In-Hand Pose Estimation Through Purely Tactile Perception and In-Hand Manipulation

A Master Thesis

written by

Victor Melbye Staven
vista17@student.sdu.dk

The code for this project is available at
`https://github.com/vmstavens/in_hand_pose_estimation`

University of Southern Denmark
The Technical Faculty

Word Count : 962
May 20, 2023

Abstract

Some abstract text explaining the goal, methods and conclusion of the project.

Contents

Acknowledgements

I would like to express my sincere gratitude to those who have supported and guided me throughout my thesis project.

First and foremost, I would like to extend my deepest appreciation to my thesis supervisor, Christoffer Sloth, Lector at SDU Robotics, for his exceptional guidance and support throughout the entire process. His expertise and knowledge in the field of robotics have been invaluable in shaping this project.

I would also like to thank Yitaek Kim for his invaluable support and sparring in solving key aspects of this project. His insights and advice have been critical to the successful completion of this thesis.

Furthermore, I would like to extend my thanks to Shadow Robotics for providing the underlying software for the project to build upon along with technical support, enabling this project. Their contribution has been essential in the successful completion of this project.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout this journey. Their love and support have been a constant source of motivation and inspiration.

Once again, I express my deepest gratitude to all those who have played a significant role in this project.

Acronyms

AEBM analytical elasticity-based models.
CAD Computer Aided Design.
cGAN conditional Generative Adversarial Network.
cobots collaborative robots.
CP correspondence problem.
CPD Coherent Point Drift.
CSGM Cross-Source Graph Matching.
CV computer vision.
DeepGMR Deep Gaussian Mixture Registration.
DL deep learning.
DOF degrees of freedom.
EE end effector.
EFM elastic foundation models.
FEM finite element models.
FGM Factorized Graph Matching.
FMT Fast Marching Tree.
FPFH Fast Point Feature Histograms.
GK grasp kinematics.
GMM Gaussian Mixture Model.
GNC Graduated Non-Convexity.
GT Ground Truth.
HF hard finger.
ICP Iterative Closest Point.
IEP Inverse Elasticity Problem.
IHM In-Hand Manipulation.
JRMPC Joint Registration of Multiple Point Sets.
MIM Matrix Inversion Method.
ML machine learning.
MLP Multi Layered Perceptron.
OMPL The Open Motion Planning Library.
PC point cloud.
PCR point cloud registration.
PD Proportional-Derivative.
PE pose estimation.
PID Proportional-Integral-Derivative.
PNP pick-and-place.
PRM Probabilistic Roadmap Method.
PwoF point-contact-without-friction.
QAP Quadratic Assignment Problem.
QCQP Quadraticaly Constrained Quadratic Programming.
QP Quadratic Programming.
RANSAC Random Sample Consensus.
RCQP Relaxed Convex Quadratic Programming.
ReLU Rectified Linear Unit.
RLS Recursive Least Squares.
ROM Range of Motion.
ROS Robot Operating System.
RRT Rapidly-exploring Random Tree.
SDH Shadow Dexterous Hand.
SDP Semi-Definite Programming.
SF soft finger.
SOCP Second Order Cone Programming.
TCP Tool Center Point.
TP Tactile Perception.
UR Universal Robots.

Terms

collaborative robots (cobots) are robots which facilitate human-robot collaboration [1].

computer vision (CV) is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs - and take actions or make recommendations based on that information [2].

correspondence problem (CP) is the problem where one aims at finding correspondences between the pixels in two (or more) images [3].

deep learning (DL) are methods that allow computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [4].

docker container , a docker container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another [5].

end effector (EE) is a generic term for all functional units involved in direct interaction of the robot system with the environment or with a given object [6].

manipulator : A serial robot mechanisms. The robot manipulator is represented by a serial chain of rigid bodies, called robot segments, connected by joints [7].

point cloud registration (PCR) is a generic term for all functional units involved in direct interaction of the robot system with the environment or with a given object [6].

pose estimation (PE) A particular instance of feature-based alignment, which occurs very often, is estimating an object's 3D pose from a set of 2D point projections. This pose estimation problem is also known as extrinsic calibration [8].

Robot Operating System (ROS) is a set of open-source software libraries and tools that help build robot applications. [9].

Chapter 1

Introduction

1.1 Context

As of 2022 most of the industrialized world has developed tools for unprecedented growth in wealth and technology on a global scale [10, Chapter 4]. In such times a great deal of consumerism and interconnection is present with people needing products produced faster and more consistently than ever before [10, Chapter 4]. As one would expect, this creates a high demand for manufacturers to reliably and consistently provide products, while also remaining flexible as the demand for different product change rapidly. To accommodate the need for ever-greater volumes of products, consistent, reliable and flexible labor is essential in assembly, transport and manipulation processes in the production pipeline. Due to these types of manual labor being largely done by unskilled workers, automation alternatives are being adopted which provide benefits [10, Chapter 4]. This different approach to manufacturing has been labeled the fourth industrial revolution or i4.0 for short. The beneficiaries are the employer and employee, with the employer having the benefits: Avoid paying monthly salaries to unskilled laborers doing manual tasks, here the automation alternative only requires electrical energy and potential supervision by a few qualified individuals. Potential risks are also involved when hiring humans as the workforce can be inconsistent due to human error [11] or left out due to illness etc. Considerations about workers' rights such as working conditions and wages also need not be considered. Workers furthermore cause production limitations in the form of stand-still hours, such as bathroom and lunch breaks along with after-work hours and holidays. This replacement of manual labor also potentially benefits the employee, as boring and physically wearing work is automated, enabling the employees to take on different and less wearing and potentially dangerous roles. While the issue of labor unemployment becomes apparent solutions that provide support to already hired workers have been developed, such as collaborative robots (cobots) [12] which would negate this problem.

When implementing automation of production lines using robotics, certain categories of problems are revealed. These include assembly, alteration and pick-and-place (PNP), the last being the one of interest in this project.

1.2 Problem Description

Pick-and-place manipulators are used in a wide variety of different fields such as sorting of waste [13] handling of food [14, 15] and factory bin picking [16, 17, 18]. The solutions in these industries are examples of subcategories under the PNP problem, namely sorting and bin picking. Since both of these are subcategories of the PNP problem, they fundamentally follow the same sequential four phases from start to end. These phases are pre-grasping, grasping, transport, and placement [17] for traditional implementations of the PNP pipeline. The pre-grasp phase involves localizing the object(s), potentially estimating their pose and executing the trajectory to move the end effector (EE) grasp, collision-free to said object(s). Here different potential grasps can be considered to determine the best pose for the EE. In the grasping phase, the EE gasps the object in such a manner that the object's entire weight is supported by the EE, and ends when the object no longer is in contact with the environment, which often is the container holding the object. The transportation phase involves the motion of the manipulator to move from the pose achieved after the grasping phase, to a pose ready for placement of the object in the desired placing area or fixture. Here considerations may be needed about how much force and torque the EE's grasp can tolerate while moving without losing the object. Finally, the goal of the placing phase is to place the object within the placing

area or fixture in a desired end pose. Here the constraints on the end pose might differ significantly based on the application, as the pose of greens in a crate might need less precision than if the manipulator hands a bolt to another robotics system in the pipeline.

While these phases make up a traditional PNP system, certain assumptions are made regarding the objects of interest for this pipeline to function. Specifically, the localization and pose estimation (PE) of the pre-grasp phase are assumed possible due to either ensured object poses or estimated poses through computer vision (CV) sensory system. Due to CV being a mature research field a wide range of solution proposals to these problems have been generated [19]. These include classic vision [20, 21], deep learning (DL) based [22] and combinations of these [23]. However, while these may be sufficient for certain tasks they fundamentally suffer from the weaknesses introduced by vision techniques. These are a great number of outliers caused by: occlusions, reflecting, transparent or homogeneous surfaces, and repetitive structures when solving the correspondence problem (CP). Within factory settings, the common ones are transparent and reflective objects, due to metallic, plastic and glass products often being the materials used. While DL solutions have been developed for both reflective [24] and transparent [25] objects, these are use case specific and show limited results in a wider range of applications.

This project suggests a different PNP pipeline for cases where the object's starting pose is unknown. In this PNP pipeline the PE is moved from the pre-grasping phase to a new phase between the grasping and transportation phase, called the PE phase. The specific goal of this project is to develop a solution to this phase using tactile sensors in the EE to determine the object's pose. By using tactile sensing rather than visual the problems presented above will be eliminated. This will be done using a humanoid gripper as the EE with tactile sensors in each finger, more specifically a Shadow Dexterous Hand (SDH) [26] with 24 joints and 20 degrees of freedom (DOF).

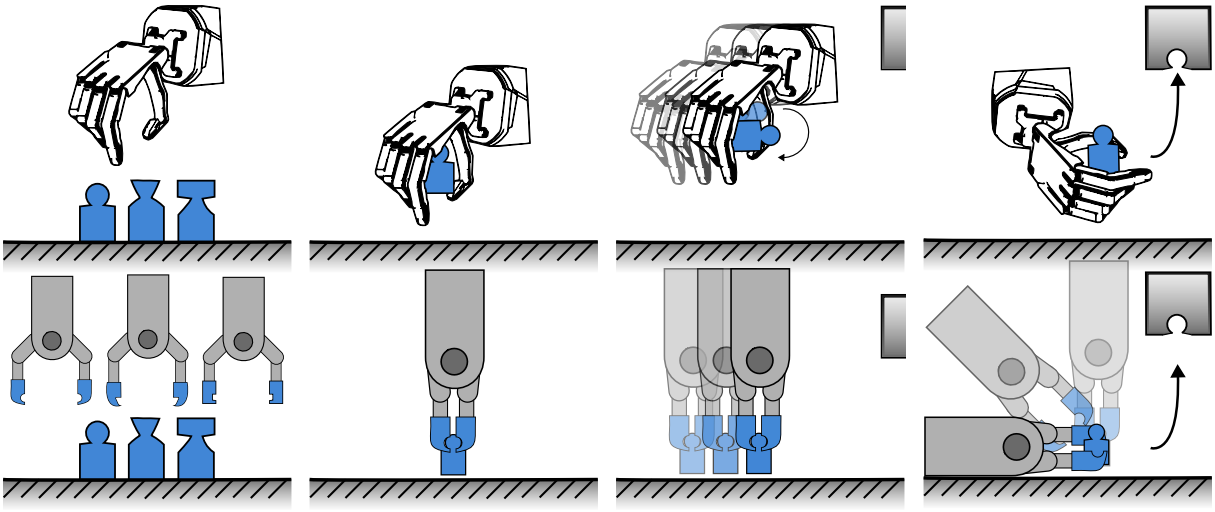
The alternative pipeline this project will enable can be seen in the upper row of Fig. 1.1 compared to the traditional pipeline in the lower row.

In Fig. 1.1(a) the pre-grasping phase can be seen for both pipelines. Here the traditional pipeline in cases of multiple objects often will employ custom fingertips or grippers entirely to facilitate form closure grasps, due to the grippers not having the flexibility to perform reliable force closure. On the contrary force closure can be performed with a humanoid gripper on a wide range of objects with no need for changing gripper equipment.

In Fig. 1.1(b) the grasping phase can be seen which introduces a greater level of complexity when using the suggested pipeline due to the humanoid gripper being a more complex physical system to represent and control. This is compared to the simplicity of executing potential binary commands in the traditional pipeline e.g.open and close.

In Fig. 1.1(c) the transportation phase can be seen, which introduces one of the benefits of using the suggested pipeline. Here tactile sensors in a humanoid gripper can pose and estimate the object and manipulations can be performed to change the object's pose such that easier placement can be performed in the following phase. This form of object manipulation is not feasible for the simple pneumatic grippers used in a traditional pipeline.

In Fig. 1.1(d) the placement phase can be seen, which demonstrates the result of the previous phase, as the traditional pipeline now has to change the grip on the object to properly place it in the socket, while the humanoid gripper simply can insert the part, as it is already oriented properly.



(a) Pre-grasp phase. **(b)** Grasping phase. **(c)** Transportation phase. **(d)** Placement phase.

Fig. 1.1: A comparison between the traditional and suggested PNP pipeline.

To solve this PE problem, three sub-problems are identified and labeled problems 1, 2 and 3.

Problem 1 involves modeling the contact between the gripper's tactile sensors and the object, also referred to as tactile perception.

Problem 2 is to convert the collected data from problem 1 to estimated pose candidates.

Problem 3 involves in-hand manipulation. Since the initial grasp of the object might not be oriented in a manner where the recognizable features make context with the tactile sensors, manipulating the object within the EE's grasp will enable further information gathering. Thus the final problem is to control the EE in such a manner that the tactile sensors make context with the object in intelligently decided areas for a better pose estimate.

To test if the developed system successfully solves the PE problem, it is hypothesized that the intelligent probing method provides a statistically significant faster average PE convergence, along with a statistically significant greater success rate when determining the correct pose. A correct pose is here defined as the pose being greater than or equal to 95 % of the ground truth pose, and statistically significant is defined by an α-level of 95 %. This hypothesis will be referred to as H_1, while the null hypothesis H_0 being that there is no statistically significant difference between intelligent and random probing's PE performance as described above.

1.3 Thesis Overview

This project is composed of 9 chapters, each listed below with an associated description.

Chapter 1: The introduction presents the historical context of the project along with a problem description and thesis overview. The problem description contains the decomposition of the project into sub-problems which will be addressed in the following chapters.

Chapter 2: The system setup presents an overview of the practical details of the project in the form of the system setup along with the code developed to execute the project.

Chapter 3: The literature review addresses the three problems identified in Chapter 1 by providing a thorough literature review on older as well as newer methods for solving each of the identified problems. To end each section, groups are compared and a method is chosen among the ones presented.

Chapter 4: The modeling chapter presents the physical modeling of the system and defines important mathematical notations and quantities as used in the following chapters.

Chapter 5: The Tactile Perception (TP) chapter addresses the method chosen in Chapter 3 to solve the first problem. This chapter expands on the method chosen and its functionality, followed by an evaluation of said method through experiments and their results. Finally, the functionality of the method is concluded.

Chapter 6: The PE chapter addresses the method chosen in Chapter 3 to solve the second problem. This chapter follows the same structure as Chapter 5.

Chapter 7: The In-Hand Manipulation (IHM) chapter addresses the method chosen in Chapter 3 to solve the third problem. This chapter follows the same structure as Chapter 5 and Chapter 6.

Chapter 8: The discussion chapter goes over the combined system and addresses problems, improvements, shortcomings and potential for future development.
Chapter 9: The conclusion chapter addresses the success of the project.
Thus the hypothesis of this projects H_1, will be testing if intelligent probing for strong features increases in-hand pose estimation performance, with the null hypothesis H_0 being that there is no statistically significant difference in the pose estimation performance of the system if the probing is done randomly or intelligently at a certainty level of 95%. Here pose estimation performance is quantified in terms of mean execution time for estimating the pose with an accuracy greater than 95%.

The development of this project is done in the docker container provided by Shadow Robotics for simulation, control and development of the hand [27]. Here a hardware-simulation agnostic Robot Operating System (ROS) [9] control [28] interface is found, which contains fundamental tools to interact with the robot hand. The dynamic simulation environment Gazebo [29] is likewise packaged as part of the docker container and is thus the one used for this project. To solve the problems presented, the ROS packages in Table **??** will be applied, where `ros_utils` and `in_hand_pose_estimation` will be developed during this project.

Chapter 2

System Setup

The SDH [26] is a sophisticated robotic hand with a wide range of sensory feedback capabilities. To develop and test algorithms for this complex system, simulation is an invaluable tool. In this chapter, the practical setup of the project is presented, which involves simulating the SDH within the dynamic simulator, Gazebo [29].

This simulation is based on the Shadow Robot Company's [30] ROS [9] packages [31], which provide a flexible and customizable framework for controlling the hand. This project uses Gazebo, a popular robot simulation environment, to simulate the physics of the hand and its interaction with the environment. To ensure reproducibility and portability, the development framework which encapsulates the simulation environment is built within a Docker container, which allows for easily distributing the code and dependencies to other researchers. Simulating the system additionally enables Ground Truth (GT) values to be easily available, and safety is less of a concern, as a broken simulated hand, and additional equipment including sensors cost are potentially easier available in the form of simulated sensor inputs.

This simulation is based on the ROS packages developed by the Shadow Robot Company [30, 9, 31]. These packages offer a flexible and customizable framework for controlling the hand. The simulation utilizes Gazebo, a widely-used robot simulation environment, to accurately model the hand's physics and its interactions with the environment. To ensure reproducibility and ease of distribution, the development framework, including the simulation environment, is encapsulated within a Docker container. This containerized approach allows for seamless sharing of the code and its dependencies with other researchers, promoting collaboration and portability.

Simulating the system provides the added benefit of readily accessible GT values, ensuring accurate analysis and evaluation of algorithms. Moreover, safety concerns are mitigated in the simulation as the risk of damaging physical hardware or incurring additional costs for equipment and sensors is eliminated. Instead, these aspects can be readily simulated through virtual sensor inputs and virtual components.

In this chapter, a detailed description of the hardware and software used in this project's simulation setup is provided, which includes the software architecture of the simulation, including the ROS packages and Gazebo setup.

2.1 Simulation Setup

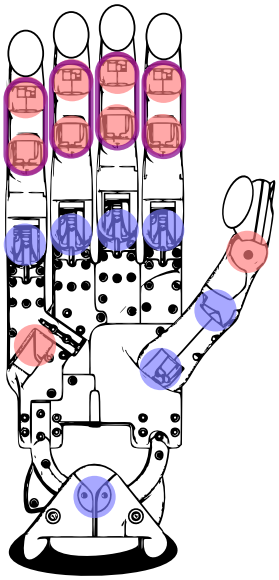
The Computer Aided Design (CAD) model of the SDH is a highly detailed and accurate representation of the physical hand. The model is based on the original design of the real-world hand, which was developed by the Shadow Robot Company. The CAD model includes precise geometry for all of the hand's components, including the finger joints, tendons, and simple tactile sensors. The model also includes detailed material properties for each component, which are used to simulate the hand's physical behavior in the simulation.

The SDH's joints are labeled based on joint location according to Appendix A.3 and placement from the fingertip i.e. the outermost joint on the middle finger is labeled `MF1` while the innermost is `MF4`. The wrist joints are labeled `WR1` and `WR2`, where `WR1` is responsible for the hand's yaw and `WR2` is responsible for the hand's pitch rotation.

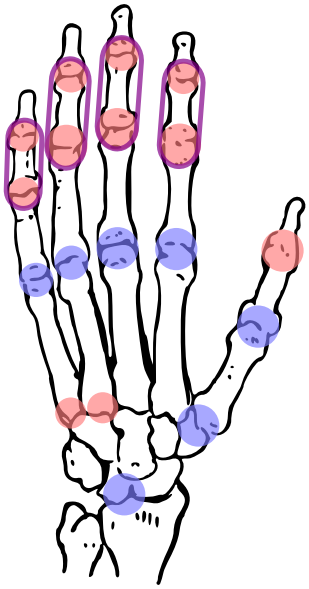
The structure of the hand can be seen in Fig. 2.1(a) compared to a human hand in Fig. 2.1(b). Here each joint is marked with a color, labeling the DOF for that particular joint. The red labels refer to joints with 1 DOF, blue refers to joints with a DOF equal to 2 and purple refers to two joints that are coupled. The coupling between joints is such that, a flexion of joint 1 imposes a constraint on joint 2. If joint 1 exceeds a certain angle, it enforces joint

2 to flex accordingly to maintain the established constraint. Both joints are mechanically linked to a single motor, functioning as a combined joint denoted as joint 0 for control purposes.

As seen here the SDH provides human-like dexterity due to its similar kinematics, see Fig. A.1, and the comparable Range of Motion (ROM) of each joint, see Table A.1 and Table A.2.



(a) SDH with joints color coded depending on the degrees of freedom. The total number of degrees of freedom can here be seen as 24. This figure is based on [32].



(b) SDH with joints color coded depending on the degrees of freedom. The total number of degrees of freedom can here be seen as 25 [33]. This figure is based on [34].

Fig. 2.1: The SDH and a human hand red here marks a joint with 1 degree of freedom, while blue marks a joint with 2.

The hand's geometry is modeled using a combination of standard shapes and custom-designed components. For example, the finger joints are modeled using a series of cylinders and spheres, which are connected by virtual tendons to simulate the motion of the real-world hand. The tactile sensors on the simulated hand, at the writing of this thesis, are purely aesthetical as representative simulated tactile sensors are yet to be supported as a standard component. To generate representative tactile data additional software is therefore required. The tactile sensors can be seen in Fig. 2.1(a) as the ellipsoids mounted at each fingertip.

Multiple hand configurations are available including a left hand, right hand, and both configurations mounted on Universal Robots (UR) manipulators [35]. The configuration chosen for this project is a Shadow Dexterous right hand without being mounted on a manipulator. Fig. 2.2 shows the CAD model of the SDH in simulation.

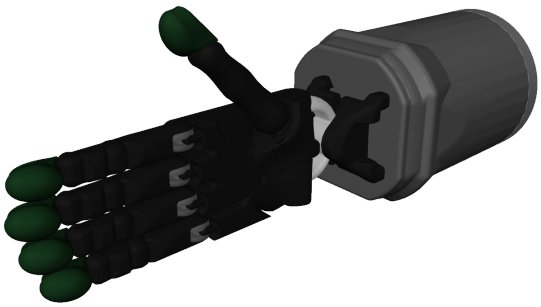


Fig. 2.2: A cutout of the simulated SDH.

2.2 Software Setup

The software for this project consists of two parts the software provided and the software produced.

