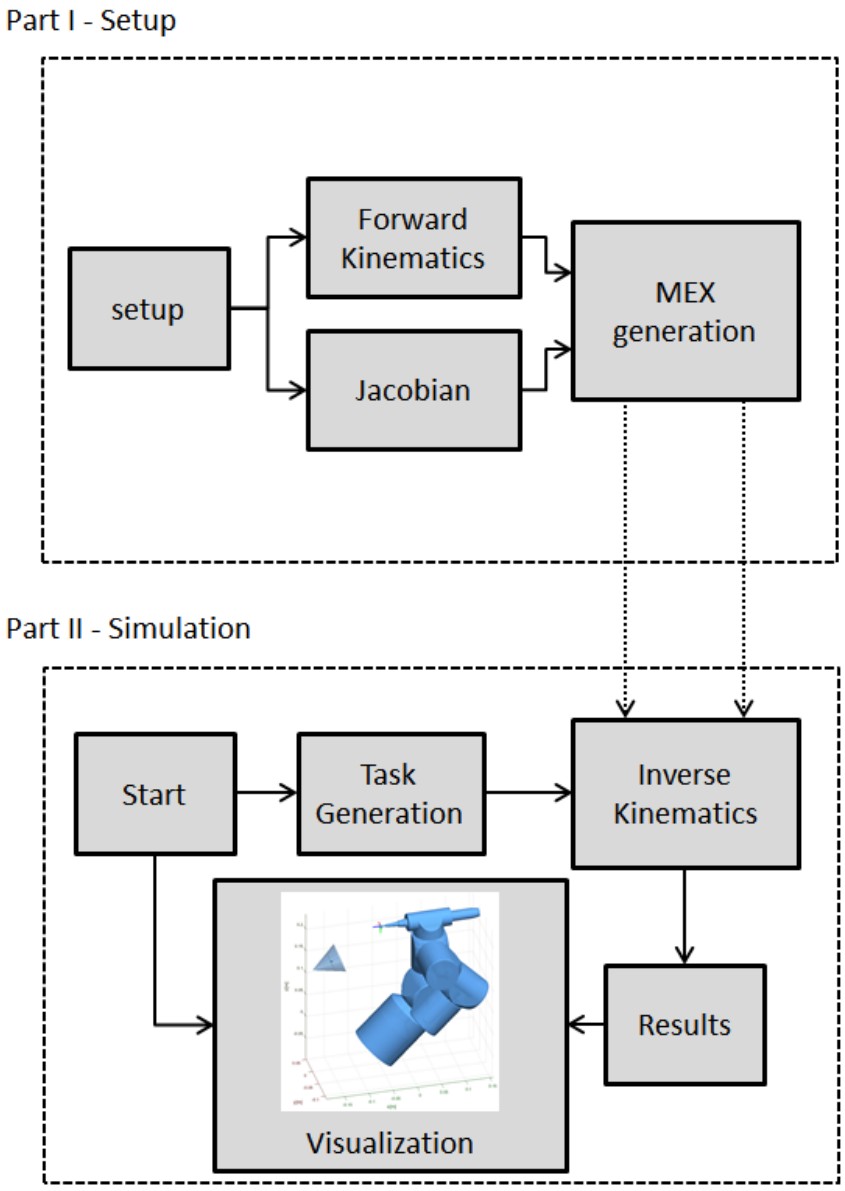

Figure 6.8: A brief schematic of the software implementation in Matlab

6.2.1 Toolbox Setup

In the setup part, the parameters of the kinematic model of the robot are set by the user. From the kinematic model, the forward kinematics and the geometric jacobian of the manipulator are computed symbolically. The computation times are around 2.5 and 2.7 seconds, respectively. The values of the parametric variables are substituted in the symbolic expressions, which are then converted into C code format. The C codes are generated automatically by the function *ModArm_generateMex* and they are compiled to generate Mex executable files that are accessible from the Matlab command line to compute the homogeneous transformation matrices and the geometric Jacobian based on a given joint configuration. The setup phase is finished after the Mex files are generated.

The setup routine needs to be run when a parameter defining the robot kinematic model or its reference pose with respect to the fixed frame is changed. Using an automated C code implementation in this phase is advantageous over using symbolic expressions for its computational efficiency. This way, the DH parameters, number of degrees of freedom, joint order and world to base transformation, which defines the position and orientation of the base of the robot, can be modified. Automatically generated Mex files for serial manipulators with different models can be stored for future use as Mex functions, which are accessible from MATLAB command line directly.

6.2.2 Simulation

In the simulation phase, parameters, which are related to the task specification, the inverse kinematics algorithm, the simulation and visualization, are defined. Firstly, the desired task task trajectories are generated. Then, the joint trajectories corresponding to the desired task are computed in the inverse kinematics part. Finally, the resulting task execution is visualized and results such as joint displacements, velocities, accelerations, the desired task trajectories, errors in task trajectory tracking, singular values of the Jacobian and manipulability measure throughout the task are plotted.

For the inverse kinematics, it is necessary to compute forward kinematics and the Jacobian at each step where the joint configuration is changed. Doing each computation symbolically within one step of the simulation has a computational duration around 0.135 seconds. When the Mex functions, which are generated in the setup phase only once, are used instead of symbolic expressions, the duration for a simulation step is reduced to 0.24 ms. As a result, a simulation with a time span of 0 to 2 seconds at 100 Hz which has 200 algorithm steps in total, takes around 27 seconds when symbolic expressions are used, whereas the same simulation takes only about 0.05 seconds with the implementation of Mex functions. A simulation step duration of approximately 0.24 ms for inverse kinematics suggests that a real time use of the current state of this software can be suitable for operation frequencies around 4000 Hz (0.25 ms per step), to compute joint velocities to track a desired task.

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

In this thesis, solutions to both the task specification and the inverse kinematics problems have been developed for the surgical bone removal robot RoBoSculpt.

With the proposed task specification method, desired task trajectories, which describe end-effector position, orientation and a constraint task on the prismatic joint, are generated for bone removal on a section of a conical approximation of a human ear. By the desired task trajectories, a section of the ear cone is routed by the end-effector, while its orientation is chosen, such that the rotary tool is clear from the walls of the cavity. Prismatic joint, which actuates the rotary tool, is also taken into account and its displacement is geometrically constrained to keep the supporting link clear from the ear to avoid collisions. As a result, the clearance problem is decoupled into one for the end-effector orientation, to avoid collision with the cavity walls, and one for the prismatic joint, to keep the end-effector's supporting link clear from the whole cavity. By introducing the one dimensional constraint task, the task space is augmented to have seven dimensions and as a result, the robot operation is not redundant any more.

A standard inverse differential kinematics algorithm is adopted and modified to incorporate augmented task space with augmented Jacobian. Using the algorithm, the inverse differential kinematics is integrated over time while the numerical errors are accounted for, using task space error feedback, and the task trajectories are tracked at the velocity level. Computed joint trajectories for sample tasks show a kinematic tracking of the desired task with a position error less than 5 μm ($5 \cdot 10^{-6}$ m) and an orientation error less than $3 \cdot 10^{-5}$ rad, which are due to the numerical integration in the algorithm and the approximations in calculating the end-effector orientation error and in setting the desired angular velocity of the end-effector at each time step. It is shown that these errors can be reduced further by setting higher gains on kinematic tracking errors. Additionally, the algorithm is computationally efficient with a duration around 0.24 ms for each step, which allows its use with an online task input at frequencies upto approximately 4 kHz.

