

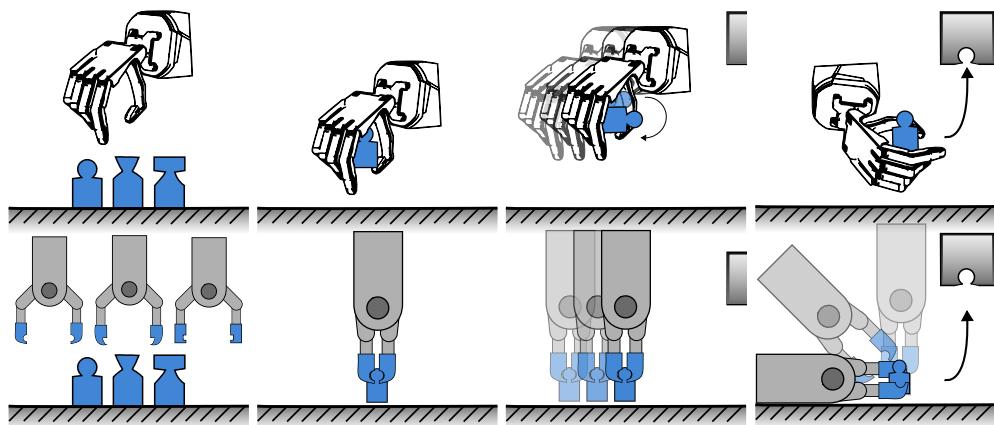


In-Hand Pose Estimation Through Purely Tactile Perception and In-Hand Manipulation

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 : 17157

June 1, 2023

Abstract

In this project, a novel pipeline for a pick-and-place (PNP) task is presented, focusing on utilizing tactile inputs instead of vision systems to overcome the limitations of vision algorithms, such as transparency and reflectance issues. The development of this pipeline involved addressing three key subproblems: Tactile Perception (TP), pose estimation (PE), and in-hand manipulation. TP involved accurately estimating the contact points, contact normals, and skew forces. The contact normals were successfully estimated using Recursive Least Squares (RLS), the contact points were found using the grasp matrix, while a deep learning (DL) model was attempted to be used for skew force estimation without success. To compensate for the missing skew forces and cases of contact normals assistance of the Gazebo physics engine was applied successfully. The PE task demonstrated promising results on synthetic data using Graduated Non-Convexity (GNC) for outlier rejection and Relaxed Convex Quadratic Programming (RCQP) for transformation estimation. This resulted in an orientation error of 3 degrees and a translation error of 0.08 cm while in the presence of 10 % outliers, which was within the criteria for success. For the in-hand manipulation problem, the use of Demo Augmented Policy Gradient (DAPG) proved successful in the MuJoCo dynamic simulator, and a comprehensive analysis of the training process and final performance was conducted. Lastly, potential enhancements for future iterations of the pipeline are discussed.

Contents

Acknowledgments	iii
Acronyms and Terms	iv
1 Introduction	1
1.1 Context	1
1.2 Problem Description	1
1.3 Thesis Overview	3
2 System Setup	5
2.1 Simulation Setup	5
2.2 Software Setup	7
2.2.1 Provided Software	7
2.2.2 Produced Software	9
3 State of the Art	11
3.1 Problem 1 - Tactile Perception	11
3.2 Problem 2 - Pose Estimation	13
3.3 Problem 3 - In-Hand Manipulation	15
4 Modeling	17
5 Tactile Perception	21
5.1 Methods	21
5.1.1 Recursive Least Squares	21
5.1.2 Network Architecture	23
5.1.3 Network Training Procedure	23
5.2 Experimental Setup	25
5.2.1 Contact Normal Estimation	25
5.2.2 Skew Force Estimation	26
5.3 Results	27
5.3.1 Contact Normals	27
5.3.2 Skew Forces	27
5.3.3 Physics Engine Comparison	30
5.4 Discussion & Conclusion	33
6 Pose Estimation	34
6.1 Problem Formulation	34
6.2 Method	36
6.2.1 Graduated Non-Convexity	36
6.2.2 Relaxed Convex Quadratic Programming	38
6.3 Experimental Setup	42
6.4 Results	43
6.4.1 Pose Estimation Performance Data	43
6.4.2 Weight Convergence Data	45

6.5	Discussion & Conclusion	47
7	In-Hand Manipulation	48
7.1	Problem Formulation	48
7.2	Method	49
7.2.1	Demo Augmented Policy Gradient	49
7.2.2	Data Collection Procedure	50
7.3	Experimental Setup	51
7.4	Results	52
7.5	Discussion & Conclusion	55
8	Discussion & Conclusion	56
A	Shadow Dexterous Hand - Technical Specifications	68
B	Tactile Perception - Simulated Electrode Activations	70
C	Pose Estimation - Weight Convergence	72

Acknowledgements

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

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

I also thank Yitaek Kim for his contribution to the optimization aspect of this project. His insights and advice have played a crucial role in its successful completion.

Furthermore, I extend my thanks to Shadow Robotics for providing the underlying software and technical support, which have enabled the completion of this project.

I am grateful to Rupam Singh for her guidance in reinforcement learning, which has greatly influenced the direction of my research and deepened my understanding of the subject matter.

Finally, I would like to acknowledge the support of my family and friends throughout this journey. Their encouragement has been a source of motivation.

Once again, I would like to thank all those who have played a significant role in this project.

Acronyms

AEBM analytical elasticity-based models.	NPG Natural Policy Gradient.
AH Adroit Hand.	OMPL The Open Motion Planning Library.
ANOVA Analysis of Variance.	PC point cloud.
BC Behavior Cloning.	PCR point cloud registration.
BR Black-Rangarajan.	PD Proportional-Derivative.
CAD Computer Aided Design.	PDE Partial Differential Equations.
cGAN conditional Generative Adversarial Network.	PE pose estimation.
cobots collaborative robots.	PID Proportional-Integral-Derivative.
CP correspondence problem.	PNP pick-and-place.
CPD Coherent Point Drift.	PnP Perspective-n-Point.
CSGM Cross-Source Graph Matching.	PRM Probabilistic Roadmap Method.
CV computer vision.	PSD Positive Semidefinite.
DAPG Demo Augmented Policy Gradient.	PwoF point-contact-without-friction.
DeepGMR Deep Gaussian Mixture Registration.	QAP Quadratic Assignment Problem.
DL deep learning.	QCQP Quadratically Constrained Quadratic Programming.
DOF degrees of freedom.	QP Quadratic Programming.
EAA Equivalent Angle Axis.	RANSAC Random Sample Consensus.
EE end effector.	RCQP Relaxed Convex Quadratic Programming.
EFM elastic foundation models.	ReLU Rectified Linear Unit.
FEM finite element models.	RL Reinforcement Learning.
FGM Factorized Graph Matching.	RLS Recursive Least Squares.
FIM Fisher Information Matrix.	ROM Range of Motion.
FMT Fast Marching Tree.	ROS Robot Operating System.
FPFH Fast Point Feature Histograms.	RPY Roll-Pitch-Yaw.
GAIL Generative Adversarial Imitation Learning.	RRT Rapidly-exploring Random Tree.
GK grasp kinematics.	SDH Shadow Dexterous Hand.
GM Geman McClure.	SDP Semi-Definite Programming.
GMM Gaussian Mixture Model.	SE Special Euclidean.
GNC Graduated Non-Convexity.	SF soft finger.
GT Ground Truth.	SLSQP Sequential Least-Squares Quadratic Programming.
HF hard finger.	SOCP Second Order Cone Programming.
ICP Iterative Closest Point.	SOIL State-Only Imitation Learning.
IEP Inverse Elasticity Problem.	SOTA state of the art.
IHM In-Hand Manipulation.	TCP Tool Center Point.
IL Immitation Learning.	TLS Truncated Least Squares.
JRMP Joint Registration of Multiple Point Sets.	TP Tactile Perception.
MDP Markov Decision Process.	TSV Task Space Vector.
MIM Matrix Inversion Method.	UR Universal Robots.
ML machine learning.	
MLP Multi Layered Perceptron.	
MPC Model Predictive Control.	

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].

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 [5].

manipulator : A serial robot mechanisms. The robot manipulator is represented by a serial chain of rigid bodies, called robot segments, connected by joints [6].

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 [5].

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 [7].

Robot Operating System (ROS) is a set of open-source software libraries and tools that help build robot applications. [8].

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 [9, 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 [9, 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 [9, 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 [10] or left out due to illness etc. Considerations regarding 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) [11] which could mitigate 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

PNP manipulators are used in a wide variety of different fields such as sorting of waste [12] handling of food [13, 14] and factory bin picking [15, 16, 17]. 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 [16] for traditional implementations of the PNP pipeline. The pre-grasp phase involves localizing the object(s) and potentially estimating their pose and executing a collision-free trajectory to move the end effector (EE) grasp to the object. Here different potential grasps can be considered to determine the best pose for the EE. In the grasping phase, the EE grasps 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 [18]. These include classic vision [19, 20], deep learning (DL) based [21] and combinations of these [22]. However, while they may be sufficient for certain tasks they fundamentally suffer from the weaknesses introduced by vision techniques. These often come in the form of 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 metal, plastic and glass often being the materials used. While DL solutions have been developed for both reflective [23] and transparent [24] 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 an anthropomorphic gripper as the EE with tactile sensors in each finger, more specifically a Shadow Dexterous Hand (SDH) [25] with 24 joints and 20 degrees of freedom (DOF).

The alternative pipeline this project will enable can be seen in the top row of Fig. 1.1 compared to the traditional pipeline in the bottom 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 or dexterity to perform reliable force closure. On the contrary force closure can be performed with an anthropomorphic gripper on a wide range of objects with no need for changing gripper equipment. It is here assumed that the system has a rough estimate of the object's position in the scene for the motion toward the object to be consistent.

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 anthropomorphic 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 an anthropomorphic gripper can PE the object and manipulations can be performed to change the object's pose such that consistent placing can be done 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.

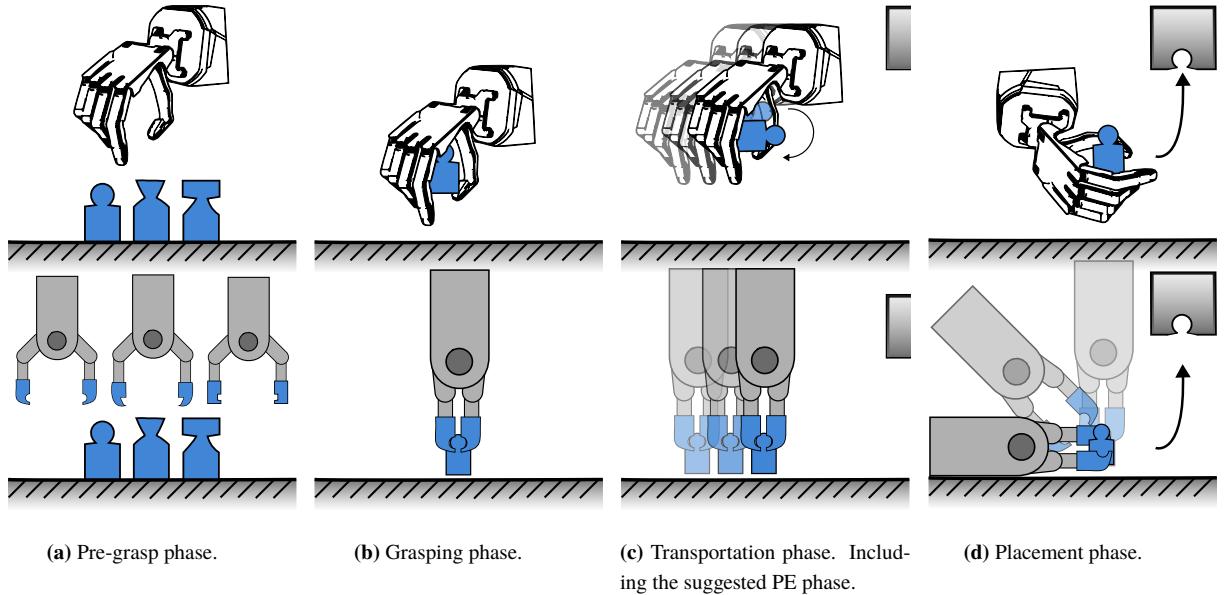


Fig. 1.1: A comparison between the traditional and suggested PNP pipeline.

To build this pipeline, 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 (TP), such that useful data can be extracted for pose estimation.

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 contact with the object through manipulation.

1.3 Thesis Overview

This project is composed of eight 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 and provided to execute the project.

Chapter 3 The literature review i.e. state of the art (SOTA), 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 TP chapter addresses the method chosen in Chapter 3 to solve problem 1. This chapter expands on the method chosen and its functionality, followed by an evaluation of said method through experiments and their results. Finally, the extent to which the method is successful in addressing problem 1 is concluded.

Chapter 6 The PE chapter addresses the method chosen in Chapter 3 to solve problem 2. 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 problem 3. This chapter follows the same structure as Chapter 5 and Chapter 6.

Chapter 8 The discussion and conclusion chapter goes over the system's components and addresses problems, improvements, shortcomings and potential for future development, followed by a conclusion that addresses the success of the project.

Chapter 2

System Setup

The SDH [25] 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 simulators, Gazebo [26] and MuJoCo [27].

This simulation is based on the Shadow Robot Company's [28] Robot Operating System (ROS) [8] packages [29], which provide a flexible and customizable framework for controlling the hand. The first simulator Gazebo, a popular robot simulation environment, simulates 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 [30] container, which allows for easily distributing the code and dependencies to other researchers. The second simulator MuJoCo, like Gazebo, is a dynamic simulator, but with a greater emphasis on accurate physics simulations, making it ideal for research involving phenomena such as friction. MuJoCo has seen applications in machine learning (ML) applications such as Reinforcement Learning (RL)-based dexterous manipulation [31, 32, 33]. The MuJoCo physics engine comes in a conda virtual environment, which like Docker, enables reproducibility and portability.

Simulating the system provides the added benefit of readily accessible Ground Truth (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 software architecture, simulated hardware and software used in this project is presented. This description separates what software is provided as well as produced, including ROS packages.

2.1 Simulation Setup

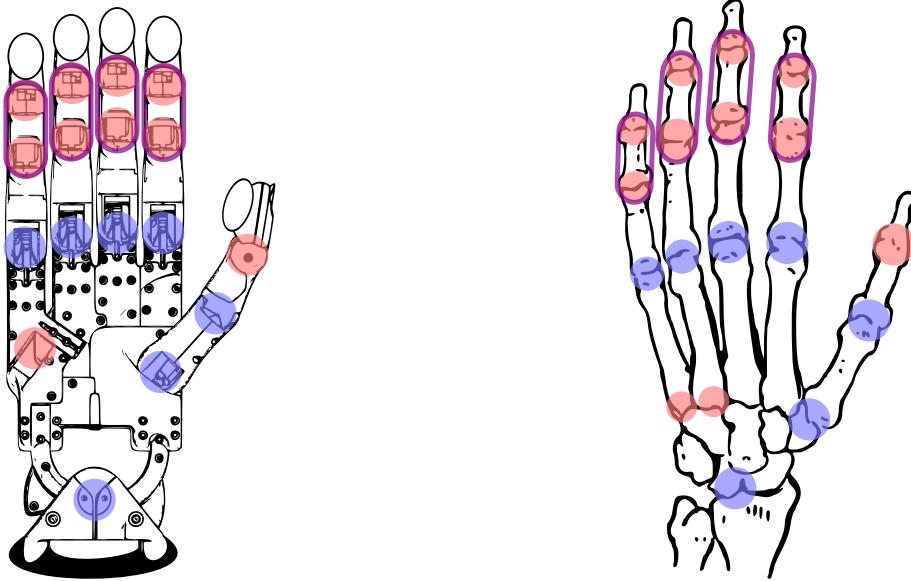
The Computer Aided Design (CAD) model of the SDH is a highly detailed and accurate representation of the physical hand. The model is based on the original design of the real-world hand, which was developed by the Shadow Robot Company. The CAD model includes precise geometry for all of the hand's components, 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 the 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 right 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 one DOF, blue refers to joints with a DOF equal to two and purple refers to two joints that are coupled. The coupling between joints is such that, a flexion of joint two imposes a constraint on joint one. If joint two exceeds a certain angle, it

enforces joint one to flex accordingly to maintain the established constraint. In the robot hand, both coupled joints are mechanically linked to a single motor.

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 DOF. The total number of controllable joints can here be seen as 24. This figure is based on [34].

(b) Human hand with joints color coded depending on the DOF. The total number of controllable joints can here be seen as 25 [35]. This figure is based on [36].

Fig. 2.1: The SDH and a human hand, red here marks a joint with one DOF, blue marks a joint with two, and purple marks coupled joints.

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 project, are purely aesthetical as representative simulated tactile sensors are yet to be supported as a standard component of the Shadow Robotics development environment. 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 [37]. 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.

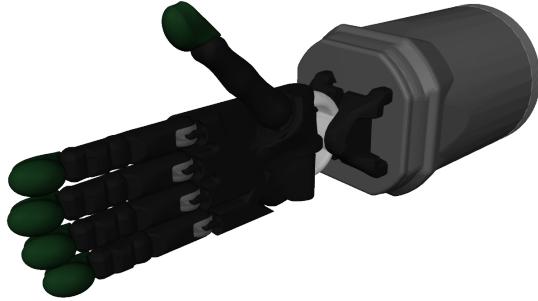


Fig. 2.2: A cutout of the simulated SDH.

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 Gazebo simulation environment is shipped in a custom Ubuntu-based docker container [30, 38] with the necessary libraries to communicate and develop applications on the simulated as well as the physical hand, wrapped within a catkin workspace [39]. Additionally, the container comes with common-use libraries for Python and C++ development in ROS including `numpy`[40], `OpenCV` [41] and `dynamic_reconfiguration` [42]. A simplified overview of 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.

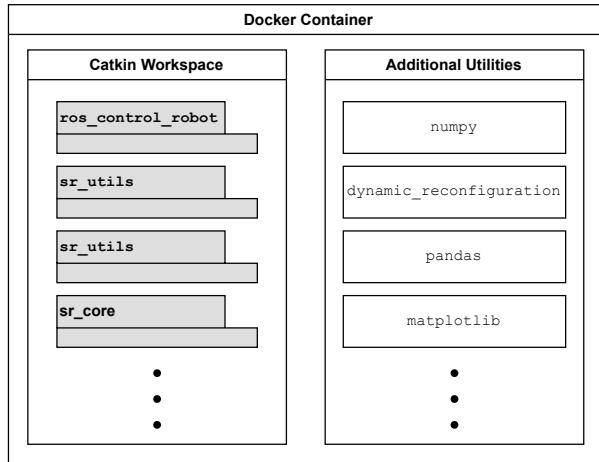


Fig. 2.3: The boxes marked with grey are ROS packages, while the white are modules.

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 $\mathbf{q}_i = [q_0, q_1, \dots, q_n]^T \in \mathbb{R}^{24}$

where $i \in \{0, 1, \dots, m_w\}$ with m_w being the number of waypoints. \mathbf{Q} can thus be written as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1^\top \\ \mathbf{q}_2^\top \\ \vdots \\ \mathbf{q}_{m_w}^\top \end{bmatrix} \in \mathbb{R}^{m_w \times 24}. \quad (2.1)$$

SrHandCommander parses the path to the `move_group` handled by MoveIt [43]. After receiving the plan, MoveIt first checks the validity of the start, goal and intermediate configurations 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 path $\mathbf{Q}_{\text{valid}} \in \mathbb{R}^{m_w \times 24}$ is parsed to The Open Motion Planning Library (OMPL) [44] where a safe high-resolution path is built using some chosen sample-based single- or multi-query path planning method to build the full high-resolution path $\mathbf{Q}_{\text{full}} \in \mathbb{R}^{m \times 24}$. 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` [45] 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 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 [46, 47, 48].

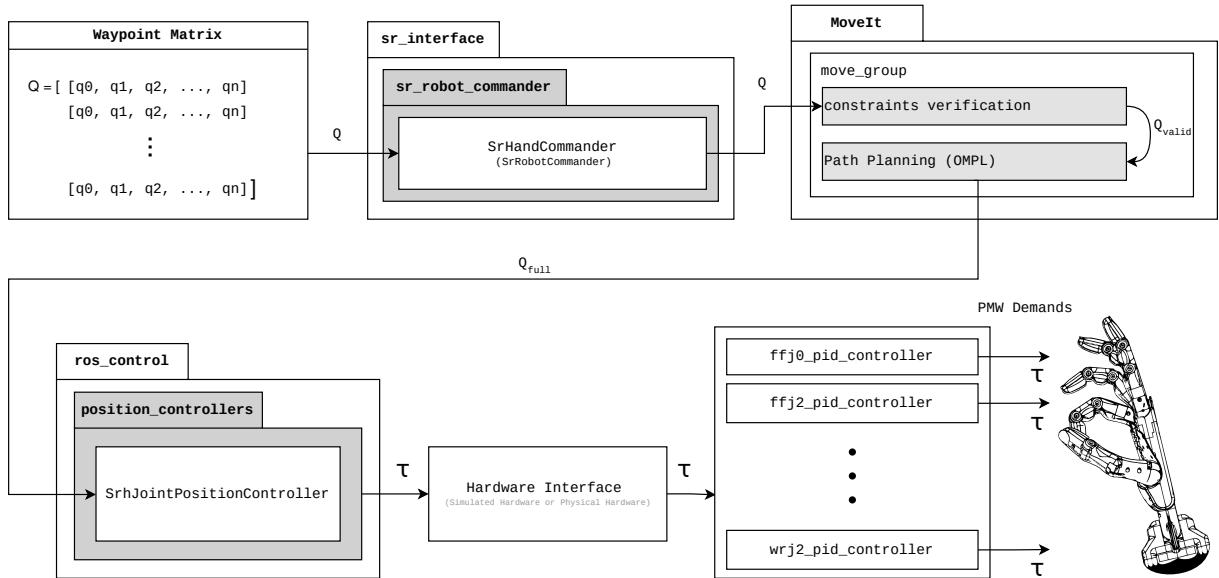


Fig. 2.4: Diagram showing the communication and control flow of interacting with the SDH.

When executing an example path, Fig. 2.5 shows joint angles during execution on the simulated hand throughout 15 s using the RRT-Connect planner. 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.

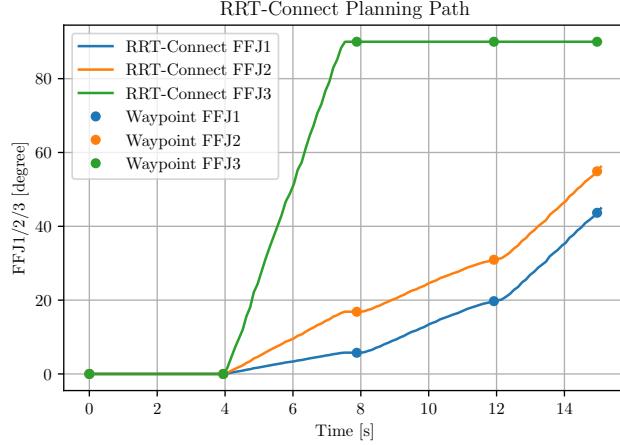


Fig. 2.5: Showing the joint trajectory of the first finger’s joints FFJ2 and FFJ3, where the coupling between FFJ1 and FFJ2 is shown.

The MuJoCo [27] simulation runs a Adroit Hand (AH) [49], which is a SDH equipped with 30 controllable parameters, 24 for the hand’s joints and 6 for the hand’s position $\mathbf{p}_{hnd} = [p_x, p_y, p_z]$ and orientation $\mathbf{u} = [\phi, \theta, \psi]$ in Roll-Pitch-Yaw (RPY) format. The system stack for MuJoCo is similar to that of Gazebo and is therefore not repeated.

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 into two ROS packages: `shadow_hand` and `sr_tactile_perception`, with an additional utility package `ros_utils`.

The `shadow_hand` package is a software wrapper structured semantically in a similar way as the hand itself, as a `ShadowHand` object contains five `ShadowFinger`, each labeled as one of TH, FF, MF, RF or LF and a `ShadowWrist`. Each finger is equipped with a reader that samples the tactile data provided by the simulator’s physics engine at 100 Hz. This wrapper is written as 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(a).

The `sr_tactile_perception` package is an interface for logging, processing and visualizing the tactile data using the `shadow_hand`. Additionally, this package is responsible for managing and executing experiments. This can be seen as the top diagram in Fig. 2.6(b).

Both of these packages are contained within a `in_hand_pose_estimation` meta package which additionally contain custom tools for easy launching and execution of experiments with live plotting of tactile data. Additionally, each of these tools provides a high degree of easy-to-access flexibility in terms of model and configuration loading.

The last package is `ros_utils` which contains important utilities used in the project such as type transformations, logging, data conversion, live plotting, encoding, decoding etc. This package contains both Python and C++ tools and can be found in [50].

The software developed in this project can be found in [51].