2.2.1 Provided Software

The simulation environment is shipped in a custom Ubuntu-based docker container [5, 36] with the necessary libraries to communicate and develop applications on the simulated as well as the physical hand, wrapped within a catkin workspace [37]. Additionally, the container comes with common-use libraries for Python and C++ development in ROS including `numpy`[38], `OpenCV` [39] and `dynamic_reconfiguration` [40]. The development environment can be seen illustrated in Fig. 2.3. The communication between the provided ROS packages and the Gazebo simulator is achieved through the ROS-Gazebo framework.

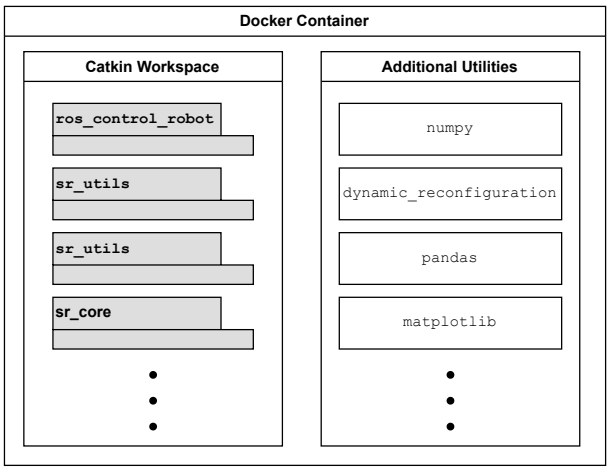


Fig. 2.3: The boxes marked with grey are ROS packages, while the white are modules.

When communicating with the SDH, the primary interface provided is the hand commander i.e. `SrHandCommander` which enables functionalities such as retrieving the current state of the hand, executing given path plans etc. To plan and execute a high-resolution path $\mathbf{Q}_{\text{full}} \in \mathbb{R}^{m\times 24}$, where m is the number of joint configurations and 24 is the number of joints in the hand, a sequence of waypoints $\mathbf{Q}$ form a low-resolution path of $\boldsymbol{q}_i = [q_0,\ q_1,\ \ldots,\ q_n]^\top \in \mathbb{R}^{24}$ where $i \in \{0, 1, \ldots, m_w\}$ with m_w being the number of waypoints. $\mathbf{Q}$ can thus be written as

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{q}_1^\top \\ \boldsymbol{q}_1^\top \\ \vdots \\ \boldsymbol{q}_{m_w}^\top \end{bmatrix} \in \mathbb{R}^{m_w\times 24}. \tag{2.1}$$

`SrHandCommander` parses the path to the `move_group` handled by MoveIt [41]. After receiving the plan, MoveIt first checks the validity of the start and goal states concerning the robot's joint limits, collision constraints, and other constraints like self-collision avoidance. This helps to ensure that the planned path is feasible and safe for the robot to execute. Once validated, the validated path $\mathbf{Q}_{\text{valid}} \in \mathbb{R}^{m_q\times 24}$ is parsed to The Open Motion Planning Library (OMPL) [42] where a safe high-resolution path is built using some chosen sample-based single- or multi-query path planning method. Some examples of these methods include Probabilistic Roadmap Method (PRM), Fast Marching Tree (FMT) and Rapidly-exploring Random Tree (RRT)-Connect, the last of which is the default used in the provided software and the one chosen for this project. To execute the path the development environment provides `SrJointPositionController` i.e. a joint space position controller by default, which is the one chosen for this project.

The controller is built using the `ros_control` [28] framework. The `ros_control` package provides a hardware-agnostic interface, to enable the same controllers on multiple different robotics platforms, along with the same controller being applicable on real hardware as well as simulated hardware. Finally, the hardware interface communicates with a 20 Proportional-Integral-Derivative (PID) controllers, one for each independent joint to ensure control. The software architecture for communicating with the robotic hand can be seen in Fig. 2.4, which was inspired by figures from [43, 44, 45].

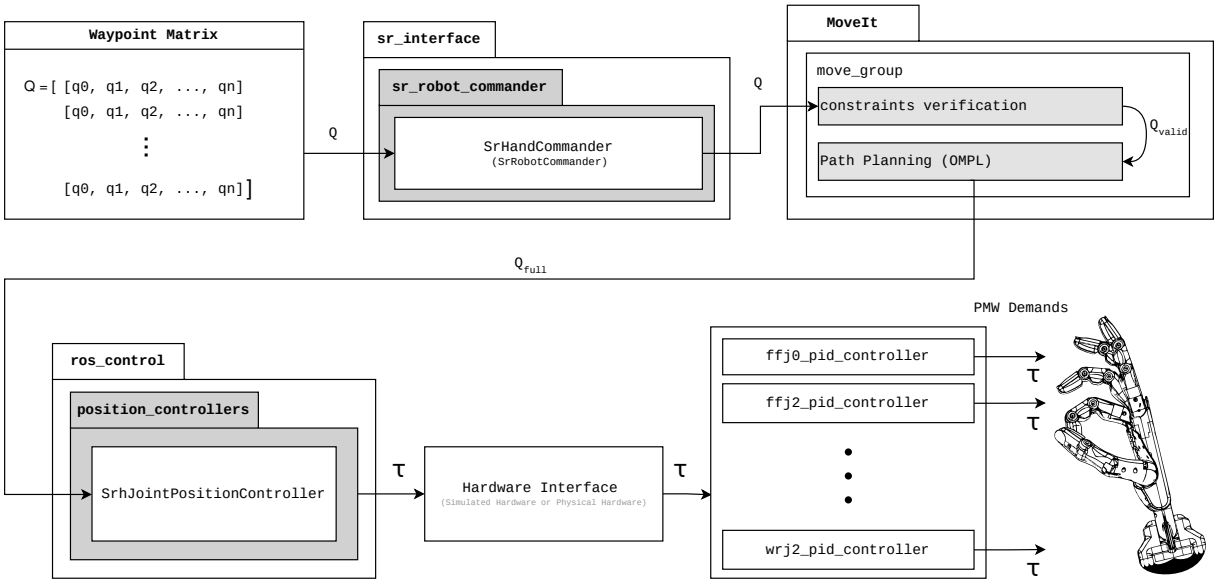


Fig. 2.4: Diagram showing the communication and control flow of interacting with the SDH.

When executing an example path Fig. 2.5 shows joint angles during execution on the simulated hand throughout 15 s. The path here consists of five waypoints for `FFJ2` and `FFJ3`, whereas none is set for `FFJ1`. This is to demonstrate the coupling as `FFJ1` still follows the motion of `FFj2` even though no motion is commanded.

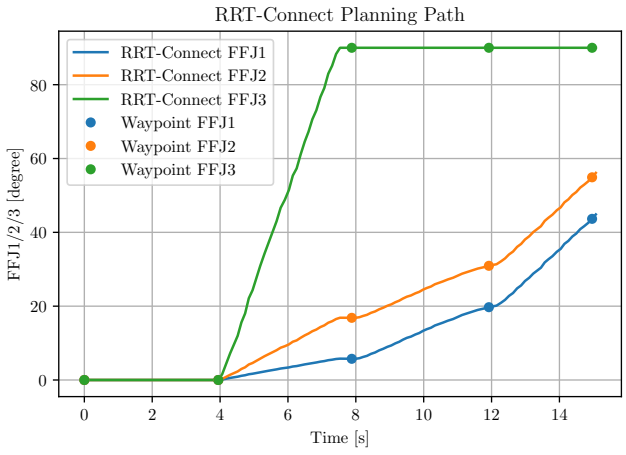


Fig. 2.5: Showing the joint trajectory of the first finger's joints `FFJ1` to `FFJ3`, where the coupling between `FFJ1` and `FFJ2` is shown.

2.2.2 Produced Software

To execute this project, software was developed to communicate with and extract data from the simulated SDH. The software is structured semantically in a similar way as the hand itself, as a `ShadowHand` contains five fingers, each labeled as one of `TH`,`FF`,`MF`,`RF` or `LF` and a `ShadowWrist` A.3. Each finger is equipped with a reader that samples the tactile data provided by Gazebo's physics engine at 100 Hz. This wrapper is written to an easy-to-use interface to the hand which provides reliable bookkeeping of sensor data. The structure can be seen illustrated in Fig. 2.6. This code is provided in a ROS package `shadow_hand` at [46]. Additionally, custom tools for easy launching and execution of experiments with live plotting of tactile data are provided with additional utilities such as base64 encoding and decoding at `ros_utils` [47].

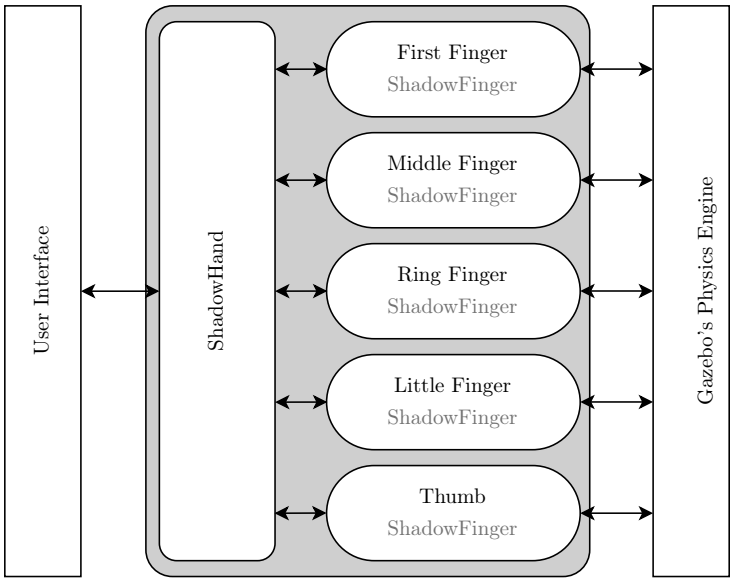


Fig. 2.6: Produced software architecture showing the wrapper and API structure.

Chapter 3

State of the Art

3.1 Problem 1 - Tactile Perception

Based on the contact model categories described in Chapter 4, the most representative was chosen to be soft finger (SF) models. Within the category of SF models, a method fit for this project's use case is to be chosen to solve problem 1. SF models can furthermore be divided up into three different categories: analytical elasticity-based models (AEBM), elastic foundation models (EFM), finite element models (FEM) [48] and machine learning (ML) models. The different categories can be seen organized in Fig. 3.1

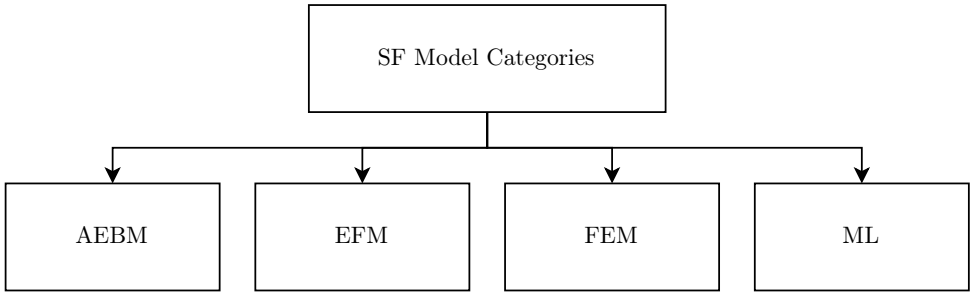


Fig. 3.1: Tree of methods for tactile perception.

AEBM are theoretical formulations of elastic contact areas and the stresses on both the surfaces and the sub-surfaces of the contacting bodies. The first of such models was introduced by Heinrich Rudolf Hertz in 1882 [49] and is still used for simple contact cases. In the formulation of the Hertzian contact model, two assumptions are made: Objects in contact are made of linear elastic materials and only small contact deformations occur compared to the dimension of an object. However, robotic EE fingertips are often made of non-linear elastic materials and for that reason, the Hertzian contact model does not represent the type of contact in this project [50, Chapter 37]. To improve on the Hertzian model, a more general formulation can be made which extends the model from linear to nonlinear elastic contacts [51, 52]. This power-law formulation subsumes the Hertzian contact theory while assuming a circular contact area. Other models have been purposed that combine the descriptions of both friction-contact and the shear-torsion as experienced by the bodies [53].

However, to more accurately describe the contacts involving robot fingers, viscoelastic soft contact model appear more relevant due to such fingers often being made of materials that show viscoelastic properties e.g., rubber, silicone and polymers. Simple models such as Kelvin-Voigt's [54] and Maxwell's [55] models describe the interaction between strain and stress as a spring-damper system in a serial or in a parallel configuration respectively. Models which expand on this idea describe the reacting force as the product of the temporal and the elastic response while incorporating previous stress responses [56]. To simplify this formulation alternatives have been developed to assume no past stress [57, 58, 59]. Upon these, more modern techniques have been developed which have seen use in similar use cases as the ones of interest in this project. One method attempts to expand the description of contacts between rigid indentors and elastic half-spaces, using the Matrix Inversion Method (MIM) as introduced by Kalker [60], to viscoelastic half-spaces as well. Assuming the surfaces are frictionless, the relationship is described in terms of the pressure distribution, the resultant force on the indenter and the penetration [61]. Attempts involving solutions to Boussinesq's problem for polynomial pressures acting over polygonal domains [62]

have also been developed and modernized by combining it with Cerruti's solution [63]. However due to numerical singularities being present, modifications are made to threshold the model. For a more complete description without singularities, Love's formulation has been added leading to a more accurate analytical representation but with the cost of an increased computational complexity [64]. For these Boussinesq-based approaches to be representative two assumptions are made 1) There exists a linear relationship between stress and strain, referred to as deformation, and 2) strains are infinitesimal [65, Chapter 6].

EFM are methods developed to build upon AEBM by allowing a simple discrete contact calculation in more general surface geometries. Here the deformable part of the contact is modeled as a layer over a rigid base with a series of discrete and independent springs in the contact normal. A widely used example of this method is Winkler's elastic foundation model [66], which has been used in structural engineering for modeling different properties of beams such as stability [67], vibrations and buckling [68]. Other EFM methods have shown accurate modeling performance when applied within the field of medical engineering. Here a comparative study between AEBM, EFM and FEM demonstrate the suggested modified EFM performs better than the alternatives in 3D knee models when predicting prosthetic knee performance [48]. A different method attempts to attain vivo contact pressure predictions for improved knee replacement designs [69] Within the field of robotics EFM have provided solutions to problems such as slip [53], compliance, sliding [70, 71], stiffness and contact mechanics [72] of anthropomorphic grippers. One such method derives friction constraints based on general expressions for non-planar contacts of elastic bodies, where the local geometry and structure of the objects in contact are taken into account. Using these, a linear complementary problem is formulated and solved, resulting in the normal and frictional forces applied at each contact, as well as the relative velocity of the bodies involved [73].

FEM are popular general tools for solving PDE [74] and have seen contact applications in a wide range of engineering disciplines due to the assumptions made in AEBM and EFM not being applicable in these cases. A great number of these cases exist within the manufacturing industry [75] whereas one example is the metal forming processes. Specifically, the estimation of wheel-rail profiles [76] has been addressed using FEM due to the estimation of contacts over a greater surface is needed than what is assumed in AEBM and EFM. Other applications such as quality control through sliding wear estimation [77], analysis of the responses of fully coupled thermo-elasto-plastic solids in contact [78] and performing diagnostics of failures in induction motors [79]. Due to the complexity of modeling the contacts within robotics, FEM have become a popular choice and enabled tactile applications such as cobots tactile skin for ensuring collaborative behavior when in contact [80], performance estimation of new tactile sensor technologies [81] and evaluating complex contact types by extending simulations and analysis systems [82]. The modeling complexity has furthermore inspired using FEM as ground truth results when synthesizing ML data in simulations for deep learning models, which has enabled execution speeds 75 times greater than simply evaluating FEM [83, 84, 85].

The use of these ML models has enabled realistic simulations of tactile sensor data. Current literature applies DL-based approaches to simulate tactile sensor data for various tasks [86, 87]. For instance, simulating realistic tactile images from simulated contact depth to bridging the reality gap for vision-based tactile sensing using a diffusion model [88]. Similarly, a conditional Generative Adversarial Network (cGAN) has been used to simulate realistic tactile sensory data for use in tactile tasks [88]. Solutions using DL models purely based on Multi Layered Perceptron (MLP) have been applied to enable real-time simulated realistic tactile data [89].

Given the methods presented above, the AEBM Boussinesq-Cerruti approach is considered along with the MLP based DL model.

Although the Boussinesq-Cerruti approach can produce precise tactile data and can be tailored to suit a particular case, it faces certain challenges. The model relies on certain assumptions regarding the materials in contact, including linear deformation and infinitesimal strains. Furthermore, evaluating the model requires complex calculations, such as multidimensional integrals, which significantly increase computation time and hinder real-time

performance. In contrast to the transparency offered by the Boussinesq-Cerruti approach, the MLP based approach is limited by the black-box nature of DL models. Despite this drawback, MLP based DL models offer several benefits, such as low execution time and high adaptability to complex systems. Due to the high adaptability and option for real-time performance, the DL model presented in [89] is chosen to solve the tactile perception problem i.e. problem 1.

3.2 Problem 2 - Pose Estimation

PE, which involves determining the position and orientation of an object in 3D space, has been the subject of many research studies. The literature has identified two main categories of methods for solving this problem: those based on DL, and those based on point cloud registration.

These can along with their subcategories be seen in Fig. 3.2 as inspired by [90].

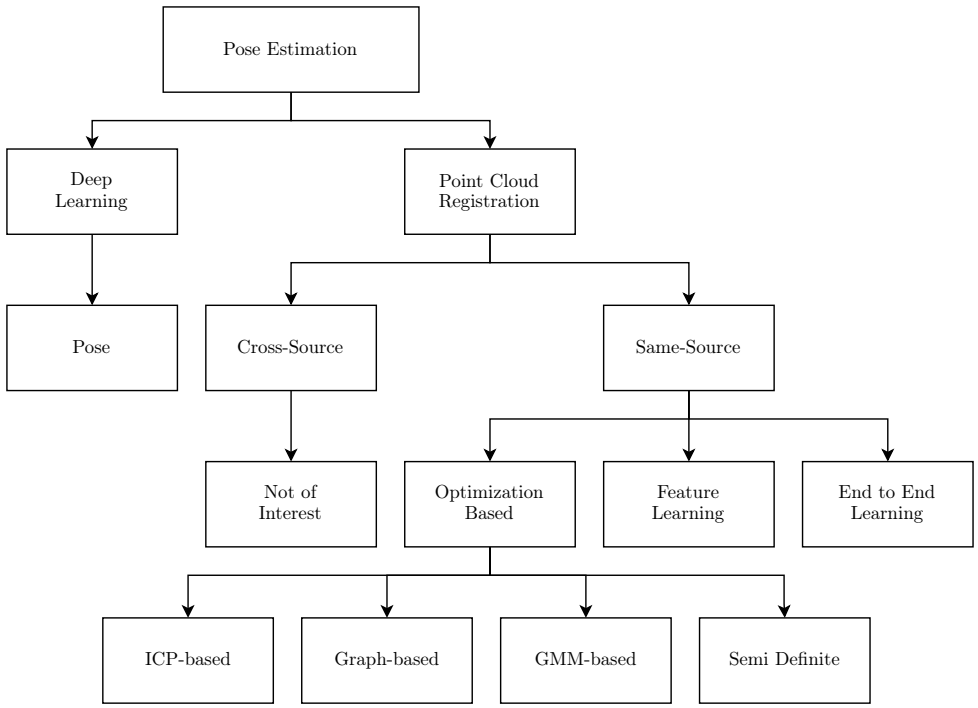


Fig. 3.2: Tree of methods for solving the point cloud registration problem. The categorization is inspired by [90].

Purely DL based methods learn feature representations of input data, often in the form of images, and use them to estimate the subject's pose. This is commonly done in the context of human pose estimation [91, 92, 93]. While these methods have shown extensive use in these cases, their applicability in this project is limited and thus excluded from consideration.

The other method group i.e. point cloud registration (PCR) methods are separated into two subgroups: cross-source and same-source point cloud (PC)s. Here cross-source refers to a PC produced by combining information from sensors of different kinds e.g. visual- and tactile sensors, while same-source methods only produce PCs based on information from the same kind of sensors e.g. only tactile sensors. While cross-source approaches have shown utility in an extensive range of applications [94, 95, 90] their applicability in this project is minimal, as purely tactile pose estimation is the problem of interest as presented in Chapter 1.

PCR methods from the same source data can be categorized into three sub-categories: end-to-end learning, feature-learning, and optimization-based methods. End-to-end learning-based methods use a neural network to estimate

the transformation matrix that aligns two point clouds. Proposed solutions include using neural networks for scene completion to estimate the relative pose between RGB-D scans [96], learning registration patterns as parametric functions through a scan completion module and pairwise matching module [97], and a fast feature-metric point cloud registration framework to minimize the feature-metric projection error without correspondences [98].

In contrast, feature-learning methods use deep neural networks to learn robust feature correspondence searches, which are then used in estimation algorithms such as Random Sample Consensus (RANSAC). In the literature, models have been developed to extract local geometric descriptors from RGB-D reconstructions [99], to learn globally informed 3D local feature descriptors [100], and to use siamese deep learning architectures with convolutional layers through a voxelized smoothed density value (SDV) representation [101].

Lastly, registration methods based on optimization are employed to estimate the transformation matrix through two stages: correspondence searching and transformation estimation. Their goal is to minimize a cost function that gauges the dissimilarity between two point clouds. Within this category, there are four sub-categories identified: Iterative Closest Point (ICP)-based, graph-based, Gaussian Mixture Model (GMM)-based, and semi-definite programming-based methods.

Since the original proposal in 1992 [102] using point-to-point correspondences, ICP-based methods have evolved and incorporated different types of correspondences to improve performance. Examples include point-to-plane [103] and plane-to-plane [104]. Modern approaches also employ complementary methods such as point cloud filtering, adaptive fireworks algorithms, and KD-Trees [105].

