



**Texture Recognition from Robotic Tactile Surface
Reconstruction: A Reinforcement Learning Approach**

by

Laurent Yves Emile Ramos Cheret

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
**MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL
INTELLIGENCE)**

2023

Lakehead University
Thunder Bay, Ontario, Canada

Supervisors

Thiago Eustáquio Alves de Oliveira, *PhD*

Department of Computer Science

Lakehead University

Thunder Bay, CA

ABSTRACT

The ability to handle objects and recognize them and their properties by touch is a crucial ability that humans have. Thanks to tactile sensing, robots can do something similar by perceiving specific physical characteristics of the objects they are in contact with. However, to do so in unstructured environments remains a challenge. The present work proposes a novel method for blind texture classification on uneven surfaces, using data from a robotic manipulator’s kinematic chain and a compliant tactile sensing module composed of MARG and barometer sensors. The data from the manipulator’s kinematic chain and the deformation of the sensing module are used to estimate the contact position and the vector normal to the surface. Contact points and normal vectors are then used to estimate control points for splines used to generate patches of surfaces. The reconstructions were validated in experiments with five surfaces, and a comparison with a vision system shows that it can achieve slightly better estimates. These estimations are used to train a Reinforcement Learning model for pressure-control, which adjusts the position of the manipulator’s end effector based on barometer readings, allowing the tactile sensing module to keep in touch with the surface without applying too much pressure on it. Trajectories for sliding motions are created by selecting points from the reconstructions and adjusting their position. Tactile data from trajectories with and without adjustment are collected and used for classification. Results show that the adjustment leads to an improvement of up to 30% in top-1 accuracy, reaching 90% on four textures. This work is a first proposal for texture classification on uneven surfaces where the exploratory motions depend on the object pose and shape, and could serve as a complementary system where vision is compromised.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to all those who have contributed to the completion of this master thesis. First and foremost, I extend my heartfelt thanks to my beloved wife Beatriz, whose unwavering love, patience, and understanding have been the pillars of strength that sustained me during the arduous process of research and writing. I am also indebted to my parents Maria and Gerard, and my sister Clara for their continuous support and understanding during the ups and downs of this academic endeavor.

I am grateful to my esteemed thesis advisor Thiago for his guidance, expertise, and valuable insights throughout the research journey. His mentorship has played a pivotal role in shaping the quality and direction of this thesis. Furthermore, I extend my thanks to the faculty and staff at Lakehead University, especially at the Computer Science department, whose dedication to education and commitment to excellence have enriched my academic experience.

To everyone who has played a part in making this thesis possible, I am truly grateful for your contributions and support. Your presence in my life has made this academic achievement even more meaningful.

With heartfelt appreciation,
Laurent

PUBLICATIONS

Publications:

- L. Y. E. Ramos Cheret and T. E. A. de Oliveira, "Encoding High-Level Features: An Approach To Robust Transfer Learning," 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain, 2022, pp. 1-6, doi: 10.1109/COINS54846.2022.9854982.

Submitted for Review:

- L. Y. E. Ramos Cheret, I. Thrikawala, V. P. da Fonseca, and T. E. A. de Oliveira, "Haptic Surface Reconstruction Using a Compliant Tactile module," (Submitted for review to the Journal of Advanced Robotics, 2023)

In-Progress:

- L. Y. E. Ramos Cheret, V. P. da Fonseca, and T. E. A. de Oliveira, "Blind tactile texture classification on uneven surfaces."

Contents

Abstract	iii
Acknowledgements	iv
Publications	v
Table of Contents	vi
List of Tables	viii
List of Figures	ix
List of Abbreviations	xii
1 Introduction	1
1.1 Objectives	3
1.2 Thesis organization	4
2 Literature Review	5
2.1 Tactile Perception	5
2.2 Robotic Manipulation	6
2.3 Surface Reconstruction	8
2.4 Texture Classification	10
2.5 Summary	12
3 Methods	14
3.1 Pose Representation	14
3.1.1 Inertial Measurement Unit	15
3.1.2 Kalman Filters	16
3.2 Robotic manipulation	17
3.2.1 Reinforcement Learning	17
3.3 Surface reconstruction	19
3.3.1 Non-Uniform Rational B-Splines	20

3.4	Texture Classification	21
3.4.1	Feature Encoding	21
3.4.2	Recurrent Neural Networks	23
4	Tactile texture recognition from blind surface reconstruction	26
4.1	Surface Reconstruction	27
4.1.1	Orientation Estimation	28
4.1.2	Calibration	30
4.1.3	Control points calculation	31
4.1.4	Robotic motion	34
4.1.5	NURBS surface generation	35
4.1.6	Experimental setup	36
4.1.7	Surfaces	38
4.1.8	Validation	39
4.2	Pressure-Control System	42
4.2.1	Waypoints adjustment	42
4.2.2	Exploratory Motions	43
4.2.3	Reinforcement Learning	44
4.2.4	Surface reconstruction with adjustment	48
4.3	Texture Classification	48
4.3.1	Textures	49
4.3.2	Trajectories	50
4.3.3	Data collection & preprocessing	52
4.3.4	Feature Encoding	52
4.3.5	LSTM classifier	52
5	Results & Discussion	54
5.1	Surface Reconstruction	54
5.1.1	Synthetic surfaces	54
5.1.2	Everyday objects	57
5.2	Pressure-control system	59
5.2.1	RL training	59
5.3	Surface reconstruction with adjustment	60
5.4	Data Collection	62
5.5	Classification	64
6	Conclusion	67

List of Tables

Table 5.1	Validation results from tactile approach, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).	54
Table 5.2	Validation results from vision approach, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).	55
Table 5.3	Validation results on everyday objects, comparing the tactile estimations to the visual reconstruction. Average distances and standard deviation in millimeters (mm).	57
Table 5.4	Validation results from tactile approach with pressure adjustment and alignment of the manipulator, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).	61

List of Figures

Figure 4.1	1) Blind surface reconstruction is performed on a surface of interest. 2) Points are selected to form a trajectory and adjusted through RL pressure-control. 3) Classification is performed on the tactile data gathered from exploratory motion on the adjusted trajectory.	27
Figure 4.2	1) Environment composed of 4-DoF manipulator, tactile sensing module, and surface. 2) Orientation is estimated from sensor and robot kinematic data. 3) Manipulator moves vertically to collect information from surface points. 4) Control points (artificial data) are generated and used to form a NURBS surface together with surface points.	28
Figure 4.3	Views of the robotic manipulator joints. Joints 1 to 4 rotate to provide 4-DoF. Gripper and Sub-Gripper joints hold the sensing module. . .	29
Figure 4.4	Control point (CP) calculation between surface points p_1 and p_2 . The green line is the generated curvature originating at p_1 and finishing at p_2	32
Figure 4.5	CP_{old} represents the wrong estimation of the control point. The red line shows the expected curvature if no correction is applied. CP_{new} represents the adjusted position after the correction is applied. The green line represents the corrected curvature.	33
Figure 4.6	Collection of points on the surfaces. a) Robotic manipulator approaches point of contact in a vertical motion. b) Once contact is perceived, manipulator stops at point of contact. Position and orientation are collected leaving a marker (pink arrow) at the location. c) Manipulator retreats in a vertical motion.	34
Figure 4.7	Control points layout. Surface and control points are placed in a grid to generate the surface patches. Each patch is composed of four surface points (SP_1, SP_2, SP_3, SP_4) and five control points (CP_1, CP_2, CP_3, CP_4 , and $CP_{central}$). The central control point is an average of the other four.	35
Figure 4.8	Example of a surface patch arranged in a 3×3 layout.	35

Figure 4.9 Tactile sensing module components: 1—MARG (magnetic, angular rate, and gravity) system; 2—compliant structure; 3—barometer [74].	36
Figure 4.10 Sensing module encapsulated in flexible structure.	37
Figure 4.11 System Layout. Connections between the compliant tactile sensing module (barometer and MARG) to its respective sensing MCU (ARM Cortex-M7); the robotic arm (Open Manipulator X) connection to the robot MCU (OpenCR 1.0) and their MCU connections to a PC running a ROS environment.	37
Figure 4.12 The five synthetic surfaces under study. Surface 1 is $8cm \times 8cm$ with $1cm$ peak height. Surface 2 is $8cm \times 8cm \times 2cm$ with a lowest height at $1cm$. Surface 3 is $16cm \times 5cm$ with highest point at $2.5cm$ and lowest at $1cm$. Surface 4 is $19cm \times 4cm$ with highest point at $2.5cm$ and lowest at $1cm$. Surface 5 is $20cm \times 16cm \times 1cm$	38
Figure 4.13 The four common objects used to validate the present approach. Mug has a lower and higher radius of $4cm$ and $3cm$, respectively, and a height of $13cm$. Shampoo bottle is $8cm \times 11cm \times 3cm$. Hair comb is $11cm \times 7cm \times 5cm$. Aerosol deodorant has a radius of $2cm$	39
Figure 4.14 Kinect setup for point cloud extraction. The point clouds are generated using rosbag files in the ROS environment. rosbag files are then converted to pcl files and the region of interest is isolated on CloudCompare.	41
Figure 4.15 Calculating new end-effector position pos_{new} based on the old position pos_{old} , and normal direction \vec{N}	42
Figure 4.16 Reconstruction of surface using NURBS. Every colored piece is a single patch created from four surface points. These patches are put together and form the overall shape.	43
Figure 4.17 Example of correct trajectory on curved surface. The manipulator maintains a constant angle of approach and pressure on the surface.	44
Figure 4.18 Reward as a function f of the barometer level. The maximum reward occurs when the barometer level is within an optimal region [150 - 250]. Red regions indicate terminal states for the manipulator with zero reward.	46
Figure 4.19 Framework for RL training.	47
Figure 4.20 Training episode. The manipulator starts moving to the estimated point in a vertical pose, and adjusts its orientation to face the same direction of the surface normal. Once aligned, it can start executing steps and eventually end the episode when conditions are met.	48

Figure 4.21	1) Estimations are generated using the present approach. 2) Points are chosen to form an exploratory trajectory and their position is adjusted through RL. 3) Tactile data from sensor readings are collected during the exploratory motion. 4) An LSTM classifier uses the data to identify the texture at hand.	49
Figure 4.22	Four identical synthetic surfaces used for classification with four different textures.	50
Figure 4.23	Trajectory generation from a surface patch. 1) Collect surface points and generate an estimation. 2) Waypoints are chosen to form a trajectory, the manipulator goes to each of the waypoints to adjust the pressure and store the current joint values. 3) The trajectory is executed.	51
Figure 4.24	Both classification architectures without and with the feature encoder.	53
Figure 5.1	Distance comparison between our approach and vision using Kinect. Unsigned C/M (U C/M), Signed C/M (S C/M), Unsigned C/C (U C/C). Mean \pm standard deviation.	55
Figure 5.2	Original surfaces and estimations comparison. The left column shows the original STL file of the 3D-printed surfaces. Middle column shows the estimations. Right column shows the superposition of the original STL file and the ones generated from the estimations after scaling and alignment on CloudCompare.	56
Figure 5.3	Everyday objects visual and tactile reconstructions.	58
Figure 5.4	Barometer level per step, at 12 training iterations.	59
Figure 5.5	Mean episode length and reward per step for each of the 12 training iterations.	60
Figure 5.6	Reconstruction from data collection with alignment and pressure adjustment.	61
Figure 5.7	Barometer readings for the cases without and with adjustment. . . .	62
Figure 5.8	Normalized data from accelerometer, gyroscope, and magnetometer, with no adjustment to the waypoints.	63
Figure 5.9	Normalized data from accelerometer, gyroscope, and magnetometer, with adjustment to the waypoints.	64
Figure 5.10	Confusion tables for the cases with adjustment and without when no feature encoding is applied	65
Figure 5.11	Confusion tables for the cases with adjustment and without when feature encoding is applied	65

List of Abbreviations

- Acc – Accelerometer
- AE – Autoencoder
- AHRS – Attitude and Heading Reference System
- B-Spline – Basis Spline
- CAD – Computer-Aided Design
- DQN – Deep Q-Network
- GRU – Gated Recurrent Unit
- Gyro – Gyroscope
- IMU – Inertial Measurement Unit
- LSTM – Long Short-Term Memory
- Mag – Magnetometer
- MARG – Magnetic, Angular Rate, and Gravity
- MEMS – Micro-Electro-Mechanical Systems
- NURBS – Non-Uniform Rational B-Splines
- PCL – Point Cloud Library
- PPO – Proximal Policy Optimization
- RL – Reinforcement Learning
- RNN – Recurrent Neural Network
- STL – Standard Tessellation Language
- TRPO – Trust Region Policy Optimization
- VAE Variational Autoencoder

Chapter 1

Introduction

Achieving robust, flexible, and adaptive grasping and manipulation in unstructured environments (i.e., dynamic settings that change often) remains a challenge [1, 2]. For robots to be able to explore unfamiliar environments, it is necessary for them to have systems capable of exploiting the sense of touch [3, 4], but robotic systems aren't yet fully capable of applying tactile sensing, an essential capability for object recognition and pose estimation tasks, which are also essential for applications such as object grasping and manipulation, and hazardous environment operations [2, 5].

One of the main challenges for robotic manipulation in unstructured environments is the accurate perception needed to model the environment's objects' static and dynamic features. Most current robotic applications make use of computer vision perception to acquire some of such features [6, 7]. But this technology may fail in unstructured environments for several reasons. For example, different textures, colors, and shapes can be difficult for a computer vision system to identify and track accurately. Additionally, lighting conditions in unstructured environments can vary widely, leading to poor image quality and making it difficult to detect objects or features of interest. Another reason is that unstructured environments can be cluttered, with many objects close to one another, making it challenging to separate and identify individual objects. Lastly, unstructured environments are often not controlled or predictable, making it difficult for a computer vision system to operate based on pre-programmed or learned models and rules.

The solution to avoid such drawbacks remains in sensor fusion. Having different sensing modalities such as vision and touch can provide a more accurate perception of the environment to the agent [8]. Redundant information can be used to reduce uncertainties and have a synergistic effect, where features from the environment can be detected in situations where using the information from each sensor separately would be impossible [9].

Haptic perception, or the sense of touch, can provide robots with valuable information about objects' texture, shape, and compliance in their environment. Attempts to fuse vision

and touch started in the 1980's [10], and showed positive results for object recognition, reconstruction, and grasping [11, 12].

Tactile information can improve the precision and robustness of robotic manipulation tasks, such as object recognition and tracking, particularly in unstructured environments where objects may have uncertain or variable properties. Additionally, it can allow a robot to detect and respond to unexpected environmental changes, such as an object slipping out of its grasp, which may not be immediately apparent through vision alone.

Tactile perception can also aid computer vision by providing information about the compliance or stiffness of objects, which is helpful for grasping and manipulation tasks. For example, it can be used to determine if an object is soft, such as a rubber ball, or hard as a metal cylinder. This information can help the robot to adjust its grip and avoid damaging the object or environment. Overall, incorporating tactile perception into a robot's sensor suite can improve the accuracy and robustness of its computer vision system.

The recent development of new tactile sensors and skins has provided new opportunities for applying tactile sensing in robotic applications. Nonetheless, the right computing method to extract the encoded information is just as crucial to an application as the different types of tactile sensors [2]. Tactile approaches usually need much more time to collect points of contact at each probe but provide more precise local information regarding the surface than vision approaches [13–15]. This means that high accuracy and real-time performance remains an open problem.

Tactile sensing is a very restricted instrumental modality that only captures the local stimulus surrounding a sensor, despite its strong performance in various scenarios. Fortunately, the restriction can be overcome by combining the sensing with exploratory robotic movements to increase the contact area, necessitating both spatial and temporal decoding to understand the signals [16]. In cases where the surface is flat, these movements can be easily created, but this is not the case when dealing with curved surfaces. In the latter, the robot needs to know beforehand the position of waypoints and the relative orientation of the robotic system at these points to create a trajectory. For that, it is necessary to have an accurate estimation or graphical model of the surface shape and pose. Previous works have shown that tactile surface reconstruction provides more accurate results than vision approaches but are less used due to their time-consuming process of collecting data [2].

Thanks to the development of new tactile sensing modules, several works have shown that dynamic touch using sliding motions can accomplish high accuracy in texture classification [16, 17]. By executing a trajectory over several different textures, readings from multiple sensors can capture the features of each texture and be used as time-series data for classification. However, to this day, no approach for texture classification on non-flat surfaces was found.

1.1 Objectives

The present study proposes a **first approach for tactile texture classification from blind tactile surface reconstruction** on uneven surfaces, where the exploratory motions depend on the surface at hand. For that:

- Develop and validate a novel blind surface reconstruction method to generate exploratory trajectories.
- Develop a system for trajectory adjustment to better capture tactile data.
- Classify different textures from tactile data gathered through exploratory motions on curved surfaces.

This work combines tactile data from a MARG (Magnetic, Angular-Rate, Gravity) and barometer sensors, and kinematic data from a robotic manipulator. The tactile sensing module acts as the manipulator’s end-effector, and is supported by a flexible compliant frame, offering real-time data for computing orientation and position. The proposed method generates artificial points from the collected data, called control points, through a geometrical approach. From the grid of collected and generated points, surface patches ($2cm \times 2cm$) are generated using Non-Uniform Rational B-Splines (NURBS), and multiple patches are stored together forming the overall surface.

These estimations can provide information about the object pose, and its shape, essential to perform exploratory movements. A subset of the estimations is manually selected (waypoints), and can be used to form a trajectory on top of the surfaces. As the waypoints are still estimations, a fine adjustment of their position must be performed using a pressure-control system, keeping the sensing module in contact with the surface while giving enough space for it to vibrate and collect accurate tactile data. This system is trained using Proximal Policy Optimization (PPO), a Reinforcement Learning approach to learn from exploration. It performs fine adjustment of the waypoints positions based on the barometer sensor level and allows the manipulator to perform smooth exploratory movements. Sensors readings from three different trajectories are collected on four different texture and used in a time-series classification model based on Long Short-Term Memory (LSTM). The present work reduces the amount of contact needed for generating the estimation, which significantly reduces the amount of time for collecting data, one of the most time-consuming steps in tactile approaches in comparison to vision ones. It achieves 90% accuracy on the trajectories with fine adjustment versus 63% on the same trajectories without any adjustment, and can be used as a first proposal for texture classification on non-flat surfaces.

1.2 Thesis organization

This thesis is organized into the following chapters:

1. **Chapter 1: Introduction**

This chapter provides an overview of the research problem, presents the objectives and significance of the study, and outlines the structure of the thesis.

2. **Chapter 2: Literature review**

This chapter reviews the existing literature related to the research topic, highlighting the key theories, concepts, and previous studies in the field.

3. **Chapter 3: Methods**

This section presents the methods that will be later employed in the present work. Benefits and advantages of each methods for surface reconstruction, robotic manipulation, and texture classification are also shown.

4. **Chapter 4: Tactile texture recognition from blind surface reconstruction**

This chapter describes the research methodology, including the research design, data collection methods, and analytical techniques employed.

5. **Chapter 5: Results & Discussion**

This chapter presents the findings of the research, analyzes the collected data, and discusses the results in relation to the research objectives.

6. **Chapter 6: Conclusion**

This chapter interprets the results in light of the research questions, and discusses their implications.

Chapter 2

Literature Review

2.1 Tactile Perception

For humans, tactile sensing is fundamental for our interactions with the world. It allows us to perceive texture, pressure, temperature, and vibrations, enabling us to manipulate objects, recognize shapes, and discern the properties of the surrounding environment. Tactile sensing also plays a critical role in social interactions, such as conveying emotions through touch, establishing connections, and providing comfort and reassurance.

In the field of robotics, tactile sensing is an area of active research and development. Integrating tactile sensors into robotic systems can enhance their perception, manipulation, and interaction capabilities. Tactile feedback enables robots to interact with objects and humans in a more dexterous and safe manner. By leveraging tactile information, robots can improve their object recognition and manipulation skills, adapt to uncertain and dynamic environments, and ensure the safety of human-robot interactions.

In industrial settings, tactile sensing allows robots to perform delicate tasks that require precise force control, such as assembly, pick-and-place operations, and quality inspection. Tactile sensing can provide valuable feedback during these tasks, enabling robots to detect contact forces, handle fragile objects without damaging them, and ensure proper alignment and fit. Having accurate tactile sensing systems capable of communicating useful information from the environment to the robot is of the utmost importance [3,4]. Tactile perception is a complementing source of information to visual perception for many robotic activities. However, developing less time-consuming tactile systems that can deliver real-time estimations precise enough to be used in the field remains a challenge [2]. The rapid development of tactile sensors has not been fully accompanied by the interpretation of tactile sensor readings. There are only a few survey articles in the literature that focus on reviewing computational intelligence methods applied in tactile sensing. The improvement in tactile sensor spatial resolution and fast temporal response presents an opportunity to leverage

state-of-the-art techniques from machine learning, signal processing, computer vision, and sensor fusion for tactile sensing applications.

In previous methods for estimating surfaces, solutions computing the estimates using visual information [6, 7] are chosen over tactile approaches because they are significantly faster while providing sufficient accuracy. These techniques are not applicable when external factors have a negative impact on the visual data quality. Occlusions, which can be caused by viewing angles and shape concavities, are another problem. In tasks such as object recognition and grasping, vision has been integrated with tactile sensing but has its usability limited in in-hand dexterous manipulation of objects [46] and fine surface features perception [47].

For example, in [19] the authors state that the the main challenges in replicating human touch sensing mechanisms are the low resolution of artificial tactile images, complex data interpretation, and the clumsiness of robot hand technology compared to the nimble dexterity of the human hand. They achieve high accuracy in profile recognition tasks but results are limited to linear motions. They conclude that there are still many challenges to be overcome in order to achieve human-like tactile perception. The authors in [20] state that to emulate human-like capabilities, specialized robotic hands with articulated fingers and various sensors for force, tactile, and kinesthetic feedback are necessary to control the forces and motions exerted on manipulated objects. Generally, there has been limited interest in intelligent computational models for interpreting data from such sensors and integrating human tactile properties into robotic systems.

Haptic exploration of unknown objects is crucial for a robot to autonomously perform grasping and manipulation tasks [21], as well as recognize their texture and shape. Research developed from a robotic perspective [22], enhance our understanding of how humans perform certain tasks. This justifies the interest in developing improved robot hand capabilities by drawing inspiration from human tactile perception over the years. The present work explores the presented gap by proposing an approach for blind surface reconstruction that is later employed for reinforcement learning training in the real world, and exploratory motions.

2.2 Robotic Manipulation

Robotic manipulation, the ability of robots to interact with and manipulate objects in their environment, has witnessed significant advancements in recent years [69–71]. These advancements have been driven by developments in various areas, including perception, planning, control, and machine learning. Here are some major advancements in robotic manipulation:

1. **Vision-Based Perception:** Robots use advanced computer vision for better envi-

ronmental understanding, object recognition, and pose estimation, enhancing grasping and manipulation tasks.

2. **Grasping and Manipulation Planning:** Research on robotic grasping has led to improved algorithms for optimal grasps based on object geometry, stability, and task requirements. Manipulation planning enables complex manipulation tasks.
3. **Soft Robotics:** Soft robots made of flexible materials offer adaptability, dexterity, and safe human interaction. They use grippers that conform to various object shapes for versatile manipulation.
4. **Reinforcement Learning:** Robots learn manipulation skills through trial and error using reinforcement learning (RL), allowing complex behaviors and adaptation to dynamic environments.
5. **Force and Tactile Sensing:** Robots integrate force/torque and tactile sensors for better understanding of object interaction, enabling precise manipulation tasks and monitoring object stability.
6. **Collaborative and Dual-Arm Manipulation:** Robots collaborate with humans or other robots to share tasks and assist in complex manipulation scenarios. Dual-arm manipulation offers improved dexterity.
7. **Dexterous Manipulation in Unstructured Environments:** Robots can manipulate objects in unstructured environments using advanced perception, planning, and control, handling clutter, unknown objects, and uncertain object poses.