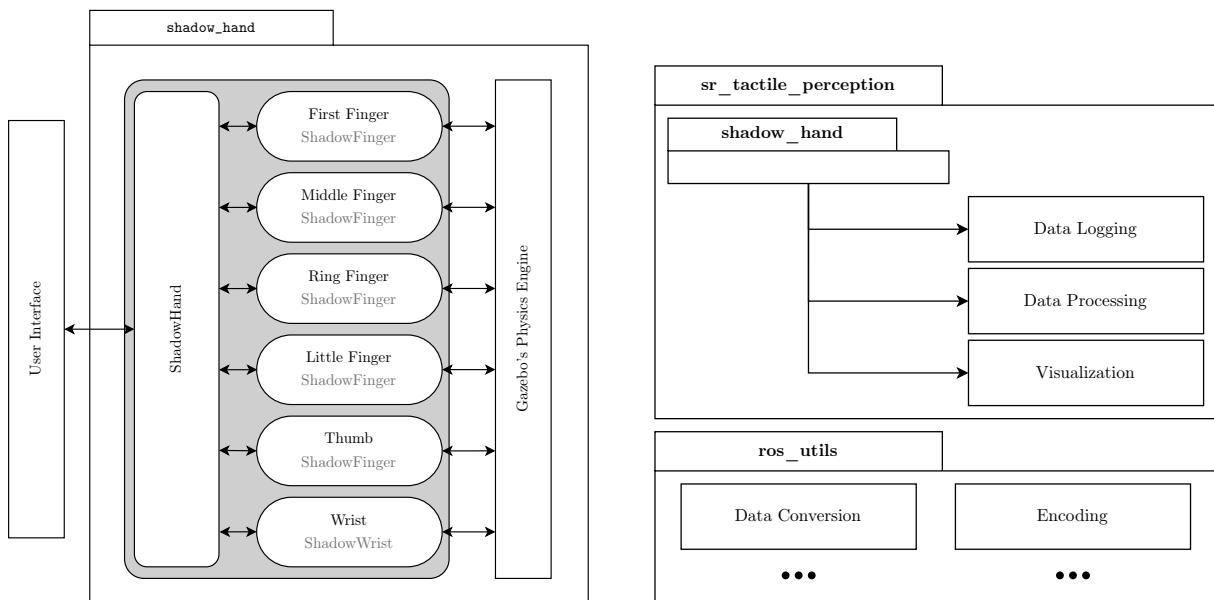


Fig. 2.6: Developed ROS packages for executing this project.

Chapter 3

State of the Art

3.1 Problem 1 - Tactile Perception

Of the contact models found during the literature review for solving problem 1 four different categories were identified: analytical elasticity-based models (AEBM), elastic foundation models (EFM), finite element models (FEM) [52] and ML models. The different categories can be seen organized in Fig. 3.1

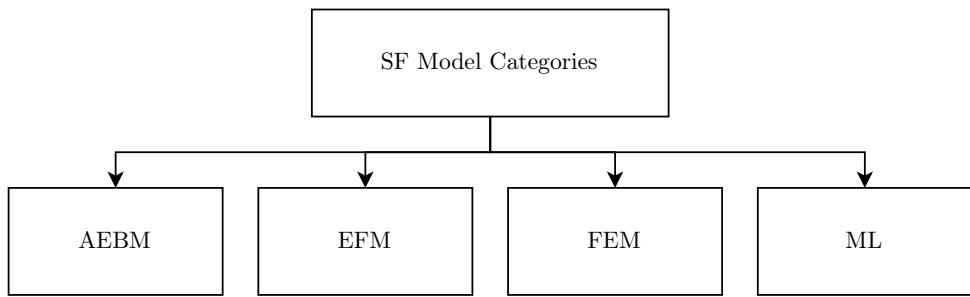


Fig. 3.1: Tree of methods for TP.

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 [53] 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 basic Hertzian contact model does not represent the type of contact in this project [54, 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 [55, 56]. This power-law formulation subsumes the Hertzian contact theory while assuming a circular contact area. Other models have been proposed that combine the descriptions of both friction-contact and the shear-torsion as experienced by the bodies [57].

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 [58] and Maxwell's [59] 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 [60]. To simplify this formulation alternatives have been developed to assume no past stress [61, 62, 63]. 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 [64], 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 [65]. Attempts involving solutions to Boussinesq's problem for polynomial pressures acting over polygonal domains [66] have also been developed and modernized by combining it with Cerruti's solution [67]. 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 [68]. 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 [69, 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 [70], which has been used in structural engineering for modeling different properties of beams such as stability [71], vibrations and buckling [72]. 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 [52]. A different method attempts to attain vivo contact pressure predictions for improved knee replacement designs [73] Within the field of robotics EFM have provided solutions to problems such as slip [57], compliance, sliding [74, 75], stiffness and contact mechanics [76] 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 [77].

FEM are popular general tools for solving Partial Differential Equations (PDE) [78] 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 [79] whereas one example is the metal forming processes. Specifically, the estimation of wheel-rail profiles [80] 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 [81], analysis of the responses of fully coupled thermo-elasto-plastic solids in contact [82] and performing diagnostics of failures in induction motors [83]. 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 [84], performance estimation of new tactile sensor technologies [85] and evaluating complex contact types by extending simulations and analysis systems [86]. The modeling complexity has furthermore inspired using FEM as ground truth results when synthesizing ML data in simulations for DL models, which has enabled execution speeds 75 times greater than simply evaluating FEM [87, 88, 89].

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 [90, 91]. For instance, simulating realistic tactile images from simulated contact depth to bridging the reality gap for vision-based tactile sensing using a diffusion model [92]. Similarly, a conditional Generative Adversarial Network (cGAN) has been used to simulate realistic tactile sensory data for use in tactile tasks [92]. Solutions using DL models purely based on Multi Layered Perceptron (MLP) have been applied to enable real-time simulated realistic tactile data [93].

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 [93] is chosen to solve the TP 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 (PCR).

These can along with their subcategories be seen in Fig. 3.2 as inspired by [94].

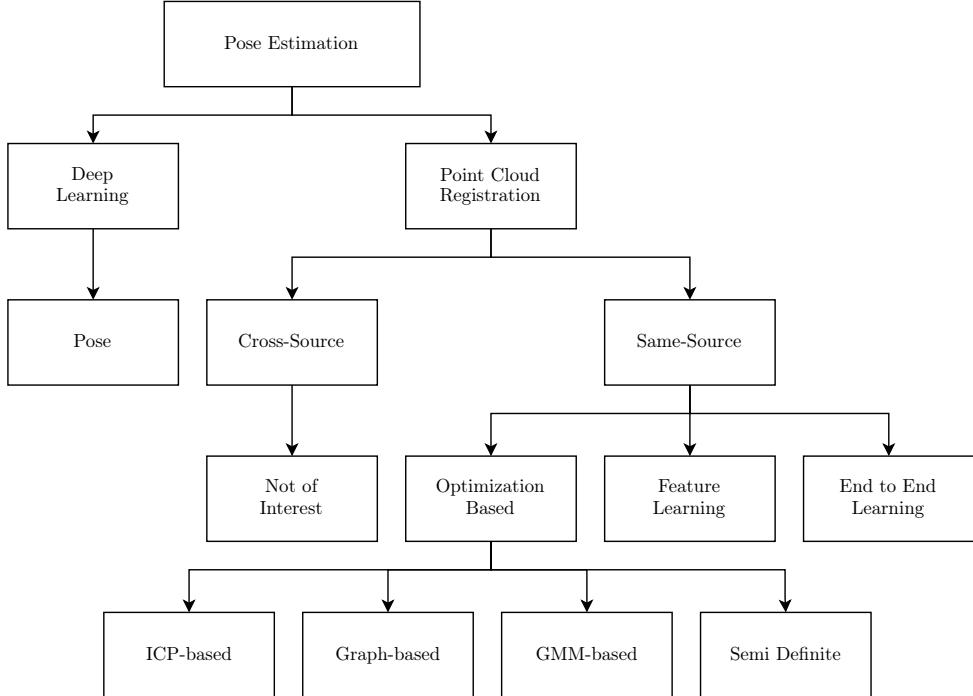


Fig. 3.2: Tree of PE methods. The categorization is inspired by [94].

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 [95, 96, 97]. 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. 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 [98, 99, 94] 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 [100], learning registration patterns as parametric

functions through a scan completion module and pairwise matching module [101], and a fast feature-metric point cloud registration framework to minimize the feature-metric projection error without correspondences [102].

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 [103], to learn globally informed 3D local feature descriptors [104], and to use siamese deep learning architectures with convolutional layers through a voxelized smoothed density value (SDV) representation [105].

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 (SDP)-based methods.

Since the original proposal in 1992 [106] using point-to-point correspondences, ICP-based methods have evolved and incorporated different types of correspondences to improve performance. Examples include point-to-plane [107] and plane-to-plane [108]. Modern approaches also employ complementary methods such as point cloud filtering, adaptive fireworks algorithms, and KD-Trees [109].

The main idea of graph-based registration methods is to use a non-parametric model [110]. In this method, correspondences between two graphs are found by considering both the vertices and edges, making it an optimization problem [110]. 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 [111]. Second-order methods include Cross-Source Graph Matching (CSGM) [98], 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) [112] factorizes the large pairwise affinity matrix into smaller matrices and solves the graph-matching problem with a simple path-following optimization algorithm. Spectral graph [113] uses a spectral relaxation method to approximate the Quadratic Assignment Problem (QAP), and SDP relaxation is used to relax the non-convex constraint using a convex semi-definite. While higher-order graph matching provide methods for [114] design a probabilistic approach to solve the high-order graph-matching problem, while [115] design a triangle similarity and convert the graph-matching problem into a tensor optimization problem. More recent work, such as [116] 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 [117] in the form of Coherent Point Drift (CPD) which imposes constraints on transformation estimation. In another approach,[118] combines GMM with the convex hull to reduce computation complexity. Furthermore, Joint Registration of Multiple Point Sets (JRMPC) [119] cast registration as a clustering problem where the transformation is optimized by solving the GMM. Recently, Deep Gaussian Mixture Registration (DeepGMR) [120] 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 SDP different optimization groups exist, such as Second Order Cone Programming (SOCP), Quadratic Programming (QP) and Quadratically 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 [121]. 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 [122]. 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 [123].

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 [123] with GNC outlier rejection [124].

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. Planning dexterous manipulation paths can be a difficult endeavor due to the high dimension and computational complexity. During this literature review, the three main solution categories found were: kinodynamics, Model Predictive Control (MPC) and RL-based solutions, as seen in Fig. 3.3.

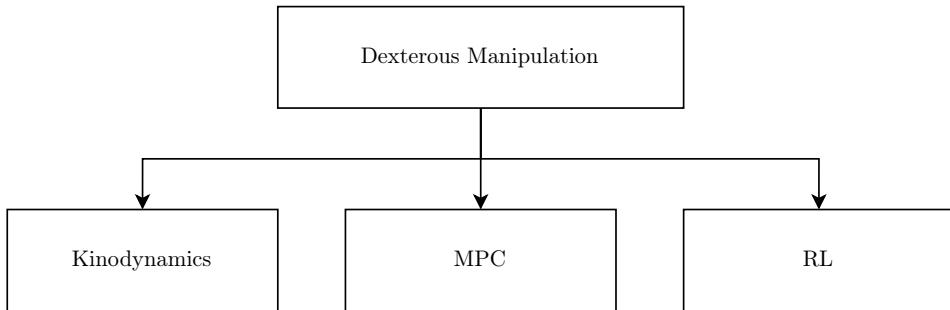


Fig. 3.3: Tree of methods for solving the in-hand manipulation problem.

For kinodynamic motion planning to address the dexterous manipulation challenges, researchers have proposed probabilistic complete sampling-based kinodynamic motion planners like SST and SST* [125]. These planners utilize sampling-based techniques and probabilistic methods to generate collision-free paths in complex environments. Due to kinodynamics both searching in configuration space and its derivative, the problem becomes PSPACE-hard [126], indicating its great computational complexity.

Despite their effectiveness, the complexity of the motion planning problem has led to the integration of motion planning algorithms with MPC. By combining these two techniques, it becomes possible to avoid solving a boundary value problem [127], reducing the computational burden associated with kinodynamic motion planning.

Furthermore, to enhance performance, researchers have explored the combination of kinodynamic motion planning algorithms with RL. This integration leverages the capabilities of RL algorithms to learn from experience and

improve the efficiency and quality of the generated motion plans [128]. This however can become expensive due to the great search space, which has pushed the research into the applications of Imitation Learning (IL) based RL. One such example [33] which presents a novel framework for integrating expert demonstrations of dexterous manipulation into a wide range of RL such as State-Only Imitation Learning (SOIL), Generative Adversarial Imitation Learning (GAIL) and Demo Augmented Policy Gradient (DAPG).

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 MPC and RL, these advancements aim to enhance the efficiency, reliability, and adaptability of motion planning algorithms to support complex robot applications.

Due to the high flexibility and the results showed in [33], which demonstrate applicable performance on a similar platform as the one chosen in this project, this method is chosen. Specifically the DAPG RL method, due to it showing the best overall results even in unseen cases.

Chapter 4

Modeling

To model the contact between the EE's tactile sensors, eight different categories exist as identified in [129]. The three most common ones within the field of robotics [54, Chapter 37] are point-contact-without-friction (PwoF), hard finger (HF) and the soft finger (SF) model as shown in Fig. 4.1.

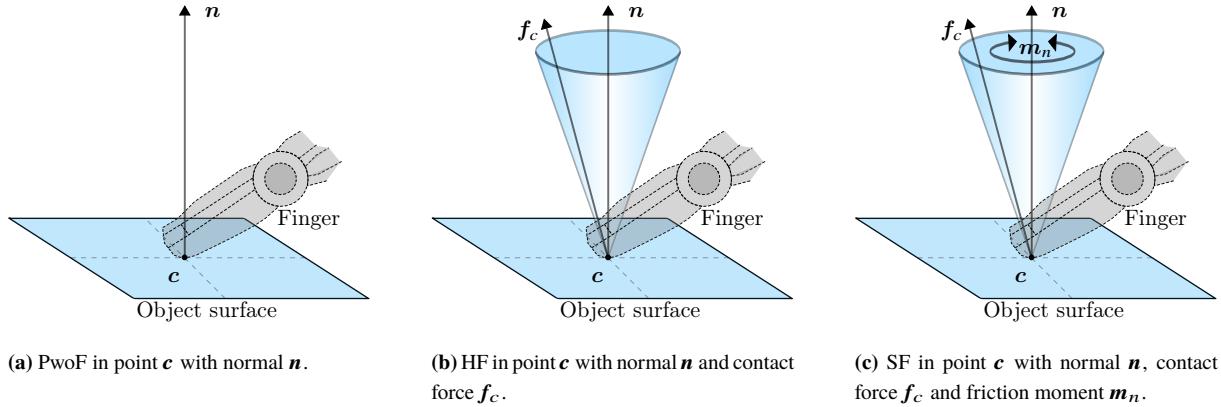


Fig. 4.1: The three most commonly used contact models.

The PwoF model, as shown in Fig. 4.1(a), can only represent forces along the surface normal $\mathbf{n} \in \mathbb{R}^3$ at the point of contact $\mathbf{c} \in \mathbb{R}^3$ 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 [54, 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 [54, 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 [130]. This model states that the frictional force acting on an object can be formulated as

$$f_f = f_N \mu, \quad (4.1)$$

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

$$C_{f,\text{HF}} = \{ f_c \mid f_t \leq \mu f_z, \mu f_z \geq 0 \} \quad , \quad f_t = \sqrt{f_x^2 + f_y^2}. \quad (4.2)$$

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 ($f_c \in \mathbb{R}^3$) and μ is the friction coefficient [54, 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 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 w_{ext} on the body can be balanced by contact forces [131]. 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 [54, 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

$$\mathcal{E}_{f,SF} = \left\{ f_c \mid f_t^2 + \frac{m_n^2}{e_n^2} \leq \mu^2 P^2 \right\} , \quad f_t = \sqrt{f_x^2 + f_y^2}. \quad (4.3)$$

This formulation forms a contact ellipsoid $\mathcal{E}_{f,SF}$ which describes the relationship between the tangential force $f_t \in \mathbb{R}^3$ and friction moment $m_n \in \mathbb{R}^3$. 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 \mathbf{n} [75, 77].

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 [132] 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.

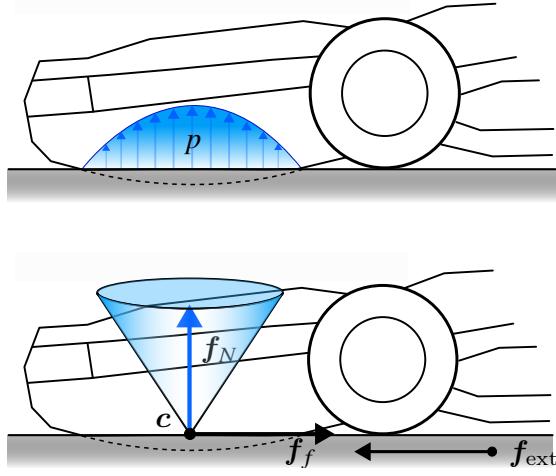


Fig. 4.2: The pressure distribution p and friction cone of a SF model experiencing an external force f_{ext} .

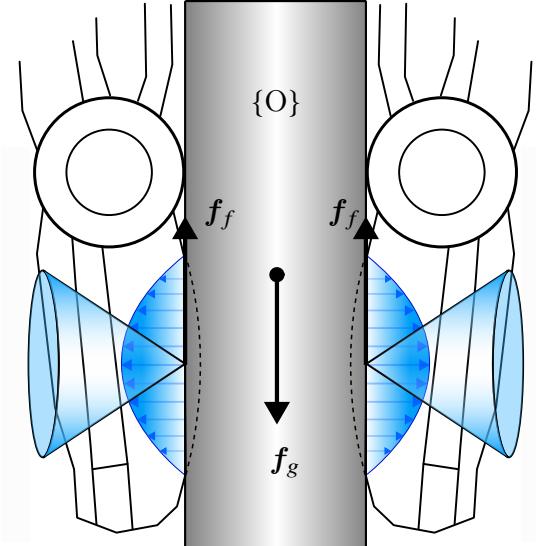


Fig. 4.3: the pressure distribution and friction cone causing force closure to prevent the object $\{O\}$ from falling due to the gravitational force f_g .

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

joints, of which it has n_q , named $\mathbf{q} = [q_1 \ q_2 \ \dots \ q_{n_q}]^\top \in \mathbb{R}^{n_q}$ each of which is revolute and can exert a torque $\boldsymbol{\tau} = [\tau_1 \ \tau_2 \ \dots \ \tau_{n_q}]^\top \in \mathbb{R}^{n_q}$. 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 EE's DOFs.

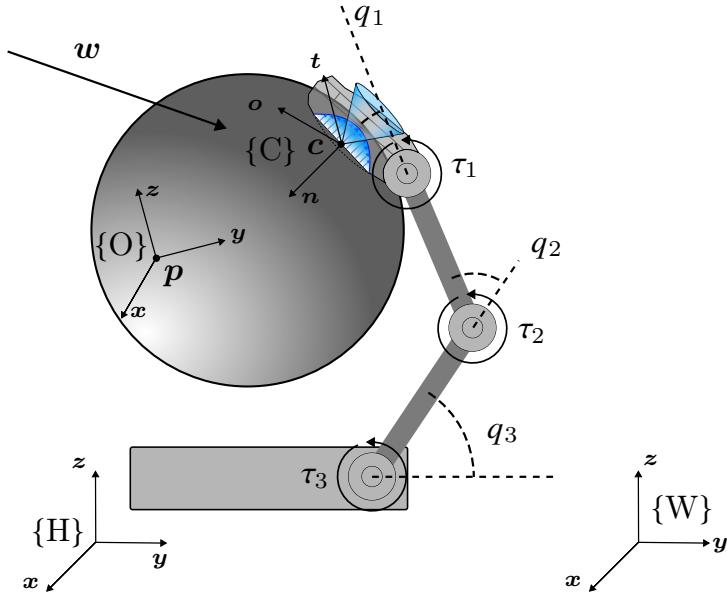


Fig. 4.4: The model of the system representation for this project.

In this system, the twists and wrenches of a contact point c_i on the object and hand, given in contact frame $\{C\}_i$ is referred to as $\boldsymbol{\nu}_{i,\xi} \in \mathbb{R}^6$ and $\boldsymbol{w}_{i,\xi} \in \mathbb{R}^6$, with $\xi = \{\text{obj}, \text{hnd}\}$. 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

$$\boldsymbol{\nu}_{c,\xi} = \left[\boldsymbol{\nu}_{1,\xi}^\top \ \boldsymbol{\nu}_{2,\xi}^\top \ \dots \ \boldsymbol{\nu}_{n_c,\xi}^\top \right]^\top \in \mathbb{R}^{6 \cdot n_c} \quad (4.4)$$

and

$$\boldsymbol{w}_{c,\xi} = \left[\boldsymbol{w}_{1,\xi}^\top \ \boldsymbol{w}_{2,\xi}^\top \ \dots \ \boldsymbol{w}_{n_c,\xi}^\top \right]^\top \in \mathbb{R}^{6 \cdot n_c} \quad (4.5)$$

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 $\{W\}$ to the twists or wrenches of the object in contact frames $\{C\}_i$. The grasp matrix thus can be expressed as

$$\mathbf{G} = [\mathbf{G}_1 \ \mathbf{G}_2 \ \mathbf{G}_3 \ \dots \ \mathbf{G}_{n_c}], \quad (4.6)$$

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

$$\boldsymbol{\nu}_{c,\text{obj}} = \mathbf{G}^\top \boldsymbol{\nu} \quad \text{and} \quad \boldsymbol{w}_{c,\text{obj}} = \mathbf{G}^\top \boldsymbol{w}. \quad (4.7)$$

While the grasp matrix describes the transformation from $\{W\}$ 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

$$\mathbf{J} = [\mathbf{J}_1^\top \mathbf{J}_2^\top \mathbf{J}_3^\top \cdots \mathbf{J}_{n_c}^\top]^\top, \quad (4.8)$$

for all contact points. Here $\mathbf{J}_i \in \mathbb{R}^{6 \times n_q}$ for $i = 1, 2, \dots, n_c$ are the individual contact points' 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

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

The modeling described above will enable the use of methods for solving the presented problems. These methods will be described in Chapter 5.

Chapter 5

Tactile Perception

To solve problem 2 and 3, the TP solution to problem 1 must provide estimates of contact positions, contact normals and skew forces, which is the goal of this chapter.

To estimate these, different techniques are considered and their performance analyzed, 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 matrix \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.

The RLS methodology is presented for normal estimation $\mathbf{n}_{c,i} = [n_{i,x}, n_{i,y}, n_{i,z}]^\top \in \mathbb{R}^3$, where $i \in \{0, 1, \dots, n_c\}$ and n_c is the number of contact points, followed by the experimental setup and results being presented.

The technique behind estimating the skew forces $\mathbf{f}_{c,i} = [f_{i,x}, f_{i,y}, f_{i,z}]^\top \in \mathbb{R}^3$, 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.

The contact positions $\mathbf{c}_i = [c_{i,x}, c_{i,y}, c_{i,z}]^\top \in \mathbb{R}^3$, are found using the grasping matrix \mathbf{G} provided by [93].

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. Due to a solution to problem 1 being a requirement for tackling the remaining problems, quantities from the physics engine are applied as enhancements or substitutes for inadequate results from the methods chosen.

The software used in this chapter is a regression neural network implemented as a Gazebo ModelPlugin [133] in C++. However, the DL model plugin used in the original publication [93] 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 [134]. 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 [135], while the fixed and updated version is available at [136].

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, the linear velocity $\hat{\mathbf{v}} \in \mathbb{R}^3$ must be estimated. 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 $\bar{\mathbf{c}}_t \in \mathbb{R}^3$ at time t is used as a representative value for each time step. Therefore, the estimated linear velocity can be computed as

$$\hat{\mathbf{v}} = \frac{\bar{\mathbf{c}}_t - \bar{\mathbf{c}}_{t-1}}{\Delta t}, \quad (5.1)$$

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 is then computed as

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

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

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

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

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

$$\hat{\mathbf{n}}_0 = \frac{\hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_0}{\|\hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_0\|_2}, \quad (5.4)$$

where $\hat{\mathbf{n}}_0 \in \mathbb{R}^3$ is the initial estimate, $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{v}}_1$ 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 $\mathbf{L}_n^1 \in \mathbb{R}^{3 \times 3}$ and $\mathbf{L}_n^2 \in \mathbb{R}^{3 \times 3}$ of the desired estimate $\hat{\mathbf{n}}_{new} \in \mathbb{R}^3$ can be found. Firstly, \mathbf{L}_n^1 is computed as

$$\mathbf{L}_n^1 = \mathbf{L}_n^1 - \beta \Delta t \mathbf{L}_n^1 + \frac{\Delta t K_L}{1 + \|\mathbf{v}_{tip}\|_2^2} \mathbf{v}_{tip} \mathbf{v}_{tip}^T, \quad (5.5)$$

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

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

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

$$\boldsymbol{\nabla} = \mathbf{f}_c \times \mathbf{v}_{tip}. \quad (5.7)$$

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

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

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

$$\boldsymbol{\omega} = \mathbf{n} \times \dot{\mathbf{n}}. \quad (5.9)$$

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

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

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

5.1.2 Network Architecture

In [93] 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 $\mathbf{c} = [c_x, c_y, c_z]$ along with three skew force vector samples $f_1 = [f_{1,x}, f_{1,y}, f_{1,z}]$, $f_2 = [f_{2,x}, f_{2,y}, f_{2,z}]$ and $f_3 = [f_{3,x}, f_{3,y}, f_{3,z}]$, and a temperature input $T \in \mathbb{R}$, which makes the input $[\mathbf{c}, f_1, f_2, f_3, T] \in \mathbb{R}^{13}$. 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 $[pdc, pac, tdc, tac, e_1, \dots, e_{19}] \in \mathbb{R}^{23}$, 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 [137], 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. This calibration information was not provided by [93].