The main idea of graph-based registration methods is to use a non-parametric model [106]. In this method, correspondences between two graphs are found by considering both the vertices and edges, making it an optimization problem [106]. To solve this optimization problem, there are two categories of graph-matching methods based on the objective functions' constraints: second-order and high-order methods [107]. Second-order methods include Cross-Source Graph Matching (CSGM) [94], which uses a linear program to solve the graph-matching problem and apply it to solve the cross-source point cloud registration task, Factorized Graph Matching (FGM) [108] factorizes the large pairwise affinity matrix into smaller matrices and solves the graph-matching problem with a simple path-following optimization algorithm. Spectral graph [109] uses a spectral relaxation method to approximate the Quadratic Assignment Problem (QAP), and Semi-Definite Programming (SDP) relaxation is used to relax the non-convex constraint using a convex semi-definite. While higher-order graph matching provide methods for [110] design a probabilistic approach to solve the high-order graph-matching problem, while [111] design a triangle similarity and convert the graph-matching problem into a tensor optimization problem. More recent work, such as [112] suggests an elastic net to control the trade-off between the sparsity and accuracy of the matching results by incorporating the Elastic-Net constraint into the tensor-based graph matching mode.

GMM-based methods commonly tackle the point cloud registration problem by transforming it into a likelihood maximization problem for the input data. This has resulted in the development of several optimization strategies aimed at maximizing the likelihood and optimizing the transformation matrix. For instance, a motion drift idea was introduced into the GMM framework by [113] in the form of Coherent Point Drift (CPD) which imposes constraints on transformation estimation. In another approach,[114] combines GMM with the convex hull to reduce computation complexity. Furthermore, Joint Registration of Multiple Point Sets (JRMPC) [115] cast registration as a clustering problem where the transformation is optimized by solving the GMM. Recently, Deep Gaussian Mixture Registration (DeepGMR) [116] employed DL to learn the correspondences between GMM components and points, enabling the estimation of both the transformation and GMM parameters in a single forward step.

Lastly, within semidefinite programming different optimization groups exist, such as Second Order Cone Programming (SOCP), Quadratic Programming (QP) and Quadraticaly Constrained Quadratic Programming (QCQP). Due to the subject of interest being a rotation in SO(3), the constraints of the rotation matrix, i.e. quadratic constraints,

must be respected. Because of this, the methods of QCQP are of interest. One such example provides estimates which are insensitive to a large fraction of spurious correspondences through decoupling the scale, rotation, and translation estimation. This decoupling enables the solving of these in cascade for the three transformations. The method is referred to as TEASER (Truncated least squares Estimation And SEmidefinite Relaxation), which solve large SDP relaxations, and additionally comes with a second fast and certifiable algorithm, named TEASER++. To decrease execution time this method uses Graduated Non-Convexity (GNC) to solve the rotation subproblem and applies Douglas-Rachford Splitting to enable efficiently certify global optimality [117]. Secondly, Invariant-based Highly Robust Point Cloud Registration (IRON) applies a similar methodology as to TEASER, but instead applies RANSIC (RANdom Samples with Invariant Compatibility) to robustly estimates the scale between two sets of point clouds [118]. Finally, Relaxed Convex Quadratic Programming (RCQP) formulates a QCQP problem with a full set of quadratic rotational constraints and obtains a Lagrangian dual relaxation, which empirically recovered a globally optimal solution in 100 % of the tested cases, although why strong relaxation seems to hold has yet to be shown [119].

Among the categories presented above, the ones of particular interest are optimization-based techniques due to their mature mathematical foundation and possible certifiable optimality and outlier rejection capabilities, and DL-based techniques due to their adaptability and low execution time. While DL-based methods can learn feature representations of point clouds and estimate the transformation between two point clouds, optimization-based approaches with outlier rejection can effectively handle noise and outliers in the data, which is a common problem within the PCR problem. Due to this, the chosen method is the optimization based RCQP method with GNC outlier rejection [120].

3.3 Problem 3 - In-Hand Manipulation

Motion planning is a critical aspect of robot applications as it involves finding a path without collisions between two configurations. However, when it comes to dexterous manipulation, where the object may not always be fully constrained, a more advanced approach called kinodynamic motion planning becomes necessary. Kinodynamic motion planning presents significant challenges, particularly in high-dimensional spaces, and is known to be PSPACE-hard [121], indicating its computational complexity.

To address these challenges, researchers have proposed probabilistic complete sampling-based kinodynamic motion planners like SST and SST* [122]. These planners utilize sampling-based techniques and probabilistic methods to generate collision-free paths in complex environments. Despite their effectiveness, the complexity of the motion planning problem has led to the integration of motion planning algorithms with model predictive control. By combining these two techniques, it becomes possible to avoid solving a boundary value problem [123], reducing the computational burden associated with kinodynamic motion planning.

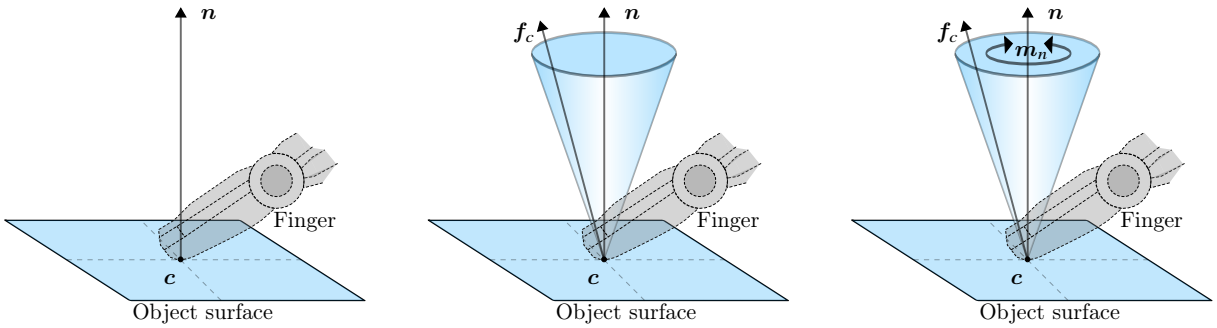
Furthermore, to further enhance performance, researchers have explored the combination of kinodynamic motion planning algorithms with reinforcement learning. This integration leverages the capabilities of reinforcement learning algorithms to learn from experience and improve the efficiency and quality of the generated motion plans [124]. By incorporating the knowledge acquired through reinforcement learning, the kinodynamic motion planning process can adapt and optimize its solutions over time, resulting in more refined and effective paths for dexterous manipulation tasks.

Overall, the field of motion planning has witnessed the development of various approaches to address the challenges posed by kinodynamic motion planning in high-dimensional spaces. From probabilistic complete sampling-based planners to the integration of model predictive control and reinforcement learning, these advancements aim to enhance the efficiency, reliability, and adaptability of motion planning algorithms in order to support complex robot applications.

Chapter 4

Modeling

To model the contact between the EE's tactile sensors, eight different categories exist as identified in [125]. The three most common ones within the field of robotics [50, Chapter 37] are point-contact-without-friction (PwoF), hard finger (HF) and the SF model as shown in Fig. 4.1.



(a) PwoF in point $\boldsymbol{c}$ with normal $\boldsymbol{n}$.

(b) HF in point $\boldsymbol{c}$ with normal $\boldsymbol{n}$ and contact force $\boldsymbol{f}_c$.

(c) SF in point $\boldsymbol{c}$ with normal $\boldsymbol{n}$, contact force $\boldsymbol{f}_c$ and friction moment $\boldsymbol{m}_n$.

Fig. 4.1: The three most commonly used contact models.

The PwoF model, as shown in Fig. 4.1(a), can only represent forces along the surface normal $\boldsymbol{n} \in \mathbb{R}^3$ at the point of contact $\boldsymbol{c} \in \mathbb{R}^3$ and thus the model does not support surface deformations between the two contacting objects. This model is applied in cases where very little deformation is present, along with the contact having a friction coefficient approximately equal to zero [50, Chapter 38].

The HF model, as shown in Fig. 4.1(b), is representative when the friction between objects is significant, while the contact deformation is small enough to ignore friction moments and deformations [50, Chapter 38]. To model the friction acting on the contact point a great number of methods exist, a common one being the Coulomb friction with different modifications depending on the use case [126]. This model states that the frictional force acting on an object can be formulated as

$$f_f = f_N \mu, \tag{4.1}$$

with f_f being the magnitude of the Coulomb friction, f_N being the magnitude of the normal force in the point of contact and $\mu \in [0, 1]$ being the friction coefficient. One visualization of this linear relationship can be seen in the cones illustrated in Fig. 4.1(b) and Fig. 4.1(c). These cones are referred to as friction cones, which for a hard finger model can be formulated as

$$C_{f,\mathrm{HF}} = \left\{ f_c \;\middle|\; f_t \leq \mu f_z,\ \mu f_z \geq 0 \right\} \qquad , \qquad f_t = \sqrt{f_x^2 + f_y^2}. \tag{4.2}$$

Here f_c is the magnitude of the contact force and f_t is the magnitude of the tangential force. f_x, f_y and f_z are the magnitudes of the x, y and z components of the contact force ($\boldsymbol{f}_c \in \mathbb{R}^3$) and μ is the friction coefficient [50, Chapter 37]. By applying a contact force that ensures the friction stays greater than the magnitude of the tangential force, neither object slips i.e. f_z must ensure that $f_z \mu$ stays greater than f_t for the objects not to slip. Visually this is the case when the contact force $\boldsymbol{f}_c$ stays within the friction cone, which enables a friction-based grasp type referred to as force closure. Specifically, force closure refers to when the composite wrench cone contains the entire wrench space so that any external wrench $\boldsymbol{w}_{\text{ext}}$ on the body can be balanced by contact forces [127]. A force that commonly contributes significantly to the external wrench, and thus to the tangential force, is gravity.

The SF model, as shown in Fig. 4.1(c), is used to represent scenarios where both friction and surface deformations are significant. Due to deformations of the finger, an additional torsional moment about the contact normal will be present [50, Chapter 38]. While an analytical formulation of the SF relation depends on the pressure distribution inside the contact, and can only be derived for a limited number of special cases, the general case can be approximated using

$$\mathcal{E}_{f,\mathrm{SF}} = \left\{ f_c \;\middle|\; f_t^2 + \frac{m_n^2}{e_n^2} \leq \mu^2 P^2 \right\} \qquad , \qquad f_t = \sqrt{f_x^2 + f_y^2}. \tag{4.3}$$

This formulation forms a contact ellipsoid $\mathcal{E}_{f,\mathrm{SF}}$ which describes the relationship between the tangential force $\boldsymbol{f}_t \in \mathbb{R}^3$ and friction moment $\boldsymbol{m}_n \in \mathbb{R}^3$. The friction parameters in this expression remain the same as for the friction cone, with the additional m_n being the magnitude of the frictional moment, e_n being the eccentricity parameter i.e. the height of the aforementioned ellipsoid and P being the magnitude of the pressure applied from the contact point along the contact normal $\boldsymbol{n}$ [71, 73].

Based on the model categories described above, the most representative for this project's case, are the SF models since these can provide information about the contact surface's shape, thus enabling the reconstruction of the contact shape from the application of a force distribution [128] i.e. the Inverse Elasticity Problem (IEP). Additionally, these models support descriptions of friction which is crucial to manipulate objects in hand. Illustrations of the system as a SF with friction cone, pressure distribution and the enabling of force closure can be seen in Fig. 4.2 and Fig. 4.3 respectively.

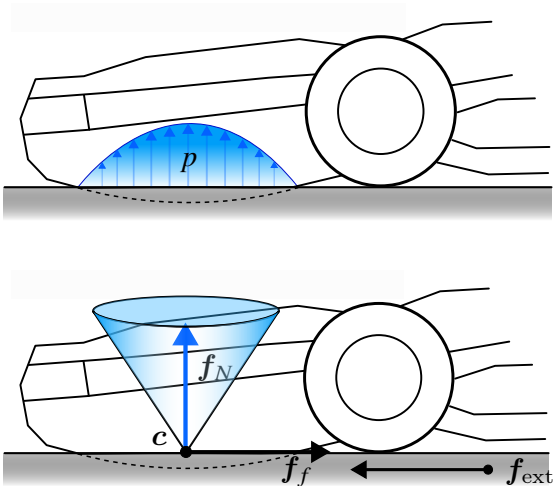


Fig. 4.2: The pressure distribution p and friction cone of a SF model experiencing an external force $\boldsymbol{f}_{\mathrm{ext}}$.

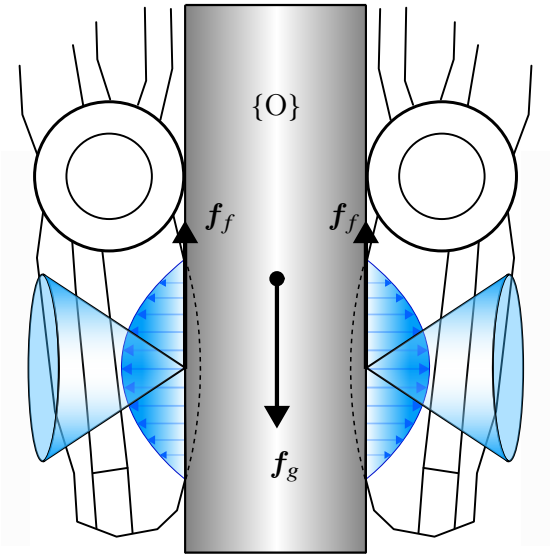


Fig. 4.3: the pressure distribution and friction cone causing force closure to prevent the object {O} from falling due to the gravitational force $\boldsymbol{f}_g$.

When modeling the kinematics of an anthropomorphic gripper with frame {H}$\in \mathbb{R}^{4\times4}$ in world frame {W}$\in \mathbb{R}^{4\times4}$ interacting with an object with frame {O}$\in \mathbb{R}^{4\times4}$ the relevant parameters must be addressed. In this system the object with position $\boldsymbol{p} \in \mathbb{R}^3$ and pose $\boldsymbol{u} \in \mathbb{R}^6$, with the orientation either being represented as a four-dimensional quaternion or a three-dimensional Euler angle, makes contact with the gripper in points $\boldsymbol{c}_i \in \mathbb{R}^3$. These contact points have frames {C}$_i \in \mathbb{R}^{4\times4}$ with axes $\{\boldsymbol{n}_i, \boldsymbol{t}_i, \boldsymbol{o}_i\} \subset \mathbb{R}^3$, where $\boldsymbol{n}_i \in \mathbb{R}^3$ points perpendicular to the contact plain towards the object, while the remaining are contained within the contact plane. For each of these parameters $i = 1, 2, \ldots, n_c$, where n_c is the number of contact points. The twist of {O} described in {W} is denoted $\boldsymbol{\nu} = \begin{bmatrix}\boldsymbol{v}^\top & \boldsymbol{\omega}^\top\end{bmatrix}^\top \in \mathbb{R}^6$ while the non-contact wrench i.e. the wrench caused by external forces such as collisions with the environment and gravity, is $\boldsymbol{w} = \begin{bmatrix}\boldsymbol{f}^\top & \boldsymbol{m}^\top\end{bmatrix}^\top \in \mathbb{R}^6$. The gripper's state is described in terms of its

joints, of which it has n_q, named $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_{n_q} \end{bmatrix}^\top \in \mathbb{R}^{n_q}$ each of which is revolute and can exert a torque $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_{n_q} \end{bmatrix}^\top \in \mathbb{R}^{n_q}$. These parameters can be seen illustrated in Fig. 4.4 showing the system model. While only a single finger here is illustrated the naming conventions and representations simply scale to all the

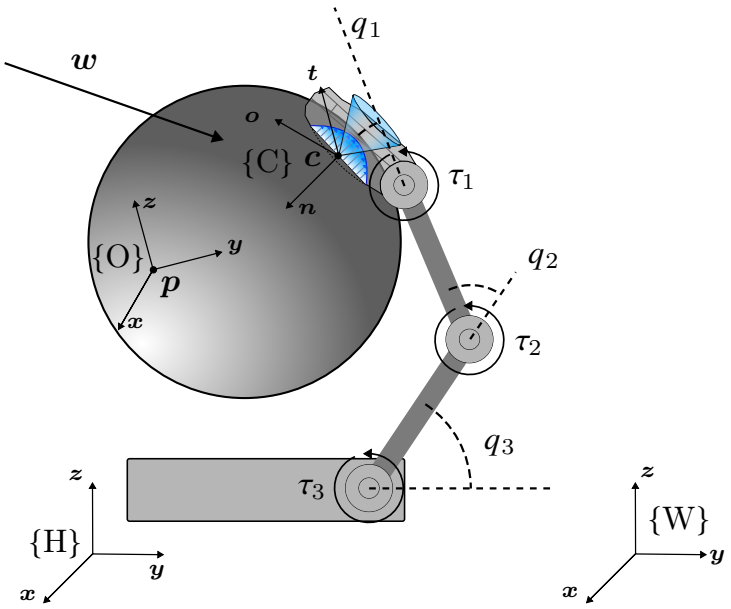


Fig. 4.4: The model of the world representation for this project.

EE's DOFs.

In this system, the twists and wrenches of a contact point $\boldsymbol{c}_i$ on the object and hand, given in contact frame $\{\mathrm{C}\}_i$ is referred to as $\boldsymbol{\nu}_{i,\xi} \in \mathbb{R}^6$ and $\boldsymbol{w}_{i,\xi} \in \mathbb{R}^6$, with $\xi = \{\mathrm{obj}, \mathrm{hnd}\}$. Given multiple contact points, complete vectors of twist and wrench can be expressed by appending each contact point's twist and wrench vector. These contain all twists and wrenches of the grasp, one for the object and one for the hand. These vectors are referred to as

$$\boldsymbol{\nu}_{c,\xi} = \begin{bmatrix} \boldsymbol{\nu}_{1,\xi}^\top & \boldsymbol{\nu}_{2,\xi}^\top & \cdots & \boldsymbol{\nu}_{n_c,\xi}^\top \end{bmatrix}^\top \in \mathbb{R}^{6 \cdot n_c} \tag{4.4}$$

and

$$\boldsymbol{w}_{c,\xi} = \begin{bmatrix} \boldsymbol{w}_{1,\xi}^\top & \boldsymbol{w}_{2,\xi}^\top & \cdots & \boldsymbol{w}_{n_c,\xi}^\top \end{bmatrix}^\top \in \mathbb{R}^{6 \cdot n_c} \tag{4.5}$$

respectively.

These definitions are used to describe and analyze the kinematics of grasping and the parameters involved in holding and manipulating objects in hand, also referred to as grasp kinematics (GK). Within GK two matrices are of special interest: the grasping matrix $\mathbf{G}$ and the hand Jacobian $\mathbf{J}$. The grasping matrix describes the transformation between the twist or wrench of the object in world frame $\{\mathrm{W}\}$ to the twists or wrenches of the object in contact frames $\{\mathrm{C}\}_i$. The grasp matrix thus can be expressed as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 & \cdots & \mathbf{G}_{n_c} \end{bmatrix}, \tag{4.6}$$

where $\mathbf{G}_i \in \mathbb{R}^{6\times6}$ describes the transformation from $\{\mathrm{W}\}$ to the individual $\{\mathrm{C}\}_i$, and thus $\mathbf{G} \in \mathbb{R}^{6\times6\cdot n_c}$ describes the transformations for all contact points. Using this grasp matrix, the object wrench and twist can be computed in all contact frames as

$$\boldsymbol{\nu}_{c,\mathrm{obj}} = \mathbf{G}^\top \boldsymbol{\nu} \qquad \text{and} \qquad \boldsymbol{w}_{c,\mathrm{obj}} = \mathbf{G}^\top \boldsymbol{w}. \tag{4.7}$$

While the grasp matrix describes the transformation from $\{\mathrm{W}\}$ to object contact frames, the hand Jacobian relates the joint velocities and torques to the contact twists and wrenches on the hand. The hand Jacobian can thus be

expressed as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1^\top & \mathbf{J}_2^\top & \mathbf{J}_3^\top & \cdots & \mathbf{J}_{n_c}^\top \end{bmatrix}^\top, \tag{4.8}$$

for all contact points. Here $\mathbf{J}_i \in \mathbb{R}^{6\times n_q}$ for $i = 1, 2, \ldots, n_c$ are the individual contact point's hand Jacobians and thus $\mathbf{J} \in \mathbb{R}^{6\cdot n_c \times n_q}$ is the complete. Using the complete hand Jacobian, the contact twists and wrenches on the hand can be related to the joint velocities and torques as

$$\boldsymbol{\nu}_{c,\text{hnd}} = \mathbf{J}\dot{\boldsymbol{q}} \qquad \text{and} \qquad \boldsymbol{\tau} = \mathbf{J}^\top \boldsymbol{w}_{c,\text{hnd}}. \tag{4.9}$$

The modeling described above will enable the use of methods for solving the presented problems. These methods will be determined in Chapter 3.

Chapter 5

Tactile Perception

To solve problem 2 and 3, the tactile perception solution must provide estimates of contact positions, contact normals and skew forces.

This chapter presents an analysis of the performance of different techniques to solve these problems, including a DL model for simulating realistic tactile skew forces, Recursive Least Squares (RLS) for estimating contact normals and the contact positions from the grasping $\mathbf{G}$. Contact normals are essential for accurately estimating the pose of an object in contact, while skew forces are critical for predicting the behavior of an object when it is grasped and manipulated by the SDH.

Firstly, the RLS methodology is presented for normal estimation $\boldsymbol{n}_{c,i} = \left[n_{i,x}, n_{i,y}, n_{i,z}\right]^\top \in \mathbb{R}^3$, where $i \in \{0, 1, \ldots, n_c\}$ and n_c is the number of contact points, followed by the experimental setup and result representation.