The end-effector manipulability is analysed in the robot workspace and throughout given tasks. The manipulability data is obtained by sampling the robot workspace with random joint displacements within specified ranges. During the collection, the manipulability data is filtered with respect to the end-effector position and orientation

at each configuration, to analyse the end-effector manipulability for specified regions of both its position and orientation. It is shown that performing a kinematic task at different locations with respect to the robot, with overall difference in the end-effector manipulability affects the performance of task execution. A task placement with higher manipulability imposes less effort for the actuators in terms of overall velocity of the joints. Hence, it can be desirable to use the manipulability analysis to investigate a given task trajectory for the end-effector manipulability and to make changes in the given task's location with respect to the robot, in order to avoid regions with lower manipulability.

Additionally, with the software tools developed during this project, design modifications can be made on the kinematic model. Modified models can be checked through forward and inverse kinematics tests and visualizations. Modified workspaces can be investigated with the manipulability analysis. Kinematic performance assessments can also be made, through task generation and inverse kinematics tests for any given task, which will hopefully be useful in future projects related to RoBoSculpt.

7.2 Recommendations

There are a number of possible improvements for future applications regarding the task specification and inverse kinematics solution for RoBoSculpt.

- The current task specification method is based on a conical approximation of an ear volume and a layer of bone to be milled is approximated as a circular section orthogonal to the cone axis. In a real operation, the shape of the chosen layer to be milled can have an irregular shape. Hence, the end-effector position trajectory generation can be extended to produce position profiles which span irregularly shaped sections of a bone volume to be milled out. In this case, the generation of the orientation trajectory, which guarantees collision-free motion, would become more complex, because of the irregularly shaped cavity. Hence, rather than a geometric solution, the orientation trajectory generation can be made with an iterative approach, since it might not be possible to find a closed-form expression for the orientation. Following a given position trajectory through an iterative algorithm, the orientation trajectory profile can be optimized to result in a continuous and collision-free motion. The optimization can be performed with respect to quality indices such as the cone-edge clearance. Additionally, if the current conical approximation is to be used in practice, then the proposed geometric solution can also be adopted in an optimization routine. Cost functions can be set on criteria such as edge clearance and ranges of angles of approach and inclination, and as a result, an optimal orientation trajectory profile can be generated. Currently, this is done manually and if collisions are detected, the user iteratively changes coefficients in the descriptions of the angular parameters which describe the orientation.

- For simple milling tasks (e.g. on planar surfaces), it can be desirable to operate with less number of joints of the robot, for precision, power consumption and rigidity, since the mechanical design of RoBoSculpt has the capability of locking the joints mechanically and the use of less joints would result in smaller accumulation of errors such as the ones from quantization and low-level control. To mill or drill on an open planar surface with no edges that could otherwise limit the motion, use

of the last three joints (RRP) can be sufficient while the other joints are mechanically locked. For these kind of practical cases, both the task specification part and inverse kinematics algorithm can be extended to include options on changing the degrees of freedom involved in task definition and robot actuation.

- Currently, the manipulability analysis is manually done by choosing ranges of joint motion and filtering the measured manipulability data by user-defined ranges of end-effector position and orientation. The analysis method can be extended to be partly automated by devising a routine which searches for the best configuration of the task placement with given limits on the joints. Gridding the workspace with the manipulability measure as explained in this thesis can also help in speeding up and simplifying the process. However, it should be noted that detection of singularities is not guaranteed since the method consists of sampling the workspace of the robot with random joint displacements (with uniform probability distribution on a set range). Hence, a high number of samples should be taken to sample the workspace with the joint displacement ranges (set by the user) as much as possible to increase the statistical reliability of the manipulability data.

- The kinematic model and modular structure of RoBoSclupt also brings up the possibility of intra-collisions between the robot parts. There are possible ways to check for intra-collisions and it is recommended that this issue is further investigated and implemented for the task planning of RoBoSculpt in future projects. One possible method is to encapsulate each moving part with a custom ellipsoid and checking for algebraic conditions for ellipsoids' separation or overlapping by using their characteristic polynomials. Note that this method is conservative since ellipsoids are used to approximate the robot parts. Additionally, this method can also be used to measure the distance to an intra-collision, which is the closest distance between robot parts that could collide with each other. As a result, it can be useful in real-time operation where it can be desirable to detect intra-collision possibilities before they actually occur. A second option is to use triangular meshes of the robot parts and check for collisions from the vertices of each triangular surface. This method is less conservative, but it can be also computationally demanding and it does not measure the distance to an intra-collision. Hence, it is suitable for task planning rather than real-time use. Another direction, which needs to be investigated, is to use the normal vectors of triangular meshes vertices of the parts. In Appendix C.1, Figures C.1-C.2 show triangular re-mesh from one of the parts' .stl files and vertex normal vectors. Using the vertex position data and normal vector directions, regions of the parts with collision possibility can be filtered with respect to their position and orientation. Then, the distance in filtered vectors' direction, which result in intersection with the vertices on the other side, can be computed.

- Finally, the software can be simplified (e.g. to become a library of functions to allow easy prototyping and testing of algorithms) and generalized for open public use for fast simulations of forward and inverse kinematics of serial manipulators with various DOFs.

Bibliography

[1] J. Bos, "Modular robotic device for precision surgical bone removal and other applications," Patent Application Number US 62/253 575, 2015.

[2] J. H. Dirckx, M.D., "Robotic surgery," *The SUM Program Obstetrics/Gynecology Advanced Medical Transcription Unit*, 2011, health Professions Institute.

[3] Y. Kwoh, J. Hou, E. Jonckheere, and S. Hayati, "A robot with improved absolute positioning accuracy for ct guided stereotactic brain surgery," *IEEE Transactions On Biomedical Engineering*, vol. 35, no. 2, pp. 153–160, 1988.

[4] S. Harris, F. Arambula-Cosio, Q. Mei, R. Hibberd, B. Davies, J. Wickham, M. Nathan, and B. Kundu, "The probot - and active robot for prostate resection," *Journal of Engineering in Medicine*, vol. 211, no. 4, pp. 317–325, 1991.

[5] Y. Wang and K. Laby, "Automated endoscope system optimal positioning," Patent US 5 657 429, 08 12, 1997.

[6] H. Yu, N. Hevelone, S. Lipsitz, K. Kowalczyk, and J. Hu, "Use, costs and comparative effectiveness of robotic assisted, laparoscopic and open urological surgery," *The Journal of Urology*, vol. 187, pp. 1392–1399, 2012.

[7] F. Matsen, J. Garbini, J. Sidles, B. Pratt, D. Baumgarten, and R. Kaiura, "Robot assistance in orthopaedic surgery: a proof of principle using distal femoral arthroplasty," *Clin. Orthop.*, no. 296, pp. 178–186, 1993.

[8] S. Delp, S. Stulberg, B. Davies, F. Picard, and F. Leitner, "Computer assisted knee replacement," *Clin. Orthop.*, no. 49, p. 354, 1998.

[9] M. Borner, U. Wiesel, and W. Ditzen, "Clinical experiences with robodoc and the duracon total knee," *Navigation and Robotics in Total Joint and Spine Surgery*, vol. Springer Verlag, no. 296, pp. 362–366, 2004.

[10] A. Wolf, B. Jaramaz, B. Lisien, and A. DiGioia, "Mbars: mini bone-attached robotic system for joint arthroplasty," *International Journal of Medical Robotics and Computer Aided Surgery*, no. 1, pp. 101–121, 2005.

[11] M. Shoham, M. Burman, E. Zehavi, L. Joskowicz, E. Batkilin, and Y. Kunicher, "Bone-mounted miniature robot for surgical procedures: concept and clinical applications," *IEEE Trans. Rob. and Autom.*, vol. 19, pp. 893–901, 2003.

[12] M. Berardoni, "A computer aided cutting guide positioner to improve bone-cutting precision in total knee arthroplasty," *Proceedings of the 3rd annual meeting of the International Society for Computer-Assisted Orthopaedic Surgery*, pp. 28–29, 2003.