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 of 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 \mathbf{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 \mathbf{D} can be represented as follows

$$\mathbf{D} = \begin{bmatrix} 0 & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \\ 1 & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \\ \vdots & & & & & & & & \vdots & & & & & \\ N_{dp} & pcd & pac & tdc & tac & e_1 & \cdots & e_{19} & f_{1,x} & \cdots & f_{3,z} & c_x & c_y & c_z \end{bmatrix} \in \mathbb{R}^{N_{dp} \times 35}. \quad (5.11)$$

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. 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

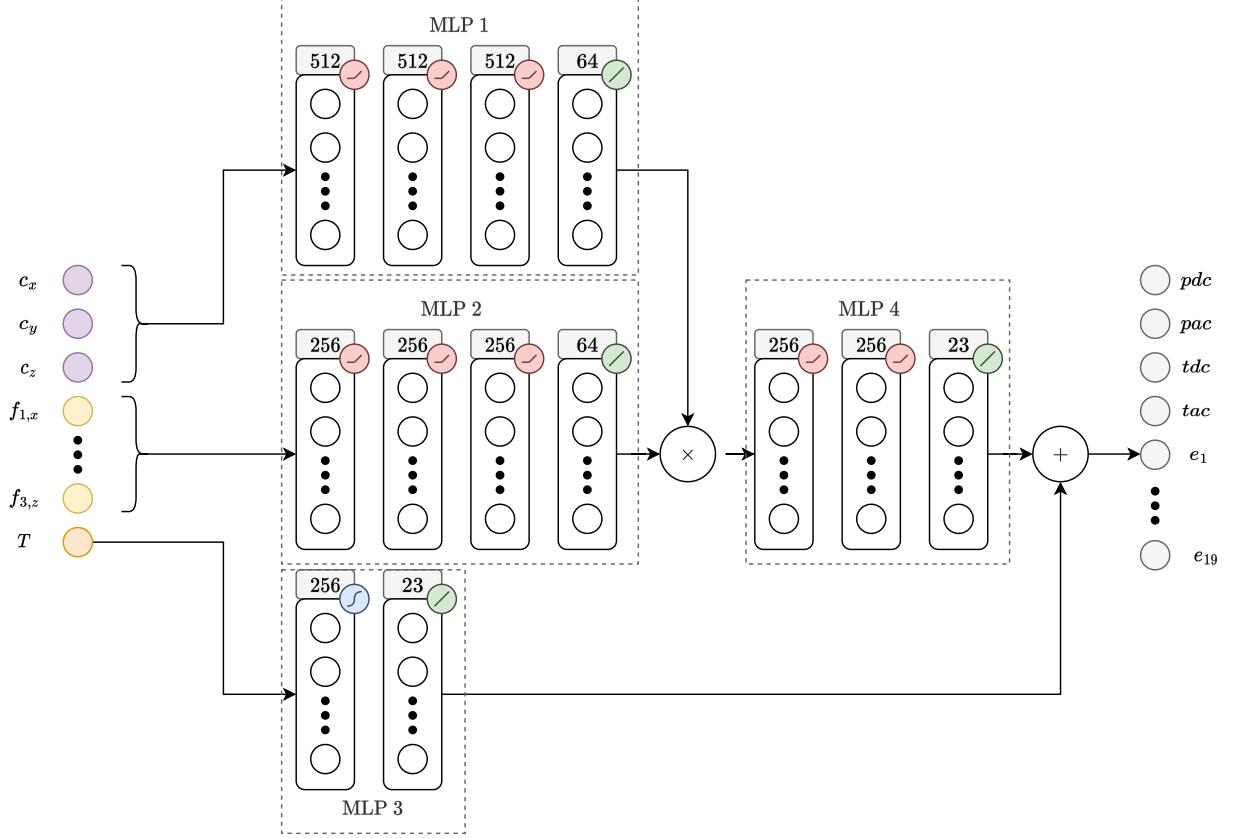


Fig. 5.1: DL model B from [93], which also has provided inspiration for this illustration.

public in neither [138] nor [93]. The network generates simulated electrode and pressure signals as outputs but currently does not simulate temperature outputs.

The forces were collected using a calibrated six-axis force-torque sensor [139] 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 [140], 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 [138].

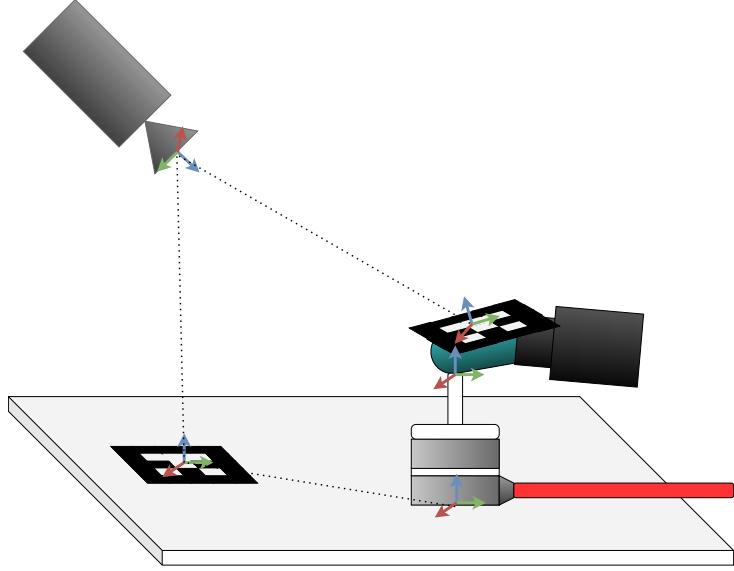
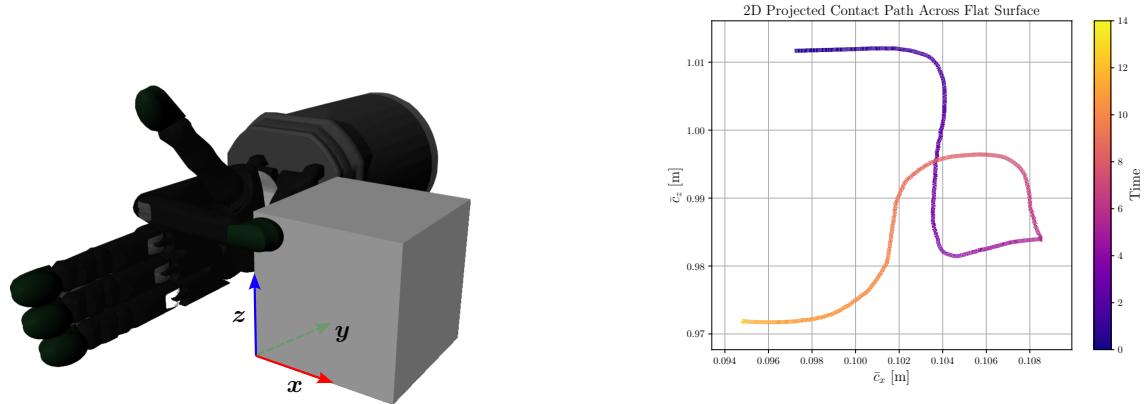


Fig. 5.2: Experimental setup for gathering data to train DL model B, as inspired by [93].

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 wrist 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 $-y$, meaning the GT normal is $\mathbf{n}_{gt} = [0, -1, 0]$ as shown in Fig. 5.3(a).



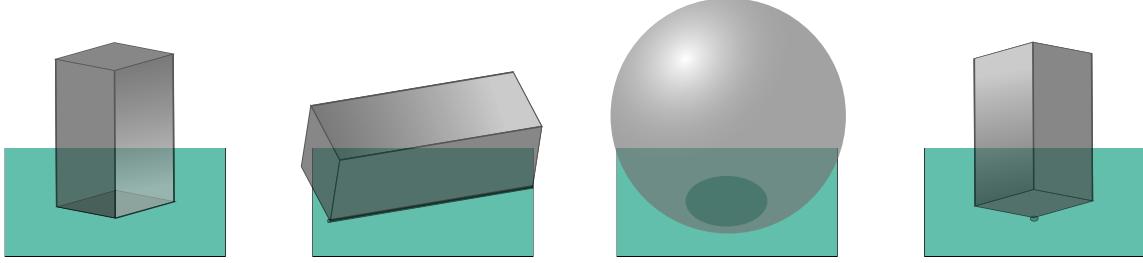
(a) Experimental setup for collecting linear velocity data to estimate contact normals using RLS.

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

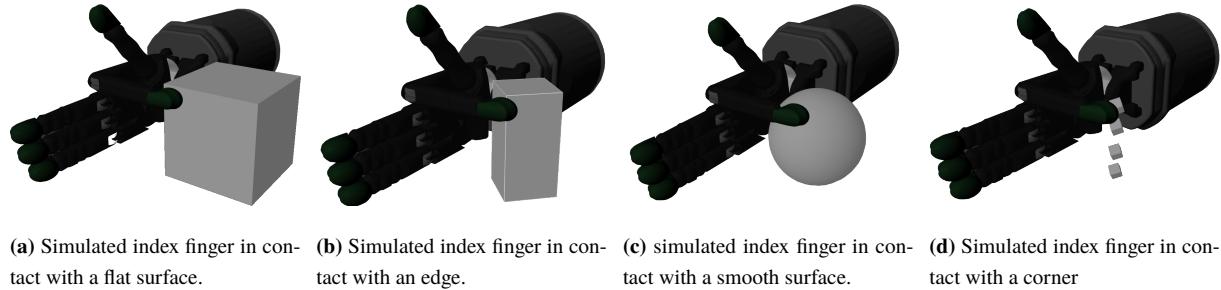
To test the performance of the DL model, four objects surfaces were used with known normals. These can be seen in Fig. 5.4 as a flat surface, an edge, a smooth surface and a corner.



(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 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.

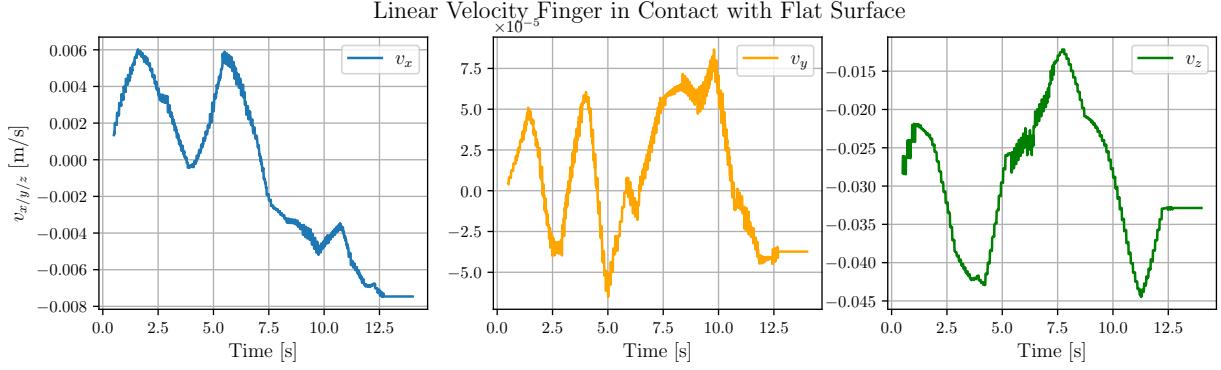


Fig. 5.6: Linear velocities of the contact points when the index finger moves across a flat surface as shown in Fig. 5.3(a).

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

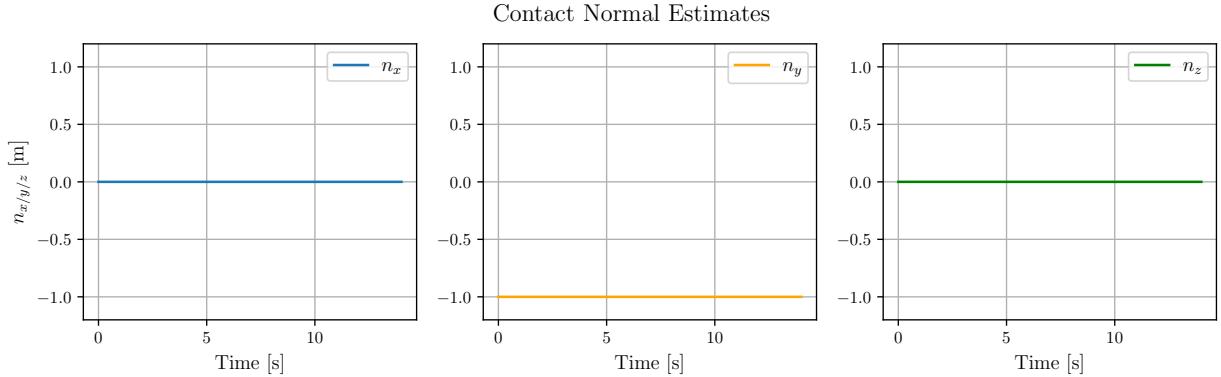
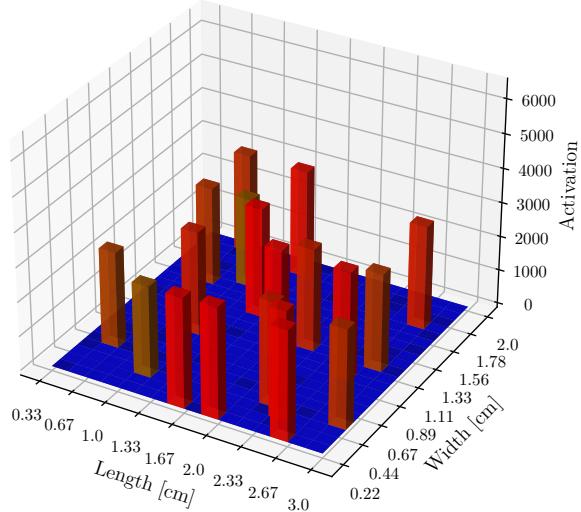


Fig. 5.7: The normal estimates across time as the experiment was executed.

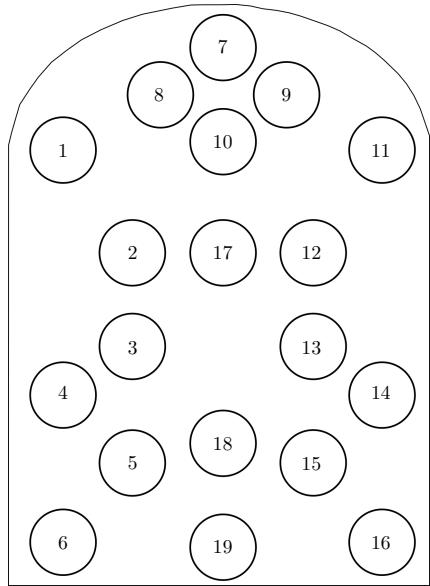
5.3.2 Skew Forces

After executing the DL model on all cases, the resulting simulated electrode activations were discovered to be infinite. Consequently, efforts were made 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.

Simulated Electrode Measurements - Flat Surface



(a) 3D plot of normalized 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.

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 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.

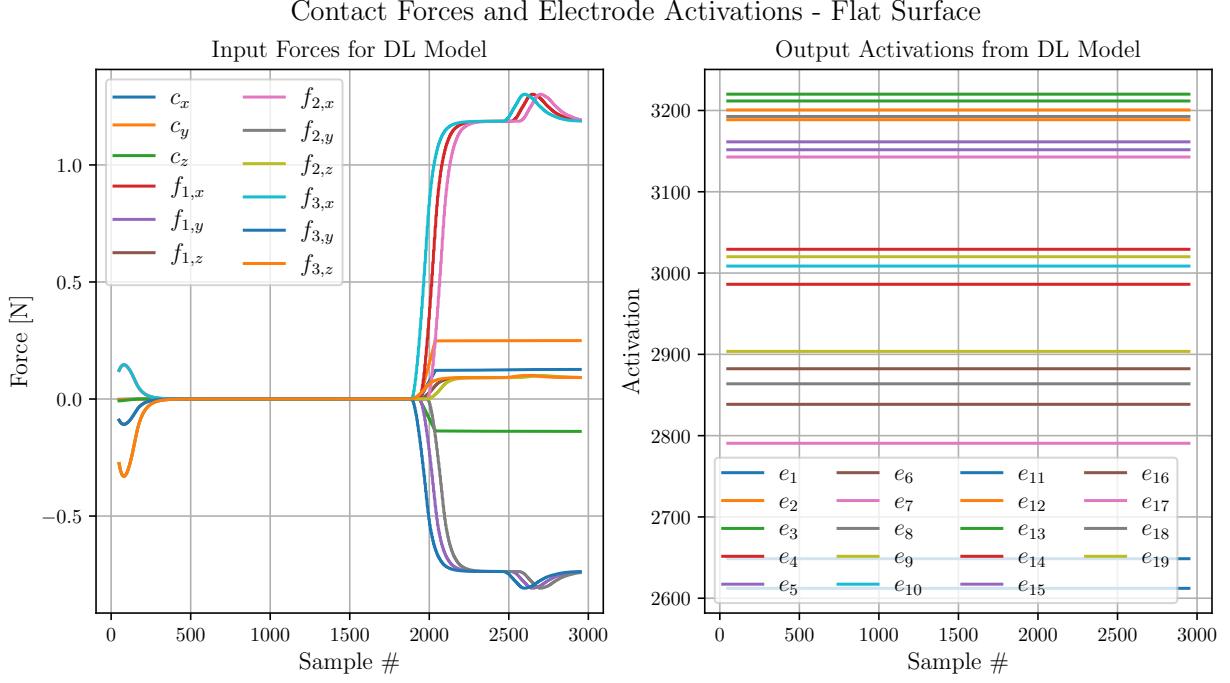


Fig. 5.9: The normalized simulated tactile electrode activations when the index finger is in contact with a flat surface.

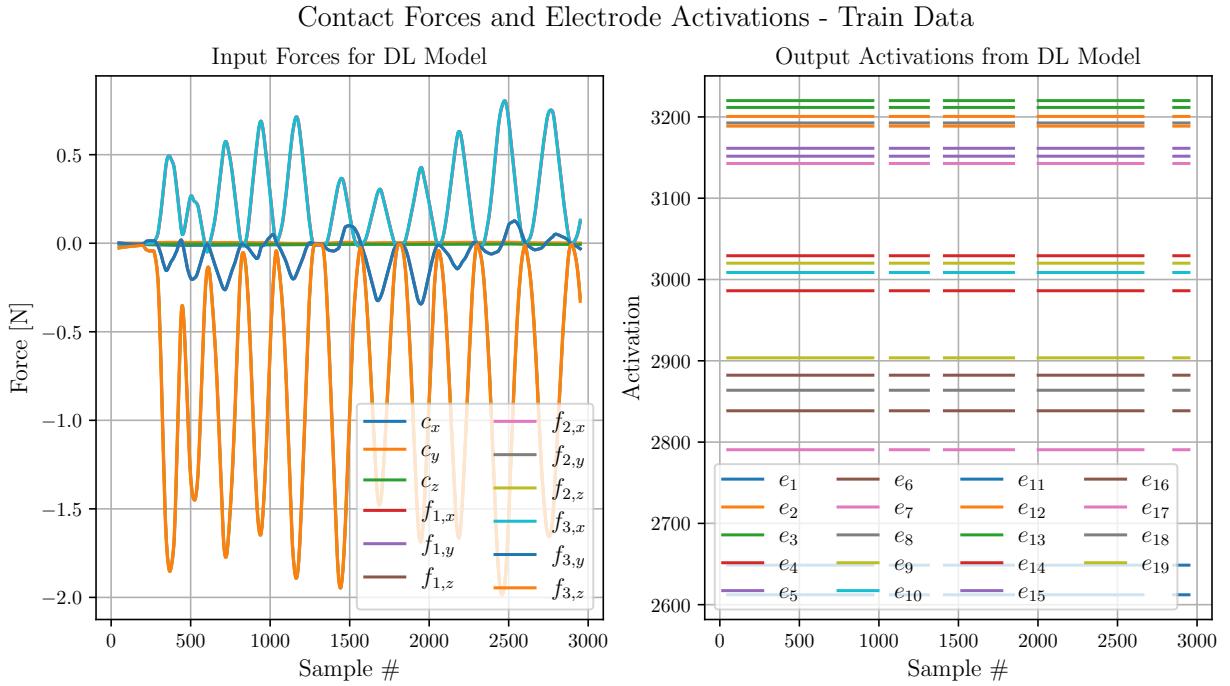


Fig. 5.10: The normalized simulated tactile electrode activations when training data was used as inputs for the DL model.

Contact Positions

The contact positions were determined by utilizing the grasping matrix \mathbf{G} and the transformation matrices for each taxel, as provided by [135]. The points in the world frame $\{\mathbf{W}\}$ were generated using Gazebo's physics engine, and the transformations were applied accordingly. To assess the engine's capability to represent contact points, a

3D mesh of the Stanford bunny [141] was used. The contact points were recorded during the sampling process, as depicted in Fig. 5.11. In Fig. 5.11(a), the contact points obtained by moving the fingers across the 3D mesh are shown. Fig. 5.11(b) displays the 3D mesh sampled to consist of 10 000 points, and Fig. 5.11(c) illustrates both the contact points and the sampled 3D mesh. The process of sampling the data shown in Fig. 5.11(a) can be found in the supplementary material `sampling-of-tactile-data.gif`.

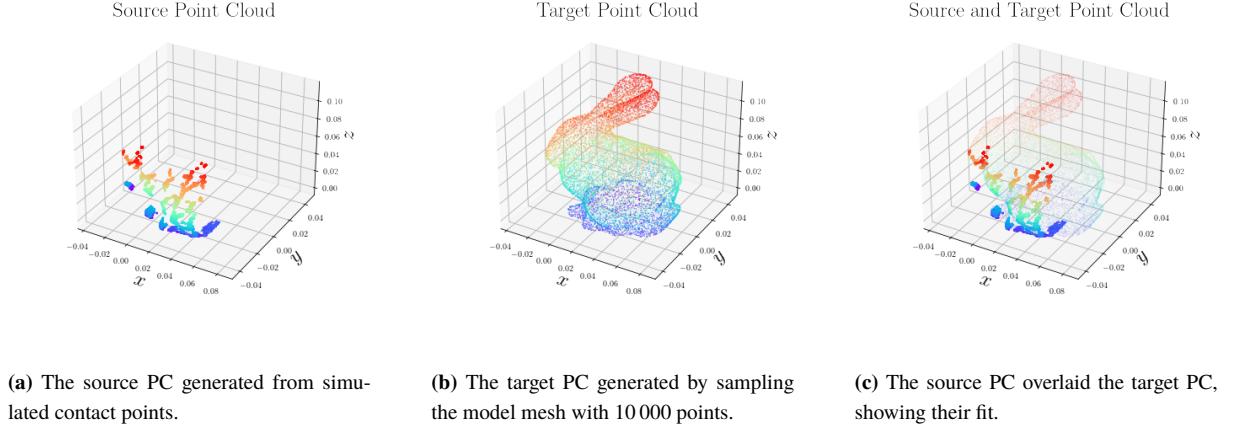


Fig. 5.11: 3D plots showing the sampled source and target PCs along with a plot showing both of them overlaid.

5.3.3 Physics Engine Comparison

Gazebo's physics engine provides interpretations of 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, depths and more. As a replacement for the lacking skew forces from the DL method and an extension of the normal estimation, Gazebo's physics engine is considered a supplement to the normal estimation and a substitute to DL method. The data sampling in this section is restricted to 100 Hz over 100 s.

The contact forces and torques generated by the engine are shown in Fig. 5.13(a) and Fig. 5.13(b), for the Shadow Dexterous finger's being in contact with an edge. The vectors here are projected into the xy-plane due to the z-component showing a negligible magnitude due to being parallel with the edge, as shown in Fig. 5.12.

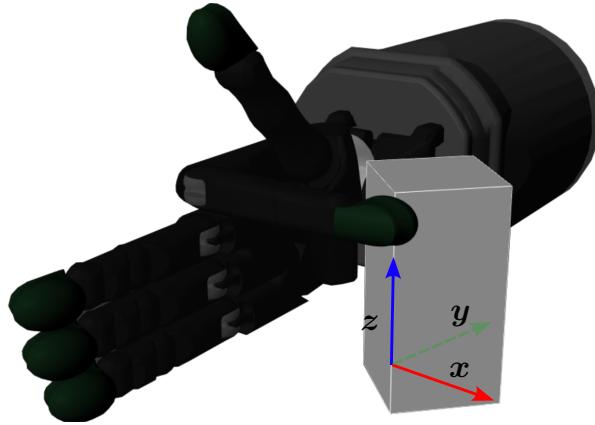
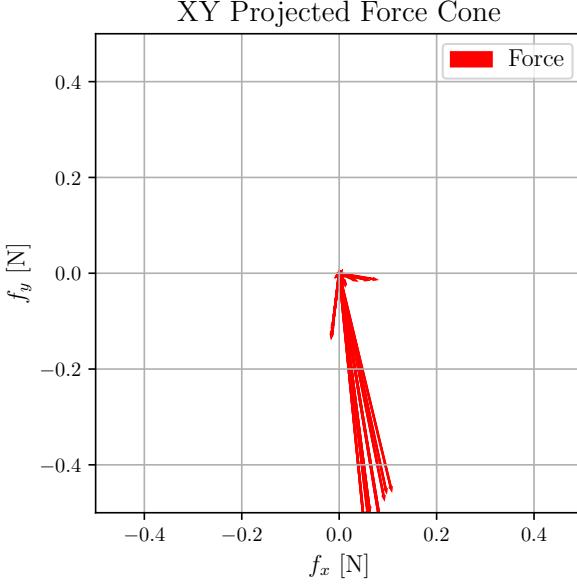
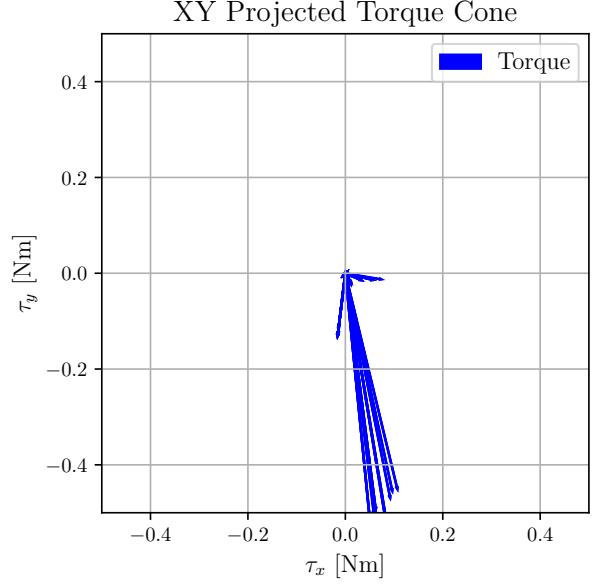


Fig. 5.12: Simulated SDH when in contact with an edge with the coordinate system.