Secondly, the technique behind estimating the skew forces $\boldsymbol{f}_{c,i} = \left[f_{i,x}, f_{i,y}, f_{iz}\right]^\top \in \mathbb{R}^3$, where i goes from 1 to n_c as with the contact normals, is presented, which includes the DL models architecture as well as the methodology used to test the network. The testing methodology involves the use of various input data, and the output is analyzed for accuracy and realism. The findings are presented and discussed, including the strengths and weaknesses of the network in simulating tactile data. Finally, an assessment is made of the network's ability to produce tactile data that is realistic.

Finally, the contact positions $\boldsymbol{c}_i = \left[c_{i,x}, c_{i,y}, c_{i,z}\right]^\top \in \mathbb{R}^3$, which has the same bounds as the skew forces and contact normals are found and evaluated from the grasping matrix $\mathbf{G}$ provided by [89].

The skew force and normal estimates are compared to the ones provided by Gazebo's physics engine, and the known GT from where conclusions are drawn.

The software used in this chapter is a regression neural network implemented as a Gazebo `ModelPlugin` [129] in C++. However, the DL model plugin used in the original publication [89] has not been updated since 2018, making the code incompatible with the current version of Gazebo API. Moreover, the licensing issues with the files in the `xmlrpc++` library, which were used for base64 encoding and decoding, necessitated their removal [130]. To address these issues, each has been resolved and the plugin has been reorganized and repackaged for compatibility with the current version of Gazebo. The original version of the plugin can be found in [131], while the fixed and updated version is available at [132].

The availability of the updated plugin ensures that the project can continue to benefit from the capabilities of the MLP based DL model for simulating realistic tactile data in the current version of Gazebo.

5.1 Methods

5.1.1 Recursive Least Squares

To use RLS to estimate the contact normal, we require the estimated linear velocity $\hat{\boldsymbol{v}} \in \mathbb{R}^3$. However, since the contact points may not be consistent across the surface during motion, some points may be missing at certain time steps. To address this issue, the centroid of the contact points $\bar{\boldsymbol{c}}_t \in \mathbb{R}^3$ at time t is used as a representative value for each time step. Therefore, we can compute the estimated linear velocity as

$$\hat{\boldsymbol{v}} = \frac{\bar{\boldsymbol{c}}_t - \bar{\boldsymbol{c}}_{t-1}}{\Delta t}, \tag{5.1}$$

where $t-1$ is the previous time step and $\Delta t \in \mathbb{R}$ is the time between samples, which in this case is 0.01 s as the sampling frequency is 100 Hz. The velocity at the fingertip in contact will then be computed as

$$\boldsymbol{v}_{tip} = \hat{\boldsymbol{v}} + [\boldsymbol{\omega}]_\times \mathbf{R}_{tcp}^{base} \boldsymbol{r}, \tag{5.2}$$

where $\hat{\boldsymbol{v}}$ is the estimated linear velocity computed from 5.1, $[\boldsymbol{\omega}]_\times \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix of the angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ i.e.

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad , \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_x, \omega_y, \omega_z \end{bmatrix}, \tag{5.3}$$

$\mathbf{R}_{tcp}^{base} \in \mathbb{R}^{3\times3}$ is the rotation matrix from the hand's base to its Tool Center Point (TCP), and $\boldsymbol{r} \in \mathbb{R}^3$ is the position of the contact point in the robot's base frame.

To compute the normal estimate, additionally, an initial guess is needed which is computed by

$$\hat{\boldsymbol{n}}_0 = \frac{\hat{\boldsymbol{v}}_1 \times \hat{\boldsymbol{v}}_0}{\|\hat{\boldsymbol{v}}_1 \times \hat{\boldsymbol{v}}_0\|_2}, \tag{5.4}$$

where $\hat{\boldsymbol{n}}_0 \in \mathbb{R}^3$ is the initial estimate, $\hat{\boldsymbol{v}}_0$ and $\hat{\boldsymbol{v}}_1$ are the linear velocity estimates for time step 0 and 1, and $\|\cdot\|_2$ is the ℓ_2-norm.

Using the quantities above, the two main components $\mathbf{L}_n^1 \in \mathbb{R}^{3\times3}$ and $\mathbf{L}_n^2 \in \mathbb{R}^{3\times3}$ of the desired estimate $\hat{\boldsymbol{n}}_{new} \in \mathbb{R}^3$ can be found. Firstly, $\mathbf{L}_n^1$ is computed as

$$\mathbf{L}_n^1 = \mathbf{L}_n^1 - \beta \Delta t \mathbf{L}_n^1 + \frac{\Delta t K_L}{1 + \|\boldsymbol{v}_{tip}\|_2^2} \boldsymbol{v}_{tip} \boldsymbol{v}_{tip}^\top, \tag{5.5}$$

with $\beta \in \mathbb{R}$ being a decay factor, $K_L \in \mathbb{R}$ is a gain and $\|\cdot\|_2^2$ is the squared ℓ_2-norm.

The second component $\mathbf{L}_n^2$ is computed by

$$\mathbf{L}_n^2 = \mathbf{L}_n^2 - \beta \Delta t \mathbf{L}_n^2 + \frac{\Delta t K_L}{1 + \|\boldsymbol{\nabla}\|_2^2} \boldsymbol{\nabla}\boldsymbol{\nabla}^\top, \tag{5.6}$$

where $\boldsymbol{\nabla} \in \mathbb{R}^3$ is the cross product between the contact force $\boldsymbol{f}_c$ and the velocity of the tip $\boldsymbol{v}_{tip}$ i.e.

$$\boldsymbol{\nabla} = \boldsymbol{f}_c \times \boldsymbol{v}_{tip}. \tag{5.7}$$

Using these components a Proportional-Derivative (PD) controller is used to compute the change in normal $\dot{\boldsymbol{n}} \in \mathbb{R}^3$ as

$$\dot{\boldsymbol{n}} = -\left(\gamma_1 \mathbf{L}_n^1 + \gamma_2 \mathbf{L}_n^2\right) \boldsymbol{n}, \tag{5.8}$$

where $\gamma_1 \in \mathbb{R}$ is the proportional gain, $\gamma_2 \in \mathbb{R}$ is the derivative gain and $\boldsymbol{n}$ is the current estimate. By computing the cross product between $\boldsymbol{n}$ and $\dot{\boldsymbol{n}}$, the angular velocity $\boldsymbol{\omega}$ is computed

$$\boldsymbol{\omega} = \boldsymbol{n} \times \dot{\boldsymbol{n}}. \tag{5.9}$$

Finally, the new normal estimate $\boldsymbol{n}_{new} \in \mathbb{R}^3$ can be computed as

$$\boldsymbol{n}_{new} = e^{\left[\frac{\Delta t}{2}[\omega]_\times\right]} \boldsymbol{n}, \tag{5.10}$$

where $e^{\left[\frac{\Delta t}{2}[\omega]_\times\right]} \in \mathbb{R}^{3\times3}$ is the exponential map of the skew-symmetric matrix represents the rotation due to the angular velocity over the time step.

5.1.2 Network Architecture

In [89] two different architectures were built, where architecture B is chosen due to its better greater accuracy and lower execution time. The network architecture B can be seen in Fig. 5.1. As inputs, the network takes one contact position $\boldsymbol{c} = \left[c_x, c_y, c_z\right]$ along with three skew force vector samples $\boldsymbol{f}_1 = \left[f_{1,x}, f_{1,y}, f_{1,z}\right]$, $\boldsymbol{f}_2 = \left[f_{2,x}, f_{2,y}, f_{2,z}\right]$ and $\boldsymbol{f}_3 = \left[f_{3,x}, f_{3,y}, f_{3,z}\right]$, and a temperature input $T \in \mathbb{R}$, which makes the input $\left[\boldsymbol{c}, \boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, T\right] \in \mathbb{R}^{13}$. The contact point and forces are extracted from Gazebo's physics engine. The outputs of the network are the data format produced by the physical sensor i.e. an output vector of $\left[pdc, pac, tdc, tac, e_1, \ldots, e_{19}\right] \in \mathbb{R}^{23}$, where pdc is the pressure DC signal, pac is the pressure AC signal, tdc is the temperature DC signal, tac is the temperature AC signal and e_1 to e_{19} are the electrode activations. The electrode activations can, according to SynTouch [**biotac-syntouch-manual**], can be related to physical quantities by equations which take raw 12-bit integer sensor readings in $[0, 4095]$ those values depend on the skin deformation and require individual calibration.

The network architecture consists of four MLPs, one for interpreting the position input, one for interpreting the force inputs, one for interpreting the temperature input, and one for combining the interpretations of force and position inputs. MLP 1, is responsible for interpreting the position data, consists of four hidden layers, three of which contain 512 neurons and uses Rectified Linear Unit (ReLU) as the activation function, while the last uses a linear activation function with 64 neurons. The activation functions can be seen marked red if they are ReLU, green if they are linear activation functions and blue if they are sigmoid activation functions in Fig. 5.1.

MLP 2 interprets the force inputs but rather than using 512, uses 256 for its hidden layers, while still having the linear activation function and the 64 neurons in its fourth layer. This MLP further differs as the 256 neuron layers apply ℓ_1 bias regularization. The MLP 3 produces a temperature correction vector using two hidden layers, one with 256 neurons and a sigmoid activation function and one with 23 neurons and a linear activation function. The last MLP, MLP 4 takes in the element-wise product of MLP 1 and 2, parses the product through two 256 neuron layers with ReLU and one 23 neuron layer with a linear activation function.

The products of MLP 3 and 4 are summed and parsed as the model's output. All os this can be seen in Fig. 5.1.

5.1.3 Network Training Procedure

The model described in 5.1.2 Network Architecture was trained using a custom dataset collected by the authors. Instead of retraining the model, the provided weights in the paper's code were utilized. The dataset, denoted as **D**, comprises $N_{dp} = 300,000$ readings from tactile sensors. It includes complete BioTac sensor data, as well as the corresponding reference forces and contact points. The structure of dataset D can be represented as follows

$$\mathbf{D} = \left[\begin{array}{c|cccccccccccc} 0 & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \\ 1 & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \\ \vdots & & & & & & & \vdots & & & & & & \\ N_{dp} & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \end{array}\right] \in \mathbb{R}^{N_{dp} \times 35}. \tag{5.11}$$

To ensure consistency and avoid bias, all inputs are standardized by normalizing them to have zero mean and unit variance, using the distribution of the captured data. In order to prevent overfitting and unrealistic reactions to high-frequency inputs that the physics simulator cannot accurately reproduce, the three force vectors are sampled at intervals of 100 ms. The BioTac sensor electrode values display a non-linear relationship with device temperature. Attempts to address this issue before inputting the data led to poor performance. Instead, the network was trained to independently compensate for this dependency. During simulation, a constant temperature was assumed, typically corresponding to the average temperature of the room when the data was collected. This temperature was not made public in neither [133] nor [89]. The network generates simulated electrode and pressure signals as outputs but currently does not simulate temperature outputs.

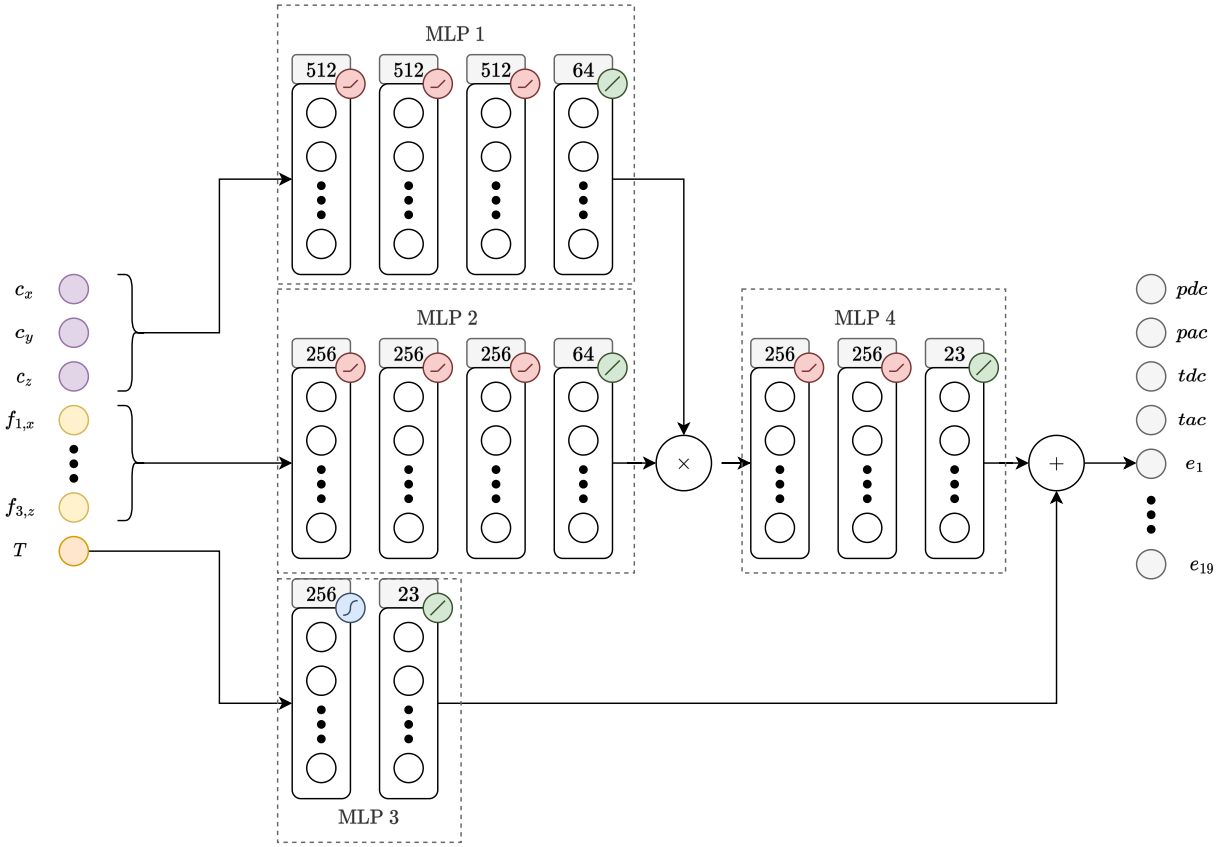


Fig. 5.1: DL model B from [89], which also has provided inspiration for this illustration.

The forces were collected using a calibrated six-axis force-torque sensor [134] with a nominal force resolution better than 0.01 N. The contact position is reconstructed optically using a calibrated HD webcam and two AprilTag markers [135], one mounted on the BioTac and one attached to the probe object. The setup for this can be seen in Fig. 5.2. Once the contact positions were collected, optimization-based calibrations were made to gain more accurate position estimates.

The data set has been made publicly available and can be found in [133].

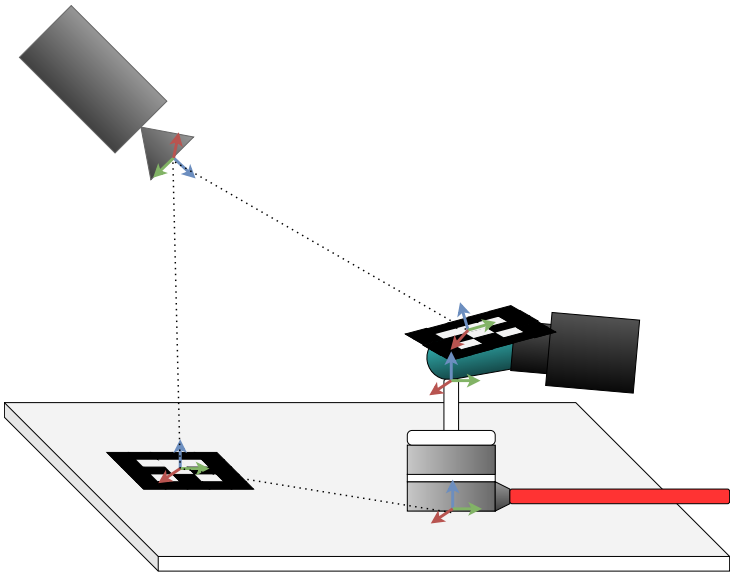
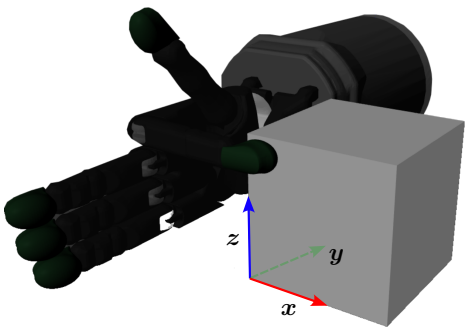


Fig. 5.2: Experimental setup for gathering data to train DL model B, as inspired by [89].

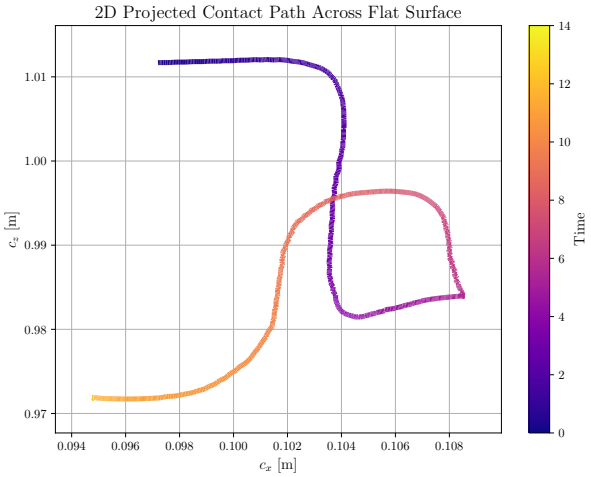
5.2 Experimental Setup

5.2.1 Contact Normal Estimation

To test the RLS method's ability to estimate contact normals the index finger makes contact with a flat surface as shown in Fig. 5.3(a), through flexion and ulnar deviation a contact path is created as shown in Fig. 5.3(b). The motion is done throughout 14 s, and due to a significant presence of noise in the simulated tactile sensors, the contact position data is filtered using a rolling low pass filter with window size 100. The experiment was conducted with the finger on the surface facing $-\boldsymbol{y}$, meaning the GT normal is $\boldsymbol{n}_{gt} = \begin{bmatrix}0, -1, 0\end{bmatrix}$ as shown in Fig. 5.3(a).



(a) Experimental setup for collecting linear velocity data to estimate contact normals using RLS.

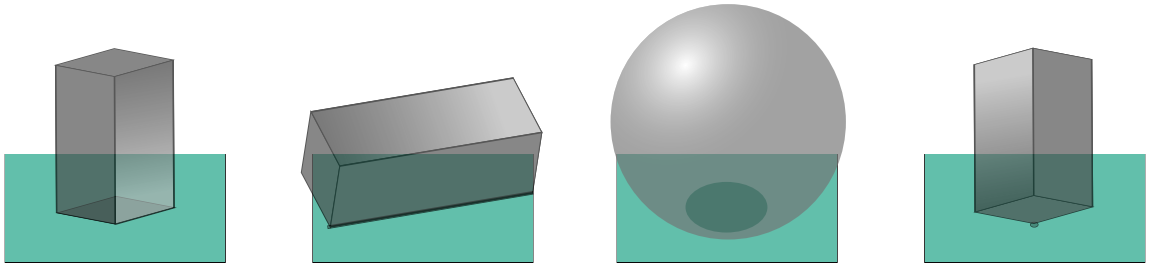


(b) 2D projection of the index finger's path across the cube's flat surface throughout the 14 s data is sampled.

Fig. 5.3: Experimental setup and index finger's contact path when sampling contact data for normal estimation.

5.2.2 Skew Force Estimation

To test the performance of the DL model, four objects surfaces were used with known normals. These can be seen in Fig. 5.4 as a flat surface, an edge, a smooth surface and a corner.



(a) Finger in contact with a flat surface.

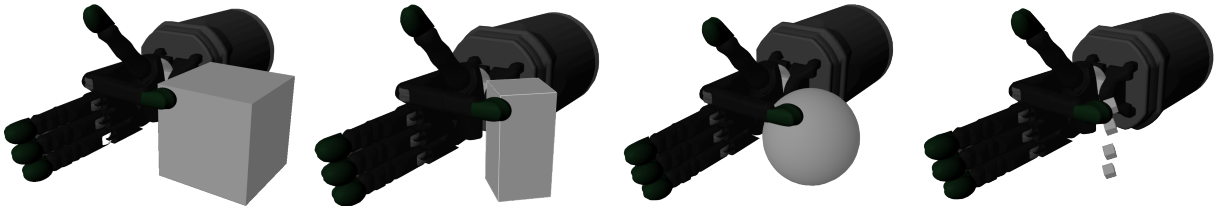
(b) Finger in contact with an edge.

(c) Finger in contact with a smooth surface.

(d) Finger in contact with a corner.

Fig. 5.4: The four surfaces used to test the performance of the DL model's ability to represent surfaces.

Within the simulation, the index finger is set to make contact with each surface, as shown in Fig. 5.5.



(a) Simulated index finger in contact with a flat surface.

(b) Simulated index finger in contact with an edge.

(c) simulated index finger in contact with a smooth surface.

(d) Simulated index finger in contact with a corner

Fig. 5.5: The simulated SDH in contact with the four surfaces used to test the performance of the DL model's ability to represent surfaces. In each case, the contact is made by the index finger.

When contact is made the inputs and outputs of the DL model are recorded over 30 s, which with a sampling frequency of 100 Hz results in 3000 samples. As inputs are collected, the contact positions and forces are given by Gazebo in {W}, which then is transformed into the contact frame {C} using the grasping matrix $\mathbf{G}$. Due to contact data in Gazebo being prone to noise, an exponential decay filter is additionally applied.

5.3 Results

5.3.1 Contact Normals

Upon estimating the linear velocities Fig. 5.6 was produced, which shows the three velocity components. Due to the great presence of noise, a rolling low pass filter was applied with a window size of 100. As one would expect v_y shows a negligible magnitude.