[13] N. Dillon, R. Balachandran, J. Fitzpatrick, M. Siebold, R. Labadie, G. Wanna, T. Withrow, and R. Webster, "A compact, bone-attached robot for mastoidectomy," *Journal of Medical Devices*, vol. 9, no. 031003, 2015.

[14] M. Mirour, Y. Nguyen, J. Szewczyk, S. Mazalaigue, E. Ferrary, O. Sterkers, and A. Grayeli, "Robotol: From design to evaluation of a robot for middle ear surgery," *IEEE/RSJ 2010 Int. Conf. on Intelligent Robots and Systems*, pp. 850–856, 2010.

[15] D. Pieper, *The Kinematics of Manipulators Under Computer Control*. Ph.D. Dissertation, Department of Mechanical Engineering, Stanford University, Stanford, CA, 1968.

[16] J. Zhao and N. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *Trans. Comput. Graph.*, vol. 13, no. 4, pp. 313–336, 1994.

[17] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, pp. 47–63, 1969.

[18] R. S. Rao, A. Asaithambi, and S. K. Agrawal, "Inverse kinematic solution of robot manipulators using interval analysis," *ASME Journal of Mechanical Design*, vol. 120, no. 1, pp. 147–150, 1998.

[19] A. Ben-Israel and T. Greville, *Generalized Inverses: Theory and Applications*. Wiley, New York, 1974.

[20] J. Baillieul, J. Hollerbach, and R. Brockett, "Programming and control of kinematically redundant manipulators," *23th IEEE Conf. Decis. Contr.*, vol. Las Vegas, pp. 768–774, 1984.

[21] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics Modelling, Planning and Control*. Springer, 2009.

[22] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," *IEEE Int. Conf. Robot. Autom.*, vol. St. Louis, pp. 722–728, 1985.

[23] P. Chang, "A closed-form solution for inverse kinematics of robot manipulators with redundancy," *IEEE J. Robot. Autom.*, vol. 3, pp. 393–403, 1987.

[24] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematics problem for redundant manipulators," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 403–410, 1988.

[25] O. Egeland, "Task-space tracking with redundant manipulators," *IEEE J. Robot. Autom.*, vol. 3, pp. 471–475, 1987.

[26] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotic Research*, vol. 4, no. 2, pp. 3–9, 1985.

[27] M. Togai, "An application of the singular value decomposition to manipulability and sensitivity of industrial robots," *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, no. 2, pp. 315–320, 1986.

[28] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 22, pp. 215–221, 1955.

[29] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2006.

Appendix A

End-effector Orientation Error

Recall the derivation of the representation of a rotation of a rigid body as a matrix exponential, by considering the velocity $\dot{q}$ of a point q when it is rotated about an axis r with constant unit angular velocity at time t

$$\dot{q}(t) = r \times q(t). \tag{A.1}$$

The cross product above can also be represented as

$$\dot{q}(t) = r \times q(t) = S(r)q(t), \tag{A.2}$$

where $S(r)$ denotes the skew-symmetric matrix which is defined as

$$S(r) = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{A.3}$$

Being a time-invariant linear differential equation, (A.2) can be integrated to give

$$q(t) = e^{S(r)t}q_0, \tag{A.4}$$

where $q_0 = q(0)$ is the initial position of the point q and $e^{S(r)t}$ is the matrix exponential given by

$$e^{S(r)t} = I + S(r)t + \frac{(S(r)t)^2}{2!} + \frac{(S(r)t)^3}{3!} + \dots . \tag{A.5}$$

Hence it follows that a rotation about an axis r by unit velocity and an amount of time θ (i.e. with the resulting displacement θ) can be represented as

$$R(r, \theta) = e^{S(r)\theta}. \tag{A.6}$$

Being an infinite series, the exponential representation of a rotation in (A.5) is not desirable from a computational point of view. Fortunately, a closed-form expression can be obtained using the following properties of skew-symmetric matrices.

Lemma A.3.1:

Given $S(r) \in so(3)$ where $so(n) = \{S \in \mathrm{R}^{n\times n} : S^T = -S\}$, the following relations are true,

$$S(r)^2 = rr^T - ||r||^2 I \ , \tag{A.7}$$

$$S(r)^3 = -||r||^2 S(r) \ , \tag{A.8}$$

and the higher powers of $S(r)$ can be calculated recursively.

Using the relations in Lemma A.3.1 and taking r to be a unit-vector $||r|| = 1$, the following expression

$$e^{S(r)\theta} = I + S(r)\theta + \frac{\theta^2}{2!}S(r)^2 + \frac{\theta^3}{3!}S(r)^3 + \dots \ , \tag{A.9}$$

becomes

$$e^{S(r)t} = I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots\right) S(r) + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots\right) S(r)^2, \tag{A.10}$$

where the infinite series can be replaced by the equivalent trigonometric expressions to give

$$e^{S(r)t} = I + S(r)\sin\theta + S(r)^2(1 - \cos\theta), \tag{A.11}$$

which is the so-called Rodrigues' formula.

Definition for the orientation error

Using the axis angle representation, the rotation needed to align a given orientation with a desired one can be expressed in two terms; r which is the axis of rotation and the angle θ which is the amount of angular displacement needed about r, as

$$R(r, \theta) = R_d^0 (R_{ee}^0(q))^T, \tag{A.12}$$

where q is the vector of joint displacement variables,

$$R_d^0 = \begin{bmatrix} x_d^0 & y_d^0 & z_d^0 \end{bmatrix} = \begin{bmatrix} a_d^0 \\ b_d^0 \\ c_d^0 \end{bmatrix}, \tag{A.13}$$

is the rotation matrix describing the desired orientation of the end-effector frame with respect to the reference frame, expressed by its column vectors (x_d^0, y_d^0, z_d^0) and row vectors (a_d^0, b_d^0, c_d^0) and

$$R_{ee}^0(q) = \begin{bmatrix} x_{ee}^0(q) & y_{ee}^0(q) & z_{ee}^0(q) \end{bmatrix} = \begin{bmatrix} a_{ee}^0(q) \\ b_{ee}^0(q) \\ c_{ee}^0(q) \end{bmatrix}, \tag{A.14}$$

is the rotation matrix describing the orientation of the end-effector frame obtained using the joint variables q, expressed by its column vectors $(x_{ee}^0(q), y_{ee}^0(q), z_{ee}^0(q))$ and row vectors $(a_{ee}^0(q), b_{ee}^0(q), c_{ee}^0(q))$. The dependency on q will be omitted for clarity.

Using Rodrigues' formula (A.11), (A.12) can be rewritten as

$$I + S(r)\sin\theta + S(r)^2(1 - \cos\theta) = R_d^0 (R_{ee}^0)^T(q). \tag{A.15}$$

Denoting $c\theta = \cos\theta$, $s\theta = \sin\theta$ and $v\theta = 1 - \cos\theta$ and expressing the right-hand side in terms of the row vectors of R_d and R_{ee}, the expression above becomes

$$\begin{bmatrix} r_1^2 v\theta + c\theta & r_1 r_2 v\theta - r_3 s\theta & r_1 r_3 v\theta + r_2 s\theta \\ r_1 r_2 v\theta + r_3 s\theta & r_2^2 v\theta + c\theta & r_2 r_3 v\theta - r_1 s\theta \\ r_1 r_3 v\theta - r_2 s\theta & r_2 r_3 v\theta + r_1 s\theta & r_3^2 v\theta + c\theta \end{bmatrix} = \begin{bmatrix} a_d^0 \cdot a_{ee}^0 & a_d^0 \cdot b_{ee}^0 & a_d^0 \cdot c_{ee}^0 \\ b_d^0 \cdot a_{ee}^0 & b_d^0 \cdot b_{ee}^0 & b_d^0 \cdot c_{ee}^0 \\ c_d^0 \cdot a_{ee}^0 & c_d^0 \cdot b_{ee}^0 & c_d^0 \cdot c_{ee}^0 \end{bmatrix}, \tag{A.16}$$

where $(a_{ee}^0, b_{ee}^0, c_{ee}^0)$ and (a_d^0, b_d^0, c_d^0) are the row vectors of R_{ee}^0 and R_d^0, respectively. Taking the sum of diagonal terms on both sides (A.16)

$$(r_1^2 + r_2^2 + r_3^2)v\theta + 3c\theta = a_d^0 \cdot a_{ee}^0 + b_d^0 \cdot b_{ee}^0 + c_d^0 \cdot c_{ee}^0, \tag{A.17}$$

with $||r|| = 1$ and $v\theta = 1 - c\theta$, it becomes

$$1 + 2\cos\theta = \text{trace}(R_d^0 (R_{ee}^0)^T). \tag{A.18}$$

Hence the solution for the angle θ is obtained as

$$\theta = \cos^{-1}\left(\frac{\text{trace}(R_d^0 (R_{ee}^0)^T) - 1}{2}\right), \tag{A.19}$$

which gives $\theta \pm 2\pi n$ and $-\theta \pm 2\pi n$ as possible solutions.