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. 5.14(a). This was solved by



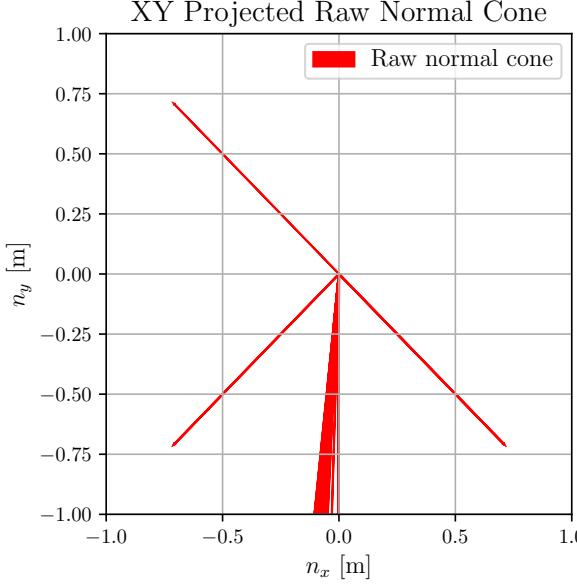
(a) XY projected force cone produced by Gazebo's physics engine when the SDH is in contact with an edge.



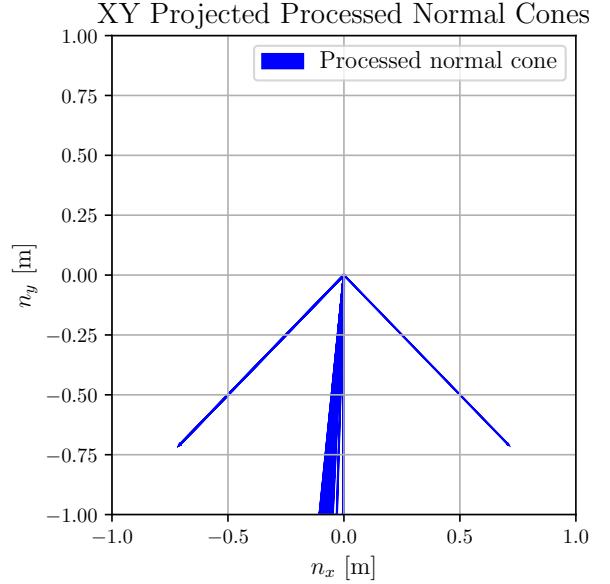
(b) XY projected torque cone produced by Gazebo's physics engine when the SDH is in contact with an edge.

Fig. 5.13: XY projected force and torque cones produced by Gazebo's physics engine when the SDH is in contact with an edge.

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 to define a cluster. The centroid of each cluster was then used as the representative and the normal cones shown in Fig. 5.14(b) were achieved.



(a) XY projected raw normal cones produced by Gazebo's physics engine when the SDH is in contact with an edge.



(b) XY projected processed normal cones produced by Gazebo's physics engine when the SDH is in contact with an edge.

Fig. 5.14: XY projected raw and processed normal cones produced by Gazebo's physics engine when the SDH is in contact with an edge.

Finally, the angle errors θ_e between the ground truth normals \mathbf{n}_{gt} and the simulated contact normals were found for each of the four presented surfaces. The probability distributions over angle errors $P(\theta_e)$ in degrees can be seen

in Fig. 5.15.

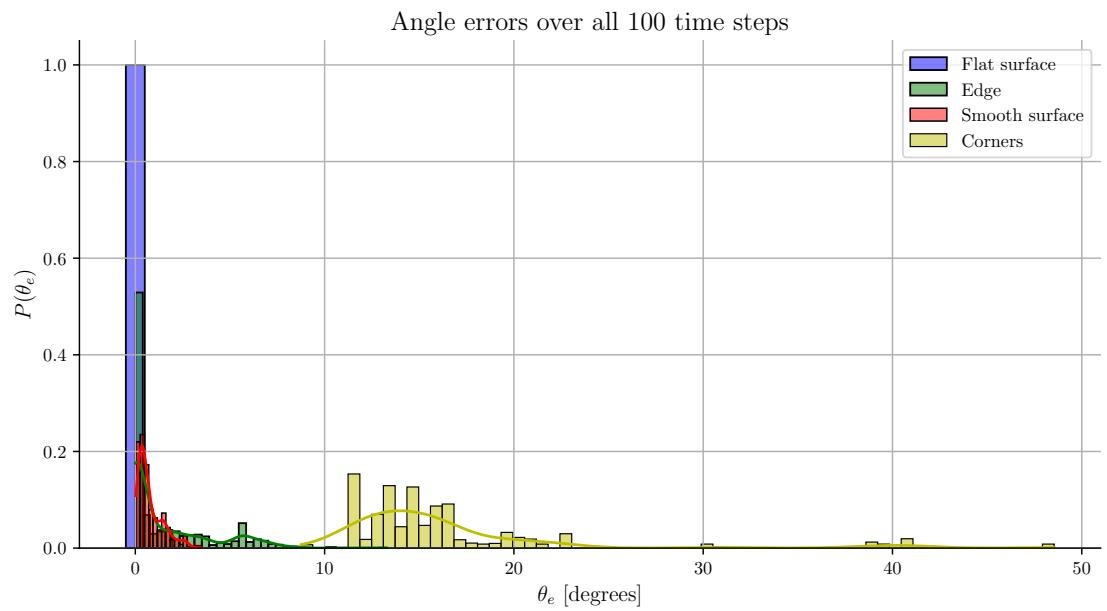


Fig. 5.15: Probability distributions $P(\theta_e)$ of the angle error θ_e between simulated and GT contact normals for each tested surface.

5.4 Discussion & Conclusion

In this chapter contact positions, normals and skew forces are successfully estimated or simulated.

The contact points have been successfully estimated using simulated contact points and the grasping matrix from [135].

The contact normals were estimated using estimated linear velocities from simulated contact points and the RLS method. This technique was tested and verified on a flat surface where the results show little error and thus were deemed satisfactory. Due to time constraints, contact normals were not estimated on other surfaces than flat, which would be necessary for stronger confirmation of the technique's applicability. Instead, Gazebo's physics engine produced acceptable results in these cases and is used as the normal estimation solution. The performance of Gazebo's physics is judged to be satisfactory. Estimating the contact normals provides a significant advantage by enabling the identification of robust surface features. This, in turn, has the potential to minimize the search space for the PE algorithm discussed in Chapter 6. This chapter has demonstrated that corners, being more complex surface features, result in the highest angle error, as one would expect.

It was at the time of this project not possible to simulate realistic tactile information from the chosen DL model B. The findings indicate that the DL model did not provide any useful information for reasons yet to be known. 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 [93]. Among the possible reasons for the lacking performance possible causes include non-representative weights, which due to the lack of transparency means no method exists to determine if the weights utility and Gazebo's API change may include unforeseen differences. To address these issues, the DL model could be retrained to ensure the legitimacy of the weights, given the dataset causes the performance presented in [93]. This was however not done due to the time constraint of this project. Instead, the forces and torques produced by Gazebo's physics engine are used as a substitute, the performance of which is considered satisfactory based on the presented force and torque cones.

In conclusion, this chapter 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

Given the results from Chapter 5 to solve problem 1, the goal of this chapter is to perform PE on a known object. This chapter focuses specifically on addressing the optimization-based PCR problem. It starts by formalizing the problem, followed by introducing the RCQP and GNC methods used to solve this problem and evaluates their performance in estimating the pose of the object of interest. The primary objective is to assess the extent to which RCQP and GNC can accurately estimate the position and orientation of the object, aiming for an absolute orientational error less than 5° and a positional error less than 1 cm. Importantly, it is assumed throughout the chapter that the object of interest is known, eliminating the need for classification.

To achieve this goal, the chapter presents the methodologies employed, the experimental setup, and the obtained results. The results obtained in this chapter are two-fold 1) synthetic source data is generated from the target data to ensure GT correspondences. This allows for the evaluation of the methods' robustness by introducing varying percentages of outliers. 2) the methods are applied on sampled source data from Chapter 5 to estimate the pose of the object based on computed correspondences. The additional challenge in 2) is thus to also solve the CP to a degree where the number of feature correspondences allows the algorithms to run.

Furthermore, in addition to solving the pose estimation problem, the chapter includes an analysis of the weight involved in GNC. Specifically the estimation of the signal-to-outlier ratio and weight convergence. If possible this is performed on both the synthetic and sampled sensor data.

Finally, the performance of the presented methods is concluded upon and the extent to which the results are satisfactory is addressed.

6.1 Problem Formulation

To solve the PE problem i.e. problem 2, the sampled source data from Chapter 5 is represented as

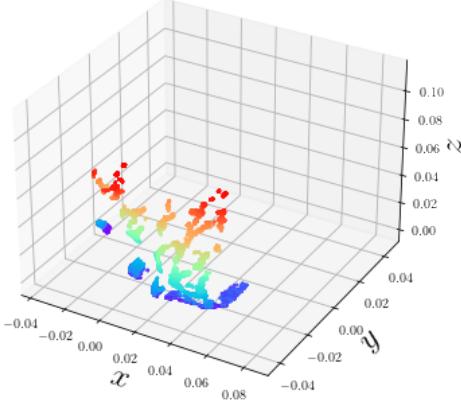
$$\mathbf{X}_{src} = \begin{bmatrix} c_x & c_y & c_z & n_x & n_y & n_z \\ c_x & c_y & c_z & n_x & n_y & n_z \\ \vdots \\ c_x & c_y & c_z & n_x & n_y & n_z \end{bmatrix} \in \mathbb{R}^{M_{src} \times 6}, \quad (6.1)$$

where $\mathbf{c} = [c_x, c_y, c_z]$ is a contact point, $\mathbf{n} = [n_x, n_y, n_z]$ is the corresponding point's normal vector and M_{src} is the number of source data points. The target data \mathbf{Y}_{tar} is structured likewise, except for the number of target data points being N_{tar} , where $N_{tar} \geq M_{src}$. For convenience, the i 'th row in source data is referred to as \mathbf{x}_i while the i 'th row in the target data is referred to as \mathbf{y}_i .

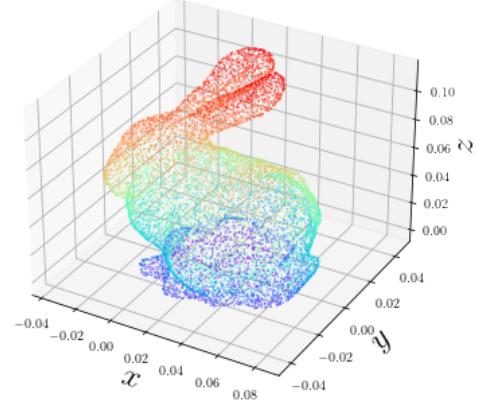
Under the assumption of already knowing the object of interest, \mathbf{Y}_{tar} is generated from the mesh of the Stanford bunny model [141] with a resolution of 10 000 data points as shown in figure Fig. 6.1(b). The sampled source data \mathbf{X}_{src} can likewise be seen in Fig. 6.1(a) as generated in Chapter 5.

To determine the transformation between the two point clouds $\mathbf{T}_{\mathbf{Y}_{tar}}^{\mathbf{X}_{src}}$, correspondences must be found according to the CP. Due to \mathbf{X}_{src} being produced by local sensing with high-density sensor regions, a PC feature which exploits this is desired. The one chosen is Fast Point Feature Histograms (FPFH) which is a feature descriptor used in 3D point cloud analysis and registration tasks and provides a feature matrix $\mathbf{F} \in \mathbb{R}^{N_{pc} \times 33}$, where N_{pc} is the number of

Source Point Cloud



Target Point Cloud



(a) The source data \mathbf{X}_{src} PC generated from simulated contact points, normals are also included, but not illustrated here for the sake of simplicity.

(b) The target data \mathbf{Y}_{tar} PC generated by sampling the model mesh with 10 000 points, normals are also included, but not illustrated here for the sake of simplicity.

Fig. 6.1: 3D plots showing source and target data.

rows in the PC. The feature descriptor calculates a histogram representation for each point by considering the local geometric properties of its neighboring points. It captures information about the distribution of surface normals and distances between points within a local neighborhood. Additionally, FPFH is computationally efficient and provides robust, scale-invariant feature representations.

Once features are computed, these are used to find row matches that make up a new source and target matrices such that,

$$\mathbf{X} \in \mathbb{R}^{M \times 6} \subseteq \mathbf{X}_{src} \in \mathbb{R}^{M_{src} \times 6} \quad \text{and} \quad \mathbf{Y} \in \mathbb{R}^{N \times 6} \subseteq \mathbf{Y}_{tar} \in \mathbb{R}^{N_{tar} \times 6}. \quad (6.2)$$

These new matrices \mathbf{X} and \mathbf{Y} are the ones used from this point on as source and target matrices.

Using the correspondences found from FPFH, the pose estimation problem can be formulated as an optimization problem for determining the optimal homogeneous transformation matrix between the two points clouds $\mathbf{T}^{\star \mathbf{X}}_{\mathbf{Y}}$, which for convenience is from now on referred to as \mathbf{T}^* , by

$$\mathbf{T}^* = \arg \min_{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3)} \sum_{i=1}^M d_{P_i}^2(\mathbf{x}_i, \mathbf{T}). \quad (6.3)$$

Here $\mathbf{T} = (\mathbf{R}, \mathbf{t}) \in \text{SE}(3)$ refers to the homogeneous transformation matrix $\mathbf{T} \in \mathbb{R}^{4 \times 4}$ from \mathbf{X} to \mathbf{Y} , which consists of a rotation matrix $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$ as members of the Special Euclidean (SE) group in 3D, and $d_{P_i}(\cdot)$ is the distance to the matching primitive P_i . The primitives of interest in this project are point-to-point and point-to-plane, due to the presence of contact normals. The distance function will differ between primitive, and thus the ones of interest are listed as

$$\min_{\mathbf{y}' \in P} \|\mathbf{x} - \mathbf{y}'\|_2^2 = \quad (6.4)$$

$$= \|\mathbf{x} - \mathbf{y}\|_2^2 = \|\mathbf{x} - \mathbf{y}\|_{\mathbf{I}_3}^2 \quad (\text{point}) \quad (6.5)$$

$$= (\mathbf{n}^\top (\mathbf{x} - \mathbf{y}))^2 = \|\mathbf{x} - \mathbf{y}\|_{\mathbf{n}\mathbf{n}^\top}^2 \quad (\text{plane}) \quad (6.6)$$

where $\mathbf{y}' \in \mathbb{R}^3$ is the term used as a substitute for each primitive, \mathbf{x} is a 3D point, $\mathbf{I}_3 \in \mathbb{R}^{3 \times 3}$ is the identity matrix and \mathbf{n} is the unit normal vector for a plane. However, these are highly sensitive to outliers and noise, and thus a

robust cost function $\rho(\cdot)$ is applied. Of the two presented in [124] i.e. Truncated Least Squares (TLS) and Geman McClure (GM), TLS was chosen, as it shows slightly better performance. The problem can thus be written as

$$\mathbf{T}^* = \arg \min_{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3)} \sum_{i=1}^M \rho(d_{P_i}^2(\mathbf{x}_i, \mathbf{T})). \quad (6.7)$$

However solving the PCR problem globally is still a challenging endeavor, even when known correspondences are available, primarily because of the non-convex nature of the rotation constraints, where $\mathbf{R} \in \text{SE}(3)$. For this reason, global approaches are applied to deal with the problem of local minima, by starting from a convex problem, and gradually increasing the non-convexity until the original problem is retrieved, this being the purpose of GNC. However, not all solutions when found globally are of interest, as the structure of the rotation matrix places certain quadratic constraints on the solution to be valid, such as orthonormality. From a then estimated optimal \mathbf{R}^* , the optimal translation \mathbf{t}^* can be found [123]. Thus the problem has been reduced to an orientation estimation problem to find \mathbf{R}^* .

6.2 Method

6.2.1 Graduated Non-Convexity

Rather than optimizing the robust cost function $\rho(\cdot)$ directly, instead a control parameter μ is introduced to instead optimize

$$\mathbf{T}^* = \arg \min_{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3)} \sum_{i=1}^M \rho_\mu(d_{P_i}^2(\mathbf{x}_i, \mathbf{T})) \quad (6.8)$$

However, since this problem cannot be solved using a non-minimal solver, the Black-Rangarajan (BR) duality can be applied, given the necessary conditions are met. These conditions are as follows

$$\lim_{z \rightarrow 0} \phi'(z) = 1 \quad \text{condition 1} \quad (6.9)$$

$$\lim_{z \rightarrow \infty} \phi'(z) = 0 \quad \text{condition 2} \quad (6.10)$$

$$\phi''(z) < 0 \quad \text{condition 3} \quad (6.11)$$

Here $\phi(z)$ is defined with respect to the robust cost function ρ in the following manner

$$\phi(z) \doteq \rho(\sqrt{z}). \quad (6.12)$$

Condition 1 indicates that as the argument z approaches 0, the derivative of $\phi(z)$ approaches 1. Geometrically, it implies that near the origin, the function $\phi(z)$ grows at a similar rate as the square root function itself. This behavior ensures that small values of z are penalized appropriately by the robust cost function $\rho(\cdot)$, allowing for robustness against outliers or deviations from the assumed distribution.

Condition 2 states that as z approaches ∞ , the derivative of $\phi(z)$ tends towards 0. This implies that as the argument becomes larger, the function $\phi(z)$ approaches a plateau or a constant value. Consequently, large values of z have a diminishing effect on the robust cost function, making it less sensitive to extreme observations.

Condition 3 states that the second derivative of $\phi(z)$ is negative, indicating that the function $\phi(z)$ is concave. This concavity is crucial as it ensures that the robust cost function $\rho(\cdot)$ is monotonically increasing. In other words, as the argument z increases, the robust cost increases as well. This property is desirable in robust estimation problems, as it helps in downplaying the impact of outliers or influential points on the overall estimation.

All of these conditions are met by the TLS [124].

With these conditions fulfilled, the BR duality enables

$$\arg \min_{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3)} \sum_{i=1}^M \rho_\mu \left(d_{P_i}^2(\mathbf{x}_i, \mathbf{T}) \right) = \arg \min_{\substack{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3), \\ w_i \in [0, 1]}} \sum_{i=1}^M \left[w_i d_{P_i}^2(\mathbf{x}_i, \mathbf{T}) + \Phi_{\rho_\mu}(w_i) \right], \quad (6.13)$$

where the rightmost side will be the problem of interest. In this expression $w_i \in [0, 1] \forall i = 1, 2, \dots, M$ are weights associated with each measurement \mathbf{x}_i and the outlier process function $\Phi_{\rho_\mu}(w_i)$, which computes a penalty based on the weight. The exact shape of $\Phi_{\rho_\mu}(w_i)$ is determined by the chosen robust cost function.

$\Phi_{\rho_\mu}(w_i)$ can simply be computed by

$$\Phi_{\rho_\mu}(w_i) = \frac{\mu(1-w_i)}{\mu+w_i} \bar{c}^2, \quad (6.14)$$

where μ is the control parameter used to dictate to what extent the $\Phi_{\rho_\mu}(w_i)$ should be convex or non-convex. For TLS, $\Phi_{\rho_\mu}(w_i)$ will become less convex as $\mu \rightarrow \infty$, while completely convex at $\mu = 0$. This effect can be seen in Fig. 6.2.

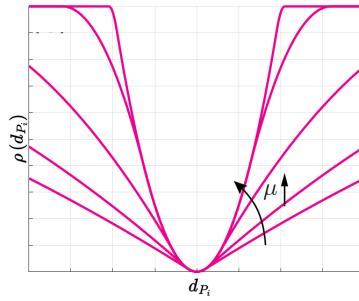


Fig. 6.2: Cost function for the TLS with control parameter μ .

\bar{c}^2 is a given truncation threshold, which in practice is set to the maximum error expected for the inliers. Finally, w_i can be computed when using TLS by

$$w_i^{(t)} = \begin{cases} 0 & \text{if } \hat{r}_i^2 \in \left[\frac{\mu+1}{\mu} \bar{c}^2, +\infty \right] \\ \frac{\bar{c}}{\hat{r}_i} \sqrt{\mu(\mu+1)} - \mu & \text{if } \hat{r}_i^2 \in \left[\frac{\mu}{\mu+1} \bar{c}^2, \frac{\mu+1}{\mu} \bar{c}^2 \right] \\ 1 & \text{if } \hat{r}_i^2 \in \left[0, \frac{\mu}{\mu+1} \bar{c}^2 \right] \end{cases} \quad (6.15)$$

where $\hat{r}_i^2 \doteq d_{P_i}^2(\cdot)$. (6.13) can now be solved through a two-step alternating optimization process. First, an outer loop is defined which increases control parameter μ from 0 to ∞ , until a stop criterion, dictated by the cost function, is met. [124] found that an increase of μ by a factor of 1.4 provided satisfactory results. The inner loop is two-fold 1) the current estimate of $\mathbf{T}^{(t)}$ is computed by optimizing over \mathbf{T} with fixed weights w_i , also referred to as the variable update step. The variable update step can thus be written as

$$\mathbf{T}^{(t)} = \arg \min_{(\mathbf{R}, \mathbf{t}) \in \text{SE}(3)} \underbrace{\sum_{i=1}^N w_i^{(t-1)} d_{P_i}^2(\mathbf{x}_i, \mathbf{T})}_{\text{constant}} + \underbrace{\Phi_{\rho_\mu}(w_i^{(t-1)})}_{\text{constant}} \quad (6.16)$$

This optimization problem is non-convex and is solved using RCQP, which is described in 6.2.2 Relaxed Convex Quadratic Programming. 2) the found \mathbf{T} is then used to update the weights $w_i \forall i = 1, 2, \dots, N$, which typically can be solved in closed form, as formulated in (6.15), which also is referred to as the weight update step.

When executing GNC in practice, values are needed for μ_0 and $\mathbf{w}^{(0)}$, i.e. the parameters' initial values, along with an update rule for μ and a stopping criterion. In this project the initial value for μ is $\mu_0 = 2r_{\max}^2/\bar{c}^2$ where $r_{\max}^2 \doteq \max_i \left(r^2(\mathbf{x}_i, \mathbf{T}^{(0)}) \right)$, meaning the maximum residual after the first variable update, while $\mathbf{w}_i^{(0)} = 1 \forall i = 1, 2, \dots, N$.

The update rule used for μ is $\mu \leftarrow 1.4\mu$, and as mentioned previously \bar{c}^2 is set to the maximum error expected for the inliers. For TLS a fitting stop criterion is chosen to be the sum weighted squared residuals, i.e.

$$\sum_{i=1}^N w_i \hat{r}_i^2. \quad (6.17)$$

These initial values, update rules and stop criterion are based on the discoveries of [124]. The GNC algorithm can be seen summarized in Algorithm 1.

Algorithm 1 GNC algorithm when using TLS as $\rho(\cdot)$

Input: matching pairs (\mathbf{X}, \mathbf{Y}) with potential outliers.

Output: $(\mathbf{X}_{in}, \mathbf{Y}_{in})$, stats

```

1:  $\mu \leftarrow 2r_{\max}^2/\bar{c}^2$ 
2: while true do // outer loop
3:    $\mu \leftarrow 1.4\mu$ 
4:    $\mathbf{R}^{(t)} \leftarrow$  Variable Update
5:    $\mathbf{w}^{(t)} \leftarrow$  Weight Update
6:   if  $\sum_{i=1}^N w_i \hat{r}_i^2$  has converged then
7:     break
8:   end if
9: end while
10: return  $(\mathbf{X}_{in}, \mathbf{Y}_{in})$ , stats

```

In Algorithm 1 the stats output generally contains information about the algorithm execution such as execution time, number of iterations, and the weights \mathbf{w} , while the source \mathbf{X}_{in} and target \mathbf{Y}_{in} inliers are subsets of \mathbf{X} and \mathbf{Y} with true correspondence according to GNC.

After GNC has converged and the inlier source \mathbf{X}_{in} and target \mathbf{Y}_{in} point matches are found, a final iteration of RCQP is executed on the inliers for a final estimate of \mathbf{T}^* .

6.2.2 Relaxed Convex Quadratic Programming

To compute the optimal transformation matrix \mathbf{T}^* , the problem definition can be expressed purely in terms of \mathbf{R} , after which the optimal translation \mathbf{t}^* can be computed. This simplifies the optimization problem and is therefore preferred. To do so, it will be derived from the optimization problem involving the full transformation matrix \mathbf{T} , formulated as

$$d_{P_i}^2(\mathbf{x}_i, \mathbf{T}) = (\mathbf{T} \oplus \mathbf{x}_i - \mathbf{y}_i)^\top \mathbf{C}_i (\mathbf{T} \oplus \mathbf{x}_i - \mathbf{y}_i). \quad (6.18)$$

Here $\mathbf{C}_i \in \mathbb{R}^{3 \times 3}$ is a symmetric matrix determined by the primitive as written in (6.4) and $\mathbf{T} \oplus \mathbf{x}_i$ is the Euclidean transformation of \mathbf{x}_i , which is an expression linear in \mathbf{R} and \mathbf{t} i.e.

$$\mathbf{T} \oplus \mathbf{x}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t} = (\tilde{\mathbf{x}}^\top \otimes \mathbf{I}_3) \text{vec}(\mathbf{T}). \quad (6.19)$$

Here $\tilde{\mathbf{x}} = [\mathbf{x}^\top, 1]^\top$ is the homogeneous coordinate of \mathbf{x} , \otimes is the Kronecker product and $\text{vec}(\mathbf{T})$ is the vectorization of \mathbf{T} i.e.

$$\text{vec}(\mathbf{T}) = [\text{vec}(\mathbf{R})^\top, \mathbf{t}^\top]^\top \quad (6.20)$$

$$= [r_{11}, r_{12}, \dots, r_{32}, r_{33}, t_1, t_2, t_3]^\top \in \mathbb{R}^{12}. \quad (6.21)$$

Using this linear relationship in \mathbf{T} , (6.18) can be reformulated as

$$d_{P_i}^2(\mathbf{x}_i, \mathbf{T}) = \tilde{\mathbf{r}}^\top \mathbf{N}_i^\top \mathbf{C}_i \mathbf{N}_i \tilde{\mathbf{r}} \quad (6.22)$$

where $\tau = \text{vec}(\mathbf{T})$, $\tilde{\tau}$ is the homogenized $\text{vec}(\mathbf{T})$ i.e. $[\text{vec}(\mathbf{T})^\top, 1]^\top$, and $\mathbf{N}_i = [\tilde{\mathbf{x}}_i^\top \otimes \mathbf{I}_3 | -\mathbf{y}_i] \in \mathbb{R}^{3 \times 10}$. Due to the quadratic nature of (6.22), it is possible to reorganize the observations and compress all the data into a single matrix $\tilde{\mathbf{M}}$, build from $\tilde{\mathbf{M}}_i = \mathbf{N}_i^\top \mathbf{C}_i \mathbf{N}_i \in \mathbb{R}^{13 \times 13}$, meaning

$$f(\mathbf{T}) = \sum_{i=1}^N d_{P_i}^2(\mathbf{x}_i, \mathbf{T}) = \tilde{\tau}^\top \left(\sum_{i=1}^N \tilde{\mathbf{M}}_i \right) \tilde{\tau} = \tilde{\tau}^\top \tilde{\mathbf{M}} \tilde{\tau}. \quad (6.23)$$

Here $f(\mathbf{T})$ is a quadratic objective function. From this expression, it can now be seen that the size of the problem has been made independent of N . By then applying marginalization to the unconstrained parts of the known \mathbf{T} i.e. the translation \mathbf{t} , further compression of the problem is achieved, such that the optimal f^* is written as

$$f^* = \min_{\mathbf{R} \in \text{SE}(3)} \tilde{\mathbf{r}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}}, \quad , \quad \tilde{\mathbf{r}} = [\text{vec}(\mathbf{R})^\top, 1]^\top. \quad (6.24)$$

Here $\tilde{\mathbf{Q}}$ is the Schur complement of the block matrix $\tilde{\mathbf{M}}_{t,t}$ in $\tilde{\mathbf{M}}$. Here the t subscript indicates the indexes corresponding to the translation variables and

$$\mathbf{Q} = \tilde{\mathbf{M}}_{t,t} - \tilde{\mathbf{M}}_{t,t} \tilde{\mathbf{M}}_{t,t}^{-1} \tilde{\mathbf{M}}_{t,t} \quad (6.25)$$

with $!t$ being the complement of t .