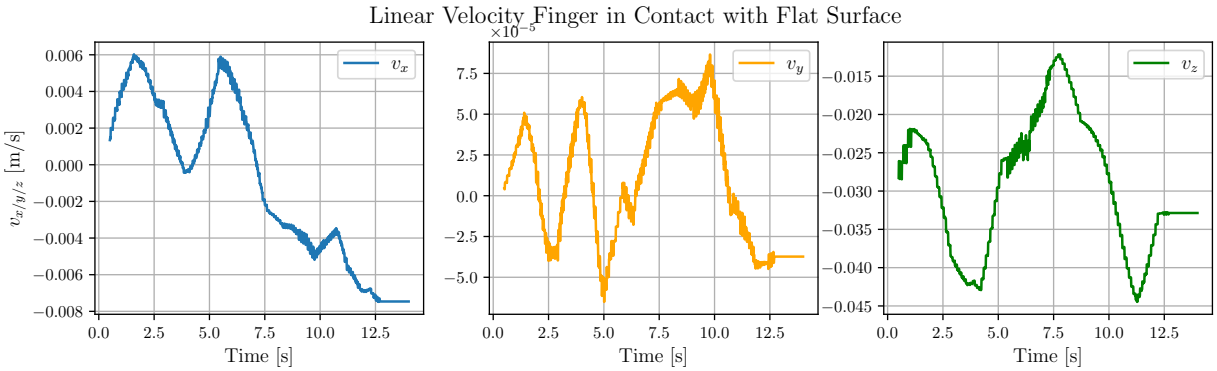


Fig. 5.6: Linear velocities of the contact points when the index finger moves across a flat surface.

Based on these the contact normals were estimated using RLS, which results in Fig. 5.7, which show great consistency and accuracy over the 14 s the experiment was performed.

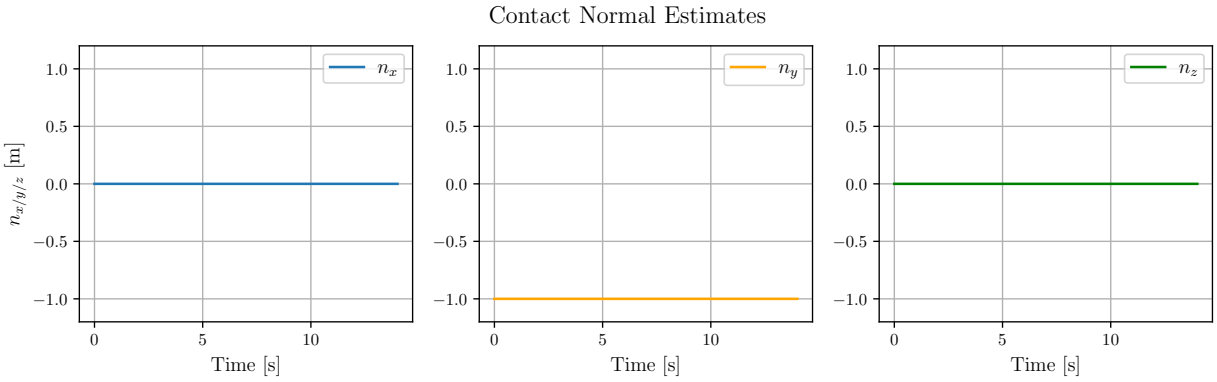
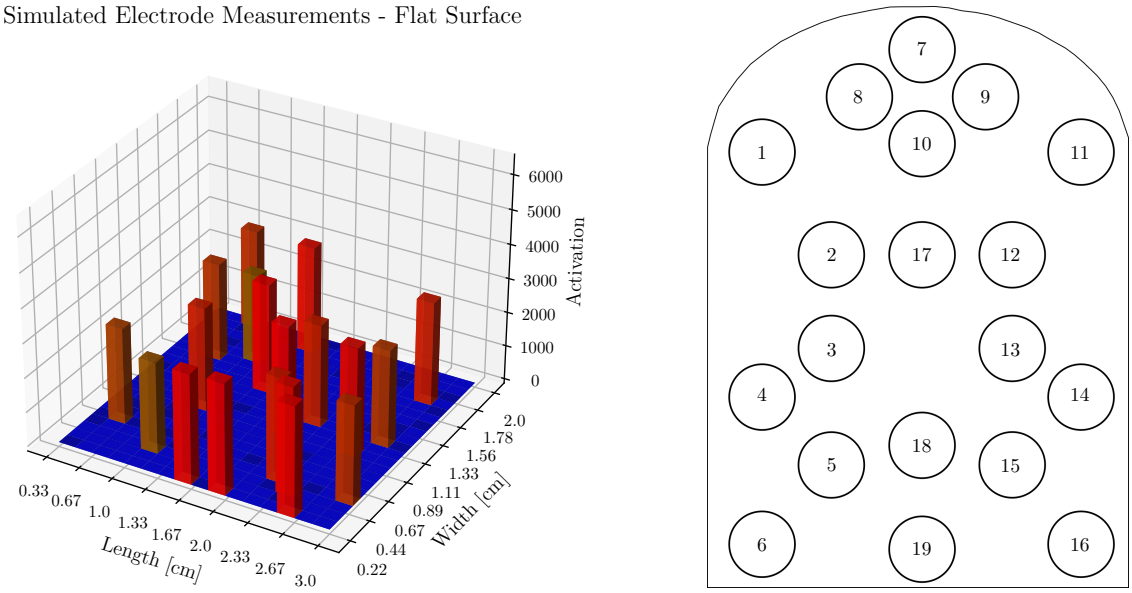


Fig. 5.7: The normal estimates across time as the experiment was executed.

5.3.2 Skew Forces

After executing the DL model on all cases, the resulting simulated electrode activations were discovered to be infinite. Consequently, efforts were undertaken to address this issue. It was determined that the model does not include layer-wise normalization to establish limits on feature responses. The addition of this normalization procedure resulted in the network outputting values within the expected range. Fig. 5.8(a) shows a 3D plot of the electrode activations after layer-wise normalization, while Fig. 5.8(b) shows a 2D projection of the finger tip with its electrodes labeled.

Fig. 5.9 illustrates the inputs and outputs of the DL model when the SDH's index finger makes contact with a flat surface. As seen here the output of the model is independent of the input. In an attempt to isolate a potential cause for this behavior, data from the custom data set on which the model had been trained, is applied and similar results



(a) 3D plot of electrode activations when the SDH's index finger makes contact with a flat surface.

(b) Map showing a 2D projection of the electrodes' positions and numbers.

Fig. 5.8: 3D plot of electrode activations and 2D electrode map with labels.

were found as seen in Fig. 5.10. The missing elements from this graph are value responses from the network of `inf`, which is a reference to the highest representable value by the system and is therefore not included.

Due to this independence of input, the electrodes, and by extension the DL model is not judged to be able to accurately simulate skew forces for a BioTac sensor in contact. The contact data from the remaining experiments can be found in Appendix B, which show a similar pattern.

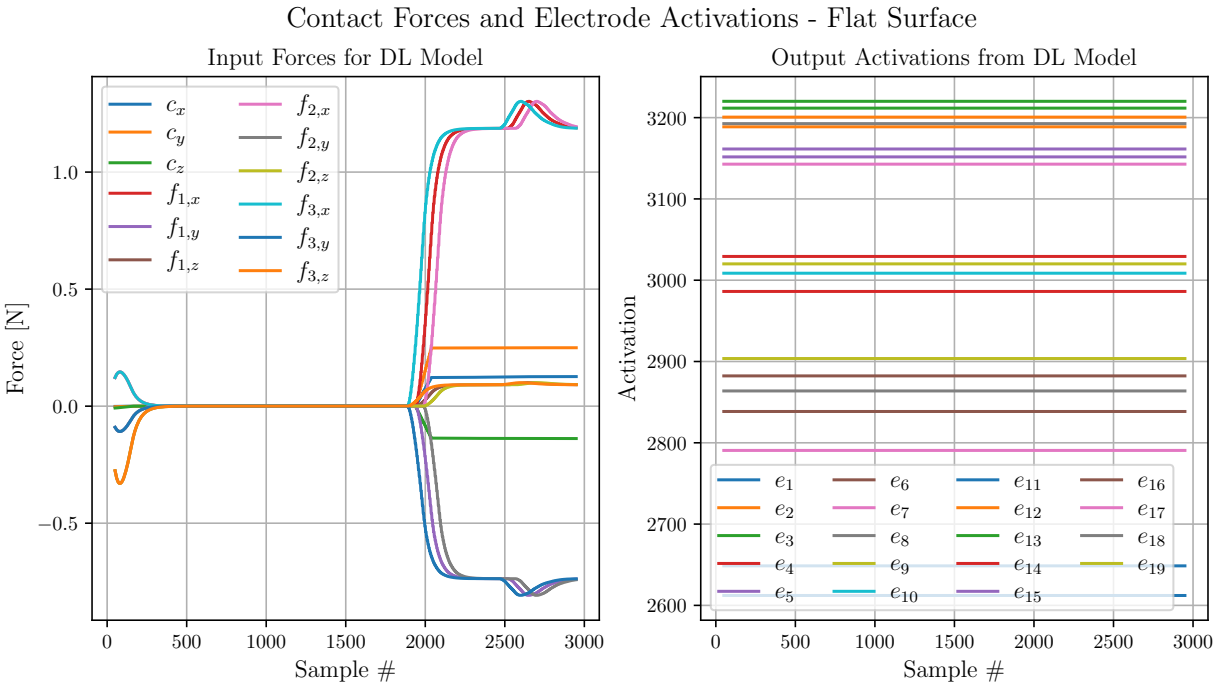


Fig. 5.9: The simulated tactile electrode activations the index finger is in contact with a flat surface.

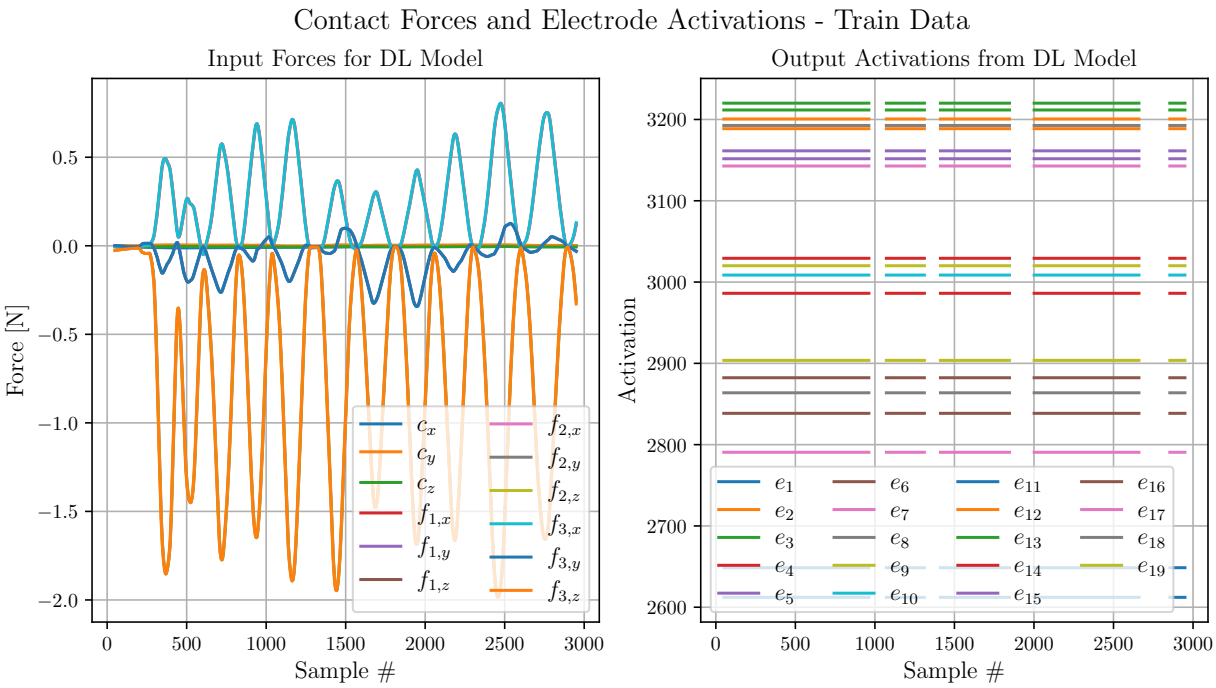
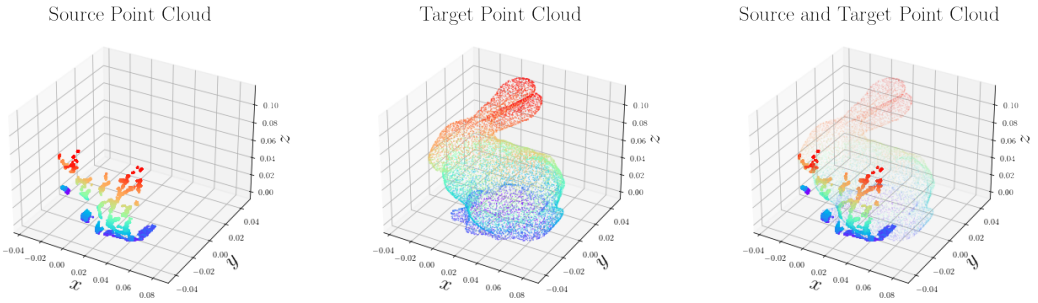


Fig. 5.10: The simulated tactile electrode activations when data the DL model was trained on is applied.

Contact Positions

The contact positions were found by applying the provided matrices for the contact sensor poses on the fingers. To test the engine's ability to represent contact points, a 3D mesh [136] was sampled using the Shadow Dexterous hand, with the contact points being recorded as shown in Fig. 5.11. Here Fig. 5.11(a) shows the contact points sampled from moving the Shadow Dexterous fingers across the 3D mesh.



(a) The source PC generated from simulated contact points.

(b) The target PC generated by sampling the model mesh with 10 000 points.

(c) The source PC overlaid the target PC, showing their fit.

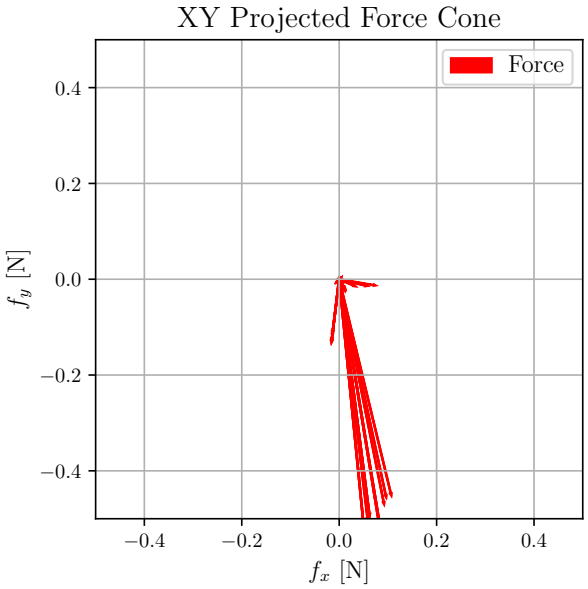
Fig. 5.11: 3D plots showing the sampled source and target PCs along with a plot showing both of them overlaid.

5.3.3 Physics Engine Comparison

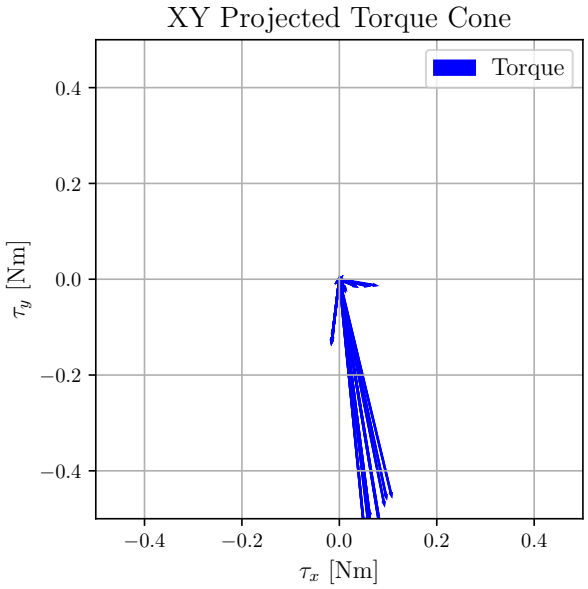
Gazebo's physics engine provides interpretations of tactile contacts for models containing a contact sensor model plugin. The data comes in the form of a `ContactState` which contain the contact points in {W}, wrenches, depth and more. As a replacement for the lacking skew forces from the DL method, and a possible extension of the

contact point and normal estimation, Gazebo's physics engine is considered a potential supplement to the methods presented.

The contact forces and torques were found likewise found, shown in Fig. 5.12(a) and Fig. 5.12(b), for the Shadow Dexterous finger's being in contact with an edge.



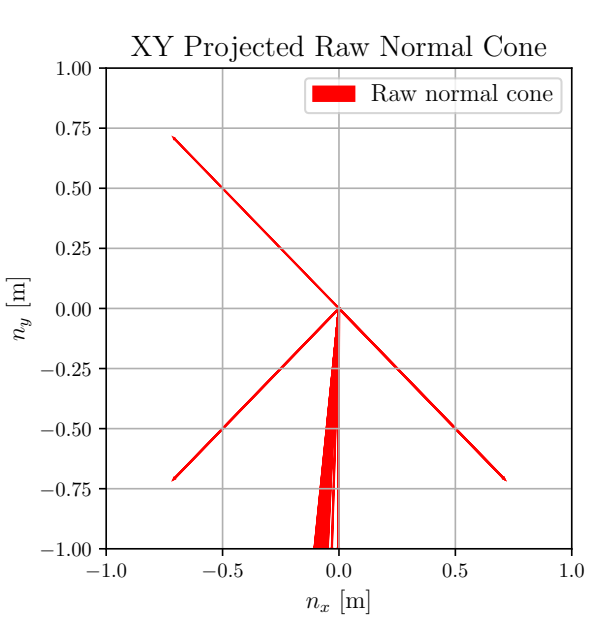
(a) XY projected force cone produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.



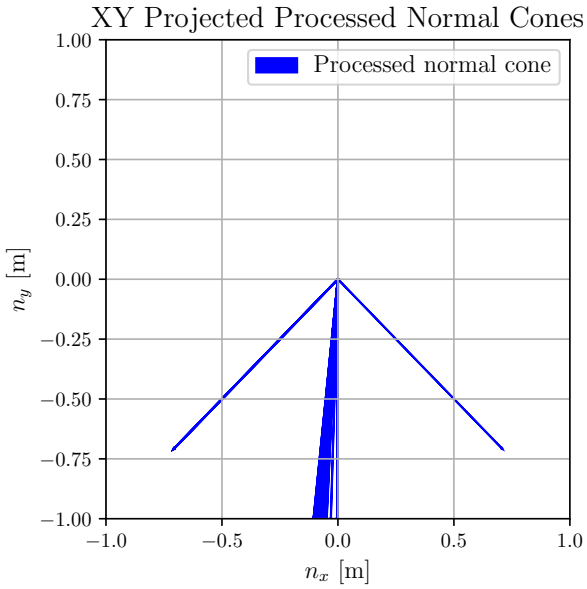
(b) XY projected torque cone produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.

Fig. 5.12: XY projected force and torque cones produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.

The normals however posed a challenge, as the colliding meshes caused a misinterpretation by the physics engine to produce contact normals which were reflected 180°. This can be seen in Fig. 5.13(a). This was solved by sampling a set of contact normals over 10 time steps, reducing the sampling frequency to 10 Hz. These were then clustered using Euclidean clustering with an ϵ of 1 cm and 3 as the minimum number of samples. The centroid of each cluster was then used as the representative and the normal cones shown in Fig. 5.13(b) were achieved.
Finally, the angle errors θ_e of the contact normals were found for each of the four presented cases. This can be seen in Fig. 5.14



(a) XY projected raw normal cone produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.



(b) XY projected processed normal cone produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.

Fig. 5.13: XY projected raw and processed normal cones produced by Gazebo's physics engine when the Shadow Dexterous hand is in contact with an edge.

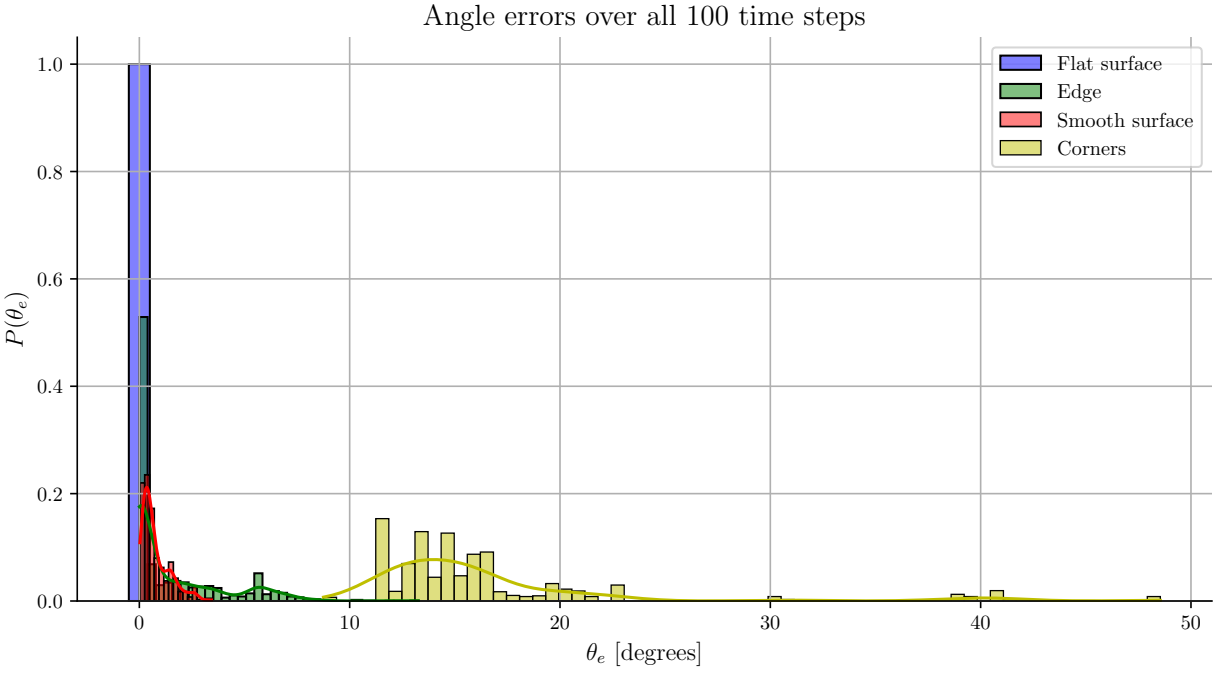


Fig. 5.14: Probability distributions of the angle error θ_e between simulated and GT contact normals for each tested surface.