By equating the off-diagonal terms in (A.16)

$$\begin{array}{l} c_d^0 \cdot b_{ee}^0 - b_d^0 \cdot c_{ee}^0 = 2r_1 s\theta \\ a_d^0 \cdot c_{ee}^0 - c_d^0 \cdot a_{ee}^0 = 2r_2 s\theta \\ b_d^0 \cdot a_{ee}^0 - a_d^0 \cdot b_{ee}^0 = 2r_3 s\theta \end{array}, \tag{A.20}$$

and assuming $\theta \neq 0$, the vector r which represents the axis of rotation, is obtained as

$$r = \frac{1}{2s\theta}\begin{bmatrix} c_d^0 \cdot b_{ee}^0 - b_d^0 \cdot c_{ee}^0 \\ a_d^0 \cdot c_{ee}^0 - c_d^0 \cdot a_{ee}^0 \\ b_d^0 \cdot a_{ee}^0 - a_d^0 \cdot b_{ee}^0 \end{bmatrix}. \tag{A.21}$$

Now that the terms r and θ which define the rotation to align an end-effector frame with a desired frame, are obtained, the orientation error based on these terms can be defined as

$$e_o = r\sin\theta. \tag{A.22}$$

Substituting the expression for r in (A.21) into (A.22)

$$e_o = \frac{1}{2}\begin{bmatrix} c_d^0 \cdot b_{ee}^0 - b_d^0 \cdot c_{ee}^0 \\ a_d^0 \cdot c_{ee}^0 - c_d^0 \cdot a_{ee}^0 \\ b_d^0 \cdot a_{ee}^0 - a_d^0 \cdot b_{ee}^0 \end{bmatrix}. \tag{A.23}$$

Expanding the dot products between the row vectors, rewriting in terms of the column vectors of R_d^0 and R_{ee}^0

$$e_o = \frac{1}{2}\begin{bmatrix} (x_d^0)_3(x_{ee}^0)_2 + (y_d^0)_3(y_{ee}^0)_2 + (z_d^0)_3(z_{ee}^0)_2 - (x_d^0)_2(x_{ee}^0)_3 - (y_d^0)_2(y_{ee}^0)_3 - (z_d^0)_2(z_{ee}^0)_3 \\ (x_d^0)_1(x_{ee}^0)_3 + (y_d^0)_1(y_{ee}^0)_3 + (z_d^0)_1(z_{ee}^0)_3 - (x_d^0)_3(x_{ee}^0)_1 - (y_d^0)_3(y_{ee}^0)_1 - (z_d^0)_3(z_{ee}^0)_1 \\ (x_d^0)_2(x_{ee}^0)_1 + (y_d^0)_2(y_{ee}^0)_1 + (z_d^0)_2(z_{ee}^0)_1 - (x_d^0)_1(x_{ee}^0)_2 - (y_d^0)_1(y_{ee}^0)_2 - (z_d^0)_1(z_{ee}^0)_2 \end{bmatrix}, \tag{A.24}$$

and regrouping the terms, the following is obtained

$$e_o = \frac{1}{2}\Bigg(\begin{bmatrix} (x_d^0)_3(x_{ee}^0)_2 - (x_d^0)_2(x_{ee}^0)_3 \\ (x_d^0)_1(x_{ee}^0)_3 - (x_d^0)_3(x_{ee}^0)_1 \\ (x_d^0)_2(x_{ee}^0)_1 - (x_d^0)_1(x_{ee}^0)_2 \end{bmatrix} + \begin{bmatrix} (y_d^0)_3(y_{ee}^0)_2 - (y_d^0)_2(y_{ee}^0)_3 \\ (y_d^0)_1(y_{ee}^0)_3 - (y_d^0)_3(y_{ee}^0)_1 \\ (y_d^0)_2(y_{ee}^0)_1 - (y_d^0)_1(y_{ee}^0)_2 \end{bmatrix} + \begin{bmatrix} (z_d^0)_3(z_{ee}^0)_2 - (z_d^0)_2(z_{ee}^0)_3 \\ (z_d^0)_1(z_{ee}^0)_3 - (z_d^0)_3(z_{ee}^0)_1 \\ (z_d^0)_2(z_{ee}^0)_1 - (z_d^0)_1(z_{ee}^0)_2 \end{bmatrix} \Bigg) \quad . \tag{A.25}$$

Recalling the definition of cross product of two vectors $a, b \in \mathrm{R}^3$ which is given as

$$a \times b = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix},$$

it is seen that (A.25) is equivalent to

$$e_o = -\frac{1}{2}\left(x_d^0 \times x_{ee}^0 + y_d^0 \times y_{ee}^0 + z_d^0 \times z_{ee}^0\right), \tag{A.26}$$

which can also be written, by using the vector cross product property $a \times b = -(b \times a)$, as

$$e_o = \frac{1}{2}\left(x_{ee}^0 \times x_d^0 + y_{ee}^0 \times y_d^0 + z_{ee}^0 \times z_d^0\right). \tag{A.27}$$

This approach is used in Section 5.1.4 and Section 5.1.5 to formulate the orientation error and the desired angular velocity of the end-effector frame, respectively.

Appendix B

Forward Kinematics

In this section, a brief summary is given for rigid body motions and coordinate vector transformations in the context of forward kinematics. See [29] for more details.

B.1 Rigid Body Rotations & Translations

In robotics, it is often necessary to represent the relative position and orientation of a rigid body with respect to another. To do this, coordinate frames are attached to each rigid body, while there is also a fixed reference frame. Geometric relations can then be specified between moving rigid bodies.

B.1.1 Position and Orientation Representation

Every coordinate frame attached to a rigid body, consists of an origin o and two or three orthogonal axes as xy or xyz, which describe position and orientation for two or three dimensional spaces, respectively.

A location of a point in three dimensional space, which is a geometric entity, can be denoted as p, while the coordinate vector which describe the position of p with respect to a coordinate frame $o_i x_i y_i z_i$ is denoted as p^i. Similary, the origin of the coordinate frame $o_i x_i y_i z_i$, denoted by o_i, can be described as a coordinate vector with respect to another frame $o_j x_j y_j z_j$ as o_i^j, which represents the position of the origin of frame $o_i x_i y_i z_i$ with respect to the frame $o_j x_j y_j z_j$.

The notation, which is used to assign coordinates to points, also applies to vectors, to describe the velocity of a point with respect to different coordinate frames.