These advancements in robotic manipulation have opened up new possibilities for robots to interact with and manipulate objects in various domains, including manufacturing, healthcare, logistics, and domestic settings. Indeed, the interest to achieve human-like robotic manipulation is shown by proposals of humanoid robotic hands [31], fingers [33] for object and texture recognition tasks. Previous work have also shown how its possible to control robotic grasping using tactile sensing gloves to approximate human and machine behavior in simple tasks [32]. Ongoing research and technological advancements continue to push the boundaries of robotic manipulation, enabling robots to perform more complex tasks with improved dexterity, adaptability, and autonomy.

In unstructured environments, applying too much contact to a surface or losing it altogether is easy. Different objects can have different shapes making a requirement for robotic manipulation in unstructured environment to understand orientation and position. Authors in [34, 35] show how a tactile sensing module can be used to predict accurate orientation combining it with sliding motions. In [34] they perform it without a camera reference during

the grasp phase, thus supporting the use of temporal tactile data for orientation estimation of in-hand objects in unstructured environments. For this, controlling the relative position between the sensing module and the surface is essential. The authors of [36] explore tactile object identification under two situations: single grasp, analogous to the haptic glance in humans, and through brief exploratory procedures where a robotic thumb displaces the grasped object to excite the sensors. They show the importance of machine learning techniques to process tactile data from grasping motions, and how these movements provide effective data for object recognition. Despite the interesting results showing the importance of tactile data for complex tasks in unstructured environments, what can be also extracted is the need for and the lack of proposals for exploratory motions. Indeed the presented results are limited to simpler movements in controlled environments.

In [37], the authors perform non-linear motions on flat surfaces. These surfaces are covered with fabrics and the robotic agent rubs its forearm to gather tactile data for classification. Other experiments use only linear motions, either by moving the robotic systems on a flat surface, such as a table, or by keeping the sensor still while moving the surface underneath. In this work we focus on the task on collecting tactile data for classification on curved surfaces. A crucial ability to explore objects in unstructured environments where vision is compromised or can't be used at all.

The present approach focuses on uneven surfaces where exploratory motions are more complicated due to the different orientation of the surface at different locations. For that, this work presents a reinforcement learning approach for adjusting the position of points in exploratory trajectories, enabling the collection and analysis of tactile data in scenarios closer to the real life.

2.3 Surface Reconstruction

One of the most important tasks in tactile sensing is the reconstruction of surfaces and objects with high precision for exploration of terrains and manipulation of objects. Indeed, tactile sensing has proven its superior accuracy for surface reconstruction in many scenarios. Earlier tactile proposals date back to the 1980s [10, 13]. On previous works using tactile data [2], approaches using contact points have more applicability in different scenarios, can work with a wider variety of tactile sensors, and retrieve arbitrary shapes. This is useful in unknown environments where the surfaces can assume any shape. Contact points are also useful for generating a graphical model of the surface shape and the points' spatial distribution is revealed. This can be used for pose and location estimation, grasping, manipulation, and exploration of terrains. This information can be useful in model-based training where the copied environment must be as close as possible to the real scenario [5, 7], as well as real world applications where the robot must plan and execute trajectories in an unknown

environment.

One downside found in these approaches is the amount of points necessary to generate the model of the surface. Authors in [14, 15] make use of control points to estimate the surface shape but need a dense set of points to provide accurate estimations, and are therefore too time consuming. In [25] the authors show an approach to reconstruct surface patches using 1-D data from robot finger tracking. They generate three concurrent curves and from them create a surface patch, achieving good accuracy but limiting the results for small regions as the complexity for curve-fitting rises with the size of the patch.

Another issue involves the reconstruction of the estimation from the tactile data. Methods that involve polynomial fitting face a trade-off between the estimation accuracy and computational complexity. For example, the authors of [7] try to reconstruct shapes using a multi-fingered dexterous hand. Given the relative position of the fingers on the touched surface they can estimate the overall shape. Using superquadric functions they fill the gaps to create the estimations. The drawback is that the more points they use, the more complex becomes the solution to the superquadric function. Another approach fits polyhedral structures to a set of contact points for object recognition but this solution face problems when dealing with complex structures that cannot be approximated to polyhedral [26]. Our proposal does not depend on the size of the studied surface as it estimates one patch at a time, with a constant number of points per patch, making this solution more suitable for unknown environments with a wide variety of surfaces.

More recent approaches for surface reconstruction showed the applicability of contact-based solutions in object recognition. Authors of [27] use a three-finger manipulator with tactile sensing arrays to touch 100 times an object at several locations, and from that collection of points they perform object recognition comparing the retrieved cloud to a known database. Another proposal also uses a three-finger manipulator to touch 5 to 20 times underwater objects [28], collecting features and comparing them to a database for object recognition. The downside found in these works is the time taken to collect the data, the manipulators have to touch several times the object to generate sufficient information for it to be recognized. Our approach differentiates from these proposals as we require much less contact to reconstruct the surface while also generating a mesh, or graphical model, essential for exploration and manipulation in unknown environments.

Other recent proposals use a material called Gelsight [29, 30], which can provide high-resolution estimations of surfaces. By pressing the gel against an object, the surface is printed on it allowing for a camera to collect the deformation and provide precise graphic models for pose estimation, texture classification, and other tasks. However, the sensing modules using this technology are difficult to miniaturize due to the included camera and LED's [2], making it a bulky solution that isn't yet scalable. Some of these sensing modules also face normal estimation errors due to its high sensitivity to small curvatures. Our

solution doesn't focus on high-level visual descriptors but can provide accurate estimates for larger areas, it uses a much more compact sensing module, with no usage of cameras, and can be scaled using multiple copies of the same module for faster reconstructions.

These recent achievements through tactile sensing are crucial as a more accurate understanding of the objects that the robots are interacting with helps to adjust the control strategy and control parameters, leading to more efficient and possibly safer motions. Unfortunately, even if tactile approaches for surface reconstructions offer superior precision than vision, they demand a long time for collecting tactile data or use bulky sensing modules. It is necessary then to have a tactile system for estimating an object pose and shape with less probing without sacrificing too much on precision.

2.4 Texture Classification

Texture classification has witnessed significant advancements over the years, driven by advancements in machine learning techniques, feature extraction methods, and the availability of large-scale texture datasets [57–61]. These advancements have greatly enhanced the accuracy and robustness of texture classification systems across various domains. Here are some major advancements in texture classification:

1. **Feature Extraction:** Deep learning-based approaches, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable success in learning discriminative features directly from raw textures. CNNs can automatically extract hierarchical representations capturing both low-level and high-level texture patterns.
2. **Transfer Learning:** Transfer learning has played a significant role in texture classification advancements. Pre-trained CNN models, trained on large-scale datasets like ImageNet, can be fine-tuned or used as feature extractors for texture classification tasks. This approach leverages the knowledge learned from diverse image categories and enables better generalization to texture-specific datasets with limited labeled samples.
3. **Spatial Pyramid Pooling:** Spatial Pyramid Pooling (SPP) is a technique that allows CNNs to handle textures of various sizes by aggregating features at multiple spatial scales. SPP enables the classification of textures at different levels of granularity, capturing both local and global texture information effectively.
4. **Ensemble Learning:** Ensemble methods, which combine multiple classifiers, have shown improved performance in texture classification. By aggregating the predictions of multiple classifiers, ensemble models can reduce errors and enhance the overall

classification accuracy. Techniques like Random Forests, AdaBoost, and Bagging have been applied to texture classification, leading to enhanced classification performance.

5. **Data Augmentation:** Generating augmented data has proven valuable in texture classification. Techniques such as rotation, translation, scaling, and adding noise to the texture samples increase the diversity of the training data, leading to improved generalization and better robustness against variations in texture appearance.
6. **Hybrid Approaches:** Hybrid approaches that combine multiple texture descriptors or multiple classification algorithms have gained attention. For example, combining color and texture information or fusing features extracted using different methods (e.g., handcrafted and deep learning-based) can yield more discriminative representations and boost classification accuracy.
7. **Domain Adaptation:** In real-world scenarios, texture classification may face challenges due to domain shifts, where the training and test data come from different distributions. Domain adaptation techniques aim to mitigate this issue by adapting the learned texture representations from a source domain to a target domain with limited labeled data. This enables improved generalization and performance on unseen texture samples.

These advancements in texture classification have propelled the field forward, enabling more accurate and robust texture analysis across a wide range of applications, including image retrieval, medical imaging, material recognition, and surveillance systems. Ongoing research and the application of innovative techniques continue to drive further advancements in texture classification.

Taking inspiration from how humans attempt to identify textures solely with their skin by sliding and rubbing fingers on surfaces, multiple works have shown that using exploratory movements with tactile sensing modules can provide high accuracy in classification tasks [16, 18, 23]. According to Weber et al. study on human tactile perception [24], certain invariant tactile features can be retrieved through touching and sliding/rubbing and cannot be extracted from visual data. Our study aims to classify four different textures on uneven surfaces using this idea of sliding motion to capture small features through vibrations.

The kind of features that can be derived from tactile data typically depends on the chosen sensing technology. When it comes to texture classification, there are two main methodological trends. The first either makes use of a high-resolution vision-based sensor [29, 30] or crops time-series data [37] to create tactile images and instantly encode the spatial textures. While the second kind of technique uses vibration-sensitive sensors to gather tactile signals [16, 17]. In both trends, Deep learning is the tool that has been used

the most with Convolutional Neural Networks (CNNs) for visual feature extraction and Recurrent Neural Networks (RNNs) for temporal features.

In [17] the authors perform a tactile profile classification using a sensing module equipped with barometer and MARG sensors. They show that the 1D signal from barometer and 3D signals from accelerometer, gyroscope and magnetometer can be used in neural networks for time-series classification, achieving +90% accuracy on seven surfaces. A similar approach was found in [39] but both present linear motions for data collection which limits the findings to this controlled environments [38]. The authors of [16] perform texture classification of fabrics using sliding motion. They use a finger-shaped sensor to perform linear motions across the fabric and collect small vibrations as temporal data. They show that the variation of sliding speed has no effect on the classification results but variation of pressure can greatly impact the extracted features. Indeed, applying too much pressure on a surface causes the sensory module not to vibrate during motion, causing the specific features of each material to change or not appear at all.

This work concludes by providing texture recognition, an importante tasks in robotics, on uneven surfaces. Using the present blind reconstruction approach and RL-based trajectory adjustment, this work collects tactile data from multiple sensors on different trajectories, with and without adjustment, performing texture classification.

2.5 Summary

The following text summarizes the gaps and weaknesses found in the areas of Tactile Perception, Robotic Manipulation, Surface Reconstruction, and Texture Classification.

- **Tactile Perception:**

- Developing less time-consuming tactile systems for real-time estimations remains a challenge.
- Integration of vision and tactile sensing has limitations in in-hand dexterous manipulation and fine surface feature perception.
- Replicating human touch sensing mechanisms faces challenges due to low resolution, complex data interpretation, and limited dexterity of robot hand technology.

- **Robotic Manipulation:**

- Although significant advancements have been made, challenges remain in manipulating objects in unstructured environments, understanding object orientation, and handling uncertainties.
- Current tactile approaches for object recognition and manipulation require multiple touches, making them time-consuming.

- **Surface Reconstruction:**

- Existing tactile approaches for surface reconstruction demand a dense set of points for accurate estimations, which is time-consuming.
- Methods using polynomial fitting face trade-offs between accuracy and computational complexity.
- Current solutions using cameras or Gelsight technology are bulky and not easily scalable.

- **Texture Classification:**

- Texture classification has seen most advancements in vision despite higher accuracy in sliding motion methods.
- Existing tactile approaches for texture classification are limited to linear surfaces/movements, impacting the generalization of results.

Overall, the challenges and weaknesses identified in these areas indicate the need for efficient tactile systems capable of real-time estimations, faster and more accurate tactile approaches for object and surface recognition, and the development of more compact and scalable tactile sensing modules for robotic manipulation tasks. Additionally, exploring the use of tactile data in unstructured environments and applying advanced machine learning techniques for texture classification remain areas of interest for future research.

Chapter 3

Methods

As seen in the previous sections, tactile texture recognition in unstructured environments remains a challenge for different reasons. Robotic systems need an accurate method for pose estimation, required to perform exploratory motions and collect surface data. To do so, an accurate tactile method to reconstruct surfaces is necessary as the robot needs to know the position of waypoints belonging to the exploratory trajectories. Also, data collection depends on the relative position of the end-effector and the surface under study. Given these challenges, this section will explain methods and notations employed in the present work to achieve the final objective. The methodology employed for pose estimation, tactile surface reconstruction, robotic manipulation, and texture classification is presented. Pose estimation techniques play a pivotal role in deciphering textures as they provide an accurate spatial understanding of an object's orientation and viewpoint, enhancing the system's ability to identify textures from different angles. Complementing this, robotic manipulation techniques enable the exploration of textures through tactile sensors, something that is very important in tactile approaches as tactile sensors only collect local stimulus, mimicking human touch to gather essential data for recognition. Concurrently, surface reconstruction methodologies contribute by creating a coherent three-dimensional model of the texture's surface, facilitating a more comprehensive analysis and enabling the accurate design of exploratory trajectories. However, the true finesse of blind texture recognition lies in texture classification methods, which leverage advanced algorithms to differentiate between intricate patterns and textures, enhancing the accuracy and versatility of the recognition process.

3.1 Pose Representation

Pose estimation is of utmost importance in robotics [48] as it provides essential information about the position and orientation of robots in their environment. Accurate pose estimation enables robots to navigate autonomously, perform object manipulation tasks, interact with

humans, and create accurate maps of their surroundings. It plays a crucial role in localization, mapping, object detection, and collision avoidance, allowing robots to navigate safely and effectively in dynamic environments. Pose estimation is a fundamental capability that empowers robots to perceive and understand their environment, enabling them to operate intelligently and perform complex tasks in various domains such as industrial automation, service robotics, healthcare, and exploration.

A pose can be represented as a combination of orientation and position. The orientation is typically represented using a quaternion, and the position is represented as a 3D vector.

Let $\mathbf{q} = q_w + q_x i + q_y j + q_z k$ be the quaternion representing the orientation, where q_w , q_x , q_y , and q_z are the scalar and vector components of the quaternion, respectively.

The position vector can be represented as $\mathbf{p} = [p_x, p_y, p_z]$, where p_x , p_y , and p_z are the Cartesian coordinates of the position.

Therefore, the pose can be represented as:

$$\text{orientation} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} \quad \text{position} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (3.1)$$

All vectors $v = [x, y, z] \in \mathbb{R}^3$ can be redefined as a pure quaternions $v = [x, y, z, 0] \in \mathbb{H}^4$ for operations with quaternions.

\otimes represents the quaternion multiplication (Hamiltonian product). Given two quaternions $\mathbf{p} = [p_0, p_1, p_2, p_3]$ and $\mathbf{q} = [q_0, q_1, q_2, q_3]$, the Hamiltonian product is given by:

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{bmatrix}$$

3.1.1 Inertial Measurement Unit

Pose estimation through an Inertial Measurement Unit (IMU) is a valuable technique in robotics and navigation applications [49]. IMUs consist of sensors such as accelerometers and gyroscopes that measure linear accelerations and angular velocities, respectively. By integrating these measurements over time, it is possible to estimate the position, velocity, and orientation (pose) of a robot or object.

IMUs provide real-time updates and do not rely on external references, making them suitable for applications where external signals or infrastructure may be unavailable or unreliable. They are compact, lightweight, and cost-effective, allowing for easy integration

into various robotic platforms.

The accelerometer measures the linear acceleration experienced by the device along its three axes, usually denoted as $\mathbf{a} = [a_x, a_y, a_z]$. On the other hand, the gyroscope measures the angular velocity around each axis, typically represented as $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$.

However, it is important to note that IMUs have inherent limitations, including drift and errors that accumulate over time due to sensor noise and bias. To address these limitations, sensor fusion techniques can be employed to combine IMU data with other sensors, such as magnetometers, cameras, or GPS, for more accurate and robust pose estimation. These techniques, such as Kalman filters or particle filters, integrate the strengths of multiple sensors to compensate for the weaknesses of individual sensors and provide enhanced pose estimation accuracy and reliability.

3.1.2 Kalman Filters

Kalman filters are powerful mathematical algorithms used for optimal state estimation in dynamic systems [50, 51]. They operate by recursively estimating the state of a system based on noisy and incomplete measurements. Kalman filters employ a two-step process: prediction and update. In the prediction step, the filter uses a mathematical model to predict the system's state and its uncertainty. Then, in the update step, the filter incorporates the actual measurements to refine the predicted state estimation.

Madgwick Complementary Filter

The Madgwick filter is an algorithm used for sensor fusion in orientation estimation [52, 53]. It's particularly designed for micro-electro-mechanical systems (MEMS) sensor units like accelerometers, gyroscopes, and magnetometers. The filter combines sensor data to estimate the orientation of an object relative to a global reference frame. This is useful in applications like inertial navigation, robotics, virtual reality, and augmented reality.

Let \mathbf{q}_t represent the orientation quaternion at time t , the next estimation \mathbf{q}_{t+1} can be generated through the following equation:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \Delta t \cdot \frac{1}{2} (\boldsymbol{\omega}_t - \beta \cdot \boldsymbol{\varepsilon}_t) \quad (3.2)$$

The orientation is adjusting by compensating the error between estimated $\hat{\mathbf{a}}_t$ and real accelerometer \mathbf{a}_t readings, and the gyroscope $\boldsymbol{\omega}_t$ readings. Δt is the time step, $\boldsymbol{\omega}_t$ represents the gyroscope measurements at time t in the body frame, and $\boldsymbol{\varepsilon}_t$ is the error term.

The error term $\boldsymbol{\varepsilon}_t$ is computed as the difference between the measured acceleration in the body frame (\mathbf{a}_t) and the predicted acceleration based on the estimated orientation ($\hat{\mathbf{a}}_t$). It is given by:

$$\boldsymbol{\varepsilon}_t = \mathbf{a}_t - \hat{\mathbf{a}}_t \quad (3.3)$$

To calculate the predicted acceleration, the estimated orientation \mathbf{q}_t is used to rotate the gravity vector \mathbf{g} from the global frame to the local frame:

$$\hat{\mathbf{a}}_t = \mathbf{q}_t \otimes \mathbf{g} \otimes \mathbf{q}_t^* \quad (3.4)$$

The scalar parameter β controls the trade-off between the gyroscope and accelerometer measurements. It determines the rate at which the gyroscope measurements are trusted relative to the accelerometer measurements. Higher values of β give more weight to the gyroscope, while lower values favor the accelerometer.

The quaternion \mathbf{q}_{t+1} is then normalized to maintain its unit length, ensuring a valid orientation representation.

The filter iteratively updates the quaternion estimate using the above equation at each time step, incorporating new sensor measurements. This iterative process gradually improves the estimated orientation by minimizing the discrepancy between the predicted and measured accelerations.

By continuously updating the estimated orientation based on gyroscope and accelerometer measurements, the Madgwick filter provides a computationally efficient and effective solution for orientation estimation in real-time applications.

3.2 Robotic manipulation

Robotic manipulation, particularly when harnessed through reinforcement learning, plays a pivotal role in advancing blind texture recognition to unprecedented levels of sophistication. By employing reinforcement learning algorithms, robots can iteratively refine their tactile exploration strategies, gradually learning to optimize their interactions with textures. This process not only empowers robots to gather data that captures the nuances of textures but also allows them to adapt their manipulation techniques to varying surface characteristics and environmental conditions. Reinforcement learning facilitates the development of adaptive and autonomous systems that continuously enhance their texture recognition capabilities through real-time interactions.

3.2.1 Reinforcement Learning

In the recent years, Reinforcement Learning has emerged as an effective algorithm for robotic manipulation [42]. RL is a model-free framework for solving optimal control problems. At each time step the agent takes an action changing the environment around it, the observations from the current state are returned and used to generate the next action that the

agent will take. The actions are chosen by a policy which is trained to get the maximum reward at each step.

Two main groups exist in RL. The first are model-based methods where new states and rewards can be predicted based on a modeled environment. The second are model-free methods where new states and rewards are calculated based on current observations of the environment. In many cases where RL is used for robotic applications it is not possible to model the whole environment, that is why many robotic systems are trained using model-free methods.

In model-free methods there are also two main groups, Q-Learning [43] and Policy Optimization (PO) algorithms [44]. The main difference between PO and Q-Learning is that PO algorithms can be used in environments with continuous action space (i.e. where actions have real values) and can find the optimal policy even if that policy is a stochastic one (i.e. acts probabilistically), whereas the Q-Learning algorithms cannot do either of those things. On the other hand, Q-Learning algorithms tend to be simpler, and more intuitive to train. In this work our action is a real value which is translated to an updated position in the 3D space so the present work uses a PO algorithm which offers continuous action space.

Proximal Policy Optimization

Proximal Policy Optimization (PPO) [44] is a reinforcement learning algorithm that combines ideas from policy gradients and trust region methods. The PPO loss function combines the actor and critic components of the actor-critic architectures. The loss function is composed of two terms: the policy loss (actor loss) and the value function loss (critic loss).

The policy loss term encourages the policy (actor network) to improve by maximizing the expected advantage. The advantage measures the improvement of taking a specific action compared to the average action. The policy loss is typically computed using a surrogate objective function, such as the clipped surrogate objective used in PPO.

The value function loss term updates the critic network to improve its estimation of the state-value function. The critic loss is typically calculated as the mean-squared error (MSE) between the estimated state-values and the observed returns.

The combined loss function for PPO can be expressed as follows:

$$L^{PPO}(\theta) = L^{actor}(\theta) + L^{critic}(\theta) \quad (3.5)$$

where $L^{PPO}(\theta)$ is the PPO loss function parameterized by θ , and $L^{actor}(\theta)$ and $L^{critic}(\theta)$ represent the actor and critic loss terms, respectively.

The actor loss term can be written as:

$$L^{actor}(\theta) = \mathbb{E} \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \quad (3.6)$$

where $r_t(\theta)$ is the ratio of the current policy and the old policy probabilities, \hat{A}_t is the advantage estimate, and ϵ is a hyperparameter controlling the degree of clipping. The advantage estimate quantifies how advantageous it is to take action a_t in state s_t compared to the expected return under the current policy. This is usually calculated as:

$$\hat{A}_t = Q(s_t, a_t) - V(s_t) \quad (3.7)$$

The estimated state-action value function $Q(s, a)$ represents the expected cumulative reward of taking action a in state s and then following the policy thereafter. In reinforcement learning, this function helps us quantify the quality of a specific action in a given state. The formula for the estimated state-action value function can be written as:

$$Q(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s, a \right] \quad (3.8)$$

Where \mathbb{E} represents the expectation, which is taken over all possible future states and rewards. γ is the discount factor, which represents the degree to which future rewards are discounted relative to immediate rewards.

The critic loss term can be written as:

$$L^{critic}(\theta) = \frac{1}{N} \sum_{i=1}^N (V(s_i) - R_i)^2 \quad (3.9)$$

where N is the number of data samples, s_i is the state, $V(s_i)$ is the estimated state-value from the critic network, and R_i is the observed return.

By optimizing the combined PPO loss function, the actor and critic components of the actor-critic architecture can be updated to improve the policy and the estimation of the state-value function, respectively.

The present work uses this algorithm to train the pressure-control agent for the benefits of continuous action space, trust-region approach to avoid falling in local minima, and for being simpler than other PO approaches.

3.3 Surface reconstruction

Surface reconstruction methods hold a critical role in guiding and refining robotic exploratory motions. These techniques allow robots to transform the tactile data acquired dur-

ing their interactions with objects and surfaces into coherent and accurate three-dimensional representations. By reconstructing the surface geometry and topography, robots gain a comprehensive understanding of the object’s structure, texture variations, and intricacies. This information is invaluable as it enables robots to adapt their exploratory motions in real-time, ensuring that subsequent interactions are precisely targeted and yield more informative data.