5.4 Discussion & Conclusion

In this study, the performance of a DL model in estimating the tactile perception of a BioTac sensor when in contact with different objects is evaluated. The findings indicate that the DL model did not provide any useful information in this study. The lack of accuracy in simulating the electrode activations limits the usefulness of

the model in applications that require tactile force sensing. This finding highlights the importance of carefully evaluating the performance of DL models in specific applications and contexts, as they may not always provide the expected benefits. It was at the time of this project not possible to replicate the tactile information from the original paper [89].

Contact normals were found using simulated contact points and the RLS method to estimate these values. However, the contact normals produced by Gazebo's physics engine were acceptable and could be used to supplement our methods.

In conclusion, this study provides insights into the limitations of DL models in estimating tactile perception and highlights the importance of considering alternative methods, such as physics engine simulations, to supplement or replace DL models when necessary. Future studies could explore other DL architectures or combinations of different methods to improve the accuracy and usefulness of tactile perception estimation.

Chapter 6

Pose Estimation

6.1 Introduction

Here we write the introduction for problem 2.
In this chapter, the RCQP method will be presented along with its performance in solving the point cloud registration problem. The data produced in Chapter 5 is a point cloud of the form

$$\mathbf{X} = \begin{bmatrix} c_x & c_y & c_z & n_x & n_y & n_z \\ c_x & c_y & c_z & n_x & n_y & n_z \\ & & \vdots & & & \\ c_x & c_y & c_z & n_x & n_y & n_z \end{bmatrix} \in \mathbb{R}^{M\times 6}, \tag{6.1}$$

where $\boldsymbol{c} = \begin{bmatrix} c_x, c_y, c_z \end{bmatrix}$ is a contact point and $\boldsymbol{n} = \begin{bmatrix} n_x, n_y, n_z \end{bmatrix}$ is the corresponding point's normals vector. Under the assumption of already knowing the object of interest, a GT point cloud $\mathbf{Y}$ is also generated, which takes the same structure except the number of points being M. One row in the data matrix $\mathbf{X}$ is referred to as $\boldsymbol{x}$.

The problem is thus to solve,

$$\mathbf{T}^{\star} = \arg \min_{(\mathbf{R},\boldsymbol{t}) \in \mathrm{SE}(3)} \sum_{i=1}^{M} d_{P_i} \left(\mathbf{T}\boldsymbol{x}_i\right)^2 \tag{6.2}$$

where SE(3) is the Special Euclidean group in 3D, $\mathbf{T}\boldsymbol{x}$ is the Euclidean transformation of the point $\boldsymbol{x}_i$ and $d_{p_i}(\cdot)$ is the distance to the primitive P_i.
To solve this problem, correspondences must be found between the two point cloud matrices $\mathbf{X}$ and $\mathbf{Y}$. Due to the points collected from

6.2 Method

The method chosen for this chapter is twofold: first outliers are rejected using GNC and RCQP for determining the optimal transformation $\mathbf{T}^{\star}$. One of the common problems of PCR is the presence of outliers, exceeding 95 % is not uncommon [137]. Outlier removal is thus necessary, which is chosen in the form GNC. To find correspondences, a point cloud feature which utilizes the clusters of points produced by the fingertips. Thus the Fast Point Feature Histograms (FPFH) descriptor is chosen. Using these features descriptors d_t are found for the target data i.e. $\mathbf{Y}$ and the source data $\mathbf{X}$ i.e. d_s. Using these matching pairs of points are found using an exhaustive search and a similarity threshold of 0.01

6.2.1 Graduated Non-Convexity

GNC refers to a phenomenon that arises in optimization problems where the objective function exhibits non-convexity and has multiple local optima. In contrast to traditional non-convex optimization, GNC exploits the presence of these local optima to find better solutions progressively.

To understand GNC, let's start by defining some terms. In optimization, a convex function is one that has a unique global minimum. Mathematically, a function $f(\boldsymbol{x})$ defined on a convex set $\mathbf{X}$ is convex if for any two points $\boldsymbol{x}_1$ and

$\boldsymbol{x}_2$ in $\mathbf{X}$ and any $\lambda \in [0, 1]$, the following inequality holds:

$$f(\lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1 - \lambda) f(\boldsymbol{x}_2) \tag{6.3}$$

This inequality essentially means that the function lies below the line segment connecting any two points on its graph. Convex optimization is a well-studied field with efficient algorithms for finding the global minimum.

However, many real-world problems involve non-convex functions, which may have multiple local minima, saddle points, or other complex structures. Traditional optimization methods struggle with non-convex problems because they can get stuck in local optima, unable to find the global optimum.

GNC takes a different approach. Instead of trying to escape local optima, it leverages them to improve the optimization process. The idea is to gradually increase the level of non-convexity in the objective function during optimization. This process allows the algorithm to refine its solution by escaping local optima at a controlled pace.

A common technique used in GNC is to introduce a parameter t that controls the level of non-convexity. As t increases, the objective function becomes more non-convex. The optimization algorithm starts with a low value of t where the objective function is approximately convex. It then gradually increases t over time, exploring the non-convex landscape.

The introduction of t is often done by incorporating a regularization term into the objective function. The regularization term helps to maintain the properties of the convex function, ensuring that the optimization algorithm does not diverge. As t increases, the regularization term becomes less significant compared to the non-convex part of the objective function, allowing the algorithm to escape local optima.

A common form of the objective function in GNC is:

$$F_t(x) = f(x) + t\phi(x) \tag{6.4}$$

where $f(x)$ is the original non-convex function to be optimized, $\phi(x)$ is a convex regularization term, and t is the parameter controlling the non-convexity level.

The choice of the regularization term $\phi(x)$ depends on the problem at hand and the desired behavior of the optimization algorithm. It should be designed in such a way that it encourages exploration of the non-convex landscape as t increases.

During the optimization process, as t gradually increases, the algorithm starts with a nearly convex objective and finds a solution that is close to a local optimum. Then, it gradually explores the non-convex landscape by increasing the influence of the non-convex part of the objective function. This exploration helps the algorithm find better solutions that are not trapped in local optima.

In summary, Graduated Non-Convexity is a technique used in optimization problems with non-convex objective functions. It gradually increases the level of non-convexity to explore the landscape and escape local optima. By incorporating a regularization term that maintains convexity at

Let's consider a specific optimization problem with a non-convex objective function $f(x)$ that we want to minimize. We introduce a parameter t to control the level of non-convexity.

First, let's define the problem in a matrix and vector form. Suppose we have a vector of variables $x \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{m \times n}$ representing the problem's constraints. The optimization problem can be formulated as:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && Ax \leq b, \end{aligned}$$

where $b \in \mathbb{R}^m$ is the vector of constraint values.

To incorporate Graduated Non-Convexity, we introduce a convex regularization term $\phi(x)$ that helps maintain the properties of a convex function. The objective function $F_t(x)$ becomes:

$$F_t(x) = f(x) + t\phi(x). \tag{6.5}$$

Here, t controls the level of non-convexity, and as t increases, the influence of the non-convex part ($t\phi(x)$) becomes more significant.

For example, let's consider a simple case where $f(x)$ is a quadratic function and $\phi(x)$ is a convex regularization term. We can express $f(x)$ using a matrix form as:

$$f(x) = \frac{1}{2}x^T Qx + c^T x, \tag{6.6}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive semi-definite matrix, and $c \in \mathbb{R}^n$ is a vector. The convex regularization term $\phi(x)$ can be written as:

$$\phi(x) = g(Dx), \tag{6.7}$$

where $D \in \mathbb{R}^{p\times n}$ is a matrix, and $g(\cdot)$ is a convex function.

The objective function $F_t(x)$ with GNC can be written as:

$$F_t(x) = \frac{1}{2}x^T Qx + c^T x + tg(Dx). \tag{6.8}$$

Now, during the optimization process, we start with a small value of t e.g., $t = 0$ where the objective function is approximately convex. We solve the optimization problem using traditional convex optimization techniques, such as quadratic programming, to find a solution x_0 that is close to a local optimum.

Then, we gradually increase t over time, which increases the non-convexity of the objective function. As t increases, the regularization term becomes less significant compared to the non-convex part $tg(Dx)$, allowing the algorithm to explore the non-convex landscape and potentially escape local optima.

The specific choice of the convex regularization term $\phi(x)$ and the function $g(\cdot)$ depends on the problem at hand. Common choices include the ℓ_1 norm, total variation, or other convex functions that encourage certain properties or structures in the solution.

In summary, the math behind Graduated Non-Convexity involves introducing a parameter t to control the level of non-convexity and incorporating a convex regularization term $\phi(x)$

Step 1: Initialize the algorithm

Set $t = t_0$ initial value of t

Set x_0 as an initial feasible solution

Set iteration counter $k = 0$

Step 2: Solve the convex subproblem

Solve the following convex subproblem to obtain a solution x_k :

$$\min_x \frac{1}{2}x^T Qx + c^T x$$

subject to $Ax \leq b$

Step 3: Update the parameter t

Step 4: Update the objective function

Compute the convex regularization term: $\phi(x_k)$

Update the objective function with the increased non-convexity:

$$F_t(x) = \frac{1}{2}x^T Qx + c^T x + t\phi(x)$$

Step 5: Solve the updated non-convex problem

Solve the following non-convex problem to obtain a new solution x_{k+1} :

$$\min_x F_t(x)$$

Step 6: Check termination criteria

If termination criteria are satisfied, stop and return the current solution x_{k+1}

Otherwise, go to Step 3

In Step 2, the convex subproblem represents a traditional convex optimization problem that can be solved using various techniques such as quadratic programming or linear programming.
In Step 4, the convex regularization term $\phi(x_k)$ can take different forms based on the problem requirements. For example, if $\phi(x_k)$ is based on the ℓ_1 norm, it can be computed as $\phi(x_k) = \|Dx_k\|_1$ where D is a matrix that determines the sparsity pattern.
In Step 6, the termination criteria can be defined based on the problem's requirements or convergence properties, such as reaching a maximum number of iterations, achieving a desired objective value, or satisfying certain optimality conditions.
The algorithm iteratively solves a sequence of convex subproblems and updated non-convex problems, gradually increasing the non-convexity level with the parameter

$$t$$

. By exploring the non-convex landscape, the algorithm aims to escape local optima and find better solutions.
Remember that the specific implementation details of the GNC algorithm may vary depending on the problem, the chosen convex regularization term, and the optimization techniques used in solving the convex subproblems.

6.2.2 Relaxed Convex Quadratic Programming

The tight dual relaxation method is a powerful approach for solving non-convex optimization problems. It leverages Lagrangian duality theory to obtain a globally optimal solution. This chapter provides an extensive explanation of the method presented in the paper, focusing on the mathematical formulations and key concepts involved. The sizes of vectors and matrices are explicitly mentioned to enhance clarity and understanding.
Before delving into the tight dual relaxation method, it is essential to understand some fundamental concepts from Lagrangian duality theory. In Lagrangian duality, we consider an optimization problem with a primal objective function, subject to constraints. The Lagrangian function is formed by introducing dual variables associated with each constraint. The dual problem seeks to maximize the Lagrangian function over the feasible region defined by the dual variables. Duality theory establishes a relationship between the primal and dual problems, providing bounds on the optimal solutions.
Let's consider a non-convex optimization problem denoted as P. The objective is to find an optimal solution to this problem. The problem involves minimizing a cost function, subject to a set of constraints. The problem can be expressed as follows:

$$
\begin{aligned}
&\text{minimize } f(\boldsymbol{x}) \\
&\text{subject to} \\
&\qquad g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m \\
&\qquad h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p
\end{aligned}
$$

Here, $\boldsymbol{x}$ represents the optimization variables, $f(\boldsymbol{x})$ is the cost function, $g_i(\boldsymbol{x})$ are the inequality constraints, and $h_j(\boldsymbol{x})$ are the equality constraints.
To apply Lagrangian duality, we introduce dual variables λ_i for the inequality constraints and ν_j for the equality constraints. The Lagrangian function for problem P is defined as:

$$
L(\boldsymbol{x}, \lambda, \nu) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{p} \nu_j h_j(\boldsymbol{x}) \tag{6.9}
$$

Here, $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ and $\nu = (\nu_1, \nu_2, \ldots, \nu_p)$ are the vectors of dual variables.
The Lagrangian dual problem seeks to maximize the Lagrangian function over the feasible region defined by the dual variables. The dual problem is formulated as follows:

$$
\begin{aligned}
&\text{maximize } \theta(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \\
&\text{subject to} \\
&\qquad \lambda \geq 0
\end{aligned}
$$

The function $\theta(\lambda, \nu)$ represents the optimal value of the Lagrangian function.
The tight dual relaxation approach aims to solve non-convex optimization problems by leveraging Lagrangian duality. The key idea is to formulate a dual problem that provides tight bounds on the optimal solution. In the tight dual relaxation, we carefully select the set of constraints to maximize the tightness of the relaxation.
In the context of the paper, we consider a specific non-convex optimization problem denoted as $\tilde{P}$. The objective is to find an optimal 3x3
rotation matrix $\mathbf{R}$ that satisfies orthonormality and determinant constraints. The primal problem $\tilde{P}$ can be formulated as a Quadratically Constrained Quadratic Program (QCPQ):

$$
\begin{aligned}
&\text{minimize } \operatorname{tr}(R^\top R - I)^2 \\
&\text{subject to} \\
&\qquad R^\top R = I \\
&\qquad \det(R) = 1
\end{aligned}
$$

Here, $\mathbf{I}$ represents the identity matrix.
To derive the dual problem for $\tilde{P}$, we construct the Lagrangian function $L(\mathbf{R}, \Lambda)$, where $\mathbf{R}$ is the rotation matrix and Λ represents the dual variables associated with the constraints. The Lagrangian function is defined as:

$$
L(R, \Lambda) = \operatorname{tr}(\mathbf{R}^\top \mathbf{R} - \mathbf{I})^2 + \operatorname{tr}(\Lambda^\top (R^\top \mathbf{R} - I)) + \lambda(\det(\mathbf{R}) - 1) \tag{6.10}
$$

Here, Λ represents a symmetric matrix of dual variables and λ is a scalar dual variable.

The tight dual relaxation approach involves constructing a penalized matrix to incorporate the penalization terms corresponding to the different kinds of constraints. The penalized matrix is denoted as $\mathcal{M}$ and defined as:

$$\mathcal{M} = 2(R^\top R - I) + \Lambda + \lambda(I - R^\top R) \tag{6.11}$$

Here, $\mathcal{M}$ is a 3x3 matrix, $R^\top R$ is a 3x3 matrix representing the orthonormality constraints, and I is the identity matrix.
The Lagrangian relaxation involves expressing the dual problem as an unconstrained problem by eliminating the equality constraints. The dual problem can be reformulated as follows:

$$\text{maximize } \mathcal{D}(\Lambda, \lambda) = \inf_R L(R, \Lambda, \lambda) \tag{6.12}$$

Here, $\mathcal{D}(\Lambda, \lambda)$ represents the optimal value of the Lagrangian function after relaxation.
The tight dual relaxation provides a dual bound function $d(\lambda)$ that estimates the optimal value of the primal problem. The dual bound function is defined as:

$$d(\lambda) = \max_\Lambda \mathcal{D}(\Lambda, \lambda) \tag{6.13}$$

Here, $d(\lambda)$ represents the dual bound.
The tight dual relaxation method is a powerful technique for solving non-convex optimization problems. By carefully selecting the set of constraints and formulating the dual problem, this method provides tight bounds on the optimal solution. The Lagrangian relaxation and dual bound function play crucial roles in the process. Understanding and applying the tight dual relaxation approach can lead to efficient and effective solutions to challenging optimization problems.

6.3 Experimental Setup

Chapter 7

In-Hand Manipulation

7.1 Introduction

Here we write the introduction for problem 3.

7.2 Method

Certainly! Here's the combined section that explains the problem definition and the MPC-MPNet method, including the relevant mathematics written in LaTeX:

7.2.1 Problem Definition and MPC-MPNet Method

In this section, we present the problem definition and propose MPC-MPNet, an end-to-end learning-based Kinodynamic Motion Planning (KMP) algorithm. MPC-MPNet aims to find collision-free trajectories connecting a given initial state to a goal region in the configuration space. We describe the problem formulation and the main components of our approach, which include an observation encoder, a neural generator, a neural discriminator, and Model Predictive Control (MPC).

Problem Definition

Let C denote a configuration space (C-space) of a mechanical system, where collision and collision-free regions are represented as C_{obs} and $C_{\text{free}} = C \setminus C_{\text{obs}}$, respectively. The state space X consists of states $x = (c, \dot{c}) \in X$ that contain a configuration $c \in C$ and its time derivative $\dot{c}$. Similarly, X_{obs} and X_{free} represent the collision and collision-free state spaces, respectively. The dynamics of the system are described by the implicit set of equations $\dot{x} = f(x, u)$, where u denotes the control input from a feasible control set U. The objective of KMP is to find a collision-free trajectory $\sigma = \left[(x, u, \tau)_t\right]_{t=0}^{T}$, consisting of a sequence of states $[x_t]_{t=0}^{T}$ and controls $[u_t]_{t=0}^{T}$ with their corresponding durations $[\tau_t]_{t=0}^{T}$, such that $x(0) = x_{\text{init}}$ and $x(T) \in X_{\text{goal}}$.

7.2.2 MPC-MPNet Method

MPC-MPNet is an end-to-end learning-based KMP algorithm that iteratively generates waypoints and local steering trajectories to construct collision-free paths between the start and goal states. The method comprises an observation encoder, a neural generator, a neural discriminator, and two planning algorithms named MPC-MPNetPath and MPC-MPNetTree. We describe each component in detail:

Observation Encoder

The observation encoder embeds the workspace information, represented as voxel maps v, into latent features Z containing critical anchor points for the neural generator and discriminator. Voxel maps are volumetric with dimensions $L \times W \times H \times C$, where L, W, H, and C represent the length, width, height, and number of channels, respectively. To address the computational inefficiency of 3D convolutional neural networks (CNNs) due to their cubic representations and empty volumes, we convert the voxel maps into voxel patches with dimensions $L \times W \times \hat{C}$, where $\hat{C} = HC$. This conversion allows us to use 2D CNNs for learning the embeddings.

Neural Generator

The neural generator G, parameterized by θ_g, is a stochastic neural model that generates intermediate waypoints $\hat{x}_{t+1}$ given the environment encoding Z, the robot's current state $x_t \in X_{\text{free}}$, and the goal state $x_{\text{goal}} \in X_{\text{test}}$
MPC-MPNet (continued)

Neural Discriminator

The neural discriminator D, parameterized by θ_d, is a binary classifier that distinguishes between valid and invalid paths. It receives the environment encoding Z and a trajectory σ and produces a probability $p(\sigma)$ that the trajectory is collision-free.

Model Predictive Control

Model Predictive Control (MPC) is a control strategy that computes a sequence of control inputs over a finite time horizon by solving a constrained optimization problem. At each time step, MPC uses the current state x_t and the goal state x_{goal} to generate a candidate trajectory, which is evaluated by the neural discriminator. MPC selects the first control input of the best-performing trajectory and applies it to the system. The process is repeated at each time step, with the initial state updated to the current state.

MPC-MPNetPath and MPC-MPNetTree

We introduce two planning algorithms that use MPC-MPNet to generate collision-free paths. MPC-MPNetPath generates a single trajectory between the initial and goal states by iteratively applying MPC and refining the path with the neural generator. MPC-MPNetTree constructs a search tree by iteratively generating and evaluating candidate trajectories using MPC and the neural discriminator. The search tree is used to construct the final path by backtracking from the goal node to the start node.

Mathematical Formulation

The MPC-MPNet algorithm can be formulated as follows:

$$\text{Find } \sigma = \left[(x, u, \tau)_t\right]_{t=0}^{T} \text{ such that } x(0) = x_{\text{init}}, x(T) \in X_{\text{goal}}, \text{ and } \sigma \in \mathcal{F}, \tag{7.1}$$

where $\mathcal{F}$ denotes the set of collision-free trajectories. The problem can be solved by iteratively applying MPC and the neural generator until a collision-free trajectory is found. The MPC optimization problem can be written as:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=0}^{T-1} l(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = f(x_t, u_t) \\ & x_0 = x_t \\ & x_T \in X_{\text{goal}} \\ & x_t \in X_{\text{free}} \\ & u_t \in U \end{aligned} \tag{7.2}$$

where $l(x_t, u_t)$ is a cost function, $f(x_t, u_t)$ is the system dynamics, x_0 is the initial state, x_T is the final state, and U is the feasible control set. The neural generator is trained to generate intermediate waypoints $\hat{x}_{t+1}$ that minimize the difference between the MPC-generated trajectory and the true trajectory. The discriminator is trained to distinguish between valid and invalid trajectories, where invalid trajectories are those that collide with obstacles.