In addition to describing the position of a moving body with respect to another frame, the relative orientation is also described, using representation of rotations. The orientation of a frame can be specified by using the coordinate vectors of its axes with respect to another frame. Hence, to describe to orientation of frame $o_1 x_1 y_1 z_1$ with respect to the frame $o_0 x_0 y_0 z_0$, the matrix

$$R_1^0 = [x_1^0 \; y_1^0 \; z_1^0] \; , \tag{B.1}$$

can be used, where x_1^0, y_1^0 and z_1^0 are the coordinates of the unit vectors x_1, y_1 and z_1 in the frame $o_0 x_0 y_0 z_0$, respectively. Such a matrix, which is formed of orthogonal unit vectors, is called a rotation matrix and has the properties

$$R^T = R^{-1}, \tag{B.2}$$

$$\det R = 1, \tag{B.3}$$

and all such $n \times n$ matrices belong to the Special Orthogonal group, denoted as

$$R \in \mathrm{SO}(n), \tag{B.4}$$

where n is typically two or three in the context of rotations in Euclidean space.

For two dimensions, a rotation by θ of frame $o_1x_1y_1$ with respect to frame $o_0x_0y_0$ is simply described as

$$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{B.5}$$

since

$$\begin{aligned} x_1^0 &= \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ y_1^0 &= \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \end{aligned}, \tag{B.6}$$

as shown in Figure B.1.

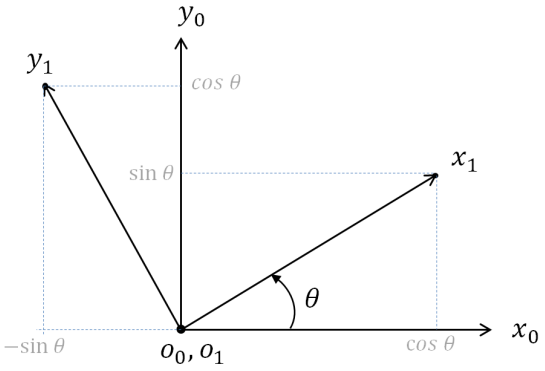

Figure B.1: Representation of a rotation between frame $o_1x_1y_1$ and $o_0x_0y_0$ by θ

Alternatively, the rotation matrix R_0^1 can obtained by the projections of the axes of frame $o_1x_1y_1$ to the axes of frame $o_0x_0y_0$ using dot product, as

$$R_0^1 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix}, \tag{B.7}$$

since

$$\begin{aligned} x_1^0 &= \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix} \\ y_1^0 &= \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix} \end{aligned}. \tag{B.8}$$

For the three dimensional case, a rotation matrix between the two frames can similary be obtained as

$$R_0^1 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}, \tag{B.9}$$

By the vector projections with dot product, basic rotations in three dimensions about each coordinate axis can be described as

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{B.10}$$

for a simple rotation about x-axis by θ

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{B.11}$$

for a simple rotation about y-axis by θ and

$$R_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{B.12}$$

for a simple rotation about z-axis by θ between two frames.

In addition to describing relative orientations of a pair of frame, rotation matrices can also be used for rotational transformation of coordinate vectors, such as the position of a point. To describe the position of a point p^0, which is expressed with respect to frame $o_0x_0y_0z_0$, in the coordinates of a different frame $o_1x_1y_1z_1$, the following relation can be used

$$p^1 = R_0^1 p^0, \tag{B.13}$$

where R_0^1 represents the orientation of frame $o_0x_0y_0z_0$ with respect to frame $o_1x_1y_1z_1$. Similarly, rotation matrices can be used to transform the coordinates of a point on a local frame, which goes under a rotation with respect to a reference frame.

B.1.2 Homogeneous Transformations

Rotations and translations can be combined into transformation matrices, which are called homogeneous transformations. A rotation with respect to one of the principal axes and a translation can be described by a homogeneous transformation matrix as

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, \tag{B.14}$$

where R and d denotes a rotational transformation matrix and a translation vector, which together describe the motion of one frame expressed in the coordinates of another frame.

B.2 Denavit-Hartenberg Parametrization

According to the DH convention [28], the homogeneous transformation between two consecutive frames, from $i-1^{\text{th}}$ to i^{th}, is defined by four parameters θ_i, d_i, a_i and α_i, as

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.15}$$

where, the first 3×3 diagonal element

$$R_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix}, \tag{B.16}$$

is the matrix defining the rotational transformation from frame $i-1$ to i and

$$p_i^{i-1} = \begin{bmatrix} a_i\cos\theta_i \\ a_i\sin\theta_i \\ d_i \end{bmatrix}, \tag{B.17}$$

is the vector defining the position of frame i with respect to frame $i-1$.

For the joint variables $q = \begin{bmatrix} q_1 & ... & q_n \end{bmatrix}^T$ where n is the number of degrees of freedom, it holds that for $i \in \{1, \ldots, n\}$, $q_i = \theta_i$ for revolute joints and $q_i = d_i$ for prismatic joints.

B.3 Forward Kinematics

As the homogeneous transformations between each pair of frames are with respect to the axes of the current local frame, the homogeneous transformation from the fixed frame to the end effector frame can be computed recursively with right multiplication as [29]

$$T_n^0 = T_1^0 \ldots T_i^{i-1} \ldots T_n^{n-1}, \tag{B.18}$$

where

$$T_n^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & (o_n^0)_x \\ r_{21} & r_{22} & r_{23} & (o_n^0)_y \\ r_{31} & r_{32} & r_{33} & (o_n^0)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.19}$$

is the matrix defining the homogeneous transformation from the fixed reference frame to the end-effector frame. It includes the matrix

$$R_n^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} x_n^0 & y_n^0 & z_n^0 \\ & & \end{bmatrix}, \tag{B.20}$$

defining the orientation of the end-effector frame described by the axes $x_n^0 y_n^0 z_n^0$, and the vector

$$o_n^0 = \begin{bmatrix} (o_n^0)_x \\ (o_n^0)_y \\ (o_n^0)_z \end{bmatrix}, \tag{B.21}$$

which gives the position of the end-effector frame origin.

By using a functional expression of T_n^0, the end-effector position o_n^0 and orientation $x_n^0 y_n^0 z_n^0$ can be obtained for a given configuration for the joint variables q.

Appendix C

Additional Figures

C.1 Intra-collisions

For the recommendations on intra-collision avoidance given in Section 7.2, Figure C.1 and C.2 are presented in this section. In Figure C.1, links 2,3 and 5 are shown with triangular re-meshes. In Figure C.2, one of the parts with a triangular re-mesh is shown with the vertex normal vectors, which are mentioned in Section 7.2.