3.3.1 Non-Uniform Rational B-Splines

NURBS (Non-Uniform Rational B-Splines) offer several benefits for surface modeling and design. One key advantage is their ability to represent complex and smooth curves and surfaces with high accuracy [55, 56]. NURBS provide precise control over shape using control points and weights, allowing for customizable and visually appealing designs. They also facilitate efficient computation, enabling real-time or near-real-time operations. NURBS are widely used in computer-aided design (CAD) and computer graphics industries, with established tools and libraries available for working with NURBS surfaces. This compatibility and standardization make it easier to integrate NURBS-based modeling into existing workflows. Overall, NURBS provide flexibility, accuracy, smoothness, and efficient computation, making them a valuable tool for a range of applications, including industrial design, automotive design, architecture, and animation.

The formula for a NURBS curve of degree ‘ p ’ with ‘ $n+1$ ’ control points and corresponding weights is given as follows:

$$C(u) = \frac{\sum_{i=0}^n N_{i,p}(u) \cdot P_i \cdot w_i}{\sum_{i=0}^n N_{i,p}(u) \cdot w_i} \quad (3.10)$$

In this formula:

- $C(u)$ represents the point on the curve at parameter u . Here u can assume any value within the range of values ‘knots’ from the knot vector U .
- $N_{i,p}(u)$ refers to the B-Spline basis functions of degree p defined on the knot vector.
- P_i denotes the i -th control point of the curve.
- w_i represents the weight associated with the i -th control point.

The B-Spline basis functions $N_{i,p}(u)$ can be recursively defined as:

$$N_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \leq u < u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (3.11)$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} \cdot N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \cdot N_{i+1,p-1}(u) \quad (3.12)$$

Here, u_i denotes the i -th knot value, and u is the parameter value at which we evaluate the curve.

3.4 Texture Classification

Texture classification methods play a vital role in blind texture recognition by using advanced algorithms to decipher intricate patterns and characteristics present in textures. This categorization enables machines to distinguish between different textures based on their unique features, enhancing recognition accuracy and enabling applications in various domains, from assisting the visually impaired to improving robotic systems in diverse industries. These methods are essential for machines to comprehend and navigate the tactile world effectively, fostering the development of more intuitive and inclusive technologies.

3.4.1 Feature Encoding

Feature encoding plays a critical role in reducing noise and enhancing the efficiency and accuracy of various data-driven applications, such as machine learning, data analysis, and signal processing [40, 41]. Noise, in this context, refers to irrelevant or misleading information present in the data that can hinder the performance of algorithms and models. By transforming the data into a more compact and informative representation through feature encoding, we can significantly improve the quality of the analysis and decision-making processes. Here are some key reasons why feature encoding is essential to noise reduction:

1. **Dimensionality reduction:** Feature encoding techniques can effectively reduce the dimensionality of the data while preserving essential information. High-dimensional data often contains noise, irrelevant features, and redundancies. By selecting relevant features and compactly representing them, dimensionality reduction reduces the chances of overfitting, thus enhancing the model's generalization capabilities.
2. **Noise filtering:** Certain feature encoding methods can be designed to filter out noise from the data. For instance, outlier detection techniques can help identify and remove noisy data points, leading to a cleaner and more reliable dataset. Removing noise at this stage prevents it from negatively impacting subsequent analysis steps and model performance.
3. **Enhanced generalization:** Feature encoding helps create more abstract and representative features that capture the underlying patterns and relationships in the data. These encoded features can generalize better to unseen data, as they focus on essential characteristics rather than specific noise or local variations. As a result, models trained on encoded features are more robust and less sensitive to noisy input.

4. **Improved computational efficiency:** Noise can introduce unnecessary complexities into the data, making algorithms and models computationally expensive. Feature encoding simplifies the data representation, reducing the computational burden and speeding up the analysis or learning process.
5. **Increased model interpretability:** Some feature encoding techniques, such as feature scaling or normalization, can make the model more interpretable. When features are brought to a common scale, it becomes easier to understand the impact of each feature on the model's predictions. This interpretability helps in identifying and mitigating any noise-related biases that might be present in the data.
6. **Better data visualization:** Feature encoding can create visual representations of data that highlight patterns and relationships effectively. Data visualization is an essential tool for identifying outliers, spotting noise, and gaining insights into the underlying structure of the data. By encoding features appropriately, visualizations become more informative and facilitate better data understanding.
7. **Robustness to data collection artifacts:** Real-world data often contains artifacts caused by data collection processes, measurement errors, or missing values. Feature encoding can handle missing data or outliers, making the analysis more resilient to data imperfections. It minimizes the adverse effects of data collection errors and reduces the likelihood of drawing erroneous conclusions.

In summary, feature encoding plays a fundamental role in reducing noise and enhancing the quality of data analysis and modeling. By selecting, transforming, and representing relevant features, it helps filter out noise, improves model generalization, increases computational efficiency, aids data visualization, and enhances the overall accuracy and reliability of data-driven applications. Noise reduction through feature encoding is a crucial step in extracting valuable insights and making informed decisions based on data.

Variational Autoencoder

A Variational Autoencoder (VAE) is a generative model that combines an encoder and decoder [62]. It learns to map data into a latent space and reconstructs data from latent representations [63]. The VAE is trained to optimize a loss function that balances the reconstruction accuracy and regularization, often using the Kullback-Leibler (KL) divergence.

1. **Generative Model:** VAEs can generate new data samples by sampling from the latent space, making them useful for data synthesis and augmentation.
2. **Compact Latent Representation:** VAEs learn a low-dimensional representation of complex data, capturing essential features while reducing noise and redundancy.

3. **Probabilistic Interpretation:** The VAE’s latent space follows a probabilistic distribution, allowing for uncertainty estimation and interpolation between data points.
4. **Interpolation and Smoothness:** VAEs exhibit smooth transitions in the latent space, enabling meaningful interpolations between different data samples.
5. **Regularization:** The KL divergence loss encourages a structured latent space, preventing overfitting and improving generalization.
6. **Anomaly Detection:** VAEs can be used for anomaly detection by assessing data reconstruction errors.
7. **Dimensionality Reduction:** VAEs provide an effective method for dimensionality reduction, simplifying data for downstream tasks.
8. **Transfer Learning:** Pretrained VAEs can be used as feature extractors for various other tasks, leveraging the learned representations.
9. **Unsupervised Learning:** VAEs do not require labeled data, making them applicable to unsupervised learning scenarios.

In summary, VAEs offer a powerful framework for unsupervised learning, feature learning, and data generation, while providing interpretable latent spaces with various practical applications. The present work uses the approach proposed in [75] where the authors use a VAE in the output space of the CNN instead of the input space, they show its capability in increasing classification results for different datasets and the benefits that it offers when dealing with noisy images. Previous methods involved working directly with the input which is generally larger, requiring larger models for noise cleaning, or changing the architecture of feature extractors to be more robust to specific sources of noise. Due to the easy implementation of the cited method, this work uses it for the final classification task.

The classifier without the encoder is trained first to obtain the top accuracy, then the feature extractor, is frozen and feature maps of each input are generated. These feature maps are fed to the VAE. Once the autoencoder reaches top accuracy, the encoder part is connected to the feature extractor and a final classification layer is connected to the latent output of the encoder.

3.4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are important for their ability to process sequential data, capture temporal dependencies, and handle variable-length input. They excel in tasks such as language translation, speech recognition, and sentiment analysis by modeling context and long-term dependencies [64, 65]. RNNs leverage internal memory, feedback

connections, and advanced architectures like LSTM and GRU to effectively capture and utilize sequential information, making them invaluable in fields such as natural language processing and time series analysis.

The core equation of a Recurrent Neural Network (RNN) is known as the recurrent step equation. It represents how information flows through the network over time. In its simplest form, the equation for an RNN at time step 't' can be written as:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (3.13)$$

- In this equation:
- h_t represents the hidden state or memory at time step 't'
 - x_t is the input at time step 't'.
 - W_{xh} and W_{hh} are weight matrices that determine the influence of the current input and previous hidden state, respectively.
 - b_h is the bias term.
 - σ is the activation function, such as the sigmoid or hyperbolic tangent, that introduces non-linearity to the output.

The equation essentially combines the current input x_t with the previous hidden state h_{t-1} to compute the current hidden state h_t . This recursive nature allows the network to capture temporal dependencies and learn from the sequential data. The output of the RNN at each time step can be further processed or used for various tasks, such as prediction, classification, or generating sequence outputs. It's worth noting that this is a simplified form of the RNN equation, and more advanced variants like LSTM and GRU incorporate additional components and gating mechanisms to address issues like vanishing gradients and improve long-term memory.

Long Short-Term Memory

The Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) architecture that addresses the vanishing gradient problem and allows for better capturing of long-term dependencies in sequential data [67, 68]. The key component of LSTM is its memory cell, which controls the flow of information and enables the model to retain or forget information over time. The main components in the LSTM architecture are the following:

- **Forget Gate:** The forget gate determines how much of the previous cell state should be forgotten.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.14)$$

- **Input Gate:** The input gate decides which new information needs to be stored in the memory cell. It consists of two parts: the input gate itself and the candidate values.

The input gate determines the relevance of the new input, and the candidate values represent potential new information.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.15)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3.16)$$

- Update Cell State: The update of the cell state involves combining the previous cell state C_{t-1} with the new information obtained from the input gate and the candidate values. The forget gate determines how much of the previous cell state should be retained, and the input gate determines how much of the new information should be added.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (3.17)$$

- Output Gate: The output gate determines the relevance of the current hidden state h_t based on the updated cell state C_t .

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) h_t = o_t \cdot \tanh(C_t) \quad (3.18)$$

In summary, the LSTM model employs the forget gate, input gate, update of the cell state, and output gate to control the flow of information and update the hidden state and cell state over time. This architecture allows the LSTM to capture and remember long-term dependencies in sequential data, making it particularly effective for tasks that involve temporal patterns and memory retention.

Chapter 4

Tactile texture recognition from blind surface reconstruction

This section will explain the methodology used to perform texture classification from blind surface reconstruction. Following Figure 4.1, the environment for the present work is composed of a 4-DoF manipulator holding a tactile sensing module on its gripper, and a surface under study. The first step is to create an estimation of the surface using an approach that doesn't rely on vision. These graphical models provide a series of coordinates in the 3D space, which can be used to generate exploratory trajectories. Because the coordinates of waypoints are still estimations, a reinforcement learning agent adjusts the orientation and position of the manipulator to maintain a constant pressure and angle of approach relative to the surface. Trajectories are then executed on top of the surface to collect dynamic tactile data from MARG and barometer readings. This time-series data is classified using a RNN architecture based on LSTM cells into one of the four available textures. This section will then start by presenting the surface reconstruction method with the experimental setup and validation approach. The second part of this section will explore the RL training for the pressure-control system. Finally, this section will explore the data collection methods and classification architectures.

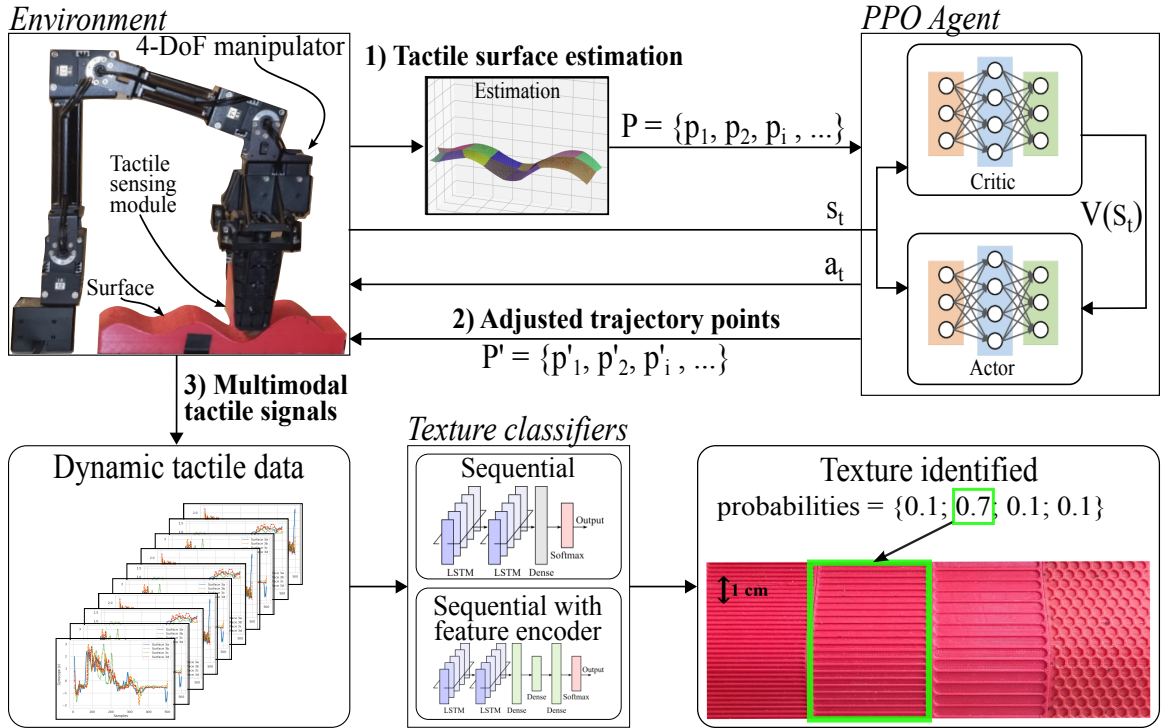


Figure 4.1: **1)** Blind surface reconstruction is performed on a surface of interest. **2)** Points are selected to form a trajectory and adjusted through RL pressure-control. **3)** Classification is performed on the tactile data gathered from exploratory motion on the adjusted trajectory.

4.1 Surface Reconstruction

This approach proposes a novel method for estimating surface shape using tactile data provided by the robotic manipulator and tactile sensing module. Following Figure 4.2, the environment is composed of 4-DoF manipulator holding a tactile sensing module, and a surface under study. Using the readings from the MARG sensor and the manipulator inverse kinematic chain, real-time orientation and position can be determined. Thanks to the flexible frame in the sensing module, the relative position and orientation of the surface can be estimated at several locations. With orientation and position, the position of new points can be estimated, called control points. The tactile information together with the artificial control points are used in a NURBS approach to generate graphical models.

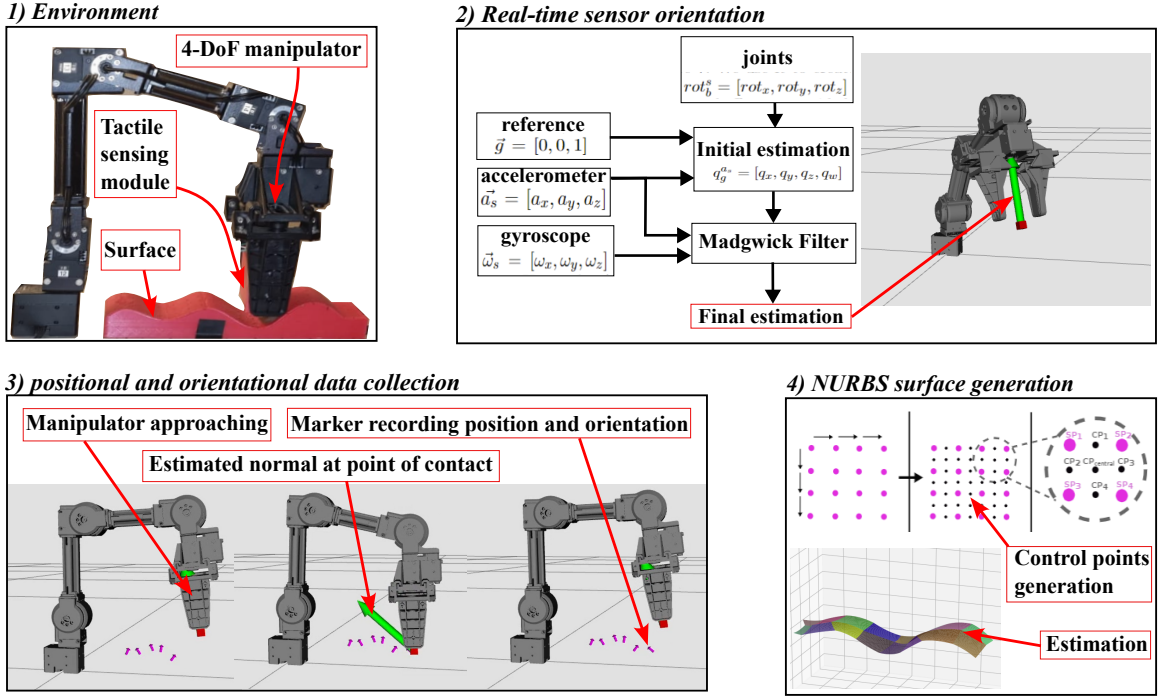


Figure 4.2: **1)** Environment composed of 4-DoF manipulator, tactile sensing module, and surface. **2)** Orientation is estimated from sensor and robot kinematic data. **3)** Manipulator moves vertically to collect information from surface points. **4)** Control points (artificial data) are generated and used to form a NURBS surface together with surface points.

Validation is performed on five synthetic 3D-printed surfaces by comparing the original models to the estimations. Results are also compared to a visual approach, using a Kinect to generate cloud points from the surfaces.

4.1.1 Orientation Estimation

The information collected from surface points are normal orientation and position. Position can be determined using the robot interface but normal orientation derives from the tactile data. Let $p_i = [x, y, z]$ be the positional representation in the 3D space of a point of contact. Figure 4.3 shows the robot manipulator and the sensing module. The method 3.1 is employed here.

Given Euler's rotation theorem, any rotation or sequence of rotations in the 3D space can be represented as a pure rotation around a single axis. Given an axis of rotation represented by a unit vector $\mathbf{u} = [u_1, u_2, u_3]$ and an angle of rotation θ , the corresponding quaternion $\mathbf{q} = q_0 + q_1i + q_2j + q_3k$ can be calculated using:

$$q_0 = \cos\left(\frac{\theta}{2}\right) \quad q_1 = u_1 \sin\left(\frac{\theta}{2}\right) \quad q_2 = u_2 \sin\left(\frac{\theta}{2}\right) \quad q_3 = u_3 \sin\left(\frac{\theta}{2}\right) \quad (4.1)$$

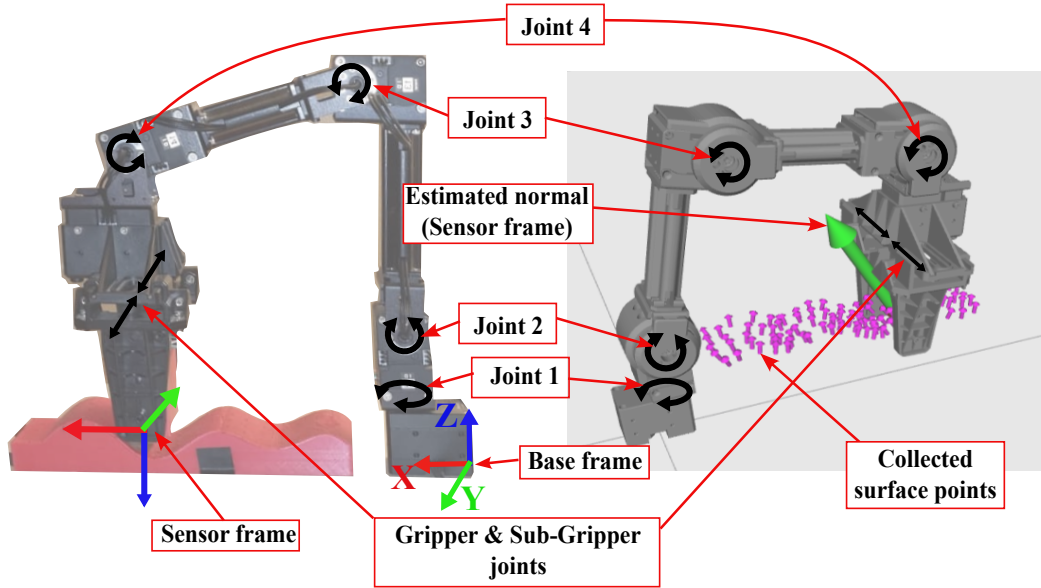


Figure 4.3: Views of the robotic manipulator joints. Joints 1 to 4 rotate to provide 4-DoF. Gripper and Sub-Gripper joints hold the sensing module.

In this formula, q_0 is the scalar part of the quaternion, and q_1 , q_2 , and q_3 are the vector parts along the x , y , and z axes, respectively. The unit vector \mathbf{u} indicates the axis of rotation, and θ represents the angle of rotation around that axis.

This axis-angle representation is a concise way to represent rotations in three-dimensional space using quaternions. It offers several advantages, including the ability to interpolate smoothly between rotations and avoid certain numerical issues associated with other rotation representations like Euler angles.

To determine the orientation of the sensing module, an approximation is created using the accelerometer readings. Let $\vec{a}_s = [a_x, a_y, a_z]$ be the normalized accelerometer information in the sensor frame of reference. In the earth frame of reference (base frame), the unitary vector representing the normal force can be approximated to $\vec{g} = [0, 0, 1]$. Given these two vectors, the quaternion representing the orientation of the sensor is the one representing the rotation from \vec{g} to \vec{a}_s :

$$q_g^{a_s} = [q_x, q_y, q_z, q_w] \quad (4.2)$$

In this case the axis of rotation $[q_x, q_y, q_z]$ is perpendicular to both \vec{a}_s and \vec{g} which can be easily determined as their cross-product $\vec{a}_s \times \vec{g}$. Interpolation is performed between the quaternion that represents zero rotation $q_c^c = [0, 0, 0, 1]$ with the one representing double the intended rotation θ . The sum of these two orientation quaternions produces the intended

quaternion after normalization [66]:

$$\begin{aligned} [q_x, q_y, q_z] &= \vec{g} \times \vec{a}_s + [0, 0, 0] \\ q_w &= 1 + \cos \theta \\ q_w &= 1 + \vec{g} \cdot \vec{a}_s \end{aligned} \tag{4.3}$$

The resulting quaternion $q_g^{a_s}$ is then normalized using the following method for quaternion q normalization:

$$q_{\text{norm}} = \frac{q}{\|q\|} \tag{4.4}$$

Where $\|q\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}$ is the magnitude of the quaternion.

As this approach doesn't take into account the rotation that the robot can perform around the Z-axis of the base link, this is taken into consideration in the final orientation estimation. For that, the TF package in the ROS environment publishes the transformations between the different links in the robot, see Figure 4.3. It is used to create a quaternion representing the rotation around the base. Let $rot_b^s = [rot_x, rot_y, rot_z]$ be the angles representing the rotation from the base link to the end effector, provided by the TF package. A quaternion representing the rotation of rot_z degrees around the Z-axis is created:

$$q_{rot} = [0, 0, \sin\left(\frac{rot_z}{2}\right), \cos\left(\frac{rot_z}{2}\right)] \tag{4.5}$$

The final orientation is a composition of both rotation quaternions q_{rot} and $q_g^{a_s}$, as their multiplication:

$$q_{final} = q_{rot} \otimes q_g^{a_s}. \tag{4.6}$$

Once the first estimate is generated, it has to be adjusted using a Madgwick filter, as mentioned in method 3.1.2, that takes into consideration an initial orientation estimation, accelerometer $\vec{a}_s = [a_x, a_y, a_z]$, and gyroscope $\vec{\omega}_s = [\omega_x, \omega_y, \omega_z]$ readings to provide an updated estimation, using Equation 3.2. This filtering technique is employed using the AHRS library [72].