Although the problem is originally non-convex, it can be relaxed by applying Lagrangian relaxation to facilitate its solution. Even though a formal proof is not presented, empirical evidence from [123] demonstrates that this relaxation method consistently produces results that are close to the globally optimal solution for the marginalized problem.

Solving this non-convex optimization problem is achieved by applying Lagrangian duality to define two problems, a primary problem \mathcal{P} and a dual problem \mathcal{D} . The primary problem is an extension of (6.24), such that the necessary constraints for legal rotation matrices solutions are enforced, while the dual problem will be corresponding to a homogeneous version of the primal problem \mathcal{P} , which makes \mathcal{D} a Semidefinite Program (SDP), meaning the problem is convex and can be solved using off the shelf solvers.

To apply Lagrangian duality, strong duality must hold between \mathcal{P} and \mathcal{D} , meaning the solution to each of the problems are equal and their objective function values coincide. While this is not proven to hold, empirically this relaxation appears always to be tight, meaning strong duality holds, even under extreme conditions as asserted by [123]. This project will continue under the assumption that strong duality holds.

To formulate \mathcal{P} , the constraints of $\mathbf{R} \in \text{SE}(3)$ state that orthonormality and positive unit determinant must hold i.e.

$$\mathbf{R}^\top \mathbf{R} = \mathbf{I}_3 \quad \text{and} \quad \det(\mathbf{R}) = +1 \quad (6.26)$$

While it is appealing to formulate \mathcal{P} as a QCQP, the determinant constraint is cubic and thus does not allow this formulation. In other solutions to this problem, it is not uncommon to discard the determinant constraint [142, 143] resulting in the problem only being constrained by the orthonormality constraint. This amounts to optimizing in $\text{O}(3)$ rather than in $\text{SO}(3)$.

Instead, by introducing additional constraints, the duality of the problem can be enhanced. Each time a new scalar constraint $c_{k+1}(\cdot)$ is incorporated into the Lagrangian, a corresponding dual variable λ_{k+1} is introduced, and the dimension of the dual problem increases by one. This causes the new \mathcal{D} bound d_{k+1}^* to be at least as good as the previous problem's bound i.e.

$$d_k^* \leq d_{k+1}^* \leq f^*. \quad (6.27)$$

Since this change in \mathcal{P} causes direct changes to \mathcal{D} , \mathcal{D} is not intrinsic, instead it is dependant on the feasible region of \mathcal{P} . By utilizing this property, adding more valid quadratic constraints has been shown to improve the quality of the dual relaxation [144, Chapter 13].

Thus additional linearly independent quadratic constraints are added such as column and row-based orthonormality, along with additional quadratic constraints, which implicitly enforce the positive unit determinant, referred to as the rotation matrix's handedness. This constraint states that a positive unit determinant is guaranteed if the column space of \mathbf{R} respects the right-hand rule i.e.

$$\mathbf{R}^{(1)} \times \mathbf{R}^{(2)} = \mathbf{R}^{(3)}, \quad (6.28)$$

where $\mathbf{R}^{(k)}$ is the k -th column of \mathbf{R} . Utilizing the three possible cyclic permutations of this equation provides exactly three independent quadratic constraints, which strengthens the duality between \mathcal{P} and \mathcal{D} . A final constraint can be added by homogenizing \mathcal{P} by introducing an auxiliary variable y with the constraint $y^2 = 1$. The primary problem \mathcal{P} can thus be expressed as

$$\begin{aligned} & \min_{\mathbf{R}} \tilde{\mathbf{r}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}} \quad , \quad \tilde{\mathbf{r}} = [\text{vec}(\mathbf{R})^\top, y]^\top, \\ & \text{s.t.} \\ & \mathbf{R}^\top \mathbf{R} = y^2 \mathbf{I}_3, \\ & \mathbf{R} \mathbf{R}^\top = y^2 \mathbf{I}_3, \\ & \mathbf{R}^{(i)} \times \mathbf{R}^{(j)} = y \mathbf{R}^{(k)} \quad , \quad i, j, k = \text{cyclic}(1, 2, 3), \\ & y^2 = 1. \end{aligned} \quad (\mathcal{P})$$

To derive the dual problem \mathcal{D} , the QCQP nature of \mathcal{P} is utilized to express its Lagrangian as

$$\mathcal{L}(\tilde{\mathbf{r}}, \tilde{\lambda}) = \gamma + \tilde{\mathbf{r}}^\top \tilde{\mathbf{Z}} \tilde{\mathbf{r}}. \quad (6.29)$$

Here $\tilde{\lambda} = [\lambda^\top, \gamma]^\top \in \mathbb{R}^{22}$ is the dual vector, where 22 is the number of constraints in \mathcal{P} , $\tilde{\mathbf{Z}} = \tilde{\mathbf{Q}} + \tilde{\mathbf{P}}(\tilde{\lambda})$ is a penalization matrix of the sum of $\tilde{\mathbf{Q}}$, which contains all the data from \mathcal{P} , and $\tilde{\mathbf{P}}(\tilde{\lambda})$ which is the accumulation of all the penalization terms corresponding to different constraint types. $\tilde{\mathbf{P}}(\tilde{\lambda})$ can be computed as

$$\tilde{\mathbf{P}}(\tilde{\lambda}) = \tilde{\mathbf{P}}_r(\Lambda_r) + \tilde{\mathbf{P}}_c(\Lambda_c) + \tilde{\mathbf{P}}_d(\{\lambda_{d_{ijk}}\}) + \tilde{\mathbf{P}}_h(\gamma), \quad (6.30)$$

which by definition is a linear function of the dual variables, where the structural pattern of the involved matrices can be found in [123]. Using this formulation of the Lagrangian, its relaxation can be solved in closed form as

$$d(\tilde{\lambda}) = \min_{\tilde{\mathbf{r}}} \mathcal{L}(\tilde{\mathbf{r}}, \tilde{\lambda}) = \min_{\tilde{\mathbf{r}}} \gamma + \tilde{\mathbf{r}}^\top \tilde{\mathbf{Z}} \tilde{\mathbf{r}} \quad (6.31)$$

$$= \begin{cases} \gamma & \text{if } \tilde{\mathbf{Z}} \geq \mathbf{0}, \\ -\infty & \text{otherwise.} \end{cases} \quad (6.32)$$

In cases where $\tilde{\mathbf{Z}}$ is not Positive Semidefinite (PSD) no lower bound exists for this problem, and thus no optimal solution can be found. But by restricting all $\tilde{\lambda}$, to the solutions of $d(\tilde{\lambda})$ which satisfy $\tilde{\mathbf{Z}} \geq 0$, the dual problem \mathcal{D} to the homogeneous primal problem \mathcal{P} is a Semidefinite Program on the form

$$\begin{aligned} d^* &= \max_{\tilde{\lambda}} \gamma, \\ &\text{s.t.} \\ \tilde{\mathbf{Z}}(\tilde{\lambda}) &= \tilde{\mathbf{Q}} + \tilde{\mathbf{P}}(\tilde{\lambda}) \geq 0, \end{aligned} \quad (\mathcal{D})$$

i.e. a convex problem that can be solved by out-of-the-box solvers.

From the now defined primal problem \mathcal{P} and dual problem \mathcal{D} , and under the assumption that strong duality holds $f^* = d^*$, which implies by duality theory that $\tilde{\mathbf{r}}^*$ must be a minimizer of the Lagrangian 6.29 evaluated at $\tilde{\lambda}^*$, such that

$$\mathbf{r}^* = \arg \min_{\mathbf{r}} \mathcal{L}(\mathbf{r}, \lambda^*) \quad \Rightarrow \quad (\tilde{\mathbf{r}}^*)^\top \tilde{\mathbf{Z}}^* \tilde{\mathbf{r}}^* = 0. \quad (6.33)$$

Assuming $\tilde{\mathbf{Z}}^*$ is PSD i.e. $\tilde{\mathbf{Z}}^* \succeq 0$, then the optimal solution $\tilde{\mathbf{r}}^*$ must lie in the nullspace of $\tilde{\mathbf{Z}}^*$ i.e. $\tilde{\mathbf{r}}^* \in \text{null}(\tilde{\mathbf{Z}}^*)$.

If this was not the case, either $\text{null}(\tilde{\mathbf{Z}}^*) = \emptyset$ or $\tilde{\mathbf{r}}^* \notin \text{null}(\tilde{\mathbf{Z}}^*)$, as a direct consequence of $\tilde{\mathbf{Z}}^*$ being PSD.

From this the nullspace basis \mathbf{V} of $\tilde{\mathbf{Z}}^*$ i.e. $\mathbf{V} = \text{null}(\tilde{\mathbf{Z}}^*)$ can be computed using QR decomposition. If it here is the case that $\text{rank}(\mathbf{V}) = 1$ then the optimal solution \mathbf{r}^* is found as the dehomogenized nullspace basis of $\tilde{\mathbf{Z}}$, due to \mathcal{P} being homogenous.

Verifying the legitimacy of this solution is simply done by evaluating the nullspace determinant to be unit, and the difference between the optimal value of \mathcal{P} and \mathcal{D} being 0 i.e.

$$\text{rank}(\text{null}(\tilde{\mathbf{Z}}^*)) = 1 \quad \text{and} \quad d^* - f^*(\mathbf{R}^*) = 0 \quad (6.34)$$

where $\mathbf{R}^* = \text{reshape}(\mathbf{r}^*)$.

The algorithm for computing the optimal \mathbf{R}^* can be seen summarized in Algorithm 2. To compute the full optimal transformation matrix \mathbf{T}^* , this result is used in Algorithm 3, and thus the optimal homogeneous transformation matrix has been estimated using RCQP.

Algorithm 2 RCQP algorithm for determining \mathbf{R}^* from $\tilde{\mathbf{r}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}}$

Input: Marginalized quadratic form of the primary problem, \mathcal{P} i.e. $\tilde{\mathbf{r}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}}$

Output: Globally optimal \mathbf{R}^* , if strong duality holds

- 1: $\tilde{\mathbf{Z}}(\tilde{\lambda}) \leftarrow \text{build}(\tilde{\mathbf{Q}} + \tilde{\mathbf{P}}(\tilde{\lambda}))$ based on 6.30
 - 2: $(\tilde{\lambda}^*, \tilde{\mathbf{Z}}^*) \leftarrow \text{cvx}(\mathcal{D}(\tilde{\lambda}, \tilde{\mathbf{Z}}))$ i.e. solve the Semidefinite Program using optimization tools such as cvx
 - 3: $\mathbf{V} \leftarrow \text{null}(\tilde{\mathbf{Z}}^*)$, from methods such as QR decomposition.
 - 4: **ASSERT** ($\text{rank}(\mathbf{V}) == 1$), ensure the nullspace rank is unit
 - 5: **ASSERT** $((\tilde{\mathbf{r}}^*)^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}}^* - d^* < \epsilon)$, where ϵ is some threshold close to 0, to ensure strong duality
 - 6: $\mathbf{r}^* \leftarrow \tilde{\mathbf{r}}^*$, dehomogenize $\tilde{\mathbf{r}}^*$
 - 7: $\mathbf{R}^* \leftarrow \text{reshape}(\mathbf{r}^*)$, reshape $\mathbf{r}^* \in \mathbb{R}^9$ to $\mathbf{R}^* \in \mathbb{R}^{3 \times 3}$
 - 8: **return** \mathbf{R}^* , return the globally optimal rotation
-

Algorithm 3 Algorithm for determining \mathbf{T}^* from $\tilde{\mathbf{r}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{r}}$

Input: matching pairs from (\mathbf{X}, \mathbf{Y})

Output: Globally optimal \mathbf{T}^*

- 1: initialize $\tilde{\mathbf{M}} \leftarrow \text{zeros}(13)$
 - 2: **for** $(\mathbf{x}_i, P_i) \in \{\mathbf{x}_i \leftrightarrow P_i\}_{i=1}^m$ **do**
 - 3: $\tilde{\mathbf{M}}_i \leftarrow \text{buildTerm}(\mathbf{x}_i, P_i)$ here build term depends on the primitive.
 - 4: $\tilde{\mathbf{M}} \leftarrow \tilde{\mathbf{M}} + \tilde{\mathbf{M}}_i$
 - 5: **end for**
 - 6: marginalize $\tilde{\mathbf{Q}} = \tilde{\mathbf{M}} / \tilde{\mathbf{M}}_{t,t}$ using Schur complement.
 - 7: compute \mathbf{R}^* from Algorithm 2
 - 8: $t^* \leftarrow -\tilde{\mathbf{M}}_{t,t}^{-1} \tilde{\mathbf{M}}_{t,t} \tilde{\mathbf{r}}$ where $\tilde{\mathbf{r}} = [\text{vec}(\mathbf{R})^\top 1]^\top$
 - 9: **return** $\mathbf{T}^*(\mathbf{R}^*, t^*)$
-

6.3 Experimental Setup

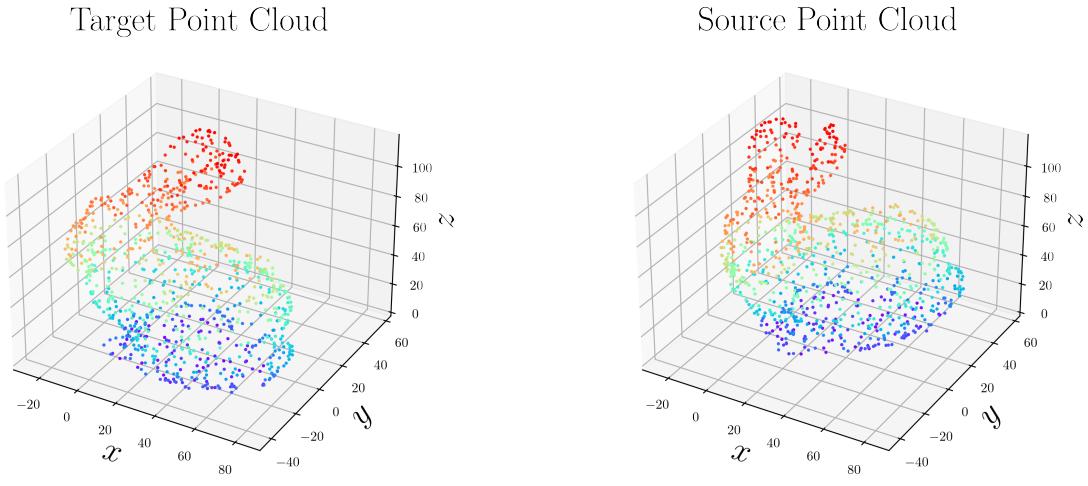
To Test the performance of the algorithms presented in 6.2 Method two setups are needed, one for synthetic source data and one for sampled source data.

The synthetic source and target data are generated by sampling the Stanford bunny mesh uniformly with 1000 data points and corresponding normals. This will ensure the index i of each PC will be true correspondences even if transformed. Once sampled, the synthetic source data is transformed using

$$\mathbf{T}_{\text{Source}}^{\text{Target}} = \begin{bmatrix} \mathbf{R}_z(\pi/4) & \mathbf{t}(0) \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (6.35)$$

Here $\mathbf{R}_z(\pi/4) \in \mathbb{R}^{3 \times 3}$ is the rotation matrix for a rotation of $\pi/4 = 45$ deg about the z -axis, $\mathbf{t}(0) \in \mathbb{R}^3$ is a translation of 0 and $\mathbf{0}_{1 \times 3} = [0, 0, 0]$. The target and synthetic source data can be seen in Fig. 6.3(b) and Fig. 6.3(a) respectively.

On these points clouds GNC+RCQP are applied over 20 experiments with varying outlier percentages $\alpha = [10\%, 20\%, \dots, 90\%]$.



(a) The target PC generated from sampling 1000 points from the CAD model.

(b) The synthetic source data based on the target data, but transformed $\mathbf{T}_{\text{Source}}^{\text{Target}}$.

Fig. 6.3: 3D plots showing the synthetic source data and target data.

The setup for the sampled source data as shown in Fig. 6.1(a), additionally includes the solving of the CP by finding feature correspondences between the sampled source PC and target PC. The target PC is the same as the one generated for the synthetic source data setup.

Due to self-collisions during the sampling process, a statistical outlier filter was applied with the number of neighbors n_{nb} being 1.000 and the standard deviation ratio σ being 0.5 to eliminate these outliers.

Using FPFH it was unfortunately found that the source data did not contain enough information to find a satisfactory number of feature matches with the target data for GNC+RCQP to function. This was found by sampling target data with a varying number of data points uniformly distributed while computing the number of feature matches with the sampled source data. These results can be seen in Fig. 6.4, where the number of target data points sampled are $\gamma = [100, 200, \dots, 900, 1000, 2000, \dots, 50000]$ and the largest number of feature matches show to be two, which

is far from sufficient. Filtering the sampled source data did not increase the number of feature matches as shown in the Fig. 6.4. Greater target data PCs were attempted but were not possible due to lack of computer memory.

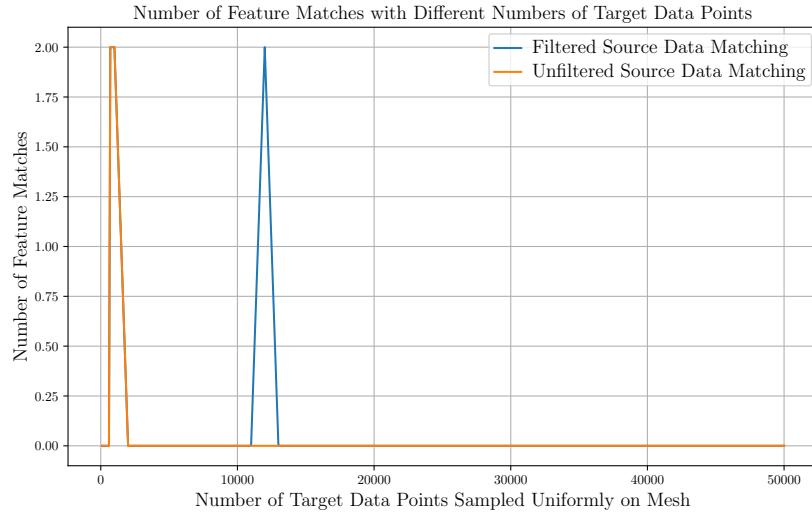


Fig. 6.4: Number of feature matches between target data containing γ data points and the sampled source data, both filtered and unfiltered.

6.4 Results

Using the synthetic source data the results presented in this section were found.

6.4.1 Pose Estimation Performance Data

The number of iterations for each outlier percentage can be seen as boxplots in Fig. 6.5, Fig. 6.6 shows boxplots of the execution times for each outlier percentage, Fig. 6.7 shows boxplots of the estimated orientation error as the angle θ_e being the angle in Equivalent Angle Axis (EAA) format. Finally, Fig. 6.8 shows boxplots of the translational error.

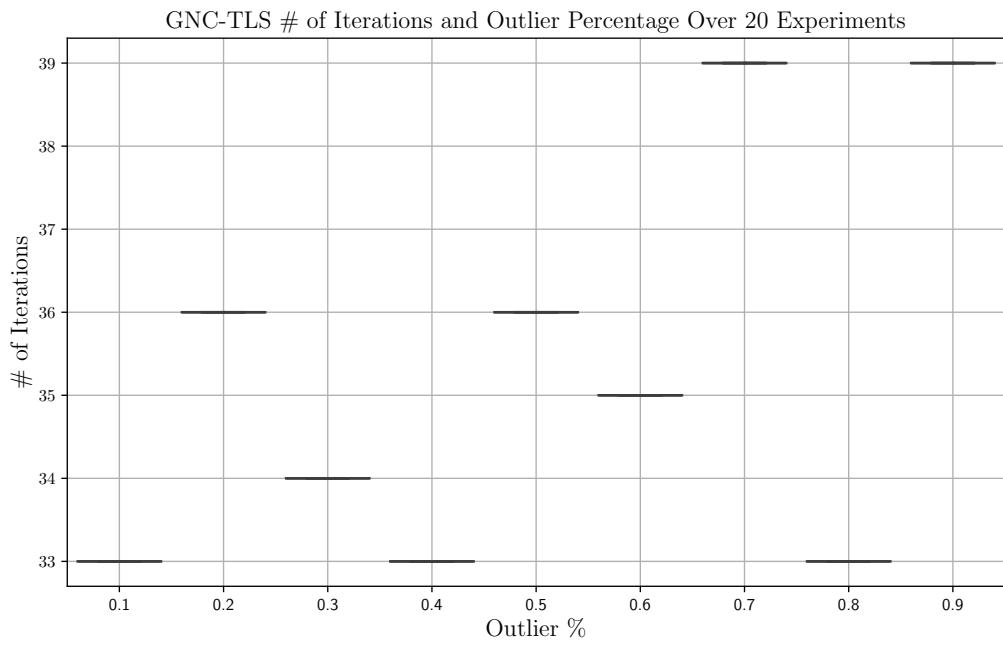


Fig. 6.5: Boxplots showing the number of GNC outer loop iterations for each outlier percentage from 10 % to 90 % in steps of 10 %, over 20 experiments.

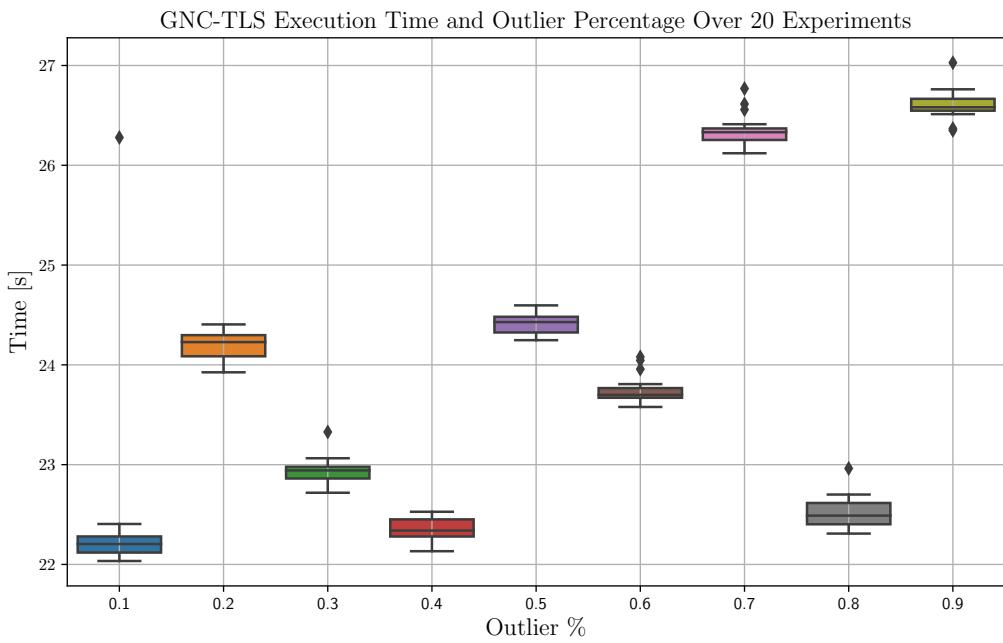


Fig. 6.6: Boxplots showing the execution time of GNC for each outlier percentage from 10 % to 90 % in steps of 10 %, over 20 experiments.