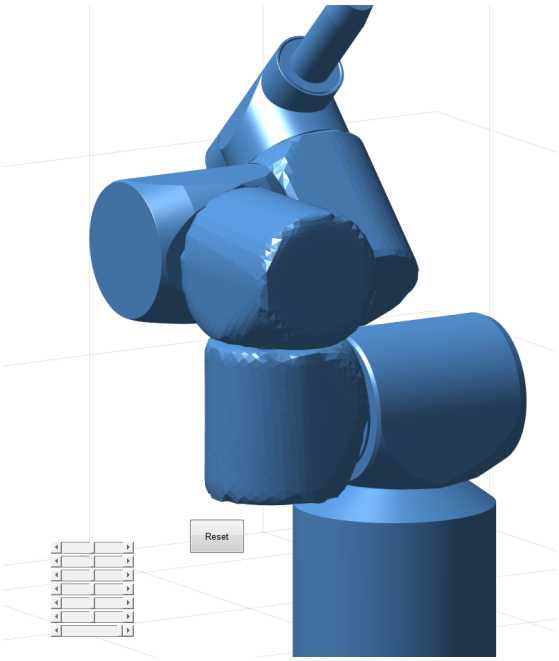

Figure C.1: RoBoSculpt with Links 2, 3 and 5 re-meshed with triangles

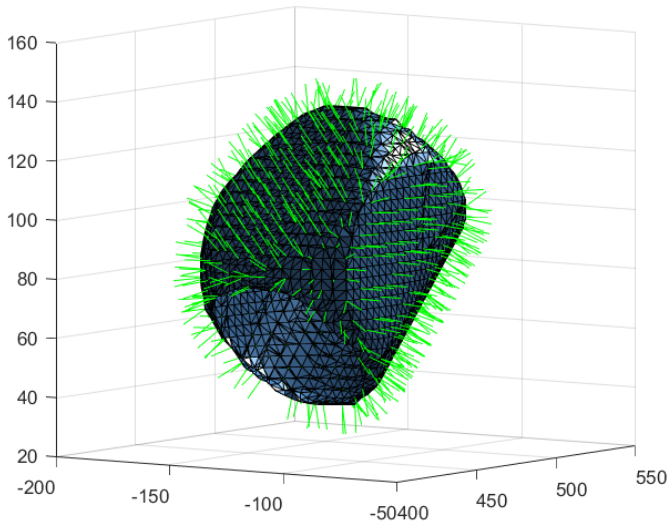

Figure C.2: A re-meshed robot part with vertex normal vectors

Appendix D

Homogeneous Transformation and Jacobian Matrices

D.1 Homogeneous Transformation Matrix of the End-effector Frame of RoBoSculpt

$$T_7^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & (o_7^0)_x \\ r_{21} & r_{22} & r_{23} & (o_7^0)_y \\ r_{31} & r_{32} & r_{33} & (o_7^0)_y \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{D.1}$$

where

$$\begin{aligned} r_{11} = \; & -c_6(c_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) - s_5(c_3s_1 + c_1c_2s_3)) \\ & +s_6(s_4(s_1s_3 - c_1c_2c_3) + c_1c_4s_2) \,, \\ r_{21} = \; & c_6(c_5(c_4(c_1s_3 + c_2c_3s_1) + s_1s_2s_4) - s_5(c_1c_3 - c_2s_1s_3)) \\ & -s_6(s_4(c_1s_3 + c_2c_3s_1) - c_4s_1s_2) \,, \\ r_{31} = \; & -c_6(c_5(c_2s_4 - c_3c_4s_2) - s_2s_3s_5) - s_6(c_2c_4 + c_3s_2s_4) \,, \end{aligned} \tag{D.2}$$

$$\begin{aligned} r_{12} = \; & s_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) + c_5(c_3s_1 + c_1c_2s_3) \,, \\ r_{22} = \; & -s_5(c_4(c_1s_3 + c_2c_3s_1) + s_1s_2s_4) - c_5(c_1c_3 - c_2s_1s_3) \,, \\ r_{32} = \; & s_5(c_2s_4 - c_3c_4s_2) + c_5s_2s_3 \,, \end{aligned} \tag{D.3}$$

$$\begin{aligned} r_{13} = \; & s_6(c_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) - s_5(c_3s_1 + c_1c_2s_3)) \\ & +c_6(s_4(s_1s_3 - c_1c_2c_3) + c_1c_4s_2) \,, \\ r_{23} = \; & -s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) + s_1s_2s_4) - s_5(c_1c_3 - c_2s_1s_3)) \\ & -c_6(s_4(c_1s_3 + c_2c_3s_1) - c_4s_1s_2) \,, \\ r_{33} = \; & s_6(c_5(c_2s_4 - c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 + c_3s_2s_4) \,, \end{aligned} \tag{D.4}$$

$$
\begin{aligned}
{}_x o_7^0 = \;& q_7\big(s_6(c_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) - s_5(c_3s_1 + c_1c_2s_3)) \\
& \quad + c_6(s_4(s_1s_3 - c_1c_2c_3) + c_1c_4s_2)\big) \\
& -d_6(s_5(c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4) + c_5(c_3s_1 + c_1c_2s_3)) \\
& +d_5(s_4(s_1s_3 - c_1c_2c_3) + c_1c_4s_2) - d_4(c_3s_1 + c_1c_2s_3) \\
& -c_1d_3s_2 + d_2s_1 \;, \\
{}_y o_7^0 = \;& -q_7\big(s_6(c_5(c_4(c_1s_3 + c_2c_3s_1) + s_1s_2s_4) - s_5(c_1c_3 - c_2s_1s_3)) \\
& \quad + c_6(s_4(c_1s_3 + c_2c_3s_1) - c_4s_1s_2)\big) \\
& +d_6(s_5(c_4(c_1s_3 + c_2c_3s_1) + s_1s_2s_4) + c_5(c_1c_3 - c_2s_1s_3)) - d_3s_1s_2 \\
& -d_5(s_4(c_1s_3 + c_2c_3s_1) - c_4s_1s_2) - c_1d_2 + d_4(c_1c_3 - c_2s_1s_3) \;, \\
{}_z o_7^0 = \;& q_7(s_6(c_5(c_2s_4 - c_3c_4s_2) - s_2s_3s_5) - c_6(c_2c_4 + c_3s_2s_4)) \\
& -d_6(s_5(c_2s_4 - c_3c_4s_2) + c_5s_2s_3) - d_5(c_2c_4 + c_3s_2s_4) \\
& +d_1 + c_2d_3 - d_4s_2s_3 \;,
\end{aligned}
\tag{D.5}
$$

where q_i, d_i, a_i and α_i are the joint variables and the DH parameters from Table 3.1 in Section 3.1.1 and s_i and c_i are abbreviations for $\sin q_i$ and $\cos q_i$, respectively.

D.2 Geometric Jacobian Matrix of the End-effector Frame of RoBoSculpt

$$
J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}, \tag{D.6}
$$

$$
\begin{aligned}
(J_v)_{1,1} = \;& c_1d_2 + d_5\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right) + d_3s_1s_2 - d_4\left(c_1c_3 - c_2s_1s_3\right) \\
& -d_6\big(s_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) + c_5\left(c_1c_3 - c_2s_1s_3\right)\big) \\
& +q_7\Big(s_6\big(c_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) - s_5\left(c_1c_3 - c_2s_1s_3\right)\big) \\
& \qquad + c_6\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right)\Big)
\end{aligned}
\;, \tag{D.7}
$$

$$
\begin{aligned}
(J_v)_{2,1} = \;& -d_6\left(s_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) + c_5\left(c_3s_1 + c_1c_2s_3\right)\right) \\
& +d_2s_1 + d_5\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right) - d_4\left(c_3s_1 + c_1c_2s_3\right) - c_1d_3s_2 \\
& q_7\Big(s_6\left(c_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) - s_5\left(c_3s_1 + c_1c_2s_3\right)\right) \\
& \qquad + c_6\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right)\Big)
\end{aligned}
\;, \tag{D.8}
$$

$$
(J_v)_{3,1} = \;\; 0 \;, \tag{D.9}
$$

$$
\begin{aligned}
(J_v)_{1,2} = \;& c_1\Big(d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) - c_2d_3 + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& \quad - q_7\left(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\right) + d_4s_2s_3\Big) \;,
\end{aligned}
\tag{D.10}
$$

$$
\begin{aligned}
(J_v)_{2,2} = \; & s_1\Big(d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) - c_2d_3 + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& - q_7\left(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\right) + d_4s_2s_3\Big) \;,
\end{aligned}
\tag{D.11}
$$

$$
\begin{aligned}
(J_v)_{3,2} = \; & -\left(c_1^2 + s_1^2\right)\Big(d_3s_2 + c_2d_4s_3 - c_4d_5s_2 - c_4c_6q_7s_2 - d_6s_2s_4s_5 + c_2c_3d_5s_4 \\
& + c_2c_5d_6s_3 - c_2c_3c_4d_6s_5 + c_2c_3c_6q_7s_4 + c_2q_7s_3s_5s_6 \\
& + c_5q_7s_2s_4s_6 + c_2c_3c_4c_5q_7s_6\Big) \;,
\end{aligned}
\tag{D.12}
$$