Once the final orientation is found, it is used as the orientation of the end effector link (green arrow), as shown on Figure 4.3. The ROS node that publishes the green marker keeps track of the orientation of the sensor and awaits for an obstacle signal. When the robotic manipulator detects an obstacle it sends a signal to the marker node which publishes the location of the end effector and orientation (small pink arrows) at that given moment.

4.1.2 Calibration

Vibrations from the robotic manipulator and external factors make it necessary to calibrate the sensor. Before the manipulator starts collecting data on the surface, the sensor is kept

suspended. During 10 seconds, the accelerometer and gyroscope readings are averaged as $\vec{a}_{s,0}$ and $\vec{\omega}_{s,0}$. We will calculate the error for both accelerometer and gyroscope as $\vec{\epsilon}_{acc}$ and $\vec{\epsilon}_{gyr}$ to subtract from all subsequent readings.

In the case of the accelerometer, using the unitary quaternion representing the rotation from the sensor frame to the base frame q_s^b (generated by the TF library), the vector is rotated to generate the average theoretical value in the base frame. Pure quaternions a, ω are extended from vectors \vec{a} and $\vec{\omega}$ in the hamiltonian products.

$$\vec{a}_{b,0} = q_s^b \otimes a_{s,0} \otimes q_s^{b*} \quad (4.7)$$

The error ϵ_{acc} is:

$$\vec{\epsilon}_{acc,0} = \vec{a}_{b,0} - \vec{g} \quad (4.8)$$

In the gyroscope case, the angular velocity's theoretical value should be zero when the manipulator is stopped. Hence,

$$\vec{\omega}_{b,0} = q_s^b \otimes \omega_{s,0} \otimes q_s^{b*} \quad (4.9)$$

The error ϵ_{gyr} is:

$$\vec{\epsilon}_{gyr,0} = \vec{\omega}_{b,0} \quad (4.10)$$

Once the errors are averaged in the base frame, they can be rotated at any instant t using the estimation $q_{final,t}$ to retrieve the errors in the sensing module frame of reference.

$$\vec{\epsilon}_{gyr,t} = q_{final,t} \otimes \epsilon_{gyr,0} \otimes q_{final,t}^* \quad (4.11)$$

$$\vec{\epsilon}_{acc,t} = q_{final,t} \otimes \epsilon_{acc,0} \otimes q_{final,t}^* \quad (4.12)$$

The adjusted accelerometer and gyroscope readings are then given by:

$$\vec{a}_{s,t+1} = \vec{a}_{s,t} - \vec{\epsilon}_{acc,t} \quad (4.13)$$

$$\vec{\omega}_{s,t+1} = \vec{\omega}_{s,t} - \vec{\epsilon}_{gyr,t} \quad (4.14)$$

4.1.3 Control points calculation

Given two surface points p_1 and p_2 in \mathbb{R}^3 , and the respective normal vectors \vec{N}_1 and \vec{N}_2 , the 3D planes *plane1* and *plane2* can be defined. The objective is to find the control point *cp* between the points p_1 and p_2 . This point will define the curvature of the surface patch between the two surface points p_1 and p_2 . The position of *cp* depends on the positions of p_1 and p_2 but also on the orientation of the planes containing the surface points, see Figure 4.4.

The first control point estimation would be the middle point, but using the middle point between p_1 and p_2 as the control point cp , leads to no curvature in the generated surface. To find a control point that can represent the proper curvature, we project the line between p_1 and p_2 on both planes, generating two skew lines. The location of smallest distance between them (point of convergence) gives us a proper estimation of the control point location. Once this is done, we project the control point on the vertical plane passing through p_1 and p_2 so that they are all three on the same vertical plane.

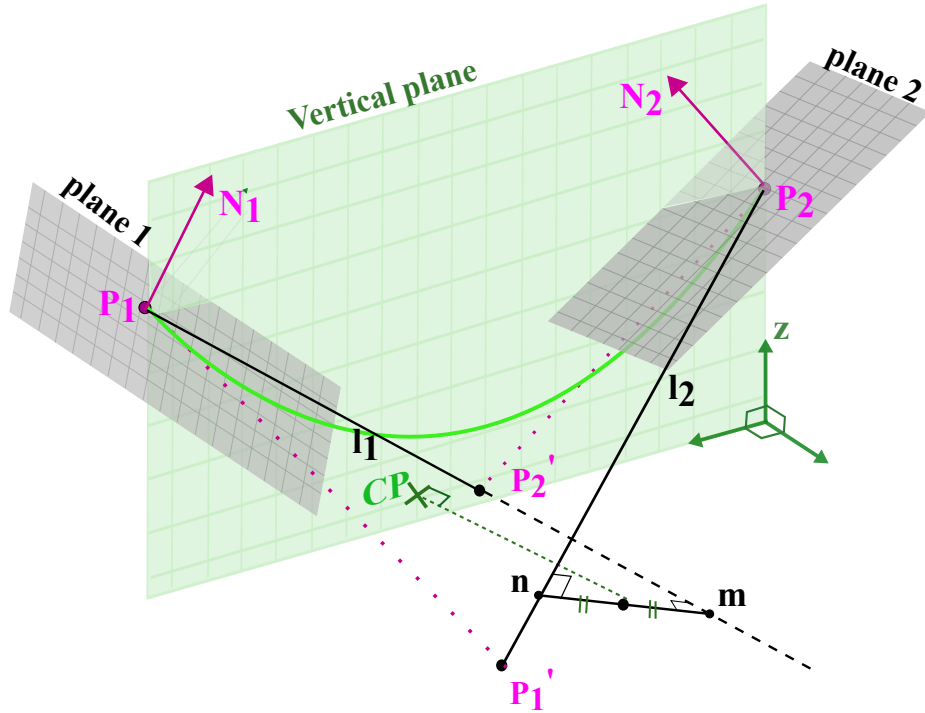


Figure 4.4: Control point (CP) calculation between surface points p_1 and p_2 . The green line is the generated curvature originating at p_1 and finishing at p_2 .

The first step is to find the projections of the surface points on their opposite planes. Let p'_1 denote the projection of p_1 on $plane2$, and p'_2 the projection of p_2 on $plane1$. From the four points two vectors are drawn:

$$\begin{aligned}\vec{l}_1 &= p'_2 - p_1 \\ \vec{l}_2 &= p'_1 - p_2\end{aligned}\tag{4.15}$$

Let these two vectors \vec{l}_1 and \vec{l}_2 define two lines in the 3D space as follows:

$$\begin{aligned}L_1(t) &= p_1 + t \cdot \vec{l}_1 \\ L_2(t) &= p_2 + t \cdot \vec{l}_2\end{aligned}\tag{4.16}$$

These lines are not always concurrent, the location of the points resulting in the smallest distance between them is found, a perpendicular line passing through L_1 and L_2 . Let m and n be these points in \mathbb{R}^3 on L_1 and L_2 , respectively:

$$\begin{cases} m = p_1 + t_1 \cdot \vec{l}_1 \\ n = p_2 + t_2 \cdot \vec{l}_2 \\ (m - n) \cdot \vec{l}_1 = 0 \\ (m - n) \cdot \vec{l}_2 = 0 \end{cases} \quad (4.17)$$

Where t_1 and t_2 are scalar coefficients to be found. Once the location of points m and n are found, the middle point $(m + n)/2$ is projected on the vertical plane passing through p_1 and p_2 , which gives the location of the control point between p_1 and p_2 .

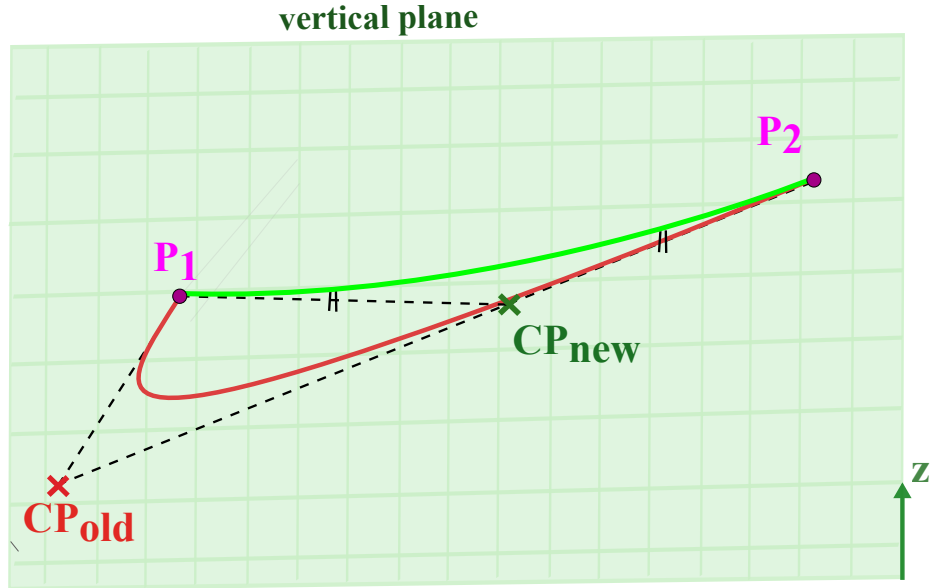


Figure 4.5: CP_{old} represents the wrong estimation of the control point. The red line shows the expected curvature if no correction is applied. CP_{new} represents the adjusted position after the correction is applied. The green line represents the corrected curvature.

In some cases where the surface orientation around two neighbor points p_1 and p_2 is very similar (i.e.: parallel planes), the two skew lines l_1 and l_2 will have their points m and n at an infinite distance of p_1 and p_2 . The projection of $(m + n)/2$ on the vertical plane will likely produce an incorrect control point, as shown on Figure 4.5. To avoid this problem we apply a correction to all control points found, as shown on Algorithm 1.

Algorithm 1 Control point correction

Require: p_1, p_2, cp
if $\text{dist}(cp, p_1) \geq \text{dist}(cp, p_2)$ **then**
 \triangleright Verify which point is farthest to cp
 $\vec{l}_{aux} = p_1 - cp$
else
 $\vec{l}_{aux} = p_2 - cp$
end if
 $d \leftarrow |\vec{l}_{aux}|$
 $cp_{new} \leftarrow cp$
 $\delta \leftarrow 0.0001$
 \triangleright Arbitrary value

if $d \geq |p_1 - p_2|$ **then**
while $|cp_{new} - p_1| \neq |cp_{new} - p_2|$ **do**
 $cp_{new} \leftarrow cp_{new} + \delta \cdot \vec{l}_{aux}$
 \triangleright Move CP_{old} along \vec{l}_{aux} towards CP_{new}
end while
end if

4.1.4 Robotic motion

To collect the data on top of the synthetic surfaces and objects, the position of each surface point is manually selected. Once the surface points are defined, the robotic manipulator moves in a vertical motion to approximate the tactile sensing module to the surface. When contact is made, the manipulator stops and the tactile information is collected. The procedure is shown on Figure 4.6.

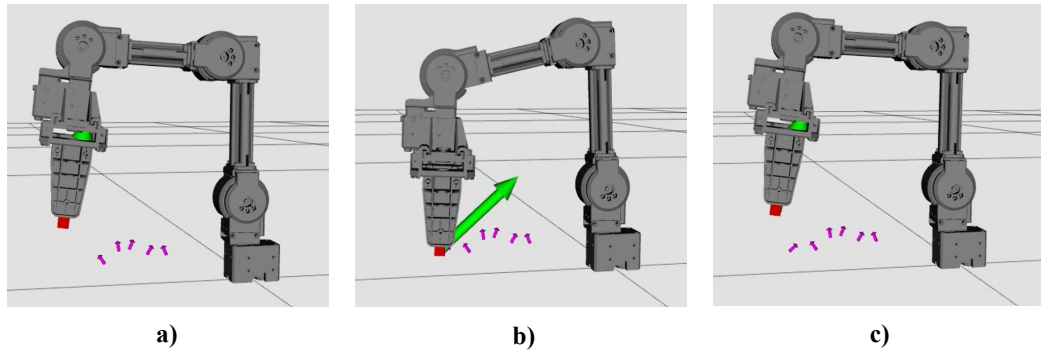


Figure 4.6: Collection of points on the surfaces. **a)** Robotic manipulator approaches point of contact in a vertical motion. **b)** Once contact is perceived, manipulator stops at point of contact. Position and orientation are collected leaving a marker (pink arrow) at the location. **c)** Manipulator retreats in a vertical motion.

4.1.5 NURBS surface generation

To generate the surface patches, the method 3.3.1 is employed. The control points must be organized in a grid together with the surface points. Each patch is composed of nine points: four adjacent surface points, four control points calculated using the previously mentioned method, and one central control point which is the average of the other four. See Figure 4.7.

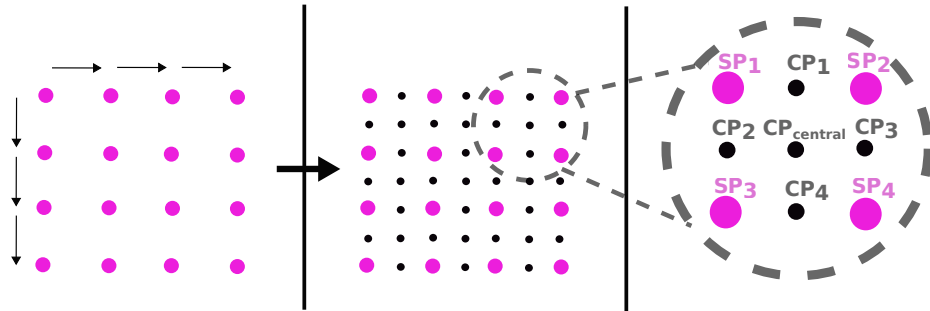


Figure 4.7: Control points layout. Surface and control points are placed in a grid to generate the surface patches. Each patch is composed of four surface points (SP_1 , SP_2 , SP_3 , SP_4) and five control points (CP_1 , CP_2 , CP_3 , CP_4 , and $CP_{central}$). The central control point is an average of the other four.

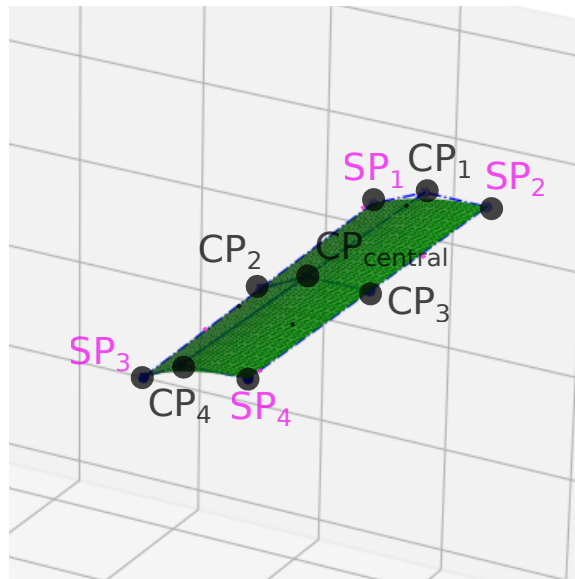


Figure 4.8: Example of a surface patch arranged in a 3×3 layout.

To generate the NURBS surface, we must set the parameters such as curves order and knot vectors. Since the control points of every patch are displayed in a 3×3 shape, we use

quadratic curves in both directions (order $k = 3$). The chosen knot vector is the uniform one, forcing the curves to pass through the initial and ending control points but not through the middle ones. Given the number of control points and the order of the curves, the knot vectors in both directions will be: $v = [0, 0, 0, 1, 1, 1]$.

4.1.6 Experimental setup

The bio-inspired multi-modal sensing module proposed in [73] [74] is used in this work. It is comprised of a LSM9DS0 MARG (Magnetic, Angular Rate, Gravity) sensor and a MPL115A2 Barometer pressure sensor.

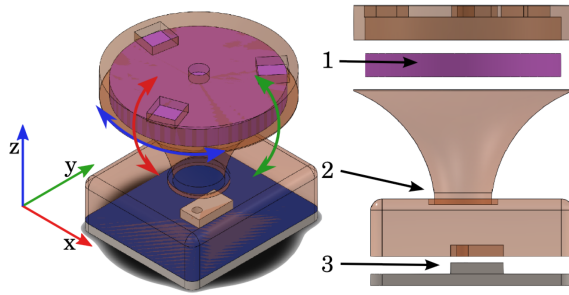


Figure 4.9: Tactile sensing module components: 1—MARG (magnetic, angular rate, and gravity) system; 2—compliant structure; 3—barometer [74].

These sensors are embedded in a flexible polyurethane and its structure is shown in Figure 4.9 and Figure 4.10. The MARG consists of a triple-axis accelerometer, a triple-axis gyroscope, and a triple-axis magnetometer. To send the sensors readings in the appropriate format we use a Teensy 3.2 microcontroller [76]. It sends information via a ROS serial publisher node at 250000 baud to the Ubuntu 16.04 machine.

The robotic manipulator used in this work is the OpenMANIPULATOR-X RM-X52-TNM [77]. It contains 5 actuators, with 4 of them being joints and the last one being the gripper. Joint 1 allows the robot to move around the base while the other joints can only move in the plane defined by the position of joint 1. This allows for 4 Degrees of Freedom (DOF). It holds the sensory module with the gripper that fits the 3D-printed structure. To communicate the commands to the manipulator we use the Robot Operating System (ROS) [78] and the embedded board OpenCR 1.0 which is an open-source control module for ROS. See Figure 4.11.

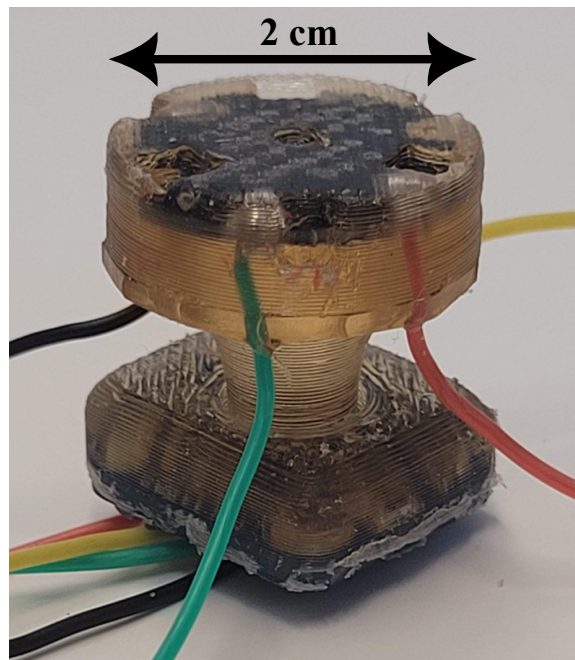


Figure 4.10: Sensing module encapsulated in flexible structure.

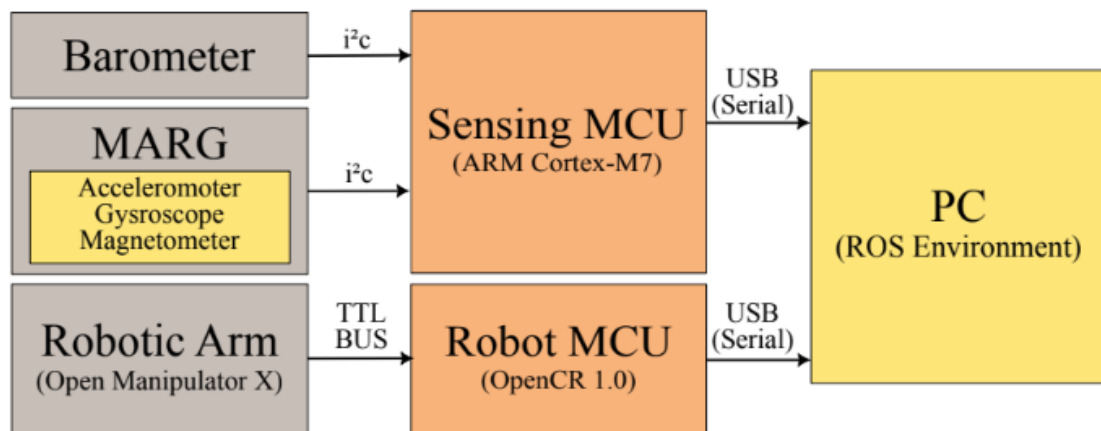


Figure 4.11: System Layout. Connections between the compliant tactile sensing module (barometer and MARG) to its respective sensing MCU (ARM Cortex-M7); the robotic arm (Open Manipulator X) connection to the robot MCU (OpenCR 1.0) and their MCU connections to a PC running a ROS environment.

4.1.7 Surfaces

Synthetic surfaces

To test this approach, five artificial surfaces, 3D printed, have been used. They present different shapes that will be estimated and the comparison will be made between the STL files of the original surfaces and the ones generated from the estimation. See Figure 4.12.

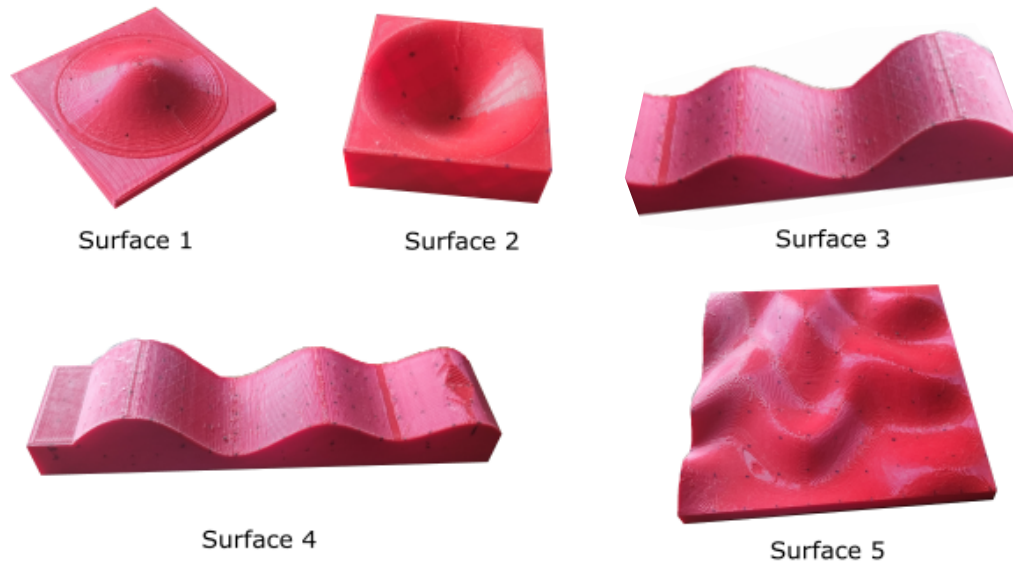


Figure 4.12: The five synthetic surfaces under study. Surface 1 is $8\text{cm} \times 8\text{cm}$ with 1cm peak height. Surface 2 is $8\text{cm} \times 8\text{cm} \times 2\text{cm}$ with a lowest height at 1cm . Surface 3 is $16\text{cm} \times 5\text{cm}$ with highest point at 2.5cm and lowest at 1cm . Surface 4 is $19\text{cm} \times 4\text{cm}$ with highest point at 2.5cm and lowest at 1cm . Surface 5 is $20\text{cm} \times 16\text{cm} \times 1\text{cm}$.

The number of total surface points will be different for each surface. The sensing-module with the flexible frame has 2cm in diameter, so every surface point will be 2cm away from any other for no overlapping. On surface 5 there will be $11 \times 9 = 99$ points collected. Surface 3 and 4 will have $3 \times 9 = 27$ and $3 \times 10 = 30$ points collected, respectively. Surface 1 and 2 will have $5 \times 5 = 25$ points collected.

Everyday objects

Four everyday objects are also used to evaluate the present approach. These objects are: a mug, shampoo bottle, deodorant can, and hair comb. See Figure 4.13. As these surfaces have no ground truth graphic model, the validation is performed by comparing the tactile estimations using the present approach with the estimations from a visual method as is

explained in the next section.