7.3 Conclusion

Chapter 8

Discussion

Chapter 9

Conclusion

this is the conclusion

Bibliography

[1] El Zaatari, Shirine et al. "Cobot programming for collaborative industrial tasks: An overview". In: *Robotics and Autonomous Systems* 116 (June 2019), pp. 162–180. DOI: 10.1016/j.robot.2019.03.003.

[2] *What is computer vision?* https://www.ibm.com/topics/computer-vision. Accessed: 2022-11-02.

[3] Zimmer, Henning. "Correspondence problems in computer vision". PhD thesis. Jan. 2012.

[4] LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[5] docker. *What is a container*. URL: https://www.docker.com/resources/what-container/.

[6] Monkman, Gareth J. et al. "Robot Grippers". PhD thesis. Jan. 2004, pp. 5–6.

[7] Mihelj, Matjaž et al. "Robotics Second Edition". PhD thesis. Jan. 2019, p. 1.

[8] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022, p. 284.

[9] Quigley, Morgan et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software* 3 (Jan. 2009).

[10] Publications, United Nations. "Inequality in Asia and the Pacific in the Era of the 2030 Agenda for Sustainable Development". In: 2018. URL: https://www.unescap.org/publications/inequality-asia-and-pacific-era-2030-agenda-sustainable-development.

[11] Wenwen, Shi et al. "Analysis and Control of Human Error". In: *Procedia Engineering* 26 (Dec. 2011), pp. 2126–2132. DOI: 10.1016/j.proeng.2011.11.2415.

[12] Galin, Rinat et al. "Cobots and the benefits of their implementation in intelligent manufacturing". In: *IOP Conference Series: Materials Science and Engineering* 862 (May 2020), p. 032075. DOI: 10.1088/1757-899X/862/3/032075.

[13] Raptopoulos, Fredy, Koskinopoulou, Maria, and Maniadakis, Michail. "Robotic Pick-and-Toss Facilitates Urban Waste Sorting". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. 2020, pp. 1149–1154. DOI: 10.1109/CASE48305.2020.9216746.

[14] Talpur, Mir Sajjad Hussain and Shaikh, Murtaza Hussain. *Automation of Mobile Pick and Place Robotic System for Small Food Industry*. 2012. DOI: 10.48550/ARXIV.1203.4475. URL: https://arxiv.org/abs/1203.4475.

[15] Yamanaka, Yuta et al. "Development of a Food Handling Soft Robot Hand Considering a High-speed Pick-and-place Task". In: *2020 IEEE/SICE International Symposium on System Integration (SII)*. 2020, pp. 87–92. DOI: 10.1109/SII46433.2020.9026282.

[16] Lee, Sukhan and Lee, Yeonho. "Real-Time Industrial Bin-Picking with a Hybrid Deep Learning-Engineering Approach". In: *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2020, pp. 584–588. DOI: 10.1109/BigComp48618.2020.00015.

[17] Mnyusiwalla, Hussein et al. "A Bin-Picking Benchmark for Systematic Evaluation of Robotic Pick-and-Place Systems". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1389–1396. DOI: 10.1109/LRA.2020.2965076.

[18] Wong, Ching-Chang et al. "Generic Development of Bin Pick-and-Place System Based on Robot Operating System". In: *IEEE Access* 10 (2022), pp. 65257–65270. DOI: 10.1109/ACCESS.2022.3182114.

[19] He, Zaixing et al. “6D Pose Estimation of Objects: Recent Technologies and Challenges”. In: *Applied Sciences* 11.1 (2021). ISSN: 2076-3417. DOI: 10.3390/app11010228. URL: https://www.mdpi.com/2076-3417/11/1/228.

[20] ., Taryudi and Wang, Ming-Shyan. “3D object pose estimation using stereo vision for object manipulation system”. In: May 2017, pp. 1532–1535. DOI: 10.1109/ICASI.2017.7988217.

[21] Oh, Jong-Kyu, Lee, Sukhan, and Lee, Chan-Ho. “Stereo vision based automation for a bin-picking solution”. In: *International Journal of Control, Automation and Systems* 10.2 (Apr. 2012), pp. 362–373. ISSN: 2005-4092. DOI: 10.1007/s12555-012-0216-9. URL: https://doi.org/10.1007/s12555-012-0216-9.

[22] Abdelaal, Mahmoud et al. “Uncalibrated stereo vision with deep learning for 6-DOF pose estimation for a robot arm system”. In: *Robotics and Autonomous Systems* 145 (2021), p. 103847. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2021.103847. URL: https://www.sciencedirect.com/science/article/pii/S0921889021001329.

[23] Nakano, Yoshihiro. “Stereo Vision Based Single-Shot 6D Object Pose Estimation for Bin-Picking by a Robot Manipulator”. In: *CoRR* abs/2005.13759 (2020). arXiv: 2005.13759. URL: https://arxiv.org/abs/2005.13759.

[24] Kozák, Viktor et al. “Data-Driven Object Pose Estimation in a Practical Bin-Picking Application”. In: *Sensors* 21.18 (2021). ISSN: 1424-8220. DOI: 10.3390/s21186093. URL: https://www.mdpi.com/1424-8220/21/18/6093.

[25] Xu, Chi et al. “6DoF Pose Estimation of Transparent Object from a Single RGB-D Image”. In: *Sensors* 20.23 (2020). ISSN: 1424-8220. DOI: 10.3390/s20236790. URL: https://www.mdpi.com/1424-8220/20/23/6790.

[26] Straszheim, Troy et al. *Shadow Robot*. URL: https://www.shadowrobot.com/dexterous-hand-series/. (accessed: 29.07.2022).

[27] Robot, Shadow. *The Shadow Robot Company*. URL: https://github.com/shadow-robot.

[28] Chitta, Sachin et al. “ros_control: A generic and simple control framework for ROS”. In: *The Journal of Open Source Software* (2017). DOI: 10.21105/joss.00456. URL: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf.

[29] Koenig, N. and Howard, A. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: 3 (2004), 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[30] Robotics, Shadow. *Shadow Robotics*. URL: https://www.shadowrobot.com/.

[31] Robotics, Shadow. *The Shadow Robot Company*. URL: https://github.com/shadow-robot.

[32] *Dimensions*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_dimensions.html. Accessed: 2023-04-26.

[33] Rahman, Akhlaquor and Al-Jumaily, Adel. “Design and Development of a Bilateral Therapeutic Hand Device for Stroke Rehabilitation”. In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 405. DOI: 10.5772/56809. eprint: https://doi.org/10.5772/56809. URL: https://doi.org/10.5772/56809.

[34] *Bones Of The Hand*. https://freesvg.org/bones-of-the-hand. Accessed: 2023-04-26.

[35] Robotics, Shadow. *Gazebo*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sim_gazebo.html%7D.

[36] *First time users*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sd_first_time_users.html. Accessed: 2023-04-26.

[37] Straszheim, Troy et al. *Conceptual overview of ROS catkin*. URL: %5Curl%7Bhttp://wiki.ros.org/catkin/conceptual_overview%7D.

[38] Beek, Bas van et al. *About Us*. URL: %5Curl%7Bhttps://numpy.org/%7D.

[39] team, OpenCV. *OpenCV*. URL: %5Curl%7Bhttps://opencv.org/%7D.

[40] Carroll, Michael. *dynamic-reconfigure*. URL: %5Curl%7Bhttp://google.com%7D.

[41] Coleman, David et al. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014. arXiv: 1404.3785 [cs.RO].

[42] Şucan, Ioan A., Moll, Mark, and Kavraki, Lydia E. "The Open Motion Planning Library". In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012). https://ompl.kavrakilab.org, pp. 72–82. DOI: 10.1109/MRA.2012.2205651.

[43] Robotics, Shadow. *Control Description*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/ed_control_description.html%7D.

[44] Robotics, Shadow. *Firmware*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sd_firmware.html%7D.

[45] Robotics, Shadow. *Controlling the Hand 2*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sd_controling_hand.html%7D.

[46] Staven, Victor Melbye. *in_hand_pose_estimation*. URL: %5Curl%7Bhttps://github.com/vmstavens/in_hand_pose_estimation%7D.

[47] Staven, Victor Melbye. *ros_utils*. URL: %5Curl%7Bhttps://github.com/vmstavens/ros_utils%7D.

[48] Pérez-González, Antonio et al. "A modified elastic foundation contact model for application in 3D models of the prosthetic knee". In: *Medical Engineering & Physics* 30.3 (2008), pp. 387–398. ISSN: 1350-4533. DOI: https://doi.org/10.1016/j.medengphy.2007.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1350453307000616.

[49] Hertz, H. "On the Contact of Rigid Elastic Solids and on Hardness". In: *Ch 6: Assorted Papers* (1882).

[50] Bruno Siciliano, Oussama Khatib, ed. *Springer Handbook of Robotics*. Springer Berlin, Heidelberg, 2016.

[51] Xydas, Nicholas and Kao, Imin. "Modeling of Contact Mechanics and Friction Limit Surfaces for Soft Fingers in Robotics, with Experimental Results". In: *The International Journal of Robotics Research* 18.9 (1999), pp. 941–950. DOI: 10.1177/02783649922066673. eprint: https://doi.org/10.1177/02783649922066673. URL: https://doi.org/10.1177/02783649922066673.

[52] Yuvaraj, S., Malayalamurthi, R., and Raja, K. Venkatesh. "The haptic and perceptional characteristics of an anthropomorphic curved soft finger structure". In: *Curved and Layered Structures* 6.1 (2019), pp. 161–168. DOI: doi:10.1515/cls-2019-0013. URL: https://doi.org/10.1515/cls-2019-0013.

[53] Howe, R.D., Kao, I., and Cutkosky, M.R. "The sliding of robot fingers under combined torsion and shear loading". In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. 1988, 103–105 vol.1. DOI: 10.1109/ROBOT.1988.12032.

[54] Flugge, Wilhelm. *Viscoelasticity*. Blaisdell Publishing Company, Waltham, MA, 1967.

[55] Maxwell, James Clerk. *On the dynamical theory of gases*. Royal Society, 1867.

[56] Fung, Yuan-Cheng. "Mechanical properties and active remodeling of blood vessels". In: *Biomechanics*. Springer, 1993.

[57] Tiezzi, Paolo and Kao, Imin. "Modeling of Viscoelastic Contacts and Evolution of Limit Surface for Robotic Contact Interface". In: *Robotics, IEEE Transactions on* 23 (May 2007), pp. 206–217. DOI: 10.1109/TRO.2006.889494.

[58] Tiezzi, P. and Kao, I. "Characteristics of contact and limit surface for viscoelastic fingers". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 2006, pp. 1365–1370. DOI: 10.1109/ROBOT.2006.1641899.

[59] Tiezzi, Paolo, Kao, Imin, and Vassura, G. "Effect of layer compliance on frictional behavior of soft robotic fingers". In: *Advanced Robotics* 21 (Oct. 2007), pp. 1653–1670. DOI: 10.1163/156855307782227390.

[60] Kalker, J. J. "On the Contact Problem in Elastostatics". In: *Unilateral Problems in Structural Analysis*. Ed. by Del Piero, Gianpietro and Maceri, Franco. Vienna: Springer Vienna, 1985, pp. 81–85. ISBN: 978-3-7091-2632-.

[61] Kozhevnikov, I.F. et al. "A new algorithm for computing the indentation of a rigid body of arbitrary shape on a viscoelastic half-space". In: *International Journal of Mechanical Sciences* 50.7 (2008), pp. 1194–1202. ISSN: 0020-7403. DOI: https://doi.org/10.1016/j.ijmecsci.2008.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0020740308000623.

[62] Marmo, Francesco and Rosati, Luciano. "A General Approach to the Solution of Boussinesq's Problem for Polynomial Pressures Acting over Polygonal Domains". In: *Journal of Elasticity* 122.1 (Jan. 2016), pp. 75–112. ISSN: 1573-2681. DOI: 10.1007/s10659-015-9534-5. URL: https://doi.org/10.1007/s10659-015-9534-5.

[63] Li, Junshan and Berger, Edward J. "A Boussinesq–Cerruti Solution Set for Constant and Linear Distribution of Normal and Tangential Load over a Triangular Area". In: *Journal of elasticity and the physical science of solids* 63.2 (May 2001), pp. 137–151. ISSN: 1573-2681. DOI: 10.1023/A:1014013425423. URL: https://doi.org/10.1023/A:1014013425423.

[64] Wasko, Wojciech et al. "Contact Modelling and Tactile Data Processing for Robot Skins". In: *Sensors* 19.4 (2019). ISSN: 1424-8220. DOI: 10.3390/s19040814. URL: https://www.mdpi.com/1424-8220/19/4/814.

[65] Slaughter, William S. *The Linearized Theory of Elasticity*. Springer, 2012.

[66] Hills, D., Nowell, D., and Barber, James. "KL Johnson and contact mechanics". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 231 (Feb. 2016). DOI: 10.1177/0954406216634121.

[67] Lee, S.Y., Kuo, Y.H., and Lin, F.Y. "Stability of a Timoshenko beam resting on a Winkler elastic foundation". In: *Journal of Sound and Vibration* 153.2 (1992), pp. 193–202. ISSN: 0022-460X. DOI: https://doi.org/10.1016/S0022-460X(05)80001-X. URL: https://www.sciencedirect.com/science/article/pii/S0022460X0580001X.

[68] Eisenberger, M. and Clastornik, J. "Vibrations and buckling of a beam on a variable winkler elastic foundation". In: *Journal of Sound and Vibration* 115.2 (1987), pp. 233–241. ISSN: 0022-460X. DOI: https://doi.org/10.1016/0022-460X(87)90469-X. URL: https://www.sciencedirect.com/science/article/pii/0022460X8790469X.

[69] Fregly, Benjamin J., Bei, Yanhong, and Sylvester, Mark E. "Experimental evaluation of an elastic foundation model to predict contact pressures in knee replacements". In: *Journal of Biomechanics* 36.11 (2003), pp. 1659–1668. ISSN: 0021-9290. DOI: https://doi.org/10.1016/S0021-9290(03)00176-3. URL: https://www.sciencedirect.com/science/article/pii/S0021929003001763.

[70] Kao, Imin and Cutkosky, Mark R. "Quasistatic Manipulation with Compliance and Sliding". In: *The International Journal of Robotics Research* 11.1 (1992), pp. 20–40. DOI: 10.1177/027364992011001102. eprint: https://doi.org/10.1177/027364992011001102. URL: https://doi.org/10.1177/027364992011001102.

[71] Howe, Robert D. and Cutkosky, Mark R. "Practical Force-Motion Models for Sliding Manipulation". In: *The International Journal of Robotics Research* 15.6 (1996), pp. 557–572. DOI: 10.1177/027364996015006003. eprint: https://doi.org/10.1177/027364996015006003. URL: https://doi.org/10.1177/027364996015006003.

[72] Kao, Imin and Yang, Fuqian. "Stiffness and Contact Mechanics for Soft Fingers in Grasping and Manipulation". In: *Robotics and Automation, IEEE Transactions on* 20 (Mar. 2004), pp. 132–135. DOI: 10.1109/TRA.2003.820868.

[73] Ciocarlie, Matei, Lackner, Claire, and Allen, Peter. "Soft Finger Model with Adaptive Contact Geometry for Grasping and Manipulation Tasks". In: *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*. 2007, pp. 219–224. DOI: 10.1109/WHC.2007.103.

[74] Sabat, Lovely and Kundu, Chinmay Kumar. "History of Finite Element Method: A Review". In: *Recent Developments in Sustainable Infrastructure*. Ed. by Das, Bibhuti Bhusan et al. Singapore: Springer Singapore, 2021, pp. 395–404. ISBN: 978-981-15-4577-1.

[75] Klocke, Fritz et al. "Examples of FEM application in manufacturing technology". In: *Journal of Materials Processing Technology* 120.1 (2002), pp. 450–457. ISSN: 0924-0136. DOI: https://doi.org/10.1016/S0924-0136(01)01210-9. URL: https://www.sciencedirect.com/science/article/pii/S0924013601012109.

[76] Telliskivi, T and Olofsson, U. "Contact mechanics analysis of measured wheel-rail profiles using the finite element method". In: *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 215.2 (2001), pp. 65–72. DOI: 10.1243/0954409011531404. eprint: https://doi.org/10.1243/0954409011531404. URL: https://doi.org/10.1243/0954409011531404.

[77] Põdra, Priit and Andersson, Sören. "Simulating sliding wear with finite element method". In: *Tribology International* 32.2 (1999), pp. 71–81. ISSN: 0301-679X. DOI: https://doi.org/10.1016/S0301-679X(99)00012-2. URL: https://www.sciencedirect.com/science/article/pii/S0301679X99000122.

[78] Pantuso, Daniel, Bathe, Klaus-Jürgen, and Bouzinov, Pavel A. "A finite element procedure for the analysis of thermo-mechanical solids in contact". In: *Computers & Structures* 75.6 (2000), pp. 551–573. ISSN: 0045-7949. DOI: https://doi.org/10.1016/S0045-7949(99)00212-6. URL: https://www.sciencedirect.com/science/article/pii/S0045794999002126.

[79] Liang, Xiaodong, Ali, Mohammad Zawad, and Zhang, Huaguang. "Induction Motors Fault Diagnosis Using Finite Element Method: A Review". In: *IEEE Transactions on Industry Applications* 56.2 (2020), pp. 1205–1217. DOI: 10.1109/TIA.2019.2958908.

[80] Ye, Zhiqiu et al. "Soft Robot Skin With Conformal Adaptability for On-Body Tactile Perception of Collaborative Robots". In: *IEEE Robotics and Automation Letters* (Mar. 2022), pp. 1–1. DOI: 10.1109/LRA.2022.3155225.

[81] Lu, Guan, Fu, Shiwen, and Xu, Yiming. "Design and Experimental Research of Robot Finger Sliding Tactile Sensor Based on FBG". In: *Sensors* 22.21 (2022). ISSN: 1424-8220. DOI: 10.3390/s22218390. URL: https://www.mdpi.com/1424-8220/22/21/8390.

[82] Ciocarlie, M., Miller, A., and Allen, P. "Grasp analysis using deformable fingers". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 4122–4128. DOI: 10.1109/IROS.2005.1545525.

[83] Narang, Yashraj S. et al. "Sim-to-Real for Robotic Tactile Sensing via Physics-Based Simulation and Learned Latent Projections". In: *CoRR* abs/2103.16747 (2021). arXiv: 2103.16747. URL: https://arxiv.org/abs/2103.16747.

[84] Narang, Yashraj S. et al. "Interpreting and Predicting Tactile Signals via a Physics-Based and Data-Driven Framework". In: *CoRR* abs/2006.03777 (2020). arXiv: 2006.03777. URL: https://arxiv.org/abs/2006.03777.

[85] Sferrazza, Carmelo et al. "Ground Truth Force Distribution for Learning-Based Tactile Sensing: A Finite Element Approach". In: *IEEE Access* PP (Nov. 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2956882.

[86] Calandra, Roberto et al. "More Than a Feeling: Learning to Grasp and Regrasp using Vision and Touch". In: *CoRR* abs/1805.11085 (2018). arXiv: 1805.11085. URL: http://arxiv.org/abs/1805.11085.

[87] Spiers, Adam et al. "Single-Grasp Object Classification and Feature Extraction with Simple Robot Hands and Tactile Sensors". In: *IEEE Transactions on Haptics* 9 (Jan. 2016), pp. 60–71. DOI: 10.1109/TOH.2016.2521378.

[88] Higuera, Carolina, Boots, Byron, and Mukadam, Mustafa. *Learning to Read Braille: Bridging the Tactile Reality Gap with Diffusion Models*. 2023. arXiv: 2304.01182 [cs.RO].

[89] Ruppel, Philipp et al. "Simulation of the SynTouch BioTac Sensor". In: (2019). Ed. by Strand, Marcus et al., pp. 374–387.

[90] Huang, Xiaoshui et al. *A comprehensive survey on point cloud registration*. 2021. arXiv: 2103.02690 [cs.CV].

[91] Oberweger, Markus, Wohlhart, Paul, and Lepetit, Vincent. *Hands Deep in Deep Learning for Hand Pose Estimation*. 2016. arXiv: 1502.06807 [cs.CV].

[92] Toshev, Alexander and Szegedy, Christian. "DeepPose: Human Pose Estimation via Deep Neural Networks". In: *Computer Vision and Pattern Recognition*. 2014.

[93] Mathis, Alexander et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature Neuroscience* 21.9 (Sept. 2018), pp. 1281–1289. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0209-y. URL: https://doi.org/10.1038/s41593-018-0209-y.

[94] Huang, Xiaoshui et al. "A Systematic Approach for Cross-Source Point Cloud Registration by Preserving Macro and Micro Structures". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3261–3276. DOI: 10.1109/TIP.2017.2695888.

[95] Huang, Xiaoshui et al. "A Coarse-to-Fine Algorithm for Matching and Registration in 3D Cross-Source Point Clouds". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.10 (2018), pp. 2965–2977. DOI: 10.1109/TCSVT.2017.2730232.

[96] Yang, Zhenpei et al. "Extreme Relative Pose Estimation for RGB-D Scans via Scene Completion". In: June 2019, pp. 4526–4535. DOI: 10.1109/CVPR.2019.00466.

[97] Wang, Lingjing et al. *Non-Rigid Point Set Registration Networks*. 2019. arXiv: 1904.01428 [cs.GR].

[98] Huang, Xiaoshui, Mei, Guofeng, and Zhang, Jian. *Feature-metric Registration: A Fast Semi-supervised Approach for Robust Point Cloud Registration without Correspondences*. 2020. arXiv: 2005.01014 [cs.CV].