$$
\begin{aligned}
(J_v)_{1,3} = \; & c_2\Big(-d_6\left(s_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) + c_5\left(c_1c_3 - c_2s_1s_3\right)\right) + d_3s_1s_2 \\
& + d_5\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right) - d_4\left(c_1c_3 - c_2s_1s_3\right) \\
& + q_7\Big(s_6\left(c_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) - s_5\left(c_1c_3 - c_2s_1s_3\right)\right) \\
& + c_6\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right)\Big)\Big) \\
& + s_1s_2\Big(d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) - c_2d_3 + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& - q_7\big(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\big) + d_4s_2s_3\Big) \;,
\end{aligned}
\tag{D.13}
$$

$$
\begin{aligned}
(J_v)_{2,3} = \; & -c_2\Big(d_6\left(s_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) + c_5\left(c_3s_1 + c_1c_2s_3\right)\right) \\
& - q_7\Big(s_6\big(c_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) - s_5\left(c_3s_1 + c_1c_2s_3\right)\big) \\
& + c_6\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right)\Big) \\
& - d_5\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right) + d_4\left(c_3s_1 + c_1c_2s_3\right) + c_1d_3s_2\Big) \;, \\
& -c_1s_2\Big(d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) - c_2d_3 + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& - q_7\big(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\big) + d_4s_2s_3\Big)
\end{aligned}
\tag{D.14}
$$

$$
\begin{aligned}
(J_v)_{3,3} = \; & -s_2\left(c_1^2 + s_1^2\right)\Big(c_3d_4 + c_3c_5d_6 - d_5s_3s_4 + c_4d_6s_3s_5 - c_6q_7s_3s_4 \\
& + c_3q_7s_5s_6 - c_4c_5q_7s_3s_6\Big) \;,
\end{aligned}
\tag{D.15}
$$

$$
\begin{aligned}
(J_v)_{1,4} = \; & -\Big(- q_7\Big(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\Big) \\
& + d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& + d_4s_2s_3\Big)\left(c_1c_3 - c_2s_1s_3\right) \\
& -s_2s_3\Big(q_7\Big(s_6\big(c_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) - s_5\left(c_1c_3 - c_2s_1s_3\right)\big) \\
& + c_6\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right)\Big) \\
& - d_6\Big(s_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) + c_5\left(c_1c_3 - c_2s_1s_3\right)\Big) \\
& + d_5\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right) - d_4\left(c_1c_3 - c_2s_1s_3\right)\Big)
\end{aligned}
\tag{D.16}
$$

$$
\begin{aligned}
(J_v)_{2,4} = \; & -\Big(d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) + d_5\left(c_2c_4 + c_3s_2s_4\right) \\
& \quad - q_7\big(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\big) \\
& \quad + d_4s_2s_3\Big)\left(c_3s_1 + c_1c_2s_3\right) \\
& -s_2s_3\Big(q_7\Big(s_6\big(c_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) - s_5\left(c_3s_1 + c_1c_2s_3\right)\big) \\
& \qquad + c_6\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right)\Big) \\
& \quad - d_6\big(s_5\left(c_4\left(s_1s_3 - c_1c_2c_3\right) - c_1s_2s_4\right) + c_5\left(c_3s_1 + c_1c_2s_3\right)\big) \\
& \quad + d_5\left(s_4\left(s_1s_3 - c_1c_2c_3\right) + c_1c_4s_2\right) - d_4\left(c_3s_1 + c_1c_2s_3\right)\Big)
\end{aligned}
\tag{D.17}
$$

$$
\begin{aligned}
(J_v)_{3,4} = \; & \left(c_1^2 + s_1^2\right)\Big(c_2c_3^2d_5s_4 + c_2d_5s_3^2s_4 - c_3c_4d_5s_2 - c_3c_4c_6q_7s_2 - c_3d_6s_2s_4s_5 \\
& \quad - c_2c_3^2c_4d_6s_5 + c_2c_3^2c_6q_7s_4 - c_2c_4d_6s_3^2s_5 + c_2c_6q_7s_3^2s_4 \\
& \quad + c_2c_3^2c_4c_5q_7s_6 + c_2c_4c_5q_7s_3^2s_6 + c_3c_5q_7s_2s_4s_6\Big)
\end{aligned}
\tag{D.18}
$$

$$
\begin{aligned}
(J_v)_{1,5} = \; & \Big(- q_7\big(s_6\left(c_5\left(c_2s_4 - c_3c_4s_2\right) - s_2s_3s_5\right) - c_6\left(c_2c_4 + c_3s_2s_4\right)\big) \\
& \quad + d_6\left(s_5\left(c_2s_4 - c_3c_4s_2\right) + c_5s_2s_3\right) \\
& \quad + d_5\left(c_2c_4 + c_3s_2s_4\right)\Big)\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right) \\
& -\Big(- d_6\left(s_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) + c_5\left(c_1c_3 - c_2s_1s_3\right)\right) \\
& \quad + q_7\Big(s_6\left(c_5\left(c_4\left(c_1s_3 + c_2c_3s_1\right) + s_1s_2s_4\right) - s_5\left(c_1c_3 - c_2s_1s_3\right)\right) \\
& \qquad + c_6\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right)\Big) \\
& \quad + d_5\left(s_4\left(c_1s_3 + c_2c_3s_1\right) - c_4s_1s_2\right)\Big)\left(c_2c_4 + c_3s_2s_4\right)
\end{aligned}
\tag{D.19}
$$

$$
\begin{aligned}
(J_v)_{2,5} = \; & \Big(- q_7\big(s_6\left(c_5\left(c_2 s_4 - c_3 c_4 s_2\right) - s_2 s_3 s_5\right) - c_6\left(c_2 c_4 + c_3 s_2 s_4\right)\big) \\
& \quad + d_6\left(s_5\left(c_2 s_4 - c_3 c_4 s_2\right) + c_5 s_2 s_3\right) \\
& \quad\quad + d_5\left(c_2 c_4 + c_3 s_2 s_4\right)\Big)\left(s_4\left(s_1 s_3 - c_1 c_2 c_3\right) + c_1 c_4 s_2\right) \\
& - \Big(q_7\Big(s_6\left(c_5\left(c_4\left(s_1 s_3 - c_1 c_2 c_3\right) - c_1 s_2 s_4\right) - s_5\left(c_3 s_1 + c_1 c_2 s_3\right)\right) \\
& \quad\quad + c_6\left(s_4\left(s_1 s_3 - c_1 c_2 c_3\right) + c_1 c_4 s_2\right)\Big) \\
& \quad - d_6\big(s_5\left(c_4\left(s_1 s_3 - c_1 c_2 c_3\right) - c_1 s_2 s_4\right) + c_5\left(c_3 s_1 + c_1 c_2 s_3\right)\big) \\
& \quad + d_5\left(s_4\left(s_1 s_3 - c_1 c_2 c_3\right) + c_1 c_4 s_2\right)\Big)\left(c_2 c_4 + c_3 s_2 s_4\right)
\end{aligned}
\tag{D.20}
$$

$$
\begin{aligned}
(J_v)_{3,5} = \; & -\left(c_1^2 + s_1^2\right)\Big(c_2 c_3^2 c_5 d_6 s_4 - c_3 c_4 c_5 d_6 s_2 + c_2 c_5 d_6 s_3^2 s_4 - c_4^2 d_6 s_2 s_3 s_5 \\
& \quad - d_6 s_2 s_3 s_4^2 s_5 + c_2 c_3^2 q_7 s_4 s_5 s_6 + c_4^2 c_5 q_7 s_2 s_3 s_6 \\
& \quad + c_2 q_7 s_3^2 s_4 s_5 s_6 + c_5 q_7 s_2 s_3 s_4^2 s_6 - c_3 c_4 q_7 s_2 s_5 s_6\Big)
\end{aligned}
\tag{D.21}
$$