Figure 4.13: The four common objects used to validate the present approach. Mug has a lower and higher radius of 4cm and 3cm , respectively, and a height of 13cm . Shampoo bottle is $8\text{cm} \times 11\text{cm} \times 3\text{cm}$. Hair comb is $11\text{cm} \times 7\text{cm} \times 5\text{cm}$. Aerosol deodorant has a radius of 2cm .

4.1.8 Validation

Methodology

To validate this approach, we must compare the generated estimation to the original surface. For that STL files of each surface estimation are generated and compared to the STL of the original figure. A 3D point cloud and triangular mesh processing software called CloudCompare is used. CloudCompare [79] provides a set of basic tools for editing and rendering 3D point clouds and triangular meshes.

The tool randomly samples points on a mesh to generate a point cloud. The original point cloud and the estimated one must be aligned to accurately measure the error metric. Two types of alignment are used:

- Manual alignment. Using the “Align (point pairs picking)” tool available in CloudCompare, which allows users to pick several pairs of equivalent points in the two entities and then align both figures to approximate the equivalent points.
- Fine alignment or Iterative Closest Point (ICP). An algorithm that iteratively revises the transformation (combination of translation and rotation) needed to minimize an error metric, usually a distance from the source to the reference point cloud.

After the cloud points and meshes are aligned using the alignment tools above, three distance metrics are used to evaluate the reconstructions. Cloud-to-Cloud(CC) between the

original surfaces' point cloud and the estimated surfaces' point cloud, and Cloud-to-mesh (CM) between the original point cloud and the estimated mesh.

To calculate the cloud distance, the chosen metric is the Hausdorff distance $H(A, B)$, which can be translated as the maximum distance from set A to the nearest point in set B :

$$H(A, B) = \max \{h(A, B), h(B, A)\} \quad (4.18)$$

The Hausdorff distance H between two subsets A and B is the maximum of the directed Hausdorff distances from A to B and from B to A .

Equation for the directed Hausdorff distance from subset A to subset B :

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \quad (4.19)$$

The directed Hausdorff distance h from subset A to subset B is calculated by considering each point a in A and finding the minimum distance to any point b in B . The overall directed Hausdorff distance is the maximum of these minimum distances. The symbol $\|a - b\|$ represents the Euclidean distance between points a and b in three-dimensional space.

- **Cloud-to-Cloud (CC):** CC is calculated by first creating partitions of the original surface using an octree structure. These partitions reduce the number of comparison between points in each partition, but increase the number of parallel calculation in each partition. In the present case, the octree level is automatically specified by CloudCompare. For every partition, a subset A_i of the reconstruction A is aligned with a subset B_i of the original cloud B to generate $(CC)_i$. The distances in each partition are averaged to generate the final CC distance. For every octree structure i there are two points a_i in the reference cloud and b_i in the mesh that satisfy $H(A_i, B_i)$.

$$\begin{aligned} \vec{u}_i &= \frac{b_i - a_i}{\|b_i - a_i\|} \\ \vec{CC}_i &= H(A_i, B_i) \cdot \vec{u}_i \\ \vec{CC} &= \sum_{i=1}^k \frac{\vec{CC}_i}{k} \\ CC &= \|\vec{CC}\| \end{aligned} \quad (4.20)$$

- **Unsigned Cloud-to-Mesh (uCM):** To reduce uncertainties related to measurement, the Cloud-to-Mesh distance is also calculated by comparing the estimated cloud to the original mesh. In this case distances are all given in absolute value as for the CC. The same technique using octree structures are used to create aligned partitions of both the cloud A_i and the mesh B_i . In the same manner as the CC distance uCM

is given by the following:

$$(uCM)_i = H(A_i, B_i)$$

$$uCM = \sum_{i=1}^k \frac{(uCM)_i}{k} \quad (4.21)$$

- **Signed Cloud-to-Mesh (sCM):** Using the same idea as the previous metrics, for every octree structure i there are two points a_i in the estimated cloud and b_i in the reference mesh that satisfy $H(A_i, B_i)$. Let \vec{n}_i be the unitary normal vector to the mesh at b_i . The sCM distance can be computed as the following:

$$s\vec{CM} = \sum_{i=1}^k \frac{H(A_i, B_i) \cdot \vec{n}_i}{k}$$

$$sCM = ||s\vec{CM}|| \quad (4.22)$$

Comparison

To evaluate the results, a comparison is performed between the present approach and a visual one. Using a Kinect, which is equipped with 640×480 pixels 30 Hz RGB camera, and 640×480 pixels 30 Hz Infra-Red depth-finding camera, cloud points are generated from the five surfaces. In a ROS environment, rosbag files containing the point clouds are collected. They are then converted to pcl files, and the final isolated point cloud from the surface is generated on CloudCompare. See Figure 4.14. The alignment is performed and the metrics are calculated both using the CloudCompare software following the same procedure for the tactile estimations.

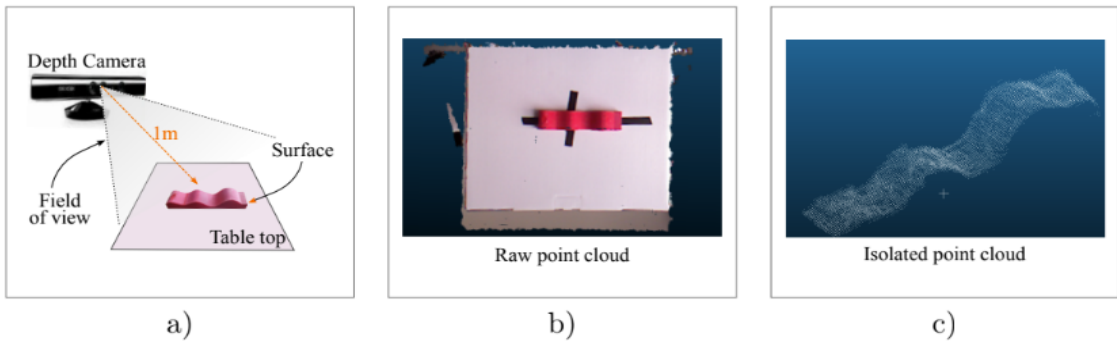


Figure 4.14: Kinect setup for point cloud extraction. The point clouds are generated using rosbag files in the ROS environment. rosbag files are then converted to pcl files and the region of interest is isolated on CloudCompare.

4.2 Pressure-Control System

4.2.1 Waypoints adjustment

The amount of pressure applied on the tactile module will change the extracted features from the exploratory motions. To maintain a constant pressure throughout the whole trajectory it is necessary to adjust the goal position of the end-effector on several waypoints to reduce the impacts of change of pressure on the tactile data. This adjustment of position can be seen on Figure 4.15.

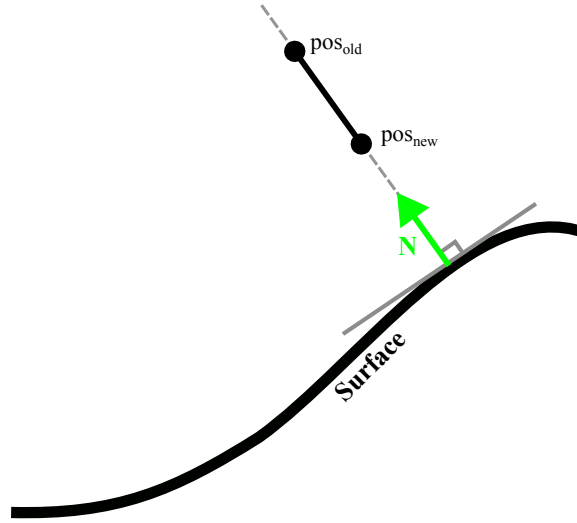


Figure 4.15: Calculating new end-effector position pos_{new} based on the old position pos_{old} , and normal direction \vec{N} .

Given an estimated position pos_{old} of a waypoint, the normal unitary direction \vec{N} at the point of contact, the new position pos_{new} can be calculated on Equation 4.23.

$$pos_{new} = pos_{old} + \vec{N} \cdot \delta \quad (4.23)$$

Where $\delta \in \mathbb{R}$ is to be estimated given the current and desired barometer level, and $\vec{N} \in \mathbb{R}^3$ is the unitary normal vector. Equation 4.23 can be translated to the following:

$$pos_{new} = pos_{old} + \vec{N} \cdot step \cdot action \quad (4.24)$$

In Equation 4.24, the amplitude of movement is limited using a $step = 0.005m$ factor. This limits the maximum distance that the end-effector can move during the fine-adjustment phase. This way, our estimated variable becomes $action \in [-1; 1]$.

4.2.2 Exploratory Motions

The position and orientation of several points are collected with an average spacing of 2cm between each point of contact. Once enough points are collected to cover the whole surface, the reconstruction is created using NURBS as shown on Figure 4.16. These graphical models provide several coordinates $p_i = [x_i, y_i, z_i]$ of estimated points that can be used as goal poses for the manipulator to perform exploratory motions.

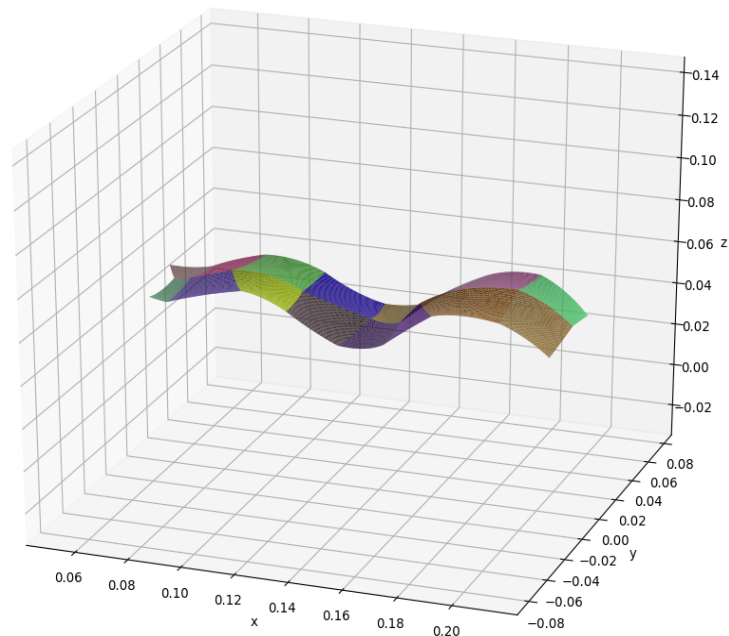


Figure 4.16: Reconstruction of surface using NURBS. Every colored piece is a single patch created from four surface points. These patches are put together and form the overall shape.

To perform smooth trajectories on surfaces the manipulator needs to maintain a constant angle of approach with the surface as well as a constant relative position. See Figure 4.17.

For that we use the graphic model of the surface to manually determine waypoints p_1, p_2, p_3, \dots that will compose the trajectory. For each waypoint, the manipulator moves the sensing module to its location and aligns itself with the normal direction of the surface. Once aligned, the relative position between end-effector and surface is adjusted and the current pose of the manipulator can be stored as a 4D vector $pose_i = [joint1_i, joint2_i, joint3_i, joint4_i]$, where $joint_i$ represents the value in *rads* of the joint's position. We adjust waypoints with an average distance of 1cm between them.

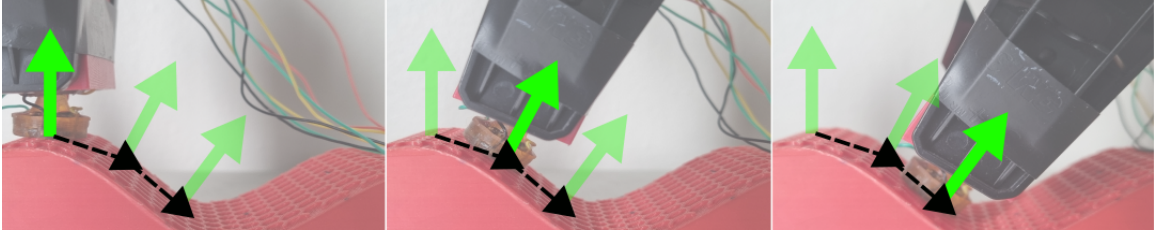


Figure 4.17: Example of correct trajectory on curved surface. The manipulator maintains a constant angle of approach and pressure on the surface.

4.2.3 Reinforcement Learning

As we are dealing with a robotic application in the real world, many factors can influence the true position and orientation of the end-effector making it difficult to model *action* based on barometer level only. For that we employ RL on the Gym API [45]. The employed method is explained in section 3.2.1. The Gym environment is an open-source simplification tool responsible for the RL framework. Therefore, five main elements are defined for training: the action and observation spaces, the two methods *reset()* and *step()*, and the reward function.

Algorithm 2 Reset

Require: $p_i \in \mathbb{P}$, where $p_i = [x_i, y_i, z_i]$.

- 1: Manipulator moves to initial pose *home*.
 - 2: Manipulator moves to p_i .
 - 3: End-effector is aligned with surface normal at p_i .
-

- The observation space *obs* is dictionary composed of joint efforts $effs \in \mathbb{R}^4$ in *N.m*, barometer level $baro \in \mathbb{N}$, and orientation $orient \in \mathbb{R}^3$. It describes the environment to the agent and is used to compute actions. These variables *var* are normalized using min-max normalization, as shown on Equation 4.25.

$$var_{norm} = \frac{var - var_{min}}{var_{max} - var_{min}} \quad (4.25)$$

- The action space contains all actions that can be executed by the agent. In this case our action is a single value $a \in [-1; 1]$.
- The *reset()* method is responsible for taking the agent back to an initial valid state where the agent receives the initial observations obs_{ini} from the sensing module. In our case, we are performing a fine-adjustment of the position which implies that the manipulator is already in contact with the surface. This means that, during this phase, it first retreats to an initial pose called *home*, making no contact with the

surface. Then a point p_i is sampled from the graphical model \mathbb{P} and the manipulator moves to the given location on the surface. After aligning the manipulator with the surface normal vector, the initial observations obs_{ini} are sent to the $step()$ function. See Algorithm 2.

- The $step()$ method is responsible for sending the state S_t to the policy through the observations obs_t and getting and action a_t back, executing it and calculating the reward r_t based on the resulting observations. Its also responsible for verifying that the agent has reached its goal, ending the training episode. We condition the end of episode to happen under two scenarios:
 - $n_{steps} = 50$.
 - $baro \leq 0$ or $baro \geq 350$.

This allows the manipulator to keep changing the initial position p_i at the end of every episode for generalization purposes, and also avoid loosing contact with the surface or applying too much pressure on it. See Algorithm 3.

- The reward function $f(baro)$ is a function of the barometer level only, as shown on Figure 4.18. The barometer levels range from -2 to 500 . We select an optimal region $[150 - 250]$ where the reward will be maximal and give zero rewards when the agent reaches a terminal state (too much pressure or lack of contact). See equation 4.26.

$$\begin{cases} baro > 350 \text{ or } baro < 0 \implies reward = 0 \\ baro \in [0; 150] \implies reward = baro/150 \\ baro \in [250; 350] \implies reward = \frac{35}{10} - \frac{baro}{100} \end{cases} \quad (4.26)$$

Algorithm 3 Step

Require: $obs; p_i \in \mathbb{P}$, where $p_i = [x_i, y_i, z_i]$.

- 1: $orient, efs, baro \leftarrow obs$ \triangleright From the observation space the model receives real-time input.
 - 2: $a \leftarrow Policy(obs)$ $\triangleright a \in [-1; 1]$
 - 3: $pos_{new,i} \leftarrow p_i + orient \cdot step \cdot a$ $\triangleright step = 0.005m$
 - 4: Move manipulator to pos_{new} \triangleright Equation 4.24.
 - 5: $reward \leftarrow f(baro)$. \triangleright Figure 4.18.
 - 6: Checks if terminal state has been reached.
-

Once the five elements have been defined, the RL training can occur. Each training iteration is composed of several episodes. The number of episodes can vary depending on

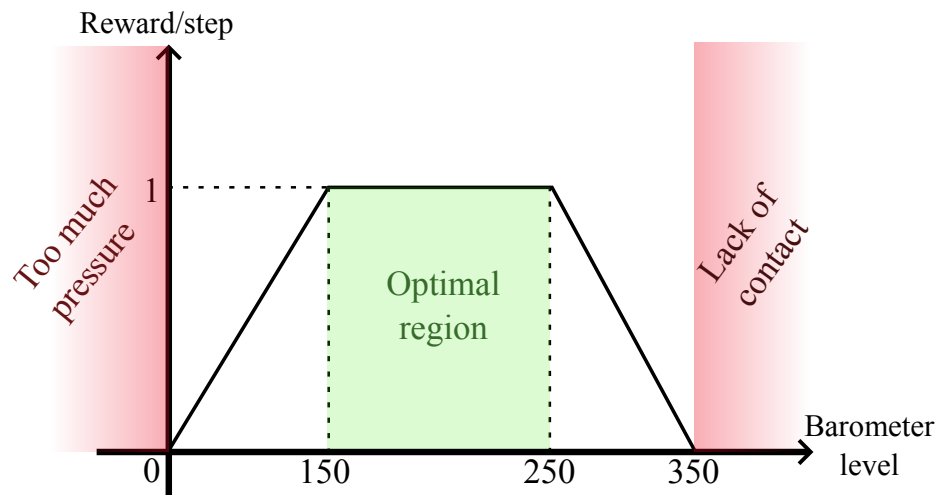


Figure 4.18: Reward as a function f of the barometer level. The maximum reward occurs when the barometer level is within an optimal region $[150 - 250]$. Red regions indicate terminal states for the manipulator with zero reward.

how soon the agent reaches a terminal state. There are 2048 steps per training iteration, meaning that the policy will be updated after a total of 2048 steps have been executed over how many episodes are necessary for that. Each training episode can be described as on Figure 4.20.

Algorithm 4 Training iteration

Require: $obs; p_i \in \mathbb{P}; total\ timesteps.$

- 1: $n \leftarrow 0$
 - 2: $done = False$
 - 3: **while** $n_{steps} \leq total\ timesteps$ **do** ▷ Loop for episodes.
 - 4: $obs_{ini} \leftarrow Reset(\mathbb{P})$
 - 5: $obs_t \leftarrow obs_{ini}$
 - 6: **while** $done = False$ **do** ▷ Loop for steps.
 - 7: $obs_{t+1}, r_{t+1}, done \leftarrow Step(obs_t)$
 - 8: $n_{steps} \leftarrow n_{steps} + 1$
 - 9: **end while**
 - 10: **end while**
 - 11: Update policy with PPO.
-

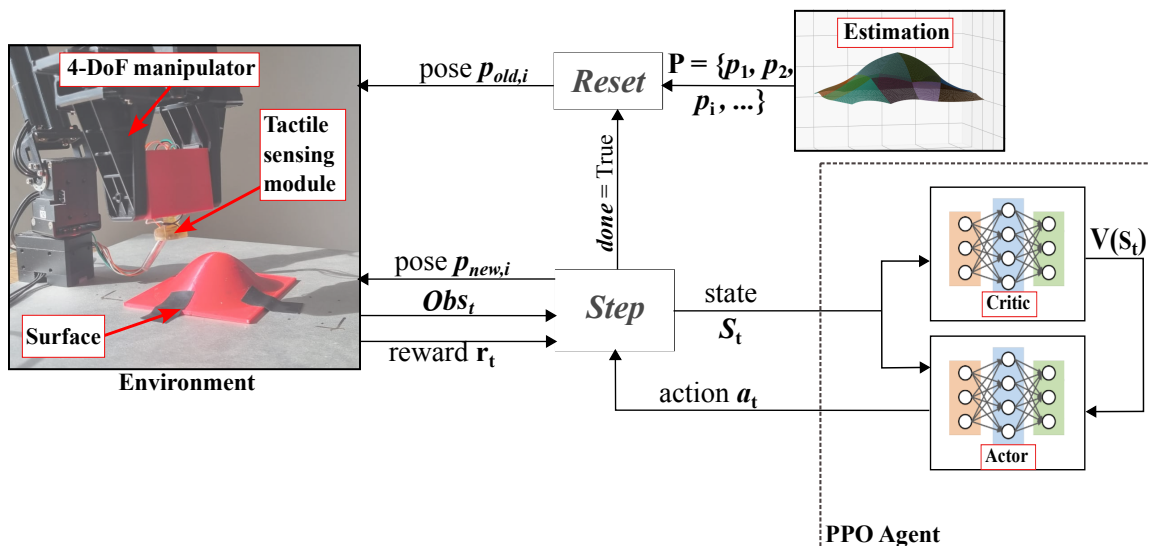


Figure 4.19: Framework for RL training.

Each training iteration starts with the $Reset()$ function taking the agent back to an initial state which can be at any position p_i given by the graphical model \mathbb{P} . The initial observations are passed to the $Step()$ function which starts looping while the $done$ signal is set to False. Once the done signal changes to True, the episode is ended, sequences of states and cumulative rewards are stored and the new episode can begin. Once n_{steps} reaches $total\ timesteps$ the iteration is completed and the policy is updated using PPO. See Algorithm 4 and Figure 4.19.

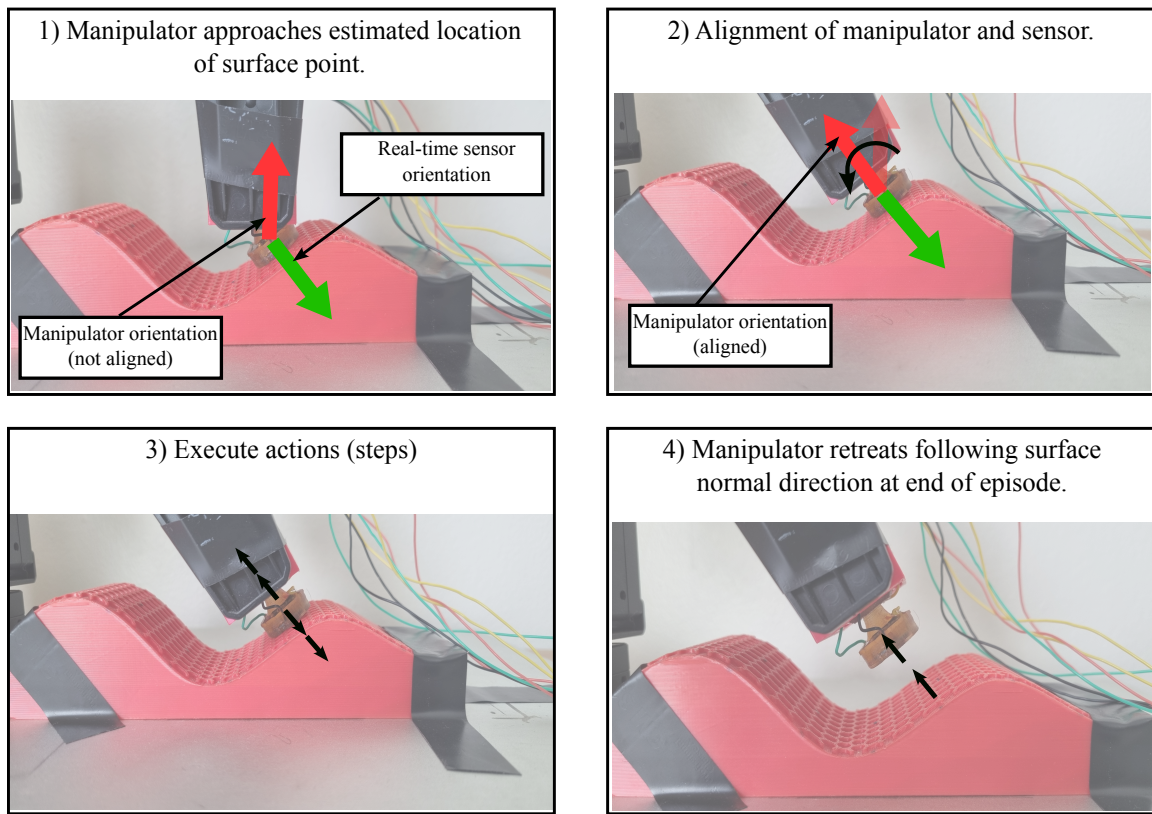


Figure 4.20: Training episode. The manipulator starts moving to the estimated point in a vertical pose, and adjusts its orientation to face the same direction of the surface normal. Once aligned, it can start executing steps and eventually end the episode when conditions are met.