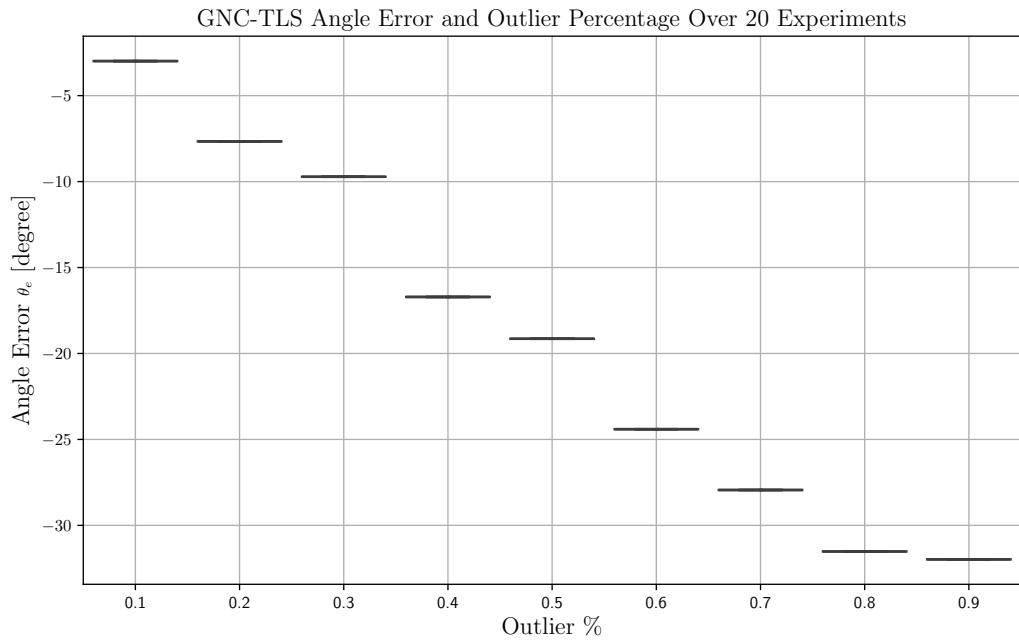


Fig. 6.7: Boxplots showing the orientation angle error θ_e of GNC+RCQP for each outlier percentage from 10 % to 90 % in steps of 10 %, over 20 experiments.

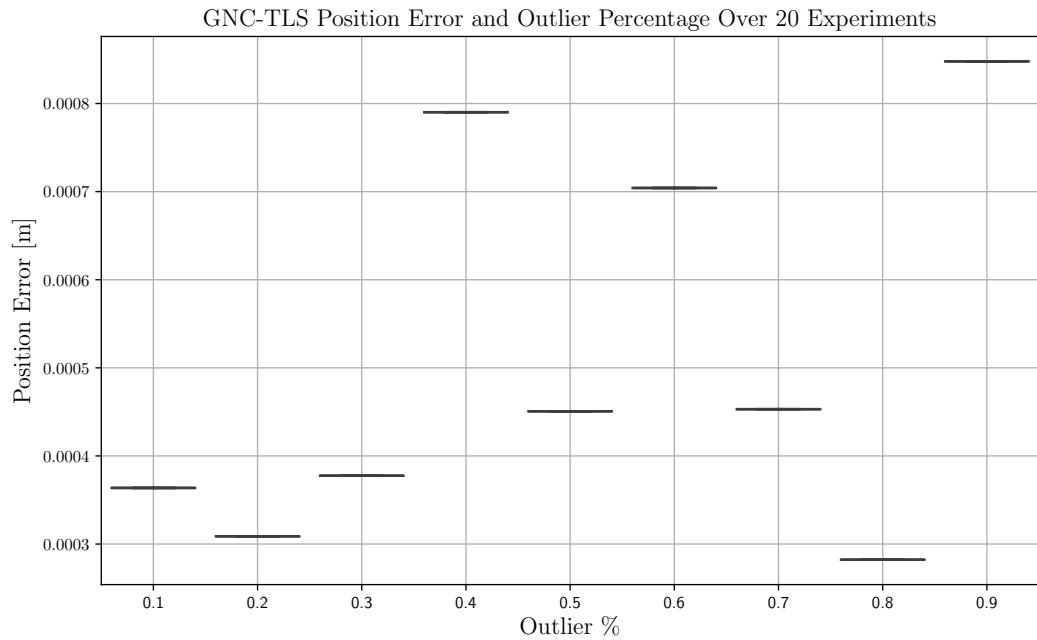


Fig. 6.8: Boxplots showing the positional error of GNC+RCQP for each outlier percentage from 10 % to 90 % in steps of 10 %, over 20 experiments.

6.4.2 Weight Convergence Data

In Fig. 6.9 the weight history is shown for a single run with 10 % outliers. Here each graph shows a weight entry and how it changes throughout the outer loop iterations. It can here be seen that the weights start at zero and

converge either to 0 or 1 as one would come to expect.

Fig. 6.10 show the number of non-zero entries for 20 experiments with 10 % outliers.

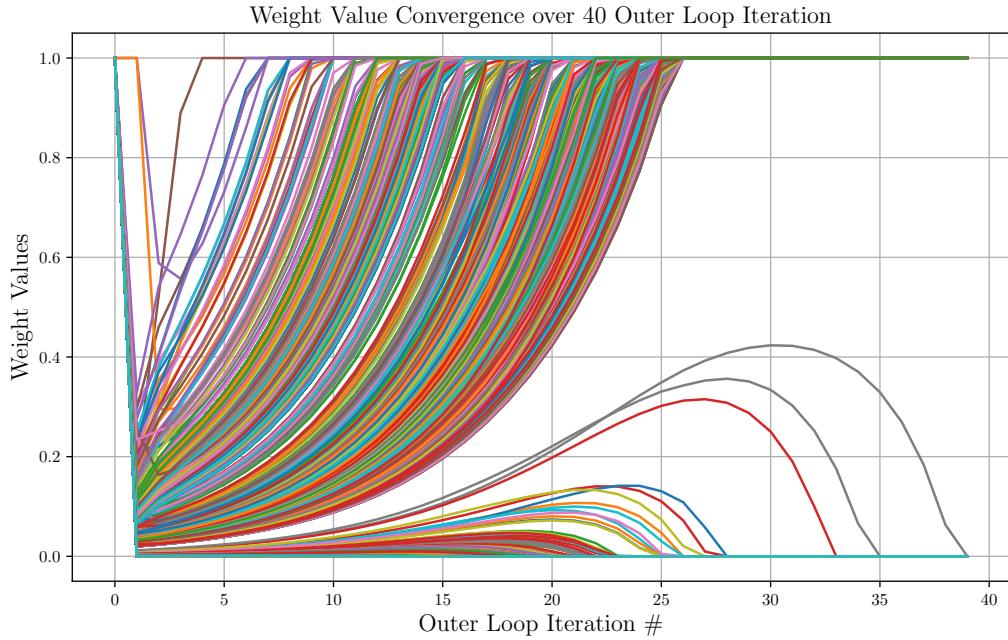


Fig. 6.9: Weight changes as outer loop iterations pass. This is performed with 10 % outliers.

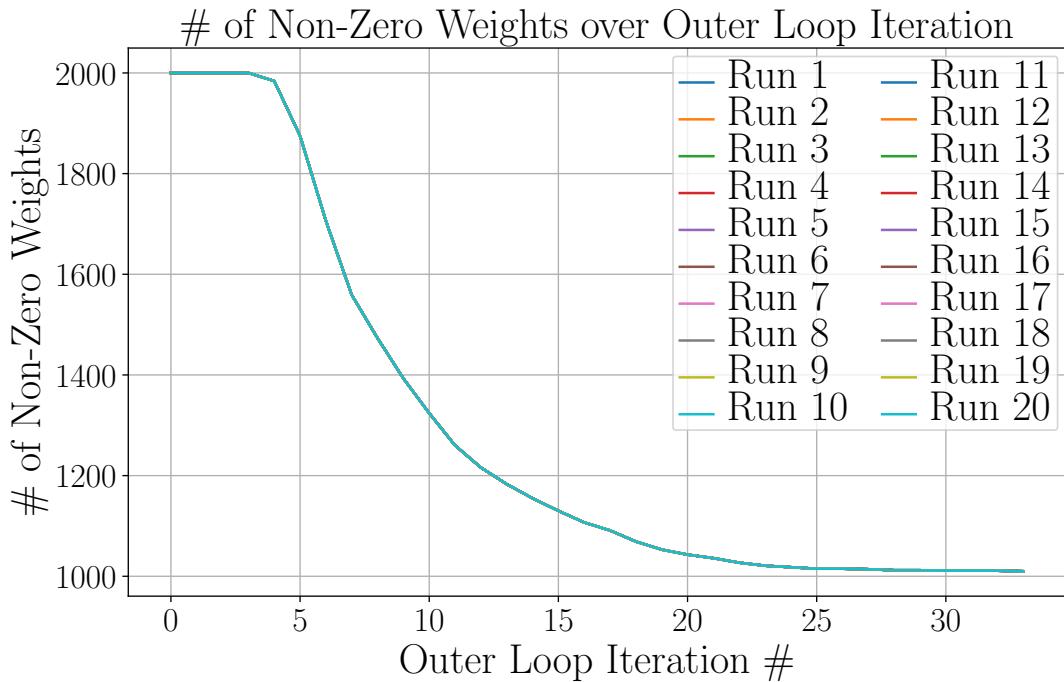


Fig. 6.10: Number of non-zero elements in w over 34 outer loop iterations.

6.5 Discussion & Conclusion

In this chapter, the pose of the known object, the Stanford bunny, has successfully been estimated from synthetic source data.

The problem of interest was successfully formalized for two cases, one considering synthetic and one considering sampled source data. Following this, both of the core methodologies GNC and RCQP were presented, along with the experimental setups for testing both cases. It was found that the sampled source data generated from Chapter 5 was insufficient to estimate the object's pose, while the synthetic source data successfully did so. The performance of which fulfilled the presented success criteria of absolute angle errors less than 5° and positional errors less than 1 cm. This was achieved with an outlier ratio of 10 % as $\theta_e \approx -3^\circ$ while the positional error, independent of the outlier ratio never exceeded 0.08 cm. However, as the outlier percentage increased to 20 %, the angle error reached -7.66° , failing to meet the desired success criteria. This trend persisted for higher outlier ratios as well.

The obtained results demonstrate the effectiveness of the GNC+RCQP method for pose estimation when applied to synthetic data. However, when dealing with sampled data, the methods faced challenges due to the lack of discernible features. One of the main difficulties encountered was the inability to identify a sufficient number of feature correspondences. This limitation can be attributed to the sparsity of the point cloud, indicating the need for intelligent probing techniques to enhance feature detection.

The execution times and number of iterations appeared consistent with trends demonstrated in the original paper [124].

The weight convergence analysis revealed an expected trend of convergence towards 0 and 1. However, the specific values at which the weights converged require further investigation and explanation. For example, in the case of a 10 % outlier scenario, the expected outcome would be the convergence of the number of non-zero weights to 900, from which an outlier ratio could be computed. Detailed graphs illustrating the weight convergence results can be found in Appendix C.

In summary, the GNC+RCQP method exhibited promising performance in pose estimation for synthetic data. However, challenges arose when working with sampled data, emphasizing the need for improved feature detection and data sampling techniques. The angle error increased with higher outlier percentages, suggesting the importance of robustness in handling outliers. Further analysis is required to understand the convergence values of the weights and refine the pose estimation algorithm accordingly.

Chapter 7

In-Hand Manipulation

The goal of this chapter is to analyze the performance of the RL-based manipulation algorithm DAPG, which employs IL to constrain the algorithm's search space when solving the relocation task, as well as compare the final performance with the DAPG algorithm trained by the authors of [33]. The relocation task requires the robot hand to pick up an object and move it to a target location.

The analysis will focus on the relevant metrics from the training process as well as the final model's performance. DAPG was chosen as the algorithm from [33], due to it showing the best performance when solving the relocation task.

The structure of this chapter follows a logical progression. It begins with the presentation of the problem formulation, providing context regarding the problem structure and mathematical expressions formulations. Next, the DAPG method is introduced and described in detail, outlining its key components and principles. Following that, the experimental setup for data collection, which is used for training the algorithm, is presented.

Once the data is collected and the algorithm is trained, the results obtained are presented and analyzed. This analysis includes evaluating the success of training the model as well as comparing the performance of the DAPG method in addressing the problem, with the performance of the model trained by the authors. The findings from the analysis are then discussed, providing insights and interpretations of the results.

Finally, the chapter concludes by summarizing the main points, discussing the implications of the findings, and highlighting potential future directions.

The simulation is done using a AH [49] in the dynamic simulator MuJoCo [27]. The hand is thus equipped with 30 controllable parameters, 24 for the hand's joints and 6 for the hand's position $\mathbf{p}_{hnd} = [p_x, p_y, p_z]$ and orientation $\mathbf{u} = [\phi, \theta, \psi]$ in RPY format. These make up the agent's actions a , while the states s also include the position of the prop when performing the relocation task.

7.1 Problem Formulation

RL aims to solve a control problem in the form of a Markov Decision Process (MDP), denoted by $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0, \gamma\}$. The MDP consists of a state space $\mathcal{S} \in \mathbb{R}^n$, an action space $\mathcal{A} \in \mathbb{R}^m$, a reward function $\mathcal{R}(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$, transition dynamics $\mathcal{T}(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, an initial state distribution ρ_0 , and a discount factor $\gamma \in [0, 1]$ that determines the importance of future rewards.

The objective in RL is to find a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}_+$ that maximizes the expected sum of rewards

$$\eta(\pi) = \mathbb{E}_{\pi, \mathcal{M}} \left[\sum_t \gamma^t r_t \right], \quad (7.1)$$

where $\mathbb{E}_{\pi, \mathcal{M}}$ denotes the expectation with respect to the policy π and the MDP \mathcal{M} . The policy parameters θ are updated to optimize this expected sum of rewards.

In practice, learning RL policies from scratch can be challenging and sample-intensive. However, by incorporating a set of demonstration data denoted as ρ_D , which contains state-action-state-reward tuples (s_t, a_t, s_{t+1}, r_t) , the sample complexity of RL can significantly be reduced. These demonstrations provide valuable guidance and can be used to improve the learning process.

7.2 Method

7.2.1 Demo Augmented Policy Gradient

The DAPG algorithm addresses the RL problem by combining the principles of Natural Policy Gradient (NPG)s and the use of demonstration data. It consists of two main components: pretraining with Behavior Cloning (BC) and RL fine-tuning with augmented loss.

The pretraining with BC works by enhancing the exploration and providing an informed policy initialization. BC aims to mimic the actions taken in the demonstrations by solving a maximum-likelihood problem. The objective is to maximize the log probability of actions given states

$$\max_{\theta} \sum_{(s,a) \in \rho_D} \log \pi_{\theta}(a|s). \quad (7.2)$$

BC can efficiently guide exploration, reducing the reliance on reward-shaping techniques commonly used to direct exploration in RL.

The process of RL fine-tuning with augmented loss involves effectively utilizing the comprehensive information provided by the demonstrations. Although BC serves as a suitable initialization, it doesn't fully utilize the full information contained within the demonstration data. Notably, different segments of the demonstration data hold value during various stages of learning, especially in tasks that require sequential behaviors. To leverage the complete range of information embedded in the demonstrations, an augmented loss is introduced.

The augmented gradient g_{aug} is defined as the sum of two terms

$$g_{aug} = \sum_{(s,a) \in \rho_{\pi_{\theta}}} \nabla_{\theta} \ln \pi_{\theta}(a|s) A^{\pi_{\theta}}(s,a) + \sum_{(s,a) \in \rho_D} \nabla_{\theta} \ln \pi_{\theta}(a|s) w(s,a). \quad (7.3)$$

The first term considers state-action pairs (s, a) sampled from the occupancy measure ρ_{π} , which represents the probability of encountering a specific state-action tuple. This term is weighted by the advantage function $A^{\pi}(s, a)$, which quantifies the advantage gained by taking action a in state s compared to the expected return from the currently best action. Additionally, the ∇_{θ} is the gradient with regards to θ .

The second term in g_{aug} sums over the state-action pairs (s, a) from the demonstrations dataset ρ_D . The gradient of the logarithm of the policy $\pi_{\theta}(a|s)$ is multiplied by a weighting function $w(s, a)$. This term allows the algorithm to leverage the knowledge provided by the demonstrations to guide the learning process.

The DAPG algorithm updates the policy parameters θ using the NPG and the demonstration-augmented gradient. The parameter update rule involves the Fisher Information Matrix (FIM) and the gradient, as shown in the update rule

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T F_{\theta_k}^{-1} g}} F_{\theta_k}^{-1} g, \quad (7.4)$$

where δ is the step size that determines the magnitude of the update.

The choice of the weighting function $w(s, a)$ in (7.3) is critical. While an ideal choice would be $w(s, a) = A^{\pi}(s, a)$ for $(s, a) \in \rho_D$, computing this quantity exactly requires additional rollouts or assumptions. Therefore, a heuristic weighting scheme is typically used. For example,

$$w(s, a) = \lambda_0 \lambda_1^k \max_{(s_0, a_0) \in \rho_{\pi}} A^{\pi}(s_0, a_0) \quad (7.5)$$

can be used, where λ_0 and λ_1 are hyperparameters, and k is the iteration counter. The decay of the weighting term via λ_1^k ensures that the guidance from demonstrations gradually diminishes as the policy's performance improves.

In summary, the DAPG algorithm combines BC with RL to effectively learn complex behaviors. By utilizing the demonstrations for policy initialization and then fine-tuning the policy using both imitation and reinforcement learning, DAPG benefits from the strengths of both approaches. This combination allows for more efficient and effective learning, reducing the sample complexity and accelerating the convergence of RL algorithms.

7.2.2 Data Collection Procedure

To train DAPG, raw video data of expert demonstrations of the relocation task is collected. While [33] provides data for additional tasks i.e. *pour* and *place inside*, these are not of interest in this chapter. The objects used are from the YCB dataset [145] and the demonstrations were recorded by two RGBD cameras.

The collected data are then parsed to a multi-state pipeline including 3D PE, for estimating the pose of both the object of interest and the hand for demonstration translation.

Object Pose Estimation

The 6-DOF object poses in the physical setup is found using the PVN3D model, which is a DL-based PE model, trained on the YCB data set.

The 6-DOF object pose is afterward optimized by minimizing the Perspective-n-Point (PnP) matching error.

Hand Pose Estimation

The hand pose is found using the DL-based PE model MANO [146], which produces three parameters to represent the hand's pose: θ_t, β_t and r_t . θ_t refers to the 3D rotations of 15 joints, β_t is the shape parameters and r_t is the hand root's global pose [146]. These can be combined through hand kinematics to compute the 3D angle of each hand joint

$$\mathbf{q}_t^{3d} = \mathbf{J}(\theta_t, \beta_t, r_t) \quad (7.6)$$

By utilizing off-the-shelf skin segmentation techniques such as Combinatorial Color Space Models [147] and hand detection models on video data, a hand mask M_t is extracted. M_t is overlayed the depth channel of the recorded data, the center of which is the used estimate of r_t . From the video data, the RGB channels are additionally parsed to a MANO which estimates 2D hand joint angles \mathbf{q}_t^{2D} , along with the hand parameters (θ_t, β_t, r_t) . Having the cameras poses $\mathbf{\Pi}$ enables the formulation of an optimization problem to determine the optimal (θ_t, β_t, r_t) i.e. $(\theta_t^*, \beta_t^*, r_t^*)$

$$\theta_t^*, \beta_t^*, r_t^* = \arg \min_{\theta_t, \beta_t, r_t} \|\mathbf{\Pi}\mathbf{J}(\theta_t, \beta_t, r_t) - \mathbf{q}_t^{2d}\|^2 + \lambda \|M_t (\mathbf{R}(\theta_t, \beta_t, r_t) - D_t)\|^2. \quad (7.7)$$

Here $\lambda = 0.001$ is a hyperparameter, \mathbf{R} is a depth rendering function and D_t is the corresponding depth map in frame t .

Demonstration Translation

Demonstration translation involves transforming the actions performed by the expert agent, with one kinematic structure, to a ML agent with some other set of actions. This is essential as, while the kinematic structure between the biological human hand and the anthropomorphic robot hand is similar, they are not identical. The similarity can be seen in the hands' kinematic trees in Appendix A.1 with their ROMs in Appendix A.1 and Appendix A.2.

Demonstration translation requires two components: hand motion retargeting and Predicting robot action. The hand motion retargeting attempts to align the human hand motion to the robot hand motion, which due to the difference in kinematics is non-trivial as mentioned above. The prediction of robot actions attempts to estimate the torque necessary for each robot motor to recover the action from only PE results.

The hand motion retargeting is solved using Task Space Vector (TSV)s to describe the optimization problem

$$\mathbf{q}_t^* = \min_{\mathbf{q}_t} \sum_{i=1}^N \| \mathbf{v}_i^{hum}(\theta_t, \beta_t, r_t) - \mathbf{v}_i^{rob}(\mathbf{q}_t) \|^2 + \alpha \|\mathbf{q}_t - \mathbf{q}_{t-1}\|^2. \quad (7.8)$$

Here $\mathbf{v}_i^{hum}(\theta_t, \beta_t, r_t)$ computes the i 'th TSV to the fingertip and finger middle joint of the human hand, while the robot's forward kinematics function $\mathbf{v}_i^{rob}(\mathbf{q})$ computes the i 'th TSV to the fingertip and finger middle joint of the robot hand. The TSVs can be seen illustrated in Fig. 7.1 from the hand root (marked with red) to the finger middle joint (marked with blue) and tip (marked with green) on the anthropomorphic robot hand.

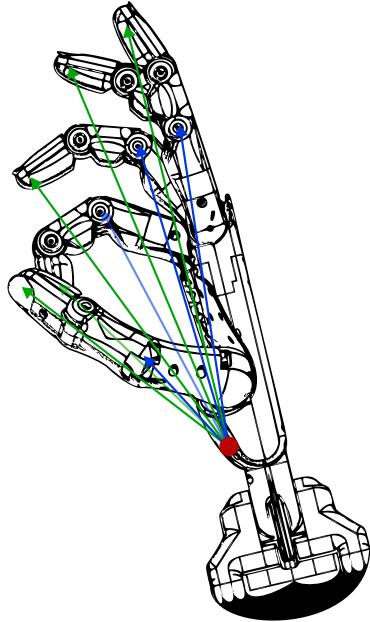


Fig. 7.1: The TSVs from the hand root (marked with red) to the finger middle joint (marked with blue) and tip (marked with green) on the anthropomorphic robot hand.

The optimization problem was solved using Sequential Least-Squares Quadratic Programming (SLSQP) with $\alpha = 8e-3$.

The robot action estimation is achieved through the model $\mathbf{q}(t)$ presented in [148], which fulfills the criteria of approximately zero jerk i.e. $\mathbf{q}'''(t) \approx 0$. The model enables $\mathbf{q}'(t)$ and $\mathbf{q}''(t)$ which through inverse dynamics can be converted to torques τ

$$\tau = f_{inv}(\mathbf{q}(t), \mathbf{q}'(t), \mathbf{q}''(t)) \quad (7.9)$$

The zero jerk here enables behavior similar to that of a human, which in turn leads to more natural robot motion, as well as ensuring low joint position errors for motors and limiting excessive wear on the physical robot.

7.3 Experimental Setup

To train and test the DAPG algorithm, the mug shown in Fig. 7.1 is chosen as the prop from YCB. Additionally, to remain consistent with the original paper [33] the number of training iterations was set to 2000 with demonstration hyperparameters λ_0 and λ_1 being 0.1 and 0.99 respectively. Furthermore, the learning rate α_{lr} was set to 1.0, the step size δ was set to 0.1 and the discount factor γ was set to 0.995. This however will result in the same performance, and thus to test the consistency of DAPG, a different seed is used for randomization. The seed used in the authors' paper [33] is 200, while the one chosen for this project is 201.

To execute the relocation task, the object is placed on a table and a goal is defined at a random location marked with green silhouette. The experimental setup can be seen in Fig. 7.2.



Table 7.1: The mug prop from YCB [145] chosen for training.

Fig. 7.2: The experimental setup used when training the DAPG algorithm. The figure contains the table platform, the mug prop, the simulated anthropomorphic hand and the goal mug position marked as a green silhouette.

7.4 Results

The evaluation score of the RL agent on a separate set of test episodes can be seen in Fig. 7.3, the running score obtained by the RL agent during training can be seen in Fig. 7.4, and finally, the expected return used in policy i.e. the improvement or increase in the surrogate objective function during the policy update can be seen in figure Fig. 7.5.

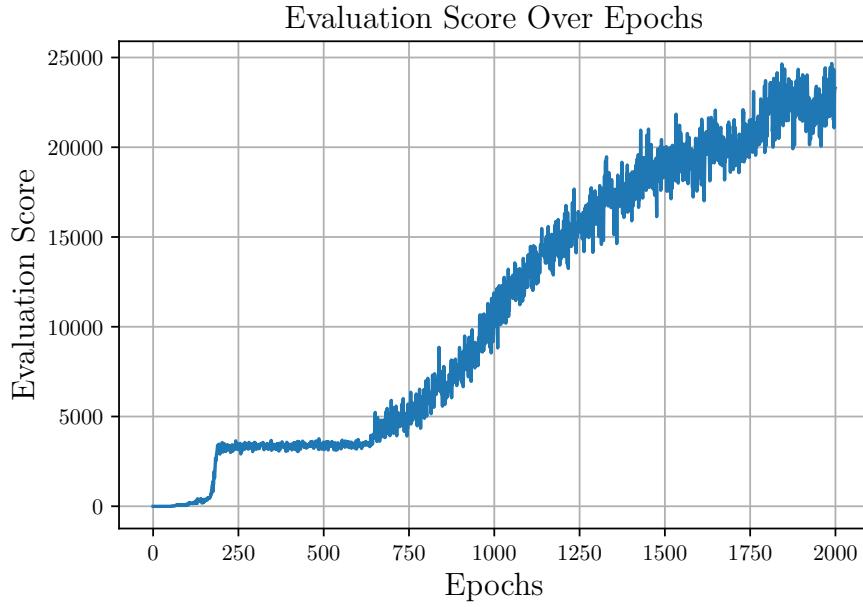


Fig. 7.3: The evaluation score of the DAPG agent on a separate set of test episodes or environments.