$$
\begin{aligned}
(J_v)_{1,6} = \; & \Big(- q_7\Big(s_6\left(c_5\left(c_4\left(c_1 s_3 + c_2 c_3 s_1\right) + s_1 s_2 s_4\right) - s_5\left(c_1 c_3 - c_2 s_1 s_3\right)\right) \\
& \quad + c_6\left(s_4\left(c_1 s_3 + c_2 c_3 s_1\right) - c_4 s_1 s_2\right)\Big) \\
& \quad + d_6\big(s_5\left(c_4\left(c_1 s_3 + c_2 c_3 s_1\right) + s_1 s_2 s_4\right) \\
& \quad\quad + c_5\left(c_1 c_3 - c_2 s_1 s_3\right)\big)\Big)\left(s_5\left(c_2 s_4 - c_3 c_4 s_2\right) + c_5 s_2 s_3\right) \\
& - \Big(- q_7\left(s_6\left(c_5\left(c_2 s_4 - c_3 c_4 s_2\right) - s_2 s_3 s_5\right) - c_6\left(c_2 c_4 + c_3 s_2 s_4\right)\right) \\
& \quad + d_6\left(s_5\left(c_2 s_4 - c_3 c_4 s_2\right) + c_5 s_2 s_3\right)\Big)\Big(s_5\left(c_4\left(c_1 s_3 + c_2 c_3 s_1\right) + s_1 s_2 s_4\right) \\
& \quad\quad + c_5\left(c_1 c_3 - c_2 s_1 s_3\right)\Big),
\end{aligned}
\tag{D.22}
$$

$$
\begin{aligned}
(J_v)_{2,6} = \; & -\Big(- d_6\Big(s_5\left(c_4\left(s_1 s_3 - c_1 c_2 c_3\right) - c_1 s_2 s_4\right) + c_5\left(c_3 s_1 + c_1 c_2 s_3\right)\Big) \\
& \quad + q_7\Big(s_6\left(c_5\left(c_4\left(s_1 s_3 - c_1 c_2 c_3\right) - c_1 s_2 s_4\right) - s_5\left(c_3 s_1 + c_1 c_2 s_3\right)\right) \\
& \quad\quad + c_6\left(s_4\left(s_1 s_3 - c_1 c_2 c_3\right) + c_1 c_4 s_2\right)\Big)\Big)\left(s_5\left(c_2 s_4 - c_3 c_4 s_2\right) + c_5 s_2 s_3\right) \\
& - \Big(- q_7\left(s_6\left(c_5\left(c_2 s_4 - c_3 c_4 s_2\right) - s_2 s_3 s_5\right) - c_6\left(c_2 c_4 + c_3 s_2 s_4\right)\right) \\
& \quad + d_6\left(s_5\left(c_2 s_4 - c_3 c_4 s_2\right) + c_5 s_2 s_3\right)\Big)\Big(s_5\left(c_4\left(s_1 s_3 - c_1 c_2 c_3\right) - c_1 s_2 s_4\right) \\
& \quad\quad + c_5\left(c_3 s_1 + c_1 c_2 s_3\right)\Big)
\end{aligned}
\tag{D.23}
$$

$$
\begin{aligned}
(J_v)_{3,6} = \; & q_7 \left(c_1^2 + s_1^2\right) \Big(c_2 s_6 c_3^2 c_4 c_5^2 + c_2 s_6 c_3^2 c_4 s_5^2 + c_2 c_6 c_3^2 c_5 s_4 - c_6 s_2 c_3 c_4 c_5 \\
& + s_2 s_6 c_3 c_5^2 s_4 + s_2 s_6 c_3 s_4 s_5^2 - c_6 s_2 c_4^2 s_3 s_5 + c_2 s_6 c_4 c_5^2 s_3^2 \;, \\
& + c_2 s_6 c_4 s_3^2 s_5^2 + c_2 c_6 c_5 s_3^2 s_4 - c_6 s_2 s_3 s_4^2 s_5 \Big)
\end{aligned} \tag{D.24}
$$

$$
\begin{aligned}
(J_v)_{1,7} = \; & s_6 \big(c_5 \left(c_4 \left(s_1 s_3 - c_1 c_2 c_3 \right) - c_1 s_2 s_4 \right) - s_5 \left(c_3 s_1 + c_1 c_2 s_3 \right) \big) \\
& + c_6 \left(s_4 \left(s_1 s_3 - c_1 c_2 c_3 \right) + c_1 c_4 s_2 \right) \;,
\end{aligned} \tag{D.25}
$$

$$
\begin{aligned}
(J_v)_{2,7} = \; & - s_6 \left(c_5 \left(c_4 \left(c_1 s_3 + c_2 c_3 s_1 \right) + s_1 s_2 s_4 \right) - s_5 \left(c_1 c_3 - c_2 s_1 s_3 \right) \right) \\
& - c_6 \left(s_4 \left(c_1 s_3 + c_2 c_3 s_1 \right) - c_4 s_1 s_2 \right) \;,
\end{aligned} \tag{D.26}
$$

$$
(J_v)_{3,7} = \; s_6 \left(c_5 \left(c_2 s_4 - c_3 c_4 s_2 \right) - s_2 s_3 s_5 \right) - c_6 \left(c_2 c_4 + c_3 s_2 s_4 \right) \;, \tag{D.27}
$$

$$
\begin{aligned}
(J_\omega)_{1,1} = \; & 0, \\
(J_\omega)_{2,1} = \; & 0, \\
(J_\omega)_{3,1} = \; & 1, \\
(J_\omega)_{1,2} = \; & s_1, \\
(J_\omega)_{2,2} = \; & -c_1, \\
(J_\omega)_{3,2} = \; & 0, \\
(J_\omega)_{1,3} = \; & -c_1 s_2, \\
(J_\omega)_{2,3} = \; & -s_1 s_2, \\
(J_\omega)_{3,3} = \; & c_2, \\
(J_\omega)_{1,4} = \; & -c_3 s_1 - c_1 c_2 s_3, \\
(J_\omega)_{2,4} = \; & c_1 c_3 - c_2 s_1 s_3, \\
(J_\omega)_{3,4} = \; & -s_2 s_3, \\
(J_\omega)_{1,5} = \; & s_4 \left(s_1 s_3 - c_1 c_2 c_3 \right) + c_1 c_4 s_2, \\
(J_\omega)_{2,5} = \; & c_4 s_1 s_2 - s_4 \left(c_1 s_3 + c_2 c_3 s_1 \right), \\
(J_\omega)_{3,5} = \; & -c_2 c_4 - c_3 s_2 s_4, \\
(J_\omega)_{1,6} = \; & -s_5 \left(c_4 \left(s_1 s_3 - c_1 c_2 c_3 \right) - c_1 s_2 s_4 \right) - c_5 \left(c_3 s_1 + c_1 c_2 s_3 \right), \\
(J_\omega)_{2,6} = \; & s_5 \left(c_4 \left(c_1 s_3 + c_2 c_3 s_1 \right) + s_1 s_2 s_4 \right) + c_5 \left(c_1 c_3 - c_2 s_1 s_3 \right), \\
(J_\omega)_{3,6} = \; & -s_5 \left(c_2 s_4 - c_3 c_4 s_2 \right) - c_5 s_2 s_3, \\
(J_\omega)_{1,7} = \; & 0, \\
(J_\omega)_{2,7} = \; & 0, \\
(J_\omega)_{3,7} = \; & 0,
\end{aligned} \tag{D.28}
$$

where q_i, d_i, a_i and α_i are the joint variables and the DH parameters from Table 3.1 in Section 3.1.1 and s_i and c_i are abbreviations for $\sin q_i$ and $\cos q_i$, respectively.