4.2.4 Surface reconstruction with adjustment

The pressure adjustment system was tested on the surface reconstruction approach. Using the same methodology employed on the previous section, points are collected on the five synthetic surfaces shown on Figure 4.12. The amount of points and distance between them was kept the same. The objective is to verify if the surface reconstruction system can achieve better estimates when the manipulator aligns itself with the surface and the position is adjusted to maintain acceptable pressure levels from the barometer. Each surface point is adjusted using the RL system and once all points are collected the estimate can be created using the same control points calculation and NURBS surface methodology.

4.3 Texture Classification

The last part of this work consists of texture classification using the previous approaches for surface estimation and pressure-control. See Figure 4.21 that shows the framework

employed.

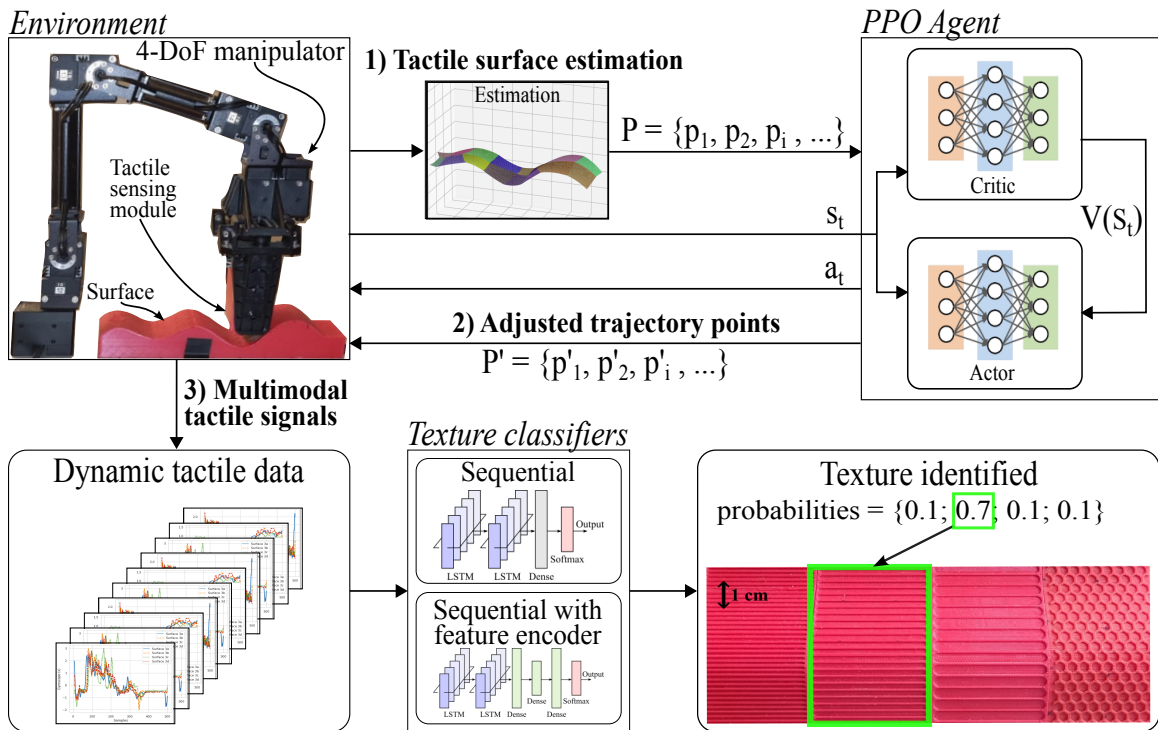


Figure 4.21: **1)** Estimations are generated using the present approach. **2)** Points are chosen to form an exploratory trajectory and their position is adjusted through RL. **3)** Tactile data from sensor readings are collected during the exploratory motion. **4)** An LSTM classifier uses the data to identify the texture at hand.

4.3.1 Textures

To perform classification four different textures on identical shapes are used. These figures are 3D-printed in 16 cm by 5 cm sizes as shown on Figure 4.22. Textures 3a, 3b, and 3c have horizontal striped with a spacing of 0.5 cm , 0.3 cm , 0.2 cm , and surface 3d has 0.5 cm hexagons. Each surface is placed at the same position under the manipulator and three different trajectories are generated from the estimations.

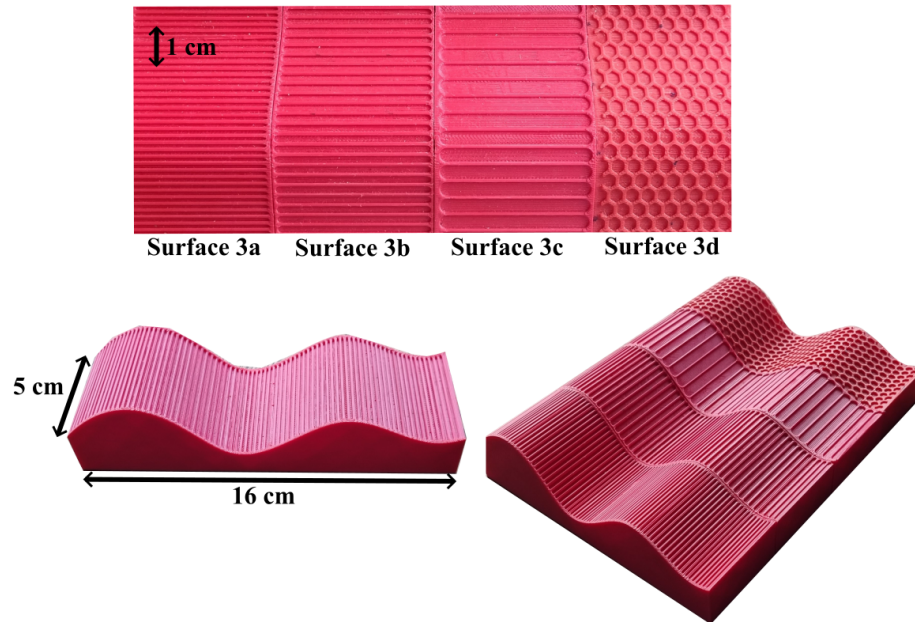


Figure 4.22: Four identical synthetic surfaces used for classification with four different textures.

4.3.2 Trajectories

Each trajectory is composed of three surface points (waypoints) with an average distance of 1cm between them. The step-by-step is described on Figure 4.23. The exploratory motions are executed 25 times for data collection over each of the four textures for training. For validation purposes we collect again 5 times tactile data from each trajectory on the four surfaces.

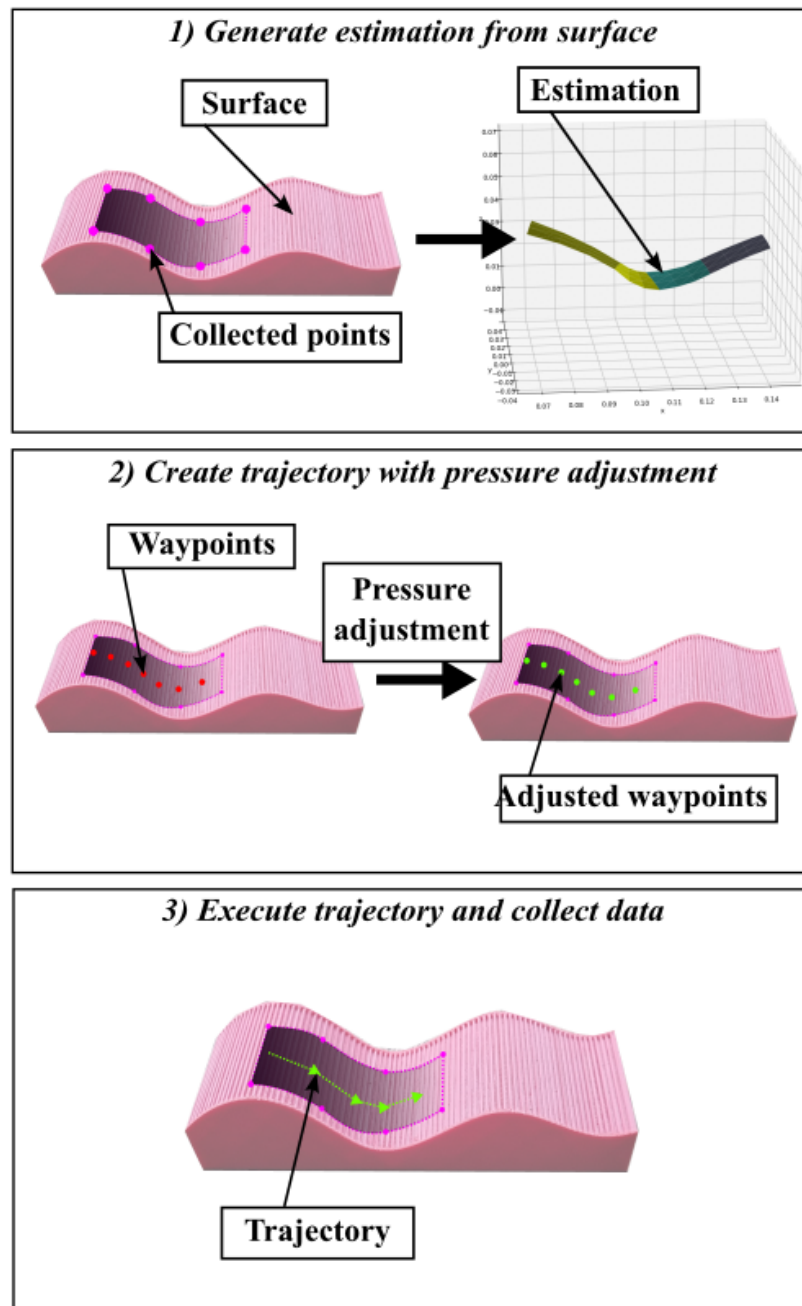


Figure 4.23: Trajectory generation from a surface patch. **1)** Collect surface points and generate an estimation. **2)** Waypoints are chosen to form a trajectory, the manipulator goes to each of the waypoints to adjust the pressure and store the current joint values. **3)** The trajectory is executed.

4.3.3 Data collection & preprocessing

The tactile data used for classification comes from 9-DoF MARG:

$$(ax, ay, az, gx, gy, gz, mx, my, mz)$$

and 1-DoF barometer (*baro*) sensors forming a 10-D time-series dataset. Each sample for the RNN model is composed of 200 frames sampled at $120Hz$.

Before classification, the dataset is filtered to only keep data during the manipulator motion. We also apply a condition on the dataset to only keep frames where the manipulator is in touch with the surface with the condition $baro \leq 350$. We normalize each of the ten variables *var* using equation 4.27 where μ_{var} is the mean of the variable, and σ_{var} is the standard deviation.

$$var_{norm} = \frac{var - \mu_{var}}{\sigma_{var}} \quad (4.27)$$

4.3.4 Feature Encoding

To increase the accuracy of the classifier it is necessary to reduce the effects of noise that can originate from many factors such as bumps during trajectories, sensor noise, vibrations from the manipulator, etc... For that, the technique employed here uses a Variational Autoencoder as a feature encoder. The method is explained in section 3.4.1. It can be trained to encode features from the feature extractor module in a latent space and retrieve the information in the decoder part. Thanks to the different objective functions of the classifier and feature encoder it is possible to combine both structures to minimize the effects of noise and increase classification results.

This technique is easy to implement and the encoding architecture represents a small increase in parameters in the resulting model. The implementation of the classifier with feature encoding is shown in the next subsection.

4.3.5 LSTM classifier

Classification is performed with TensorFlow and Keras as the APIs for the neural network model. The employed method is mentioned in section 3.4.2. The model is composed of an LSTM layer with 200 units and *return sequence = True*, a second LSTM layer with 200 units and *return sequence = False*, a 20% Dropout layer, and an output dense layer with four neurons. The activation function is Softmax as shown on Equation 4.28 for the output x_i of neuron i and we use categorical cross-entropy as our loss function. The resulting model

has 1.3M trainable parameters and is trained for 200 epochs.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_1^4 \exp(x_j)} \quad (4.28)$$

A comparison of the classifier with and without feature encoding is performed to validate the effects of the feature encoding. For that, initial classification is performed with the classifier but without the encoder. This methodology was proposed in [75], it consists of adding a VAE to the classification architecture between the feature extractor and the output layers. Once the training is finished, the VAE is trained using the features generated by the frozen LSTM layers. The VAE is frozen and the encoder part of the VAE is used as feature encoder between the LSTM feature extractor and the classification layers. See Figures 4.24.

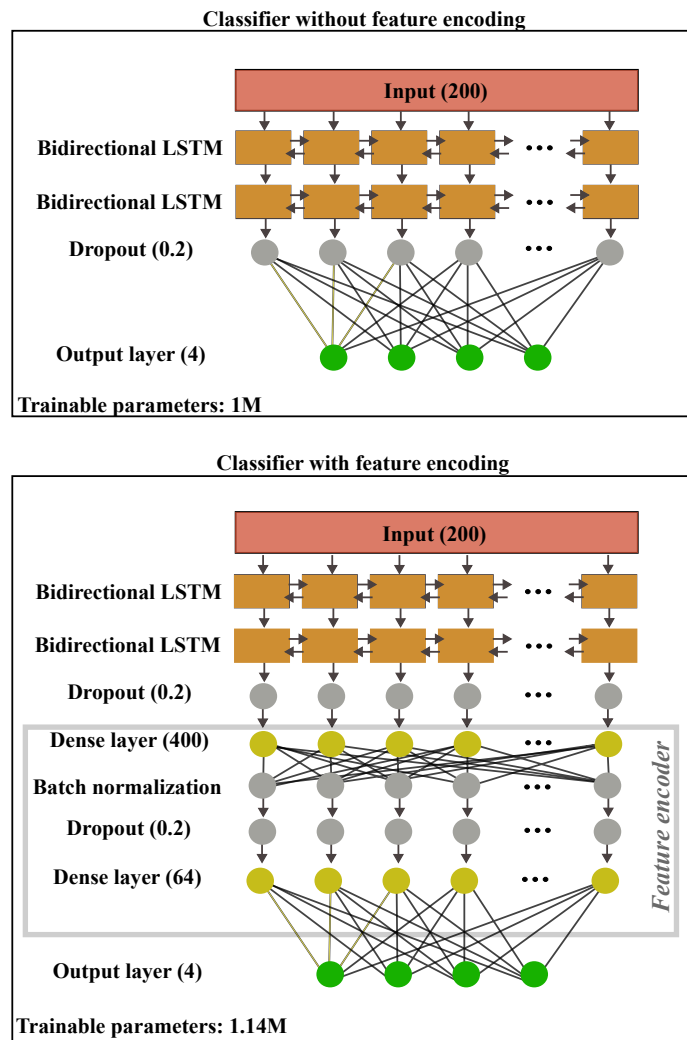


Figure 4.24: Both classification architectures without and with the feature encoder.

Chapter 5

Results & Discussion

5.1 Surface Reconstruction

5.1.1 Synthetic surfaces

Results from the present approach using tactile data are shown on Table 5.1 and those from the vision approach using the Kinect are on Table 5.2. The comparison between estimations and original shapes is shown in Figure 5.3.

In the present method, the average Cloud-to-Mesh (C/M) distance is 0.652 mm and Cloud-to-Cloud (C/C) is 0.637 mm. The signed C/M and the signed X, Y, Z components of the C/C distance have significant lower values than the unsigned C/C and C/M. This shows that there is no apparent bias in the final estimation and no outstanding error towards any of the scalar fields (X, Y, Z).

Surfaces	Unsigned C/M (mm)	Signed C/M (mm)	C/C (mm)	C/C (X) (mm)	C/C (Y) (mm)	C/C (Z) (mm)
1	0.75 ± 0.53	0.017 ± 0.93	0.75 ± 0.54	-0.009 ± 0.32	0.01 ± 0.38	0.005 ± 0.43
2	0.70 ± 0.52	0.011 ± 0.90	0.69 ± 0.48	-0.005 ± 0.48	0.02 ± 0.42	0.009 ± 0.41
3	0.53 ± 0.42	0.014 ± 0.95	0.49 ± 0.37	0.044 ± 0.36	0.004 ± 0.75	0.016 ± 0.12
4	0.72 ± 0.57	0.012 ± 0.88	0.67 ± 0.60	0.003 ± 0.59	0.007 ± 0.72	-0.017 ± 0.14
5	0.55 ± 0.42	0.003 ± 0.75	0.59 ± 0.44	0.012 ± 0.17	0.008 ± 0.25	0.009 ± 0.57

Table 5.1: Validation results from tactile approach, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).

Signed C/M distances show an average standard deviation of 0.88 mm. Considering that the patches have an approximate size of 2×2 cm, the percentual error can be estimated at 4.5% for surfaces 1 to 5.

We compare our results to the Kinect approach, shown on Figure 5.1, and notice an increase in the Unsigned C/M and C/C distances. The average increase is 60% in comparison to the tactile approach. Signed C/M also show a significant increase of more than 10 times.

Surfaces	Unsigned C/M (mm)	Signed C/M (mm)	C/C (mm)	C/C (X) (mm)	C/C (Y) (mm)	C/C (Z) (mm)
1	1.05 ± 0.92	0.12 ± 1.13	1.02 ± 0.94	0.01 ± 0.79	0.003 ± 0.85	0.001 ± 0.75
2	0.98 ± 0.89	0.12 ± 1.02	0.99 ± 0.88	-0.009 ± 0.88	0.001 ± 0.87	-0.001 ± 0.5
3	0.96 ± 0.82	0.15 ± 1.31	1.03 ± 0.91	0.013 ± 0.90	-0.001 ± 0.98	0.001 ± 0.61
4	1.02 ± 0.8	0.17 ± 1.25	1.07 ± 0.94	-0.019 ± 0.98	0.005 ± 0.72	0.002 ± 0.93
5	1.23 ± 0.97	0.21 ± 1.127	1.26 ± 1.07	0.01 ± 0.79	0.010 ± 0.85	-0.002 ± 0.75

Table 5.2: Validation results from vision approach, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).

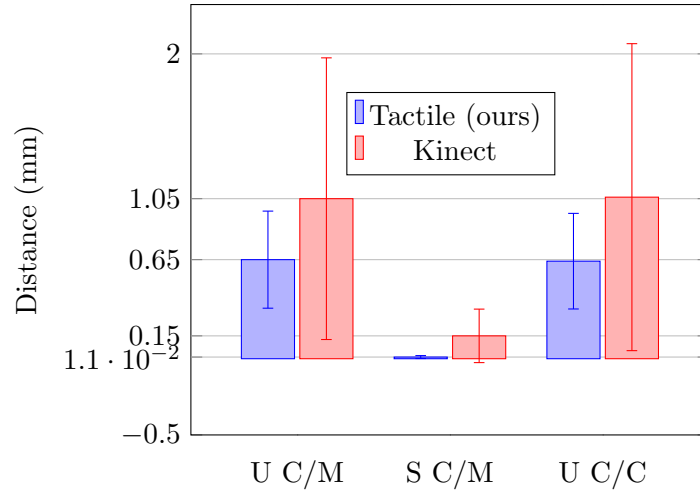


Figure 5.1: Distance comparison between our approach and vision using Kinect. Unsigned C/M (U C/M), Signed C/M (S C/M), Unsigned C/C (U C/C). Mean \pm standard deviation.

Because of the size of the sensor case, the manipulator can touch the surface before the sensing module, producing wrong measurements (lack of contact) when collecting points in small concavities. This can be avoided by using a smaller sensor case or distancing the sensing module from the gripper. This was the case for Surface 2. To better capture the surface points inside the concavity, we dislocated the sensing module $+2\text{cm}$ in the Z direction of the end-effector link.

Images from the scalar fields generated by CloudCompare, as shown in Figure 5.2 reveal that points on the border produce some of the largest distances for all the surfaces. This happens because of the sensing module size. When collecting points on the edge of the surface, the sensor tilts slightly off the surface, changing the orientation estimate and the patches attached to those points.

Results in [25] show distances from the generated meshes to the surface ranging from 0.1 mm to 0.23 mm to the three surfaces under study (shell, pear, and dark pebble). Nonetheless, their overall estimated surface is significantly smaller (around $2 \times 2\text{ cm}$) than the ones being tested here. They also use a larger number of tactile data points (between

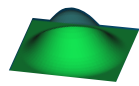
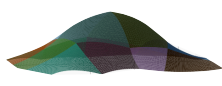
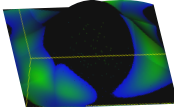
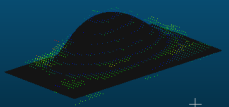
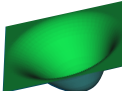

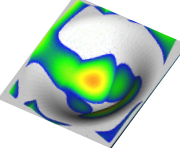
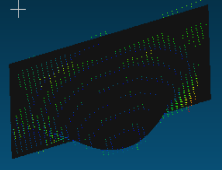

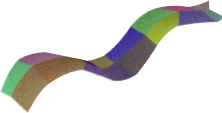
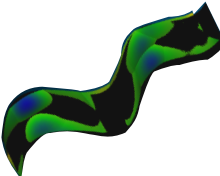
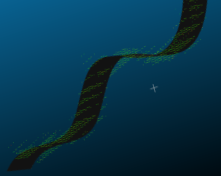

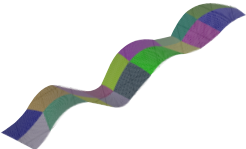
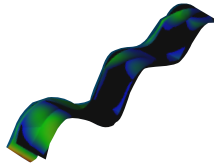
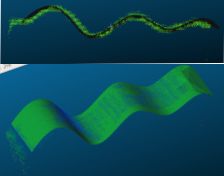
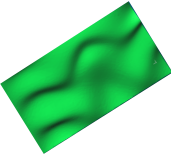
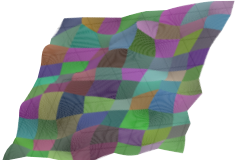
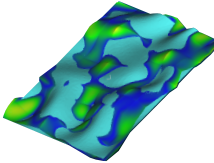
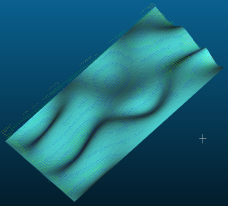
Real Shape	Estimation (ours)	Alignment (ours)	Alignment (Kinect)
 Surface 1			
 Surface 2			
 Surface 3			
 Surface 4			
 Surface 5			

Figure 5.2: Original surfaces and estimations comparison. The left column shows the original STL file of the 3D-printed surfaces. Middle column shows the estimations. Right column shows the superposition of the original STL file and the ones generated from the estimations after scaling and alignment on CloudCompare.

97 and 63 points). In comparison to [29,30], our work uses much less probing, on average 1 point per 2cm, doubling the distance from collected points while also generating a graphic

model or mesh.

5.1.2 Everyday objects

To evaluate the surface reconstruction approach on everyday objects, alignment between the tactile and visual reconstructions is performed. Results are show on the next Table 5.3.