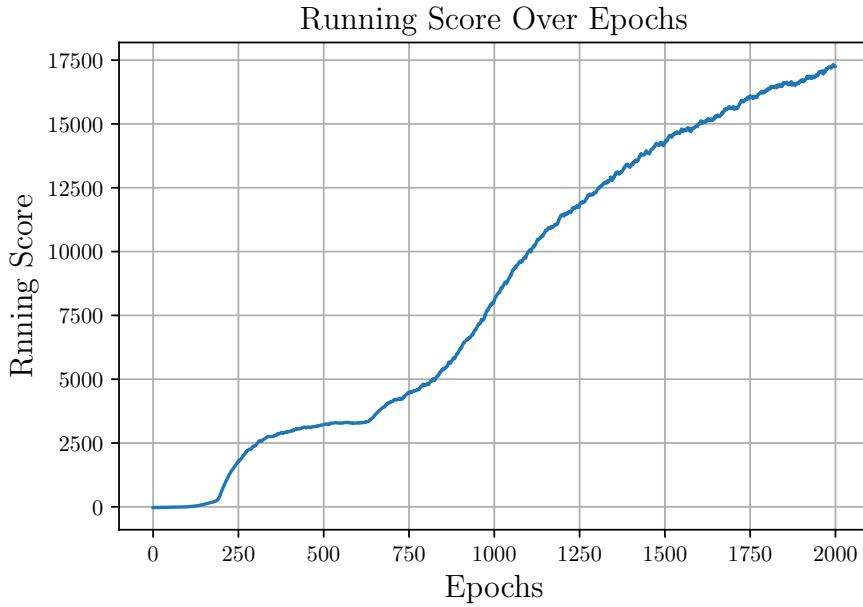


Fig. 7.4: The running score shows the cumulative reward obtained by the DAPG agent during training.

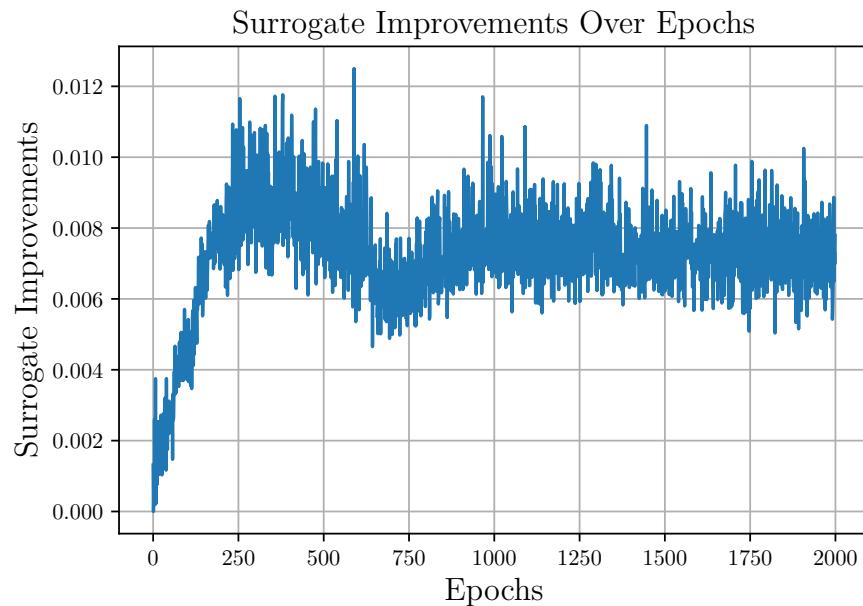


Fig. 7.5: The increase in the surrogate objective function during the policy update.

Once trained, the best policy was saved and the reward was sampled over 100 iterations, both for the model trained and the one provided by the authors resulting in Fig. 7.6, showing the probability distributions for each algorithm's best policy rewards.

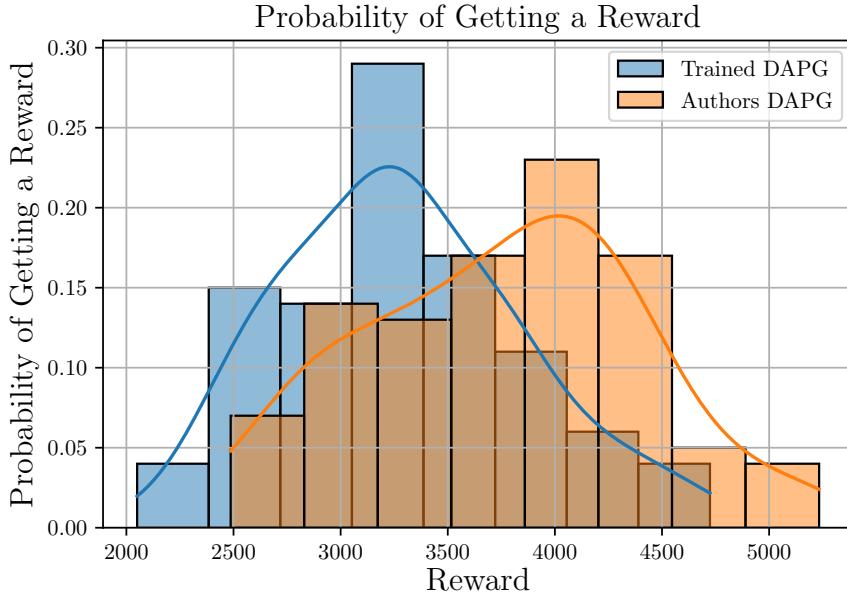


Fig. 7.6: Histograms showing the probability distributions of getting certain rewards over 100 iterations for both the authors' and the trained DAPG algorithms.

The mean of each distribution is found to be $\mu_{trained} \approx 3276.1$ and $\mu_{authors} \approx 3759.4$.

Fig. 7.7 show the three stages throughout the PNP process of the relocation task as executed when the distributions were sampled. The execution of the relocation task can be seen for both the trained and the authors' agent in the supplementary material as `trained-dapg.gif` and `authors-dapg.gif` respectively.



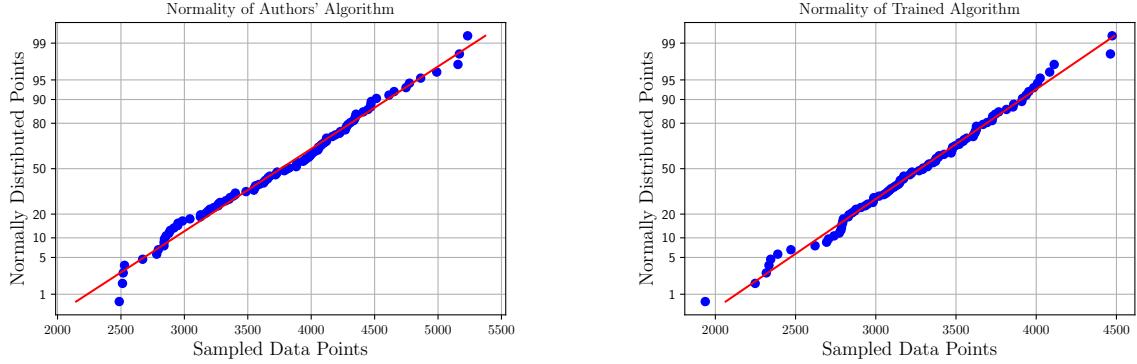
(a) The pre-grasping phase, where the AH is moving towards the mug. (b) The grasping phase, where a force closure grasp is made to hold the mug. (c) The placing phase, where the mug is placed onto the silhouette.

Fig. 7.7: The PNP phases executed by the trained DAPG in MuJoCo

To compare the trained algorithm to the one provided by the authors, it is of interest to test if the reward distributions resulting from experiments come from the same underlying distribution. To determine this a Analysis of Variance (ANOVA) test is applied, which requires the presence of normality and homoscedasticity. The normality of each algorithm's performance is tested using normal plots as shown in Fig. 7.8(a). Based on these plots, normality is assumed.

When testing for homoscedasticity the Bartlett test is applied, resulting in the p-value $p_{bart} \approx 0.0044$. Due to the common threshold for rejecting homoscedasticity being 0.05, which here is considered sufficiently close to 0.0044 to attempt a One-Way ANOVA test.

The One-Way ANOVA resulted in a p-value of $p_{ANOVA} \approx 1.13 \cdot 10^{-8}$.



(a) Normal plot, showing the extent to which the authors DAPG algorithm gets rewards that are normally distributed.

(b) Normal plot, showing the extent to which the trained DAPG algorithm gets rewards that are normally distributed.

Fig. 7.8: Normal plots showing the extent to which the performance of the DAPG algorithms tested in this chapter is normally distributed.

7.5 Discussion & Conclusion

The findings of this chapter provide strong evidence that the DAPG algorithm successfully learns from demonstrations and demonstrates its capability to perform the dexterous manipulation task of relocation. The trained algorithm exhibits promising results, with outcomes that follow a distribution resembling a normal distribution.

However, upon closer analysis, it becomes apparent that the distribution of the algorithm's performance does not strictly adhere to the assumption of homoscedasticity. Homoscedasticity assumes that the variances of the performance values are equal across the distribution. In this case, the observed departure from homoscedasticity suggests the presence of variability or heterogeneity in the algorithm's performance across different relocation tasks. Even so, a One-Way ANOVA test was performed which concluded that the reward probability distributions showed statistically significant evidence for having different means. Due to $\mu_{authors} > \mu_{trained}$, it can be concluded that the authors-trained method performed statistically significantly better than the one trained in this project.

As both agents were trained using the same setup but with different seeds, the result would indicate the randomization being the cause of this difference.

In summary, the chapter highlights the successful learning capabilities of the DAPG algorithm from demonstrations, specifically in the context of the dexterous manipulation task of relocation. The outcomes generated by the trained algorithm exhibit a distribution that resembles a normal distribution, although it deviates from strict homoscedasticity. By conducting further analysis and comparing underlying distributions, it was determined that the agent trained by the authors performs statistically significantly better than the alternative approaches.

Chapter 8

Discussion & Conclusion

This project set out to develop an alternative pipeline for the PNP problem, which uses tactile inputs rather than visual. To design this pipeline, three subproblems were identified

1. To model the contact between the gripper's tactile sensors and the object, also referred to as TP, such that useful data can be extracted for pose estimation.
2. To convert the collected data from problem 1 to estimated pose candidates.
3. To perform in-hand manipulation.

In the TP estimation chapter, the focus was on estimating contact positions, contact normals, and skew forces. Successful estimation was achieved using simulated contact points and the grasping matrix obtained from Ruppel et al. [93]. This approach proved effective in estimating contact points accurately. The estimation of contact normals was accomplished by utilizing estimated linear velocities from simulated contact points and applying the RLS method. The results obtained on flat surfaces demonstrated low error rates and were considered satisfactory. However, due to time constraints, contact normals were not estimated on surfaces other than flat ones, which limited the validation of the technique's applicability. In these cases, the Gazebo physics engine provided acceptable results and served as a substitute for normal estimation. The performance of Gazebo's physics engine was deemed satisfactory. Estimating contact normals offers a significant advantage as it enables the identification of robust surface features. This, in turn, minimizes the search space for the pose estimation algorithm discussed in a previous chapter. It was observed that more complex surface features, such as corners, resulted in higher angle errors, which aligned with initial expectations.

The findings in this chapter additionally highlight the limitations and challenges associated with simulating realistic tactile information using the chosen DL model. The DL model did not provide useful information for tactile force sensing due to inaccurate simulation of electrode activations. This emphasizes the need for careful evaluation of DL models in specific applications, as they may not always deliver the expected benefits. The project was unable to replicate the tactile information from the original paper, indicating possible reasons such as non-representative weights and potential differences in Gazebo's API. Retraining the DL model with a legitimate dataset could address these issues, but it was not feasible within the project's time constraints. Instead, the project relied on the forces and torques generated by Gazebo's physics engine, which were considered satisfactory based on the presented force and torque cones. While the DL model did not meet expectations, the project managed to find an alternative solution to approximate tactile information for the given constraints.

The PE chapter focused on estimating the pose of a known object, specifically the Stanford bunny, using synthetic and sampled source data. The problem was effectively formalized for both cases and the core methodologies, namely GNC and RCQP, were presented along with the experimental setups. Synthetic source data proved to be successful in estimating the object's pose, meeting the predefined success criteria of angle errors below 5° and positional errors below 1 cm with an outlier ratio of 10 %. However, when dealing with sampled data, the methods faced challenges due to the sparsity of strong features, which hindered the identification of sufficient feature correspondences. This limitation indicated the need for intelligent probing techniques to enhance feature detection through a great pool of data. The execution times and number of iterations were consistent with the trends observed in the original paper. Additionally, a weight convergence analysis revealed an expected trend of convergence towards zero and one, but further investigation is required to fully understand the specific ratio of weights converging to zero or one. Despite

these challenges, the GNC+RCQP method exhibited promising performance in PE for synthetic data, highlighting its potential for robust and accurate estimation.

In the dexterous manipulation chapter, the focus shifted to the DAPG algorithm and its capabilities for learning from demonstrations in the context of dexterous manipulation tasks, specifically the relocation task. The findings provided strong evidence that the DAPG algorithm successfully learned from demonstrations and demonstrated its capability to perform the relocation task. The trained algorithm exhibited promising results, and the outcomes followed a distribution that resembled a normal distribution. However, upon closer analysis, it was discovered that the assumption of homoscedasticity, which assumes equal variances across the performance values, was not strictly adhered to. This departure from homoscedasticity indicated the presence of variability or heterogeneity in the algorithm's performance across different relocation tasks. Additional investigation was conducted to compare the underlying distributions, which revealed a statistically significant difference between the agent trained by the authors and the alternative approaches. This significant difference demonstrated that the agent trained by the authors outperformed the alternative in terms of its ability to successfully accomplish the relocation task.

In summary, the project provided valuable insights into the limitations and potential of DL models in tactile perception, PE, and dexterous manipulation. It highlighted the importance of considering alternative methods, such as physics engine simulations, to supplement or replace DL models when realistic tactile information is challenging to simulate. Improving feature detection and data sampling techniques is crucial for enhancing the accuracy of PE, particularly when working with sampled data. The DAPG algorithm demonstrated effective learning capabilities for dexterous manipulation tasks, although the presence of variability in performance across different randomization seeds needs to be considered.

These findings contribute to the ongoing development of robust and efficient algorithms for tactile perception and manipulation tasks. Future studies could explore different DL architectures or combinations of methods to further improve accuracy and usefulness in estimating tactile perception. One potential approach is to combine the solutions for problem 1 and problem 2 using time-dependent state estimation methods like the Kalman filter, which would involve continuously collecting tactile sensor data and estimating the object pose in a loop. By incorporating the time dependency of the system, this approach would have the potential to enhance PE convergence and optimize computation time for real-time performance.

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](https://doi.org/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](https://doi.org/10.1038/nature14539). url: <https://doi.org/10.1038/nature14539>.
- [5] Monkman, Gareth J. et al. “Robot Grippers”. PhD thesis. Jan. 2004, pp. 5–6.
- [6] Mihelj, Matjaž et al. “Robotics Second Edition”. PhD thesis. Jan. 2019, p. 1.
- [7] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022, p. 284.
- [8] Quigley, Morgan et al. “ROS: an open-source Robot Operating System”. In: *ICRA Workshop on Open Source Software* 3 (Jan. 2009).
- [9] 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>.
- [10] 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](https://doi.org/10.1016/j.proeng.2011.11.2415).
- [11] 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](https://doi.org/10.1088/1757-899X/862/3/032075).
- [12] 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](https://doi.org/10.1109/CASE48305.2020.9216746).
- [13] 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](https://arxiv.org/abs/1203.4475). url: <https://arxiv.org/abs/1203.4475>.
- [14] 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](https://doi.org/10.1109/SII46433.2020.9026282).
- [15] 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](https://doi.org/10.1109/BigComp48618.2020.00015).
- [16] 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](https://doi.org/10.1109/LRA.2020.2965076).
- [17] 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](https://doi.org/10.1109/ACCESS.2022.3182114).

- [18] 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](https://doi.org/10.3390/app11010228). url: <https://www.mdpi.com/2076-3417/11/1/228>.
- [19] ., 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](https://doi.org/10.1109/ICASI.2017.7988217).
- [20] 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](https://doi.org/10.1007/s12555-012-0216-9). url: <https://doi.org/10.1007/s12555-012-0216-9>.
- [21] 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>.
- [22] 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](https://arxiv.org/abs/2005.13759). url: <https://arxiv.org/abs/2005.13759>.
- [23] 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](https://doi.org/10.3390/s21186093). url: <https://www.mdpi.com/1424-8220/21/18/6093>.
- [24] 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](https://doi.org/10.3390/s20236790). url: <https://www.mdpi.com/1424-8220/20/23/6790>.
- [25] Straszheim, Troy et al. *Shadow Robot*. url: <https://www.shadowrobot.com/dexterous-hand-series/>. (accessed: 29.07.2022).
- [26] 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](https://doi.org/10.1109/IROS.2004.1389727).
- [27] Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. doi: [10.1109/IROS.2012.6386109](https://doi.org/10.1109/IROS.2012.6386109).
- [28] Robotics, Shadow. *Shadow Robotics*. url: <https://www.shadowrobot.com/>.
- [29] Robotics, Shadow. *The Shadow Robot Company*. url: <https://github.com/shadow-robot>.
- [30] docker. *What is a container*. url: <https://www.docker.com/resources/what-container/>.
- [31] Lopez, Patricio Rivera et al. “Dexterous Object Manipulation with an Anthropomorphic Robot Hand via Natural Hand Pose Transformer and Deep Reinforcement Learning”. In: *Applied Sciences* 13.1 (2023). issn: 2076-3417. doi: [10.3390/app13010379](https://doi.org/10.3390/app13010379). url: <https://www.mdpi.com/2076-3417/13/1/379>.
- [32] Charlesworth, Henry and Montana, Giovanni. *Solving Challenging Dexterous Manipulation Tasks With Trajectory Optimisation and Reinforcement Learning*. 2021. arXiv: [2009.05104 \[cs.R0\]](https://arxiv.org/abs/2009.05104).
- [33] Qin, Yuzhe et al. *DexMV: Imitation Learning for Dexterous Manipulation from Human Videos*. 2021.
- [34] Dimensions. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_dimensions.html. Accessed: 2023-04-26.
- [35] Rahman, Akhlaqur 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](https://doi.org/10.5772/56809). eprint: <https://doi.org/10.5772/56809>. url: <https://doi.org/10.5772/56809>.

- [36] *Bones Of The Hand*. <https://freesvg.org/bones-of-the-hand>. Accessed: 2023-04-26.
- [37] Robotics, Shadow. *Gazebo*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sim_gazebo.html%7D.
- [38] *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.
- [39] Straszheim, Troy et al. *Conceptual overview of ROS catkin*. URL: %5Curl%7Bhttp://wiki.ros.org/catkin/conceptual_overview%7D
- [40] Beek, Bas van et al. *About Us*. URL: %5Curl%7B<https://numpy.org/%7D>
- [41] team, OpenCV. *OpenCV*. URL: %5Curl%7B<https://opencv.org/%7D>
- [42] Carroll, Michael. *dynamic-reconfigure*. URL: %5Curl%7B<http://google.com%7D>
- [43] Coleman, David et al. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014. arXiv: 1404.3785 [cs.RO].
- [44] Sucan, 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](https://doi.org/10.1109/MRA.2012.2205651).
- [45] 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](https://doi.org/10.21105/joss.00456). URL: <http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf>.
- [46] Robotics, Shadow. *Control Description*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/ed_control_description.html%7D
- [47] Robotics, Shadow. *Firmware*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sd_firmware.html%7D
- [48] Robotics, Shadow. *Controlling the Hand 2*. URL: %5Curl%7Bhttps://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/sd_controlling_hand.html%7D
- [49] Rajeswaran, Aravind et al. “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations”. In: *arXiv preprint arXiv:1709.10087* (2017).
- [50] Staven, Victor Melbye. *ros_utils*. URL: %5Curl%7Bhttps://github.com/vmstavens/ros_utils%7D
- [51] Staven, Victor Melbye. *in_hand_pose_estimation*. URL: %5Curl%7Bhttps://github.com/vmstavens/in_hand_pose_estimation%7D
- [52] 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>.
- [53] Hertz, H. “On the Contact of Rigid Elastic Solids and on Hardness”. In: *Ch 6: Assorted Papers* (1882).
- [54] Bruno Siciliano, Oussama Khatib, ed. *Springer Handbook of Robotics*. Springer Berlin, Heidelberg, 2016.
- [55] 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](https://doi.org/10.1177/02783649922066673). eprint: <https://doi.org/10.1177/02783649922066673>. URL: <https://doi.org/10.1177/02783649922066673>.

- [56] Yuvaraj, S., Malayalamurthi, R., and Raja, K. Venkatesh. “The haptic and perceptual characteristics of an anthropomorphic curved soft finger structure”. In: *Curved and Layered Structures* 6.1 (2019), pp. 161–168. doi: [10.1515/cls-2019-0013](https://doi.org/10.1515/cls-2019-0013). URL: <https://doi.org/10.1515/cls-2019-0013>.
- [57] 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](https://doi.org/10.1109/ROBOT.1988.12032).
- [58] Flugge, Wilhelm. *Viscoelasticity*. Blaisdell Publishing Company, Waltham, MA, 1967.
- [59] Maxwell, James Clerk. *On the dynamical theory of gases*. Royal Society, 1867.
- [60] Fung, Yuan-Cheng. “Mechanical properties and active remodeling of blood vessels”. In: *Biomechanics*. Springer, 1993.
- [61] 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/TR.2006.889494](https://doi.org/10.1109/TR.2006.889494).
- [62] 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](https://doi.org/10.1109/ROBOT.2006.1641899).
- [63] 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](https://doi.org/10.1163/156855307782227390).
- [64] 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-.
- [65] 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>.
- [66] 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](https://doi.org/10.1007/s10659-015-9534-5). URL: <https://doi.org/10.1007/s10659-015-9534-5>.
- [67] 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](https://doi.org/10.1023/A:1014013425423). URL: <https://doi.org/10.1023/A:1014013425423>.
- [68] 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](https://doi.org/10.3390/s19040814). URL: <https://www.mdpi.com/1424-8220/19/4/814>.
- [69] Slaughter, William S. *The Linearized Theory of Elasticity*. Springer, 2012.
- [70] 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](https://doi.org/10.1177/0954406216634121).

- [71] 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](https://doi.org/10.1016/S0022-460X(05)80001-X). URL: <https://www.sciencedirect.com/science/article/pii/S0022460X0580001X>.
- [72] 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](https://doi.org/10.1016/0022-460X(87)90469-X). URL: <https://www.sciencedirect.com/science/article/pii/0022460X8790469X>.
- [73] 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](https://doi.org/10.1016/S0021-9290(03)00176-3). URL: <https://www.sciencedirect.com/science/article/pii/S0021929003001763>.
- [74] 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: [10.1177/027836499201100102](https://doi.org/10.1177/027836499201100102). eprint: <https://doi.org/10.1177/027836499201100102>. URL: <https://doi.org/10.1177/027836499201100102>.
- [75] 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: [10.1177/027836499601500603](https://doi.org/10.1177/027836499601500603). eprint: <https://doi.org/10.1177/027836499601500603>. URL: <https://doi.org/10.1177/027836499601500603>.
- [76] 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](https://doi.org/10.1109/TRA.2003.820868).
- [77] 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](https://doi.org/10.1109/WHC.2007.103).
- [78] Sabat, Lovely and Kundu, Chinmay Kumar. “History of Finite Element Method: A Review”. In: *Recent Developments in Sustainable Infrastructure*. Ed. by Das, Bibhuti Bhushan et al. Singapore: Springer Singapore, 2021, pp. 395–404. isbn: 978-981-15-4577-1.
- [79] 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](https://doi.org/10.1016/S0924-0136(01)01210-9). URL: <https://www.sciencedirect.com/science/article/pii/S0924013601012109>.
- [80] 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](https://doi.org/10.1243/0954409011531404). eprint: <https://doi.org/10.1243/0954409011531404>. URL: <https://doi.org/10.1243/0954409011531404>.
- [81] 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](https://doi.org/10.1016/S0301-679X(99)00012-2). URL: <https://www.sciencedirect.com/science/article/pii/S0301679X99000122>.

- [82] 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](https://doi.org/10.1016/S0045-7949(99)00212-6). URL: <https://www.sciencedirect.com/science/article/pii/S0045794999002126>.
- [83] 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](https://doi.org/10.1109/TIA.2019.2958908).
- [84] 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](https://doi.org/10.1109/LRA.2022.3155225).
- [85] 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](https://doi.org/10.3390/s22218390). URL: <https://www.mdpi.com/1424-8220/22/21/8390>.
- [86] 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](https://doi.org/10.1109/IROS.2005.1545525).
- [87] 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](https://arxiv.org/abs/2103.16747). URL: <https://arxiv.org/abs/2103.16747>.
- [88] 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](https://arxiv.org/abs/2006.03777). URL: <https://arxiv.org/abs/2006.03777>.
- [89] 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](https://doi.org/10.1109/ACCESS.2019.2956882).
- [90] 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](https://arxiv.org/abs/1805.11085). URL: [http://arxiv.org/abs/1805.11085](https://arxiv.org/abs/1805.11085).
- [91] 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](https://doi.org/10.1109/TOH.2016.2521378).
- [92] Higuera, Carolina, Boots, Byron, and Mukadam, Mustafa. *Learning to Read Braille: Bridging the Tactile Reality Gap with Diffusion Models*. 2023. arXiv: [2304.01182](https://arxiv.org/abs/2304.01182) [cs.R0].
- [93] Ruppel, Philipp et al. “Simulation of the SynTouch BioTac Sensor”. In: (2019). Ed. by Strand, Marcus et al., pp. 374–387.
- [94] Huang, Xiaoshui et al. *A comprehensive survey on point cloud registration*. 2021. arXiv: [2103.02690](https://arxiv.org/abs/2103.02690) [cs.CV].
- [95] Oberweger, Markus, Wohlhart, Paul, and Lepetit, Vincent. *Hands Deep in Deep Learning for Hand Pose Estimation*. 2016. arXiv: [1502.06807](https://arxiv.org/abs/1502.06807) [cs.CV].
- [96] Toshev, Alexander and Szegedy, Christian. “DeepPose: Human Pose Estimation via Deep Neural Networks”. In: *Computer Vision and Pattern Recognition*. 2014.
- [97] 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](https://doi.org/10.1038/s41593-018-0209-y). URL: <https://doi.org/10.1038/s41593-018-0209-y>.