[99] Zeng, Andy et al. *3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions*. 2017. arXiv: 1603.08182 [cs.CV].

[100] Deng, Haowen, Birdal, Tolga, and Ilic, Slobodan. *PPFNet: Global Context Aware Local Features for Robust 3D Point Matching*. 2018. arXiv: 1802.02669 [cs.CV].

[101] Gojcic, Zan et al. *The Perfect Match: 3D Point Cloud Matching with Smoothed Densities*. 2019. arXiv: 1811.06879 [cs.CV].

[102] Besl, Paul J. and McKay, Neil D. "A Method for Registration of 3-D Shapes". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14 (1992), pp. 239–256.

[103] Chen, Yang and Medioni, Gérard. "Object modelling by registration of multiple range images". In: *Image and Vision Computing* 10.3 (1992). Range Image Understanding, pp. 145–155. ISSN: 0262-8856. DOI: https://doi.org/10.1016/0262-8856(92)90066-C. URL: https://www.sciencedirect.com/science/article/pii/026288569290066C.

[104] Segal, Aleksandr, Hähnel, Dirk, and Thrun, Sebastian. "Generalized-ICP". In: June 2009. DOI: 10.15607/RSS.2009.V.021.

[105] Shi, Xiaojing, Liu, Tao, and Han, Xie. "Improved Iterative Closest Point(ICP) 3D point cloud registration algorithm based on point cloud filtering and adaptive fireworks for coarse registration". In: *International Journal of Remote Sensing* 41.8 (2020), pp. 3197–3220. DOI: 10.1080/01431161.2019.1701211. eprint: https://doi.org/10.1080/01431161.2019.1701211. URL: https://doi.org/10.1080/01431161.2019.1701211.

[106] Zhu, Hao et al. "A Review of Point Set Registration: From Pairwise Registration to Groupwise Registration". en. In: *Sensors (Basel)* 19.5 (Mar. 2019).

[107] Livi, Lorenzo and Rizzi, Antonello. "The Graph Matching Problem". In: *Pattern Anal. Appl.* 16.3 (Aug. 2013), pp. 253–283. ISSN: 1433-7541. DOI: 10.1007/s10044-012-0284-8. URL: https://doi.org/10.1007/s10044-012-0284-8.

[108] Zhou, Feng and De la Torre, Fernando. "Factorized Graph Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9 (2016), pp. 1774–1789. DOI: 10.1109/TPAMI.2015.2501802.

[109] Leordeanu, Marius and Hebert, Martial. "A spectral technique for correspondence problems using pairwise constraints". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* 2 (2005), 1482–1489 Vol. 2.

[110] Zass, Ron and Shashua, Amnon. "Probabilistic graph and hypergraph matching". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587500.

[111] Duchenne, Olivier et al. "A Tensor-Based Algorithm for High-Order Graph Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12 (2011), pp. 2383–2395. DOI: 10.1109/TPAMI.2011.110.

[112] Zhu, Hu et al. "Elastic Net Constraint-Based Tensor Model for High-Order Graph Matching". In: *IEEE Transactions on Cybernetics* 51.8 (2021), pp. 4062–4074. DOI: 10.1109/TCYB.2019.2936176.

[113] Müller, Meinard. *Information Retrieval for Music and Motion*. Jan. 2007. ISBN: 978-3-540-74047-6. DOI: 10.1007/978-3-540-74048-3.

[114] Fan, Jingfan et al. "Convex hull indexed Gaussian mixture model (CH-GMM) for 3D point set registration". In: *Pattern Recognition* 59 (2016). Compositional Models and Structured Learning for Visual Recognition, pp. 126–141. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2016.02.023. URL: https://www.sciencedirect.com/science/article/pii/S0031320316000947.

[115] Evangelidis, Georgios D. et al. "A Generative Model for the Joint Registration of Multiple Point Sets". In: *Computer Vision – ECCV 2014*. Ed. by Fleet, David et al. Cham: Springer International Publishing, 2014, pp. 109–122. ISBN: 978-3-319-10584-0.

[116] Yuan, Wentao et al. *DeepGMR: Learning Latent Gaussian Mixture Models for Registration*. 2020. arXiv: 2008.09088 [cs.CV].

[117] Yang, Heng, Shi, Jingnan, and Carlone, Luca. "TEASER: Fast and Certifiable Point Cloud Registration". In: *IEEE Transactions on Robotics* 37.2 (2021), pp. 314–333. DOI: 10.1109/TRO.2020.3033695.

[118] Sun, Lei. *IRON: Invariant-based Highly Robust Point Cloud Registration*. 2021. arXiv: 2103.04357 [cs.CV].

[119] Briales, Jesus and Gonzalez-Jimenez, Javier. "Convex Global 3D Registration With Lagrangian Duality". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[120] Yang, Heng et al. "Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 1127–1134. DOI: 10.1109/lra.2020.2965893. URL: https://doi.org/10.1109%2Flra.2020.2965893.

[121] LaValle, Steven M. *Planning Algorithms*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.

[122] Li, Yanbo, Littlefield, Zakary, and Bekris, Kostas E. *Asymptotically Optimal Sampling-based Kinodynamic Planning*. 2016. arXiv: 1407.2896 [cs.RO].

[123] Li, Linjun et al. "MPC-MPNet: Model-Predictive Motion Planning Networks for Fast, Near-Optimal Planning under Kinodynamic Constraints". In: *CoRR* abs/2101.06798 (2021). arXiv: 2101.06798. URL: https://arxiv.org/abs/2101.06798.

[124] Khandate, Gagan et al. *Sampling-based Exploration for Reinforcement Learning of Dexterous Manipulation*. 2023. arXiv: 2303.03486 [cs.RO].

[125] Salisbury, J. Kenneth and Craig, John J. "Articulated Hands: Force Control and Kinematic Issues". In: *The International Journal of Robotics Research* 1.1 (1982), pp. 4–17. DOI: 10.1177/027836498200100102. eprint: https://doi.org/10.1177/027836498200100102. URL: https://doi.org/10.1177/027836498200100102.

[126] Jonker, Ben, Waiboer, Rob, and Aarts, Ronald. "Modelling of joint friction in robotic manipulators with gear transmissions". In: Jan. 2007, pp. 221–243. ISBN: 10-1-4020-5683-4.

[127] Lynch, Kevin M and Park, Frank C. *Modern Robotics: Mechanics, Planning, and Control*. English (US). Cambridge Univeristy Press, 2017. ISBN: 978-1107156302.

[128] Ghaednia, Hamed et al. "Contact Mechanics". In: Nov. 2013, pp. 93–140. ISBN: 978-1-4614-1944-0. DOI: 10.1007/978-1-4614-1945-7_3.

[129] Gazebo. *Control Description*. URL: %5Curl%7Bhttps://classic.gazebosim.org/tutorials?tut=plugins_hello_world&cat=write_plugin%7D.

[130] Vries, Maarten de. *Switch to libb64 for base64 encoding/decoding*. URL: %5Curl%7Bhttps://github.com/ros/ros_comm/pull/1046%7D.

[131] Görner, Michael and Ruppel, Philipp. *biotac_gazebo_plugin*. Version 1.0.0. URL: https://github.com/TAMS-Group/biotac_gazebo_plugin.

[132] Melbye Staven, Victor. *biotac_sim_plugin*. Version 1.0.0. URL: https://github.com/vmstavens/biotac_sim_plugin.

[133] Görner, Michael and Ruppel, Philipp. *Biotac Single Contact Response*. URL: %5Curl%7Bhttps://tams.informatik.uni-hamburg.de/research/datasets/index.php#biotac_single_contact_response%7D.

[134] E-MOTION. *ATI: 6-Axis Force and Torque Sensor (Nano17 Series) 9105-TW-NANO17-E-1.8*. URL: %5Curl%7Bhttps://www.e-motionsupply.com/product_p/9105-tw-nano17-e-1.8.htm%7D.

[135] Olson, Edwin. "AprilTag: A robust and flexible visual fiducial system". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3400–3407. DOI: 10.1109/ICRA.2011.5979561.

[136] University, Stanford. *The Stanford 3D Scanning Repository*. http://graphics.stanford.edu/data/3Dscanrep/.

[137] Parra, Álvaro and Chin, Tat-Jun. "Guaranteed Outlier Removal for Point Cloud Registration with Correspondences". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (Nov. 2017), pp. 1–1. DOI: 10.1109/TPAMI.2017.2773482.

[138] *Ranges*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_ranges.html. Accessed: 2023-04-26.

[139] Ngeo, Jimson, Tamei, Tomoya, and Shibata, Tomohiro. "Continuous and simultaneous estimation of finger kinematics using inputs from an EMG-to-muscle activation model". In: *Journal of neuroengineering and rehabilitation* 11 (Aug. 2014), p. 122. DOI: 10.1186/1743-0003-11-122.

[140] Palmer, A K et al. "Functional wrist motion: a biomechanical study". en. In: *J Hand Surg Am* 10.1 (Jan. 1985), pp. 39–46.

[141] *Finger*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_finger.html. Accessed: 2023-04-26.

[142] Mavrogiannis, Christoforos. "Grasp Synthesis Algorithms for Multifingered Robot Hands". PhD thesis. Mar. 2013.

[143] *Kinematics*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_kinematics.html. Accessed: 2023-04-26.

Appendix A

Shadow Dexterous Hand - Technical Specifications

Table A.1 shows the ROM for the Shadow Dexterous Hand and Table A.2 shown the ROM for a human hand. The shorthand abbreviations used in these tables can be seen listed in Table A.3. The joints are numbered from fingertip to base and thus FF1 refers to the first joint after the fingertip on the first finger i.e. the index finger.

ROM - Shadow Dexterous Hand					
Joint(s)	**Min deg**	**Max deg**	**Min rad**	**Max rad**	**Notes**
FF1, MF1, RF1, LF1	0	90	0	1.571	Coupled
FF2, MF2, RF2, LF2	0	90	0	1.571	
FF3, MF3, RF3, LF3	-15	90	-0.262	1.571	
FF4, MF4, RF4, LF4	-20	20	-0.349	0.349	
LF5	0	45	0	0.785	
TH1	-15	90	-0.262	1.571	
TH2	-40	40	-0.698	0.698	
TH3	-12	12	-0.209	0.209	
TH4	0	70	0	1.222	
TH5	-60	60	-1.047	1.047	
WR1	-40	28	-0.698	0.489	
WR2	-28	10	-0.489	0.174	

Table A.1: The ranges of motion for each joint in the Shadow Dexterous Hand [138].

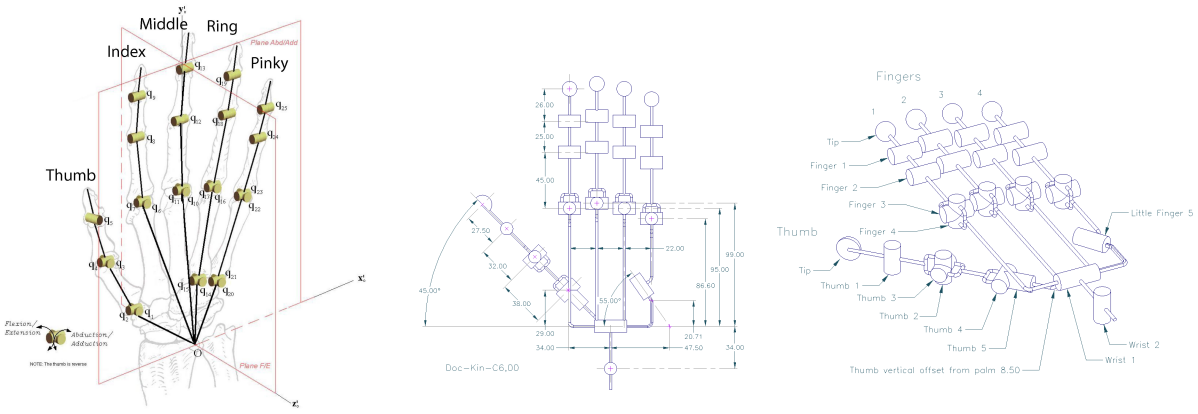
ROM - Human Hand					
Joint(s)	**Min deg**	**Max deg**	**Min rad**	**Max rad**	**Latin Name**
TH1	-15	80			Interphalangeal (IP)
TH2 + TH3	-10	55			Metacarpophalangeal (MCP)
TH4 +TH5	-10	55			Carpometacarpal (CMC)
FF1, MF1, RF1, LF1	0	80			Distal interphalangeal (DIP)
FF2, MF2, RF2, LF2	0	100			Proximal interphalangeal (PIP)
FF3, MF3, RF3, LF3	-45	90			Metacarpophalangeal (MCP)
WR1	-80	80			Radiocarpal
WR2	-28	20			Radiocarpal

Table A.2: The theoretical ROM for each finger joint in human hand [139] and found ROM for the wrist joints [140].

To compare the kinematic structure of the Shadow Dexterous Hand and a human hand, see Fig. A.1.

Joint Name Abbreviation	
Abbreviation	**Full Name**
FF	First Finger
MF	Middle Finger
RF	Ring Finger
LF	Little Finger
WR	Wrist

Table A.3: The abbreviations used to reference [141].



(a) The kinematic tree of a human hand according to [142].

(b) The kinematic tree of the Shadow Dexterous Hand according to [143].

Fig. A.1: The kinematic trees of a human hand and the Shadow Dexterous hand.

Appendix B

Tactile Perception - Simulated Electrode Activations

Below three

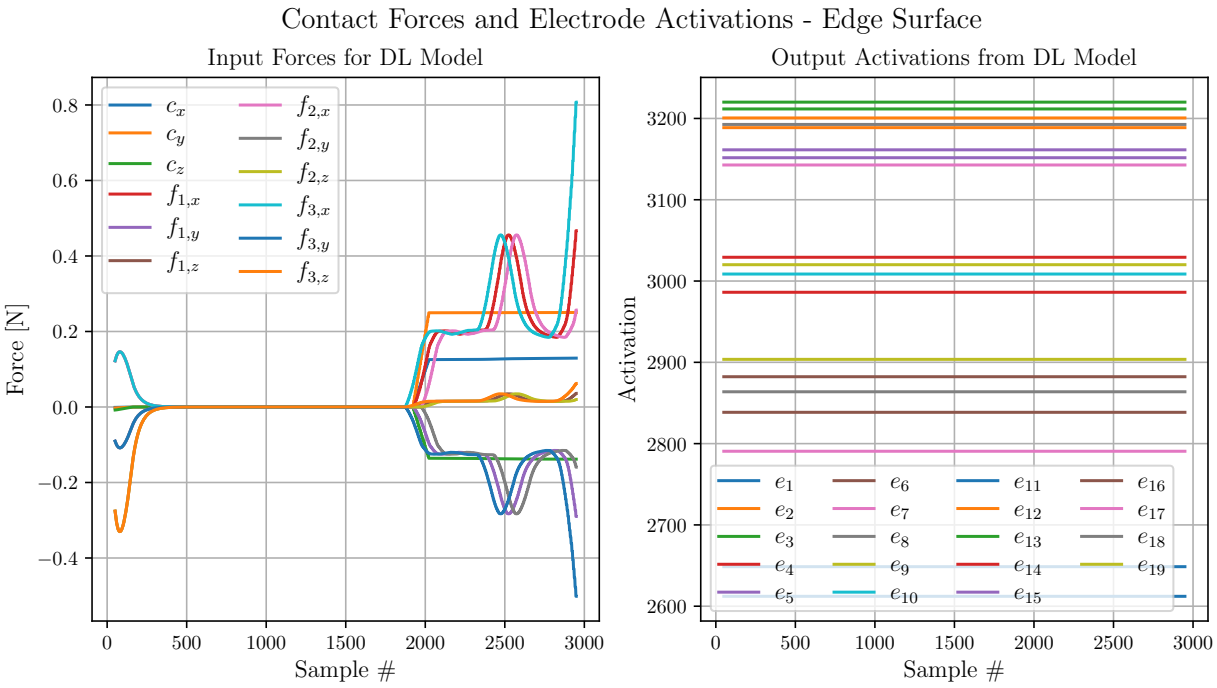


Fig. B.1: he simulated tactile electrode activations when the simulated Shadow Dexterous hand's index finger is in contact with an edge.

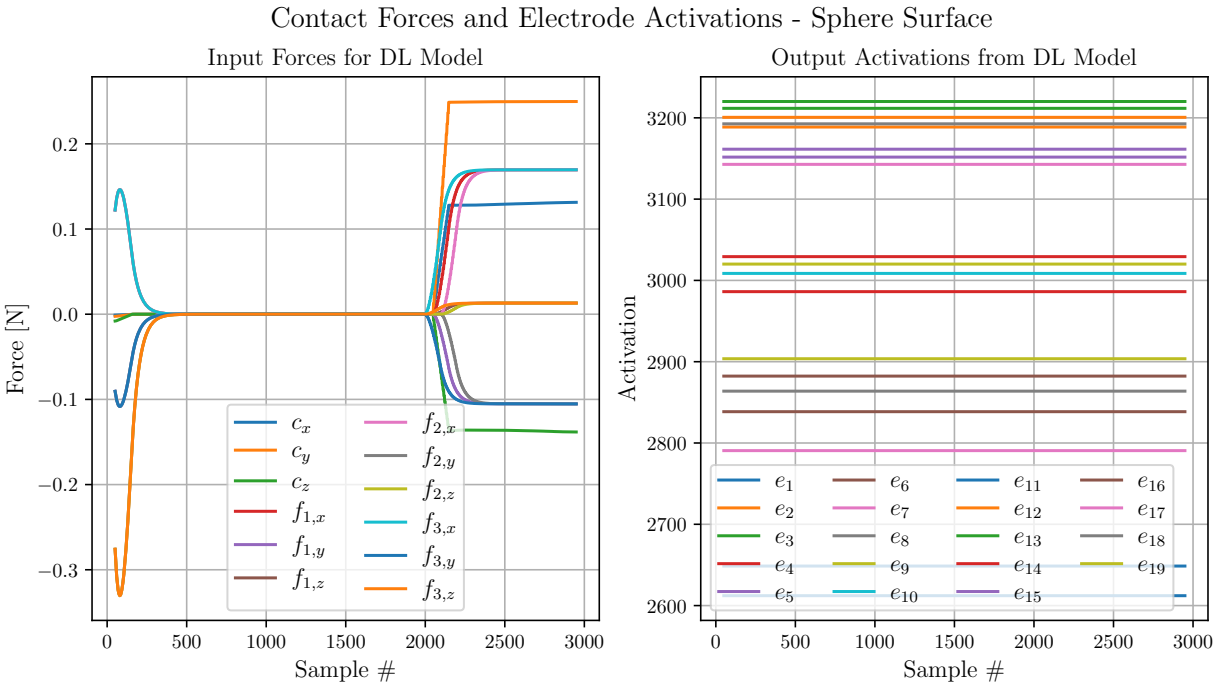


Fig. B.2: The simulated tactile electrode activations when the simulated Shadow Dexterous hand's index finger is in contact with a smooth surface.

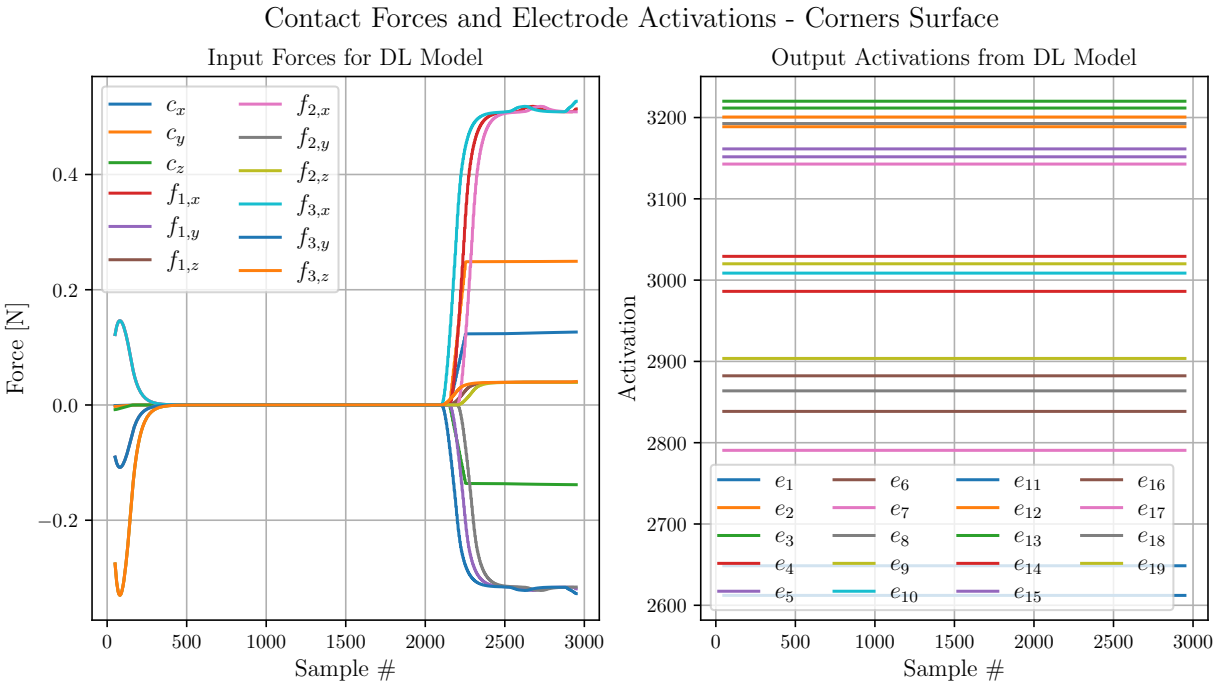


Fig. B.3: The simulated tactile electrode activations when the simulated Shadow Dexterous hand's index finger is in contact with a corner.