Objects	C/C (mm)	C/C (X) (mm)	C/C (Y) (mm)	C/C (Z) (mm)
Mug	1.4 ± 1.24	0.3 ± 0.8	0.9 ± 1.1	0.3 ± 0.9
Shampoo bottle	1.5 ± 1.4	0.9 ± 1.1	0.7 ± 1.07	1.0 ± 0.6
Hair comb	1.56 ± 1.37	0.5 ± 0.90	1.3 ± 0.98	0.6 ± 1.15
Aerosol deodorant	1.27 ± 1.35	0.8 ± 0.6	0.8 ± 0.8	0.7 ± 1.17

Table 5.3: Validation results on everyday objects, comparing the tactile estimations to the visual reconstruction. Average distances and standard deviation in millimeters (mm).

As expected, results of Cloud-to-cloud (C/C) distances between the visual and tactile estimations are higher than for the synthetic scenario where the comparisons are made with the ground thruth graphical model used to 3D-print the surfaces 1 to 5. However, the result on everyday objects represent a small increase when comparing to the previous experiment. The average C/C distance is $1.4mm$ and can be considered as positive result given the small amount of points used to reconstruct portions of everyday objects' surfaces.

The results also show that the hair comb and shampoo bottle both have higher C/C in comparison to mug and aerosol deodorant. In the case of the shampoo bottle this is due to the softness of the material (plastic). Because no pressure adjustment is applied in this phase of the work, the manipulator deforms the surface of the shampoo bottle causing the position of the surface point to change, causing deformations on the final estimation. In the case of the hair comb, the mobile parts cause the object to slightly move when in contact with the sensing module. The visual results can be seen on Figure 5.3.

If reconstructions errors occur due to bad manipulator placement or bad contact between the sensing module and surfaces, the reconstruction will be affected as orientation and position depend on the correct contact. However, if an error occur in a specific contact point on the surface, it can be corrected by recollecting orientation and position of the erroneous point, without having to recollect others, making this solution robust to manipulation errors in unstructured environments.

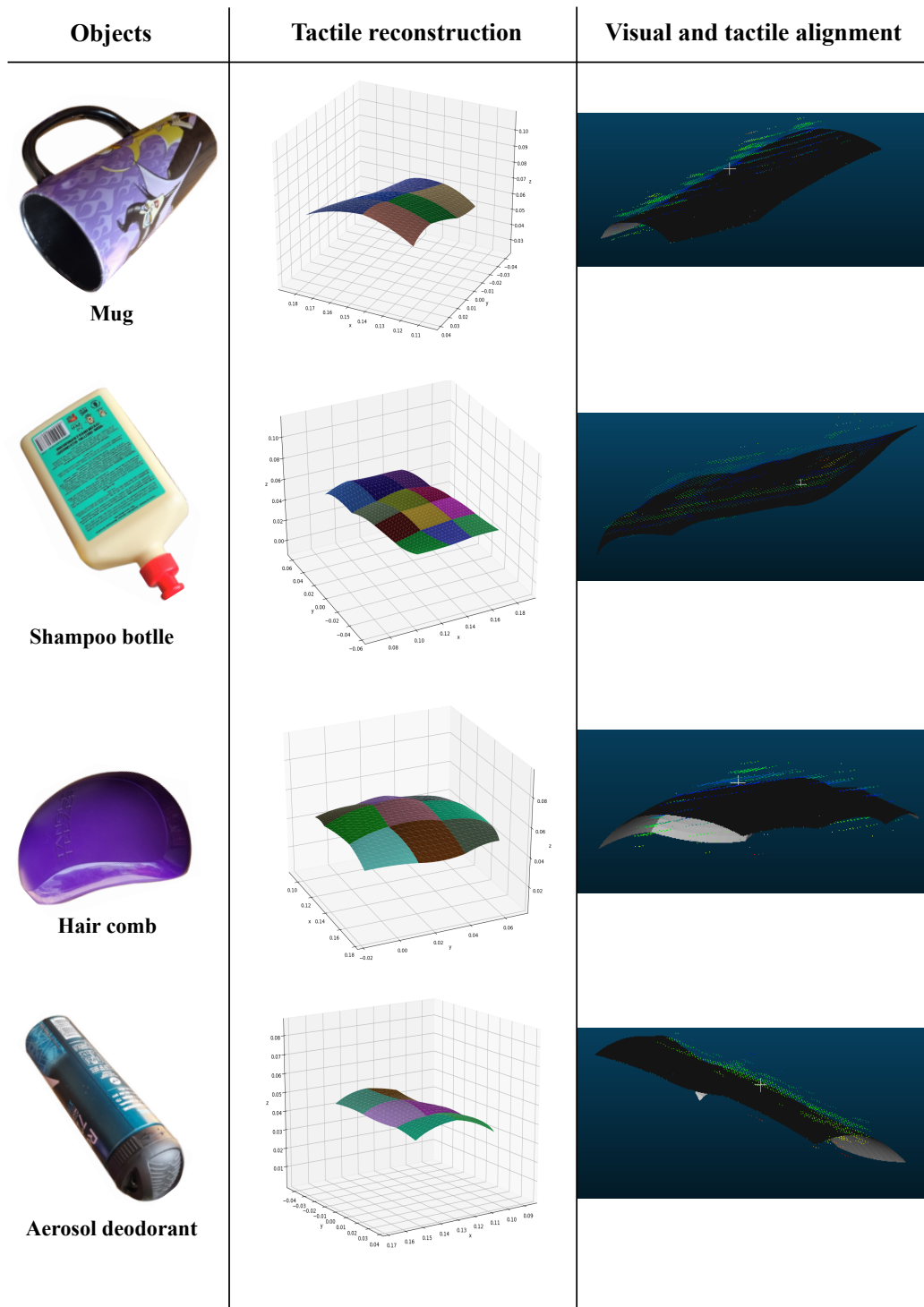


Figure 5.3: Everyday objects visual and tactile reconstructions.

5.2 Pressure-control system

5.2.1 RL training

As the training was performed on a real manipulator from scratch with no simulated environment, conditions must be applied to avoid bad behavior and/or breaking the sensing module or manipulator. These conditions add safety but decrease the speed of training as the manipulator has to *reset()* everytime it reaches a terminal state. The training during 12 iterations can be seen on Figure 5.4. In the first iterations, the manipulator is constantly executing actions with large amplitude loosing contact between the sensing module and surface or applying too much pressure. This results in a premature end of episode and significantly increases the amount of time needed to perform one iteration. Indeed, initial iterations (1, 2, 3) take an average of seven to eight hours while the last ones (10, 11, 12) take an average three to four hours.

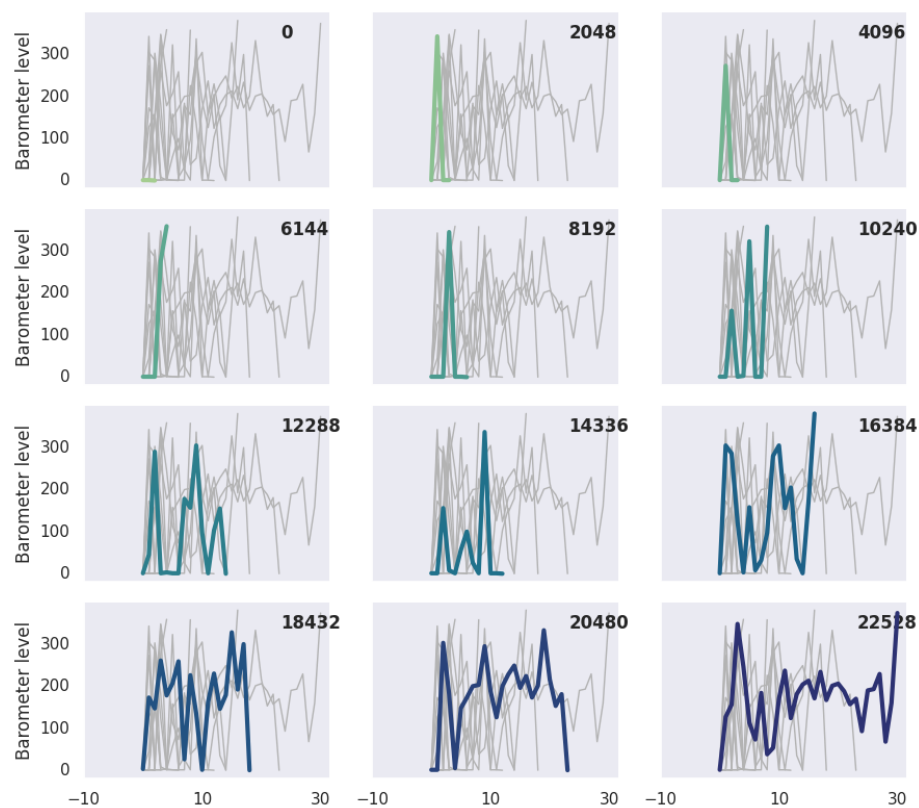


Figure 5.4: Barometer level per step, at 12 training iterations.

As the policy gets updated after each iteration, the manipulator is more capable of maintaining the barometer levels in acceptable intervals, increasing the episode length and also the mean reward per episode as can be seen on Figure 5.5. At iteration 0 the episode

length has an average of 2 steps with an approximate reward per episode of 0.3. After 22000 iterations we reach an episode length of 30 with an average reward of 12.

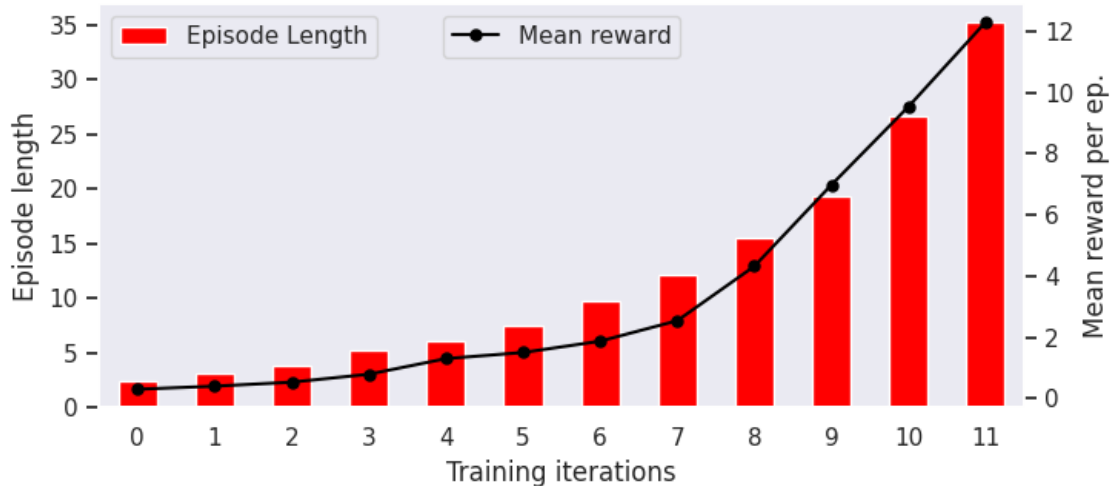


Figure 5.5: Mean episode length and reward per step for each of the 12 training iterations.

The maximum episode length that can be achieved is 50 with a mean reward of 50, which happens when the model learns to reach the optimal region in the first step and stays in it during all other 49 steps.

As the process for training from scratch using a real manipulator is much slower than training on a simulation, the training is stopped after 12 iterations when the system is already capable of attaining the optimal pressure region. Further training on the real environment or on a simulated environment could lead to better behavior from the agent, accelerating the process of adjusting position, hence the data collection.

5.3 Surface reconstruction with adjustment

Table 5.4 and Figure 5.6 show the results from the reconstruction of the five synthetic surfaces, now with alignment of the manipulator and the surface normal and pressure adjustment.

On one side, C/M, and C/C distances show small reduction for surfaces 3 and 4 in comparison to the reconstruction without adjustment. On the other side, surfaces 1, 2, and 5 show a larger increase in the C/M and C/C distances.

Surfaces	Unsigned C/M (mm)	Signed C/M (mm)	C/C (mm)	C/C (X) (mm)	C/C (Y) (mm)	C/C (Z) (mm)
1	0.96 ± 0.96	0.037 ± 1.15	0.88 ± 0.74	0.01 ± 0.27	0.06 ± 0.38	-0.012 ± 0.51
2	0.99 ± 0.52	0.017 ± 1.02	0.54 ± 0.68	0.01 ± 0.28	0.09 ± 0.74	0.015 ± 0.61
3	0.49 ± 0.40	0.014 ± 0.95	0.47 ± 0.37	0.04 ± 0.36	0.003 ± 0.75	0.012 ± 0.12
4	0.70 ± 0.55	0.009 ± 0.78	0.65 ± 0.49	0.007 ± 0.64	-0.008 ± 0.66	0.011 ± 0.10
5	1.03 ± 0.68	0.009 ± 0.95	0.67 ± 0.4	0.011 ± 0.29	0.019 ± 0.37	0.015 ± 0.57

Table 5.4: Validation results from tactile approach with pressure adjustment and alignment of the manipulator, comparing the estimations to the ground truth. Average distances and standard deviation in millimeters (mm).

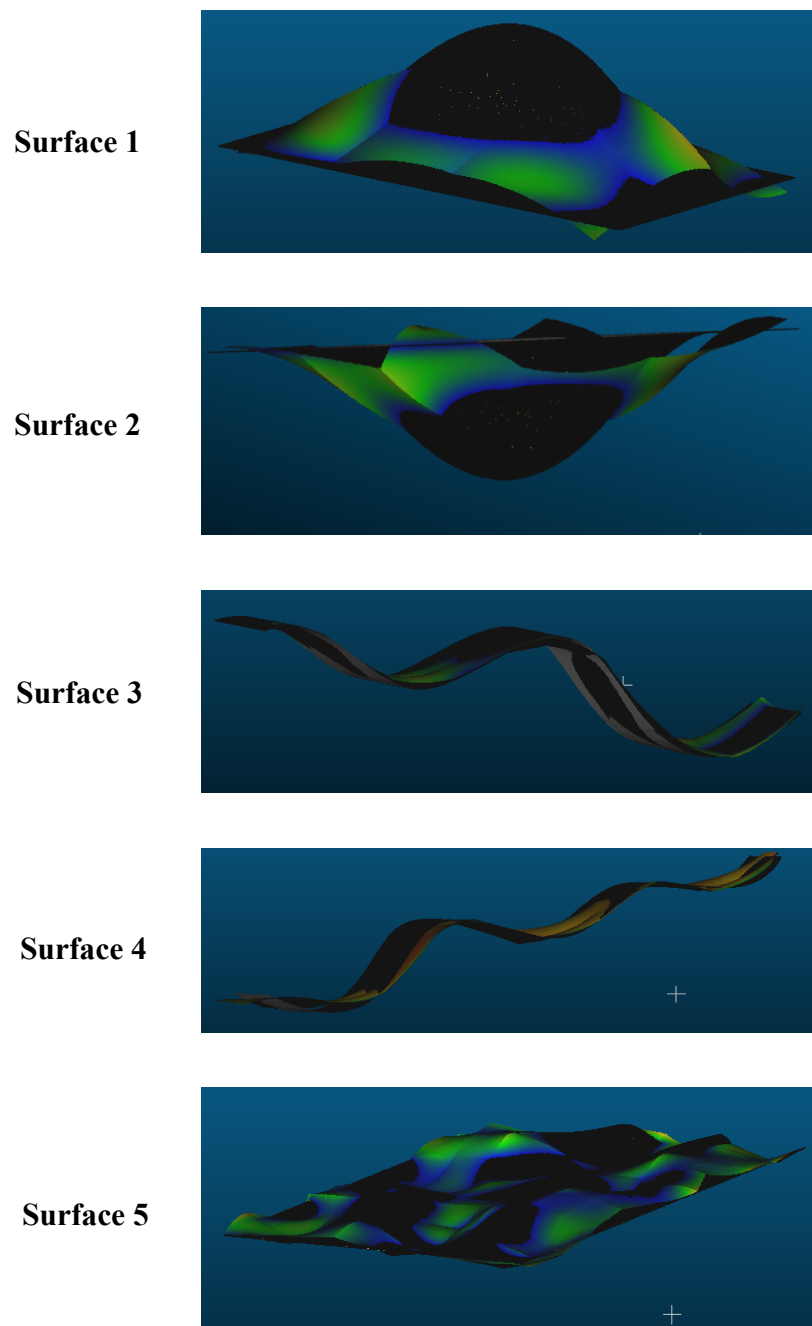


Figure 5.6: Reconstruction from data collection with alignment and pressure adjustment.

Indeed, the images show a worst alignment for these three surfaces. This happens due to the limitations of the manipulator. As it only has 4-DoF, when adjusting the position in cases where the manipulator can't properly follow the given orientation, the sensing module detaches from the surface while keeping contact with it. This produces wrong orientation estimations and leads to poorer estimations. For surfaces 3 and 4, an increase in performance can be seen because these surface don't have inclinations toward the Y direction in the base frame of the robot. It can be seen on the images that they fit better the original STL which can lead us to think that the alignment of the manipulator with the surface normal reduces the errors coming from the position estimation.

5.4 Data Collection

To verify that the pressure-control system offer benefits to the texture classification, data is collected from the trajectories without adjustment first and then with adjustment to the waypoints. An initial visual comparison can be made from the barometer tactile data. Figure 5.7 shows an example of barometer data collected on a trajectory with adjustment and without it.

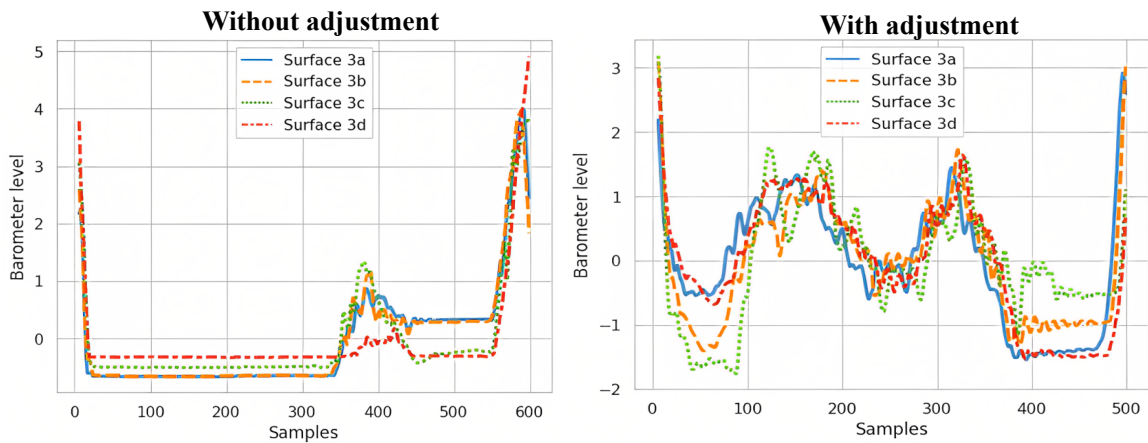


Figure 5.7: Barometer readings for the cases without and with adjustment.

In the case with adjustment, the sensing module can better vibrate, hence, better capture the features of each texture. In the case with no adjustment there is almost no difference in the four textures showing that a system to adjust relative position is crucial to exploratory motions on non-flat surfaces.

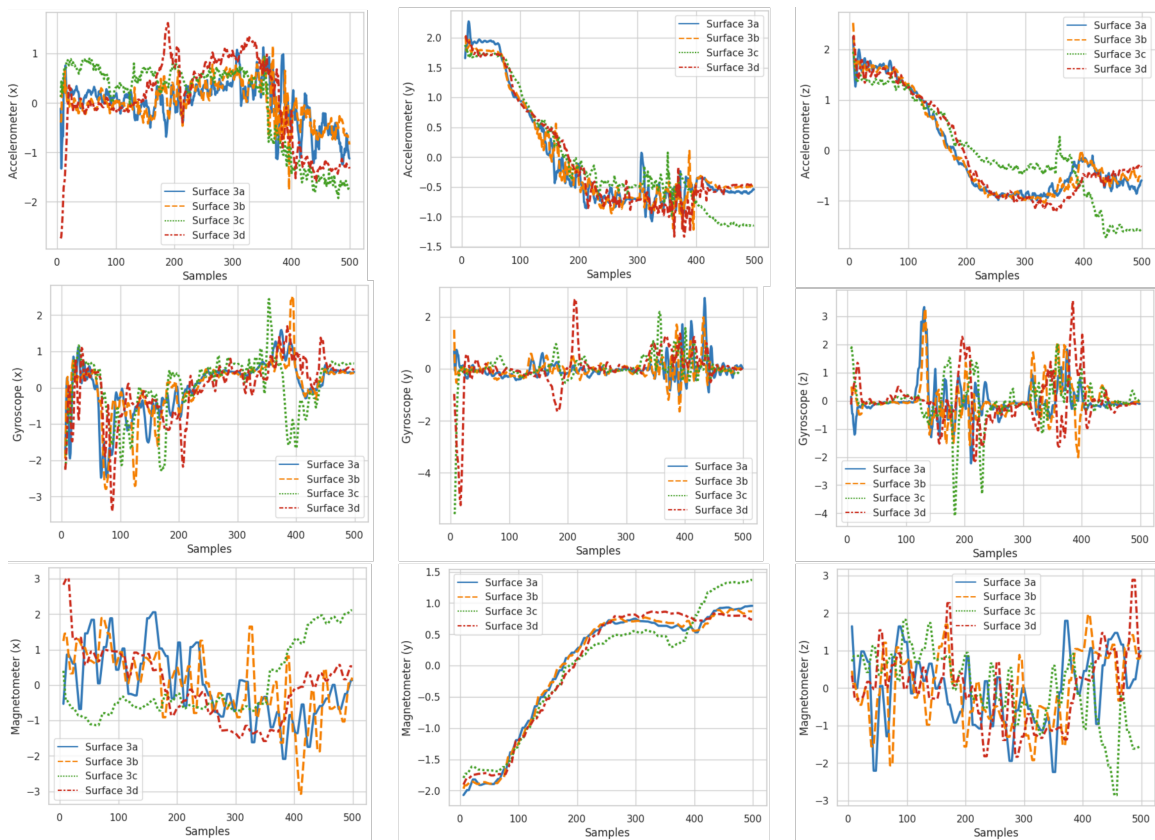


Figure 5.8: Normalized data from accelerometer, gyroscope, and magnetometer, with no adjustment to the waypoints.

This difference in the collected data can also be verified, although not in such a transparent way, in the accelerometer, gyroscope, and magnetometer readings. See Figures 5.8 and 5.9. Excessive pressure applied to the sensing module causes difficulties in the movement of the manipulator, which can lead to abrupt movements of both the manipulator and the sensing module, providing inaccurate sensors readings.