- [98] 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](https://doi.org/10.1109/TIP.2017.2695888).
- [99] 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](https://doi.org/10.1109/TCSVT.2017.2730232).
- [100] 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](https://doi.org/10.1109/CVPR.2019.00466).
- [101] Wang, Lingjing et al. *Non-Rigid Point Set Registration Networks*. 2019. arXiv: [1904.01428](https://arxiv.org/abs/1904.01428) [cs.GR].
- [102] 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](https://arxiv.org/abs/2005.01014) [cs.CV].
- [103] Zeng, Andy et al. *3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions*. 2017. arXiv: [1603.08182](https://arxiv.org/abs/1603.08182) [cs.CV].
- [104] Deng, Haowen, Birdal, Tolga, and Ilic, Slobodan. *PPFNet: Global Context Aware Local Features for Robust 3D Point Matching*. 2018. arXiv: [1802.02669](https://arxiv.org/abs/1802.02669) [cs.CV].
- [105] Gojcic, Zan et al. *The Perfect Match: 3D Point Cloud Matching with Smoothed Densities*. 2019. arXiv: [1811.06879](https://arxiv.org/abs/1811.06879) [cs.CV].
- [106] 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.
- [107] 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](https://doi.org/10.1016/0262-8856(92)90066-C). URL: <https://www.sciencedirect.com/science/article/pii/026288569290066C>.
- [108] Segal, Aleksandr, Hähnel, Dirk, and Thrun, Sebastian. “Generalized-ICP”. In: June 2009. doi: [10.15607/RSS.2009.V.021](https://doi.org/10.15607/RSS.2009.V.021).
- [109] 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](https://doi.org/10.1080/01431161.2019.1701211). eprint: <https://doi.org/10.1080/01431161.2019.1701211>. URL: <https://doi.org/10.1080/01431161.2019.1701211>.
- [110] Zhu, Hao et al. “A Review of Point Set Registration: From Pairwise Registration to Groupwise Registration”. en. In: *Sensors (Basel)* 19.5 (Mar. 2019).
- [111] 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](https://doi.org/10.1007/s10044-012-0284-8). URL: <https://doi.org/10.1007/s10044-012-0284-8>.
- [112] 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](https://doi.org/10.1109/TPAMI.2015.2501802).
- [113] 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.
- [114] 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](https://doi.org/10.1109/CVPR.2008.4587500).

- [115] 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](https://doi.org/10.1109/TPAMI.2011.110).
- [116] 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](https://doi.org/10.1109/TCYB.2019.2936176).
- [117] Müller, Meinard. *Information Retrieval for Music and Motion*. Jan. 2007. ISBN: 978-3-540-74047-6. doi: [10.1007/978-3-540-74048-3](https://doi.org/10.1007/978-3-540-74048-3).
- [118] 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>.
- [119] 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.
- [120] Yuan, Wentao et al. *DeepGMR: Learning Latent Gaussian Mixture Models for Registration*. 2020. arXiv: [2008.09088 \[cs.CV\]](https://arxiv.org/abs/2008.09088).
- [121] 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](https://doi.org/10.1109/TRO.2020.3033695).
- [122] Sun, Lei. *IRON: Invariant-based Highly Robust Point Cloud Registration*. 2021. arXiv: [2103.04357 \[cs.CV\]](https://arxiv.org/abs/2103.04357).
- [123] 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.
- [124] 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](https://doi.org/10.1109/LRA.2020.2965893). URL: <https://doi.org/10.1109/LRA.2020.2965893>.
- [125] Li, Yanbo, Littlefield, Zakary, and Bekris, Kostas E. *Asymptotically Optimal Sampling-based Kinodynamic Planning*. 2016. arXiv: [1407.2896 \[cs.RO\]](https://arxiv.org/abs/1407.2896).
- [126] LaValle, Steven M. *Planning Algorithms*. Cambridge University Press, 2006. doi: [10.1017/CBO9780511546877](https://doi.org/10.1017/CBO9780511546877).
- [127] 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](https://arxiv.org/abs/2101.06798). URL: <https://arxiv.org/abs/2101.06798>.
- [128] Khandate, Gagan et al. *Sampling-based Exploration for Reinforcement Learning of Dexterous Manipulation*. 2023. arXiv: [2303.03486 \[cs.RO\]](https://arxiv.org/abs/2303.03486).
- [129] 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](https://doi.org/10.1177/027836498200100102). eprint: <https://doi.org/10.1177/027836498200100102>. URL: <https://doi.org/10.1177/027836498200100102>.
- [130] Jonker, Ben, Waiboe, 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.
- [131] Lynch, Kevin M and Park, Frank C. *Modern Robotics: Mechanics, Planning, and Control*. English (US). Cambridge University Press, 2017. ISBN: 978-1107156302.
- [132] 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](https://doi.org/10.1007/978-1-4614-1945-7_3).

- [133] Gazebo. *Control Description*. URL: %5Curl%7Bhttps://classic.gazebosim.org/tutorials?tut=plugins_hello_world&cat=write_plugin%7D.
- [134] Vries, Maarten de. *Switch to libb64 for base64 encoding/decoding*. URL: %5Curl%7Bhttps://github.com/ros/ros_comm/pull/1046%7D.
- [135] Görner, Michael and Ruppel, Philipp. *biotac_gazebo_plugin*. Version 1.0.0. URL: https://github.com/TAMS-Group/biotac_gazebo_plugin.
- [136] Melbye Staven, Victor. *biotac_sim_plugin*. Version 1.0.0. URL: https://github.com/vmstavens/biotac_sim_plugin.
- [137] Fishel, J.A. et al. *BioTac Product Manual*. English. Version Version 21. SynTouch. August 2018.
- [138] 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.
- [139] 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.
- [140] 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.
- [141] University, Stanford. *The Stanford 3D Scanning Repository*. http://graphics.stanford.edu/data/3Dscanrep/.
- [142] Carlone, Luca et al. *Lagrangian Duality in 3D SLAM: Verification Techniques and Optimal Solutions*. 2015. arXiv: 1506.00746 [cs.RO].
- [143] Carlone, Luca et al. “Planar Pose Graph Optimization: Duality, Optimal Solutions, and Verification”. In: *IEEE Transactions on Robotics* 32 (May 2016), pp. 1–21. doi: 10.1109/TRO.2016.2544304.
- [144] Nesterov, Yuri, Wolkowicz, Henry, and Ye, Yinyu. “Semidefinite Programming Relaxations of Nonconvex Quadratic Optimization”. In: *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Ed. by Wolkowicz, Henry, Saigal, Romesh, and Vandenberghe, Lieven. Boston, MA: Springer US, 2000, pp. 361–419. ISBN: 978-1-4615-4381-7. doi: 10.1007/978-1-4615-4381-7_13. URL: https://doi.org/10.1007/978-1-4615-4381-7_13.
- [145] Çalli, Berk et al. “Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols”. In: *CoRR* abs/1502.03143 (2015). arXiv: 1502.03143. URL: http://arxiv.org/abs/1502.03143.
- [146] Romero, Javier, Tzionas, Dimitrios, and Black, Michael J. “Embodied hands”. In: *ACM Transactions on Graphics* 36.6 (Nov. 2017), pp. 1–17. doi: 10.1145/3130800.3130883. URL: https://doi.org/10.1145%2F3130800.3130883.
- [147] Shah, Khaled et al. “Combinatorial Color Space Models for Skin Detection in Sub-continental Human Images”. In: Nov. 2009, pp. 532–542. ISBN: 978-3-642-05035-0. doi: 10.1007/978-3-642-05036-7_50.
- [148] Todorov, Emanuel and Jordan, Michael I. “Smoothness Maximization Along a Predefined Path Accurately Predicts the Speed Profiles of Complex Arm Movements”. In: *Journal of Neurophysiology* 80.2 (1998). PMID: 9705462, pp. 696–714. doi: 10.1152/jn.1998.80.2.696. eprint: https://doi.org/10.1152/jn.1998.80.2.696. URL: https://doi.org/10.1152/jn.1998.80.2.696.
- [149] *Ranges*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_ranges.html. Accessed: 2023-04-26.

- [150] 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.
- [151] Palmer, A K et al. “Functional wrist motion: a biomechanical study”. en. In: *J Hand Surg Am* 10.1 (Jan. 1985), pp. 39–46.
- [152] *Finger*. https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_finger.html. Accessed: 2023-04-26.
- [153] Mavrogiannis, Christoforos. “Grasp Synthesis Algorithms for Multifingered Robot Hands”. PhD thesis. Mar. 2013.
- [154] *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 SDH and Table A.2 show 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 - SDH					
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 ROM for each joint in the SDH [149].

ROM - Human Hand					
Joint(s)	Min deg	Max deg	Min rad	Max rad	Latin Name
TH1	-15	80	-0.262	1.396	Interphalangeal (IP)
TH2 + TH3	-10	55	-0.175	0.96	Metacarpophalangeal (MCP)
TH4 + TH5	-10	55	-0.175	0.96	Carpometacarpal (CMC)
FF1, MF1, RF1, LF1	0	80	0.0	1.396	Distal interphalangeal (DIP)
FF2, MF2, RF2, LF2	0	100	0.0	1.745	Proximal interphalangeal (PIP)
FF3, MF3, RF3, LF3	-45	90	-0.785	1.571	Metacarpophalangeal (MCP)
WR1	-80	80	-1.396	1.396	Radiocarpal
WR2	-28	20	-0.489	0.349	Radiocarpal

Table A.2: The theoretical ROM for each finger joint in human hand [150] and found ROM for the wrist joints [151].

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 [152].

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

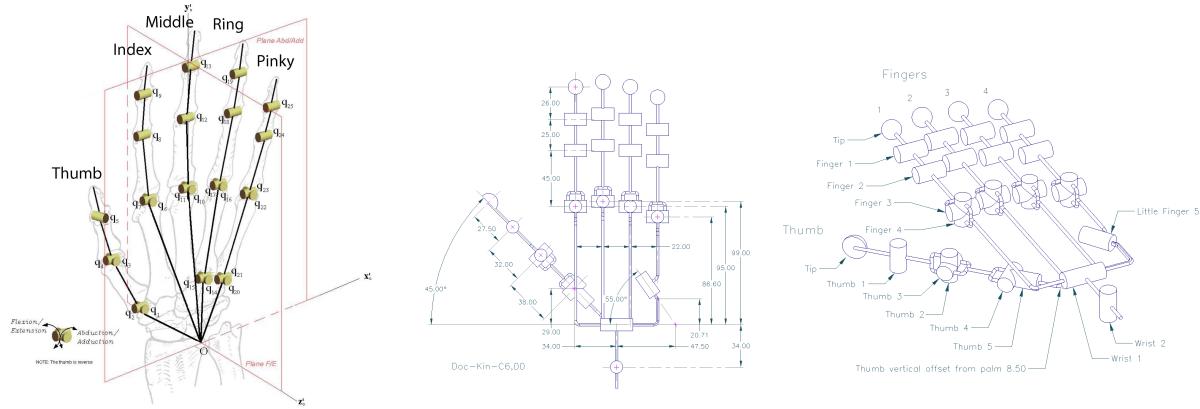


Fig. A.1: The kinematic trees of a human hand and the SDH.

Appendix B

Tactile Perception - Simulated Electrode Activations

Below three figures are found showing the DL model inputs and outputs as described in 5.1 Methods. Fig. B.1 shows the input and output graphs for the case when the SDH index finger is in contact with an edge. Fig. B.2 shows the same graph but for contact with a smooth surface and Fig. B.3 shows the graph but for contact with a corner.

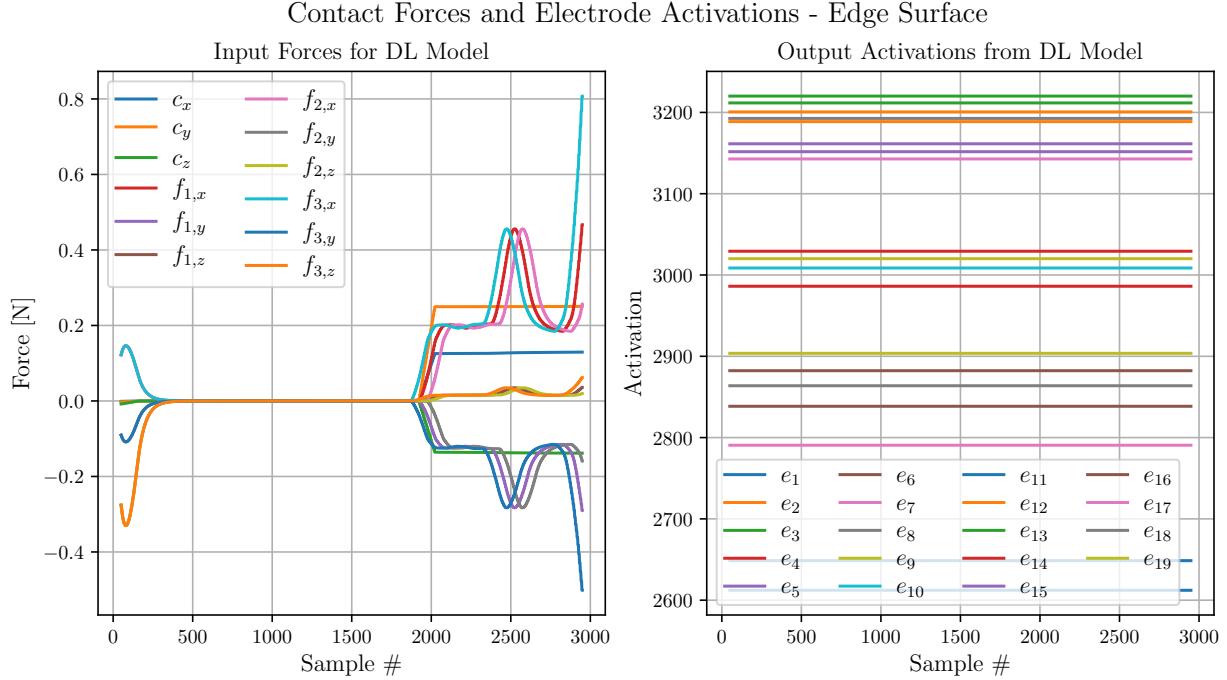


Fig. B.1: The simulated tactile electrode activations when the simulated SDH's index finger is in contact with an edge.

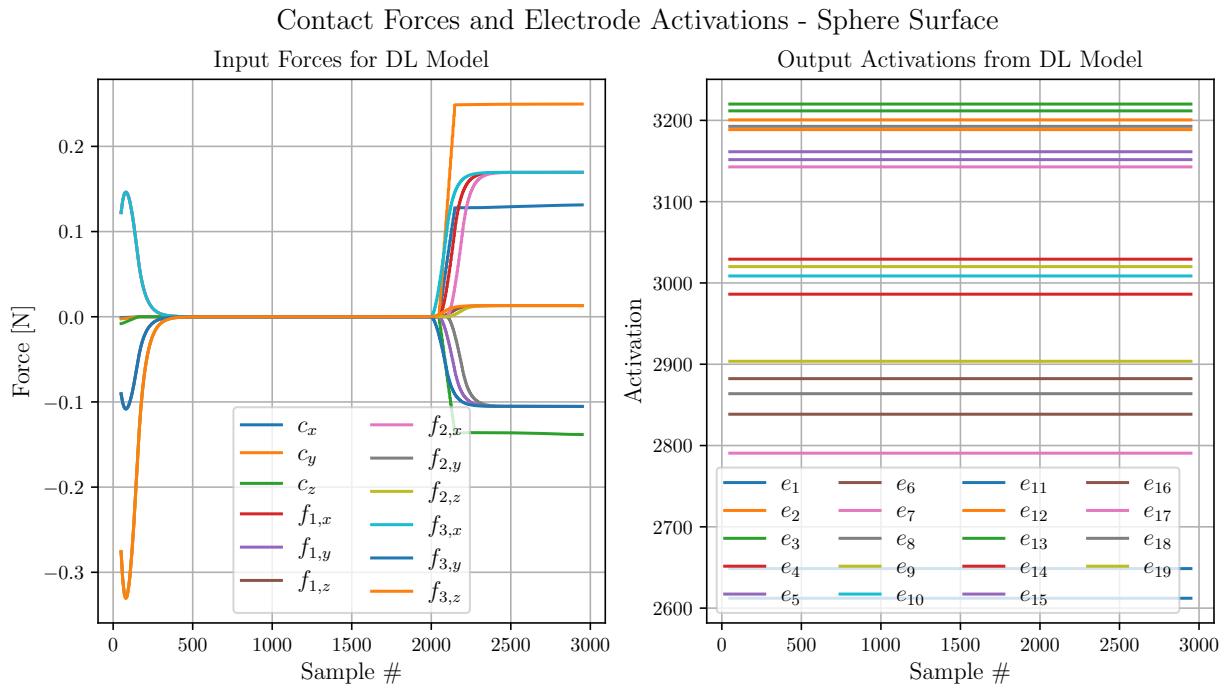


Fig. B.2: The simulated tactile electrode activations when the simulated SDH's index finger is in contact with a smooth surface.

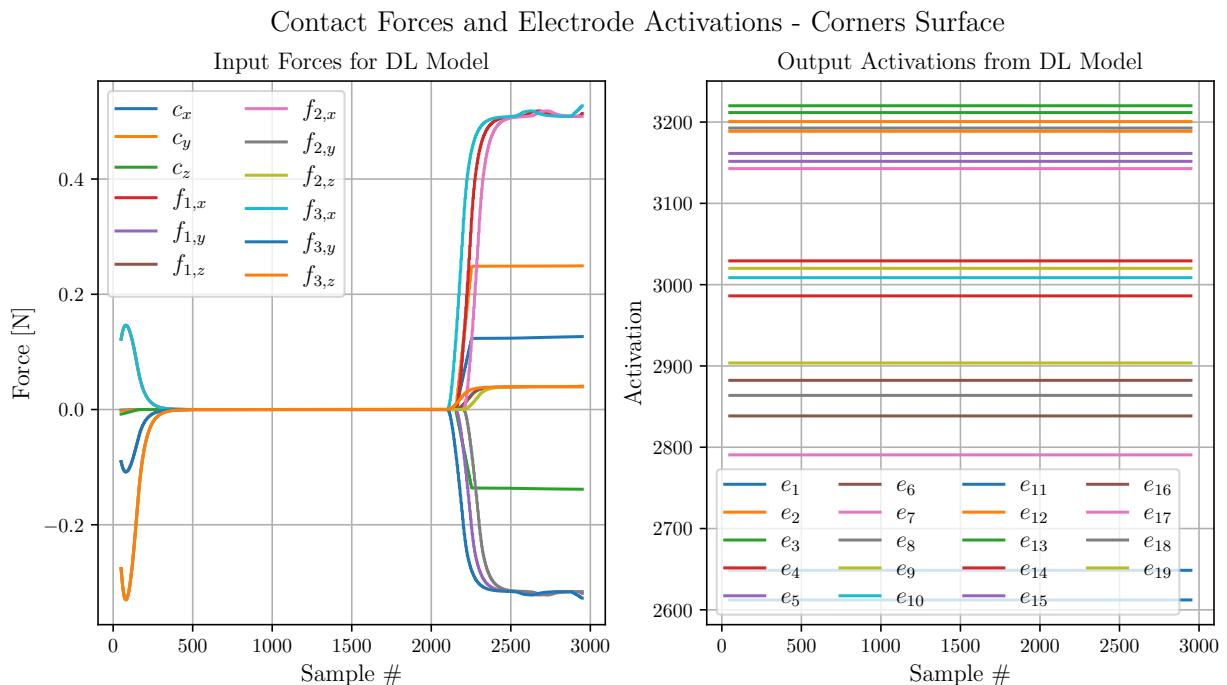
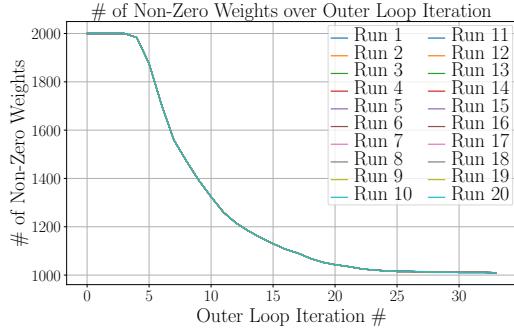


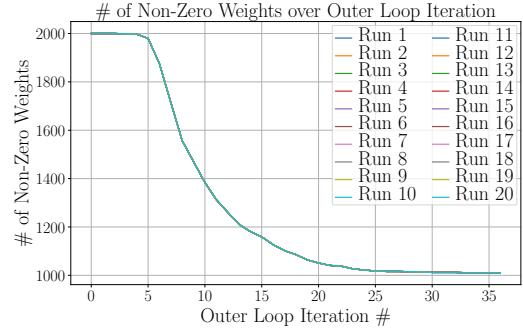
Fig. B.3: The simulated tactile electrode activations when the simulated SDH's index finger is in contact with a corner.

Appendix C

Pose Estimation - Weight Convergence

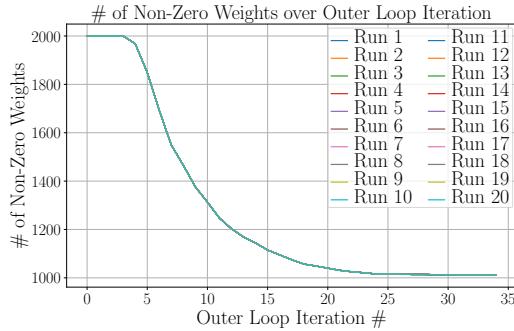


(a) Number of non-zero weight with 10 % outliers over outer loop iterations.

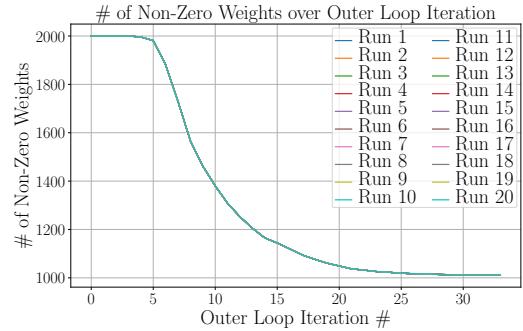


(b) Number of non-zero weight with 20 % outliers over outer loop iterations.

Fig. C.1: Number of non-zero weight with 10 % and 20 % outliers over outer loop iterations.

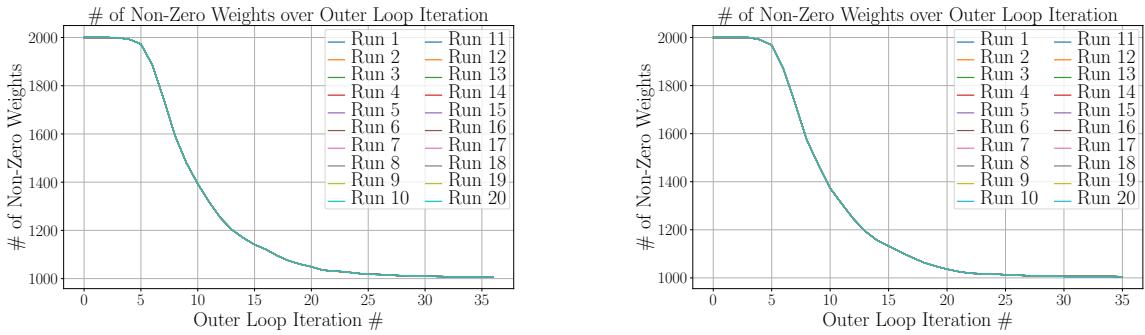


(a) Number of non-zero weight with 30 % outliers over outer loop iterations.



(b) Number of non-zero weight with 40 % outliers over outer loop iterations.

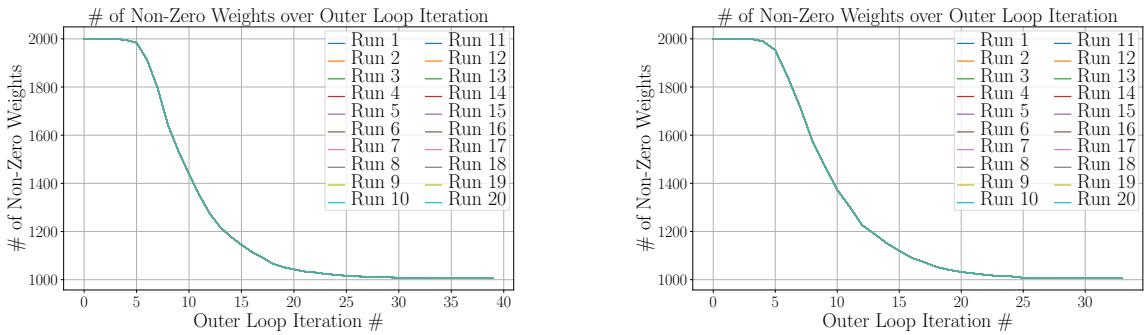
Fig. C.2: Number of non-zero weight with 30 % and 40 % outliers over outer loop iterations.



(a) Number of non-zero weight with 50 % outliers over outer loop iterations.

(b) Number of non-zero weight with 60 % outliers over outer loop iterations.

Fig. C.3: Number of non-zero weight with 50 % and 60 % outliers over outer loop iterations.



(a) Number of non-zero weight with 70 % outliers over outer loop iterations.

(b) Number of non-zero weight with 80 % outliers over outer loop iterations.

Fig. C.4: Number of non-zero weight with 70 % and 80 % outliers over outer loop iterations.

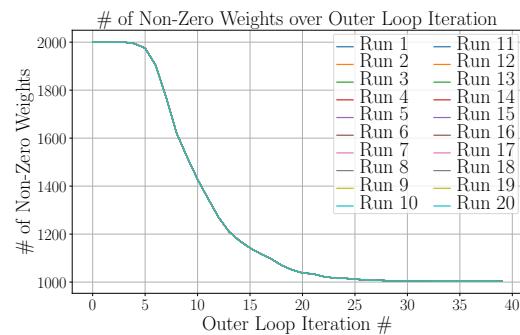


Fig. C.5: Number of non-zero weight with 90 % outliers over outer loop iterations.