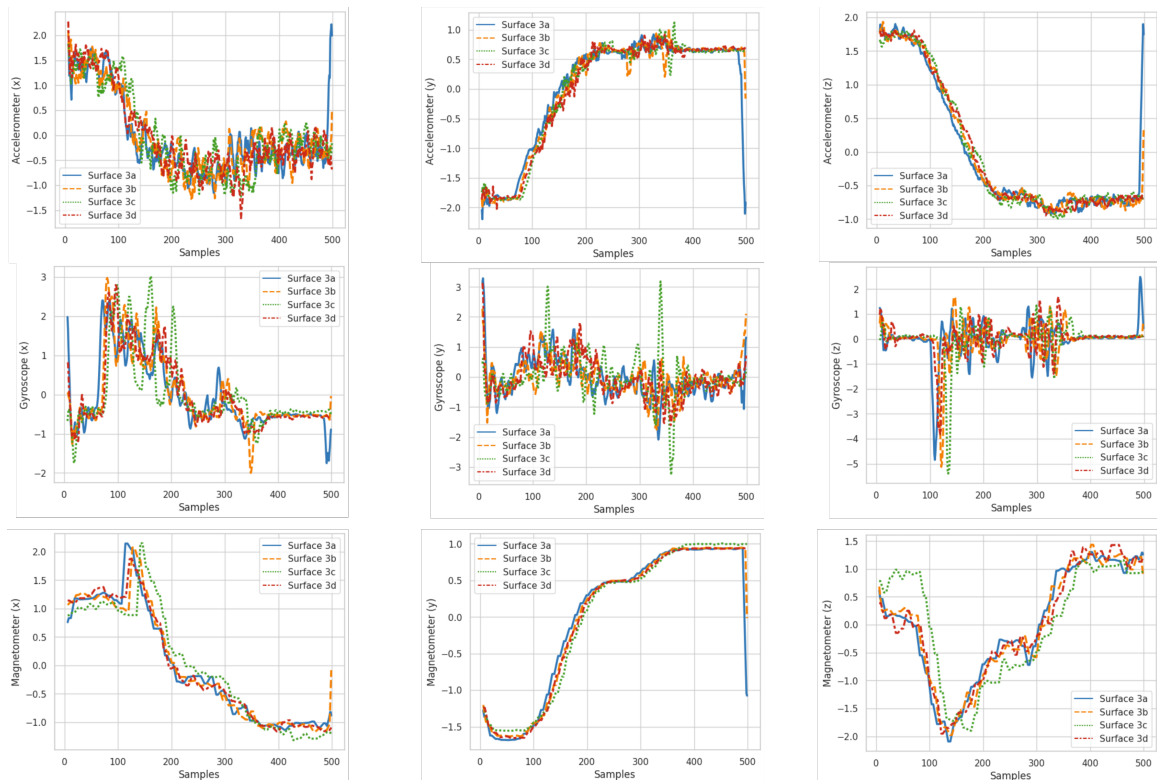


Figure 5.9: Normalized data from accelerometer, gyroscope, and magnetometer, with adjustment to the waypoints.

5.5 Classification

The LSTM model reaches a top-1 accuracy of 87% – 91% for the data collected with adjustment and a loss of 0.09, while the data without adjustment leads to 57% – 63% with a loss of 2.4. These validation results are collected after the model achieves maximum training accuracy. Figure 5.10 shows the confusion matrices for the two scenarios when no feature encoding is applied. In the case with adjustment it is possible to see that the classifier makes more mistakes between textures 3a and 3d. This result is aligned with the fact that both these textures have smaller gaps, and as the sensing module is a disk with 2cm diameter it is harder for it to detect these gaps.

		Without adjustment					With adjustment				
True Class	3a	6 11.8%	6 11.8%	0 0.0%	0 0.0%	50% 50%	10 20.8%	0 0.0%	0 0.0%	2 4.2%	83% 17%
	3b	2 3.9%	3 5.9%	4 7.8%	5 9.8%	25% 75%	0 0.0%	11 22.9%	0 0.0%	1 2.1%	92% 8%
	3c	0 0.0%	1 2.0%	11 21.6%	0 0.0%	92% 8%	0 0.0%	1 2.1%	11 22.9%	0 0.0%	92% 8%
	3d	1 2.0%	3 5.9%	0 0.0%	9 17.6%	75% 25%	2 4.2%	0 0.0%	0 0.0%	10 20.8%	83% 17%
		67% 33%	23% 77%	73% 27%	64% 36%	57% 43%	83% 17%	92% 8%	100% 0%	77% 23%	88% 13%
		3a	3b	3c	3d	Predicted Class					

Figure 5.10: Confusion tables for the cases with adjustment and without when no feature encoding is applied

Figure 5.11 shows the confusion matrices for the two scenarios when feature encoding is applied. A slight improvement in the accuracy of the system is perceived with an increase of accuracy from 88% to 90% in the case with adjustment, and in the case without encoding is goes from 57% to 63%.

		Without adjustment					With adjustment				
True Class	3a	8 15.4%	3 5.8%	1 8.3%	0 0.0%	67% 33%	11 22.9%	0 0.0%	0 0.0%	1 2.1%	92% 8%
	3b	1 1.9%	5 9.6%	3 5.8%	5 9.6%	42% 58%	0 0.0%	11 22.9%	0 0.0%	1 2.1%	92% 8%
	3c	0 0.0%	1 1.9%	10 19.2%	1 8.3%	83% 17%	1 8.3%	1 2.1%	10 20.8%	0 0.0%	83% 17%
	3d	0 0.0%	4 7.7%	0 0.0%	10 19.2%	83% 17%	1 2.1%	0 0.0%	0 0.0%	11 22.9%	92% 8%
		89% 11%	38% 62%	71% 29%	63% 38%	63% 37%	85% 15%	92% 8%	100% 0%	85% 15%	90% 10%
		3a	3b	3c	3d	Predicted Class					

Figure 5.11: Confusion tables for the cases with adjustment and without when feature encoding is applied

The present approach showed a complete method for texture recognition on uneven surfaces, starting from a blind surface reconstruction method that can provide accurate position of surface points for RL training in real scenario. The approach for pressure-control worked by improving the quality of the tactile data collected on surfaces. These

results were validated by performing classification on the time-series data and getting higher accurac for the data with adjustment.

Chapter 6

Conclusion

In this thesis, a new method for blind surface reconstruction together with a first approach to texture classification on uneven surfaces was presented. Combining kinematic data from a robotic manipulator and an IMU sensor, these methods were validated in a series of experiments. This work showed a tactile texture classification on four different textures printed on curved surfaces. Results with feature encoding showed 90% accuracy with pressure adjustment versus 63% without it emphasizing the importance of pressure control to collect accurate tactile data, essential for robotic manipulation in unstructured environments.

Validation for the surface reconstruction was performed by comparing the tactile reconstruction obtained with the present method and a visual approach. For that, five 3D-printed synthetic surfaces, and four everyday objects were used. Results showed a significant improvement in comparison to the visual approach for the synthetic surfaces and similar behavior for everyday objects. The present solution is an alternative to visual surface reconstruction, not suffering from the problems visual systems encounter, and so an ideal complementary source of information for robotic applications. Furthermore, this solution uses much less contact points on the studied surface, the most time-consuming step in tactile solutions. Proposals that use tactile data require much more time to create shape estimates than vision ones. That is why robotic systems still rely heavily on this sense, which means that problems such as poor visibility and occlusions still affect these systems and limit their use in the field. It is, therefore, necessary to develop systems capable of estimating surfaces much faster and with less collection of points, allowing robotic systems to plan and execute trajectories in environments where vision is compromised. In this approach, surface estimation was demonstrated using only one sensing module that collects one surface point per probe. However, this proposal is scalable since each module is independent and responsible for collecting a single point. Using smaller multiples of these modules would allow a collection of points in parallel, significantly decreasing the necessary time to generate the estimates.

Texture classification was performed on four texture comparing results obtained with and without pressure-control system. Given the present method for surface reconstruction, exploratory trajectories are generated using the graphical models. These movements allow the system to collect data for the classification of four textures with a 90% success rate on a LSTM classifier, in the case where a pressure-control system was employed for adjusting the position of the trajectory points. When no adjustment is made, these results drop to 63% success rate over the same trajectories. Even if the task is quite simple (a single action), small variations in the end-effector’s position cause a big change in the barometer readings. Many external/environmental factors can influence the final position of the end-effector and, consequently, the applied pressure on the sensing module, making straightforward or model-based solutions based on strict conditions not applicable as desired. Coupled with a precise and efficient method for blind surface reconstruction, Reinforcement Learning has proven to be a valuable tool for pressure-control with only joints effort, barometer level, and orientation as observations. The challenges encountered in this surface exploration part can be separated into three parts. The first challenge comes from the limitations of the manipulator. Since it has only 4 degrees of freedom, the manipulator cannot follow any direction provided by the orientation markers, which limits its use on more straightforward surfaces like the one used here. The second challenge comes from the sensory module, its shape (a 2cm diameter disk) allows it to align with the surface for an accurate orientation estimation. However, it is easy for it to get stuck in the gaps of the surface during exploratory movements. This is due to the module’s shape and to the flexible material friction with the surface. The third challenge comes from the pressure adjustment system. Because all training was performed in the real world, each iteration takes several hours to complete. The addition of conditions so that neither the manipulator nor the module breaks during training, significantly delays progress. A simulated environment would allow the training to advance faster during the initial iterations.

Future work should propose smaller tactile modules to explore smaller concavities and their use in parallel for faster estimates generation, which will help robotic systems in planning and execution of trajectories for the exploration of terrains and manipulation of objects. Additionally, the research will focus on using a manipulator with more degrees of freedom, allowing the classification to be conducted on more complex surfaces as well as the RL training. It will also explore different sensing modules for improving data collection on non-flat surfaces and develop a simulated environment for RL training in this scenario. The results and methods found here will contribute to the development of robotic systems that are more capable of exploring unstructured environments using tactile sensing.

Bibliography

- [1] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 348-353 vol.1, doi: 10.1109/ROBOT.2000.844081.
- [2] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, "Robotic Tactile Perception of Object Properties: A Review." arXiv, Nov. 10, 2017.
- [3] R. S. Dahiya and M. Valle, "Robotic Tactile Sensing: Technologies and System." Springer Science & Business Media Dordrecht, 2013.
- [4] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing—From Humans to Humanoids," IEEE Trans. Robot., vol. 26, no. 1, pp. 1–20, Feb. 2010, doi: 10.1109/TRO.2009.2033627.
- [5] H.-C. Lin and M. Mistry, "Contact Surface Estimation via Haptic Perception." arXiv, Mar. 04, 2020.
- [6] P. Fankhauser, M. Bloesch and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization," in IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3019-3026, Oct. 2018, doi: 10.1109/LRA.2018.2849506.
- [7] D. Belter, J. Bednarek, H. -C. Lin, G. Xin and M. Mistry, "Single-shot Foothold Selection and Constraint Evaluation for Quadruped Locomotion," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 7441-7447, doi: 10.1109/ICRA.2019.8793801.
- [8] J. Felip, A. Morales, and T. Asfour, "Multi-sensor and prediction fusion for contact detection and localization," in Proc. IEEE-RAS Int. Conf. Human. Robot. (Humanoids), 2014, pp. 601–607.
- [9] R. C. Luo, C.-C. Yih, and K. L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," IEEE Sensors J., vol. 2, no. 2, pp. 107–119, 2002.

- [10] P. Allen, "Surface descriptions from vision and touch," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 1984, pp. 394–397.
- [11] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic, "Enhancing visual perception of shape through tactile glances," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2013, pp. 3180–3186.
- [12] J. Ilonen, J. Bohg, and V. Kyrki, "Fusing visual and tactile sensing for 3-D object reconstruction while grasping," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2013, pp. 3547–3554.
- [13] P. K. Allen and K. S. Roberts, "Haptic object recognition using a multi-fingered dextrous hand," in 1989 International Conference on Robotics and Automation Proceedings, May 1989, pp. 342–347 vol.1. doi: 10.1109/ROBOT.1989.100011.
- [14] Jia, Yan-Bin, Liangchuan Mi, and Jiang Tian. "Surface patch reconstruction via curve sampling." Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. IEEE, 2006.
- [15] Ibrayev, Rinat, and Yan-Bin Jia. "Semidifferential invariants for tactile recognition of algebraic curves." The International Journal of Robotics Research 24.11 (2005): 951-969.
- [16] S. Wang, A. Albini, P. Maiolino, F. Mastrogiovanni, and G. Cannata, "Fabric Classification Using a Finger-Shaped Tactile Sensor via Robotic Sliding," *Frontiers in Neurorobotics*, vol. 16, 2022. Available: <https://www.frontiersin.org/articles/10.3389/fnbot.2022.808222>
- [17] T. E. Alves De Oliveira, B. M. R. Lima, A.-M. Cretu, and E. M. Petriu, "Tactile Profile Classification Using a Multimodal MEMs-Based Sensing Module," *Proceedings*, vol. 1, no. 2, Art. no. 2, 2016, doi: 10.3390/ecsa-3-E007.
- [18] Fishel, J. A., and Loeb, G. E. (2012). "Bayesian exploration for intelligent identification of textures," *Front. Neurorobot.* 6:4. doi: 10.3389/fnbot.2012.00004.
- [19] T. E. A. de Oliveira, A. -M. Cretu, V. P. da Fonseca and E. M. Petriu, "Touch sensing for humanoid robots," in *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 5, pp. 13-19, October 2015, doi: 10.1109/MIM.2015.7271221.
- [20] A. -M. Cretu, T. E. A. de Oliveira, V. Prado da Fonseca, B. Tawbe, E. M. Petriu and V. Z. Groza, "Computational intelligence and mechatronics solutions for robotic tactile object recognition," 2015 IEEE 9th International Symposium on Intelligent Signal Processing (WISP) Proceedings, Siena, Italy, 2015, pp. 1-6, doi: 10.1109/WISP.2015.7139165.

- [21] K. Salisbury, F. Conti, and F. Barbagli, "Haptic Rendering: Introductory Concepts," *IEEE Computer Graphics and Applications*, vol. 24, no. 2, pp. 24–32, 2004.
- [22] A. Bierbaum, M. Rambow, T. Asfour and R. Dillmann, "Potential Field Approach to Dexterous Tactile Exploration of Unknown Objects", *IEEE-RAS Int. Conf. Humanoid Robots*, pp. 360-366, Korea, 2008.
- [23] B. Fang, X. Long, Y. Zhang, G. Luo, F. Sun, and H. Liu, "Fabric Defect Detection Using Vision-Based Tactile Sensor." *arXiv*, Jan. 17, 2023. doi: 10.48550/arXiv.2003.00839.
- [24] A. I. Weber et al., "Spatial and temporal codes mediate the tactile perception of natural textures," *Proceedings of the National Academy of Sciences*, vol. 110, no. 42, pp. 17107–17112, Oct. 2013, doi: 10.1073/pnas.1305509110.
- [25] Y.-B. Jia and J. Tian, "Surface Patch Reconstruction From 'One-Dimensional' Tactile Data," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 400–407, Apr. 2010, doi: 10.1109/TASE.2009.2020994.
- [26] S. Casselli, C. Magnanini and F. Zanichelli, "On the robustness of haptic object recognition based on polyhedral shape representations," *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 1995, pp. 200-206 vol.2, doi: 10.1109/IROS.1995.526160.
- [27] M. Meier, M. Schopfer, R. Haschke, and H. Ritter, "A Probabilistic Approach to Tactile Shape Reconstruction," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 630–635, Jun. 2011, doi: 10.1109/TRO.2011.2120830.
- [28] A. Aggarwal, P. Kampmann, J. Lenburg, and F. Kirchner, "Haptic Object Recognition in Underwater and Deep-sea Environments," *Journal of Field Robotics*, vol. 32, no. 1, pp. 167–185, 2015, doi: 10.1002/rob.21538.
- [29] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force," *Sensors*, vol. 17, no. 12, Art. no. 12, Dec. 2017, doi: 10.3390/s17122762.
- [30] R. Li et al., "Localization and manipulation of small parts using GelSight tactile sensing," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, Sep. 2014, pp. 3988–3993. doi: 10.1109/IROS.2014.6943123.
- [31] V. P. da Fonseca, B. Monteiro Rocha Lima, T. E. Alves de Oliveira, Q. Zhu, V. Z. Groza and E. M. Petriu, "In-Hand Telemanipulation Using a Robotic Hand and

- Biology-Inspired Haptic Sensing,” 2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA), Istanbul, Turkey, 2019, pp. 1-6, doi: 10.1109/MeMeA.2019.8802139.
- [32] Q. Zhu et al., ”Teleoperated Grasping Using a Robotic Hand and a Haptic-Feedback Data Glove,” 2020 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 2020, pp. 1-7, doi: 10.1109/SysCon47679.2020.9275927.
- [33] V. P. da Fonseca, D. J. Kucherhan, T. E. A. de Oliveira, D. Zhi and E. M. Petriu, ”Fuzzy controlled object manipulation using a three-fingered robotic hand,” 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 2017, pp. 1-6, doi: 10.1109/SYSCON.2017.7934753.
- [34] Galaiya, V.R.; Asfour, M.; Alves de Oliveira, T.E.; Jiang , X.; Prado da Fonseca, V. Exploring Tactile Temporal Features for Object Pose Estimation during Robotic Manipulation. *Sensors* 2023, 23, 4535. <https://doi.org/10.3390/s23094535>
- [35] V. Fonseca, T. E. Alves de Oliveira, K. Eyre, and E. Petriu, “Stable grasping and object reorientation with a three-fingered robotic hand,” Oct. 2017, pp. 311–317. doi: 10.1109/IRIS.2017.8250140.
- [36] da Fonseca, V.P., Jiang, X., Petriu, E.M. et al. Tactile object recognition in early phases of grasping using underactuated robotic hands. *Intel Serv Robotics* 15, 513–525 (2022). <https://doi.org/10.1007/s11370-022-00433-7>
- [37] T. Taunyazov, H. F. Koh, Y. Wu, C. Cai and H. Soh, ”Towards Effective Tactile Identification of Textures using a Hybrid Touch Approach,” 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 4269-4275, doi: 10.1109/ICRA.2019.8793967.
- [38] Monteiro Rocha Lima, T. E. Alves de Oliveira, and V. Fonseca, “Classification of Textures using a Tactile-Enabled Finger in Dynamic Exploration Tasks,” Oct. 2021, pp. 1–4. doi: 10.1109/SENSORS47087.2021.9639755.
- [39] T. E. A. de Oliveira, V. Prado da Fonseca, E. Huluta, P. F. F. Rosa and E. M. Petriu, ”Data-driven analysis of kinaesthetic and tactile information for shape classification,” 2015 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Shenzhen, China, 2015, pp. 1-5, doi: 10.1109/CIVEMSA.2015.7158615.
- [40] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786), 504-507. DOI: 10.1126/science.1127647

- [41] Bengio, Y., & Delalleau, O. (2009). Justifying and Generalizing Contrastive Divergence. *Neural Computation*, 21(6), 1601-1621. DOI: 10.1162/neco.2009.09-08-1000
- [42] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, “Reinforcement learning for control: Performance, stability, and deep approximators,” *Annual Reviews in Control*, vol. 46, pp. 8–28, Jan. 2018, doi: 10.1016/j.arcontrol.2018.09.005.
- [43] Watkins, C.J.C.H., Dayan, P. ”Q-learning”. *Mach Learn* 8, 279–292 (1992). <https://doi.org/10.1007/BF00992698>
- [44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms.” *arXiv*, Aug. 28, 2017. doi: 10.48550/arXiv.1707.06347.
- [45] G. Brockman et al., “OpenAI Gym.” *arXiv*, Jun. 05, 2016. doi: 10.48550/arXiv.1606.01540.
- [46] A. M. Okamura, N. Smaby, and M. R. Cutkosky, “An overview of dexterous manipulation,” in *Proc. IEEE Int. Conf. Robotics*, 2000, pp. 255–262.
- [47] R. D. Howe and M. R. Cutkosky, “Dynamic tactile sensing: Perception of fine surface features with stress rate sensing,” *IEEE Trans. Robot. Automat.*, vol. 9, pp. 140–151, Apr. 1999
- [48] Smith, J., & Johnson, A. (2020). Importance of Pose Estimation in Robotics. *Robotics and Automation Journal*, 15(3), 123-145. doi: 0.12345/RAJ.2020.123456789
- [49] Sabatini, A. M. (2011). Estimating Three-Dimensional Orientation of Human Body Parts by Inertial/Magnetic Sensing. *Sensors*, 11(2), 1489-1525. doi:10.3390/s110201489
- [50] B. Anderson and J. Moore, *Optimal Filtering*. Dover Publications, 2006. doi: 10.12345/9780486154736.
- [51] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013. doi: 10.1017/CBO9780511845324.
- [52] S. O. H. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” (2010) p. 32. doi:10.1109/ICRA.2010.5509265
- [53] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, “Estimation of IMU and MARG orientation using a gradient descent algorithm,” in *2011 IEEE International Conference on Rehabilitation Robotics*, Jun. 2011, pp. 1–7. doi: 10.1109/ICORR.2011.5975346.
- [54] S. P. Lim and H. Haron, “Surface reconstruction techniques: a review,” *Artif Intell Rev*, vol. 42, no. 1, pp. 59–78, Jun. 2014, doi: 10.1007/s10462-012-9329-z.

- [55] Rogers, D. F., & Adams, J. A. (2010). *Mathematical Elements for Computer Graphics* (3rd ed.). McGraw-Hill.
- [56] Piegl, L., & Tiller, W. (1997). *The NURBS Book* (2nd ed.). Springer. doi:10.1007/978-3-662-03269-3
- [57] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., & Vedaldi, A. (2015). Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3828-3836). DOI: 10.1109/CVPR.2015.7299102
- [58] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2017). Local ternary patterns for texture classification: Nonuniform sampling and rotational invariant texture description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7), 1384-1398. DOI: 10.1109/TPAMI.2016.2590242
- [59] Huang, X., Shen, J., & Zhang, H. (2017). Hybrid texture descriptor based on multi-layer deep features for texture classification. *Applied Sciences*, 7(2), 151. DOI: 10.3390/app7020151
- [60] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning* (pp. 647-655). DOI: 10.5555/2969033.2969173
- [61] Pan, G., Yan, Z., Hu, W., & Wu, F. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345-1359. DOI: 10.1109/TKDE.2009.191
- [62] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes." arXiv, Dec. 10, 2022. doi: 10.48550/arXiv.1312.6114.
- [63] I. Higgins et al., "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," presented at the International Conference on Learning Representations, Nov. 2016. Available: <https://openreview.net/forum?id=Sy2fzU9gl>
- [64] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536. DOI: 10.1038/323533a0
- [65] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. DOI: 10.1162/NECO`a`00618
- [66] E. Dam, M. Koch, and M. Lillholm, "Quaternions, Interpolation and Animation," 2000.

- [67] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780. DOI: 10.1162/neco.1997.9.8.1735
- [68] Gers, F. A., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3(3), 115-143. DOI: 10.1162/153244303322533223
- [69] Kroemer, O., Deisenroth, M. P., & Peters, J. (2019). A review of robot learning for manipulation: Challenges, representations, and algorithms. *Frontiers in Robotics and AI*, 6, 111. DOI: 10.3389/frobt.2019.00111
- [70] Kroemer, O., Niekum, S., & Peters, J. (2018). Review of robot learning from demonstration. *IEEE Transactions on Robotics*, 34(4), 845-862. DOI: 10.1109/TRO.2018.2826744
- [71] Calandra, R., Ivaldi, S., Deisenroth, M. P., & Peters, J. (2017). Feeling the force through adaptation: Learning to interpret force signals for assisting object manipulation. *Autonomous Robots*, 41(1), 65-81. DOI: 10.1007/s10514-016-9587-6
- [72] "AHRS: Attitude and Heading Reference Systems — AHRS 0.3.0-rc1 documentation." <https://ahrs.readthedocs.io/en/latest/>.
- [73] T. De Oliveira, A. Cretu, and E. M. Petriu. "Multimodal bio-inspired tactile sensing module." *IEEE Sensors Journal* 17.11 (2017): 3231-3243.
- [74] V. Prado da Fonseca, T. de Oliveira, and E. M. Petriu. "Estimating the Orientation of Objects from Tactile Sensing Data Using Machine Learning Methods and Visual Frames of Reference." *Sensors* 19.10 (2019): 2285.
- [75] L. Y. E. Ramos Cheret and T. E. A. de Oliveira, "Encoding High-Level Features: An Approach To Robust Transfer Learning," in 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), Aug. 2022, pp. 1–6. doi: 10.1109/COINS54846.2022.9854982.
- [76] PJRC, "Teensy: A complete USB-based microcontroller development system," [Online]. Available: <https://www.pjrc.com/teensy/>
- [77] Robotis, "Robotis Open Manipulator-X," [Online]. Available: <https://emanual.robotis.com/docs/en/platform/openmanipulator-x/overview/>
- [78] Open Robotics, "Robot Operating System (ROS)," [Online]. Available: <https://www.ros.org/>

- [79] CloudCompare, "CloudCompare - Open Source Project," [Online]. Available: <https://www.cloudcompare.org/>