

This electronic thesis or dissertation has been downloaded from the King's Research Portal at <https://kclpure.kcl.ac.uk/portal/>



Touch Based Object Pose Estimation For Robotic Grasping

Bimbo, Joao Maria

Awarding institution:
King's College London

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without proper acknowledgement.

END USER LICENCE AGREEMENT



Unless another licence is stated on the immediately following page this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International licence. <https://creativecommons.org/licenses/by-nc-nd/4.0/>

You are free to copy, distribute and transmit the work

Under the following conditions:

- Attribution: You must attribute the work in the manner specified by the author (but not in any way that suggests that they endorse you or your use of the work).
- Non Commercial: You may not use this work for commercial purposes.
- No Derivative Works - You may not alter, transform, or build upon this work.

Any of these conditions can be waived if you receive permission from the author. Your fair dealings and other rights are in no way affected by the above.

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

KING'S COLLEGE LONDON

DOCTORAL THESIS

Touch Based Object Pose
Estimation For Robotic Grasping

Author:

João BIMBO

Primary Supervisor:

Dr. Hongbin LIU

Secondary Supervisor:

Prof. Kaspar ALTHOEFER

*A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy*

in the

Centre for Robotics Research
Department of Informatics

May 2016

Declaration of Authorship

I, João BIMBO, declare that this thesis titled, 'Touch Based Object Pose Estimation For Robotic Grasping' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

KING'S COLLEGE LONDON

Abstract

Centre for Robotics Research

Department of Informatics

Doctor of Philosophy

Touch Based Object Pose Estimation For Robotic Grasping

by João BIMBO

Robot grasping and manipulation require very accurate and timely knowledge of the manipulated object's shape and pose to successfully perform a desired task. One of the main reasons current systems fail to carry out complex tasks in a real, unstructured environment is their inability to accurately determine where in the object the fingers are touching. Most systems use vision to detect the pose of an object, but the performance of this sensing modality deteriorates as soon as the robot grasps the object. When the robot hand contacts an object, it partially occludes it, which makes it difficult for vision systems to track the object's location. This thesis presents algorithms to use the robot's available tactile sensing to correct the visually determined pose of a grasped object. This method is extended to globally estimate the pose of the object even when no initial estimate is given.

Two different tactile sensing strategies have been employed: single-point and distributed, and measurement models for these two strategies are presented. Different optimisation algorithms are developed and tested to minimise the output of these measurement models and find one or more poses that satisfy current tactile measurements. Results show that the method is able to successfully estimate the pose of a grasped object with high accuracy, even for objects with a high degree of geometric complexity. Other applications of the method are proposed, such as determining grasp stability or identifying the grasped object, as well as future research directions.

Acknowledgements

I would like to thank all the friends I have made during the course of these years at King's College London. I will always remember fondly the lunch breaks with Andreas, Ankur, Angela, Dimitris, Greg, Hugo, Neil and Vahid and our endless discussions on the balcony of Somerset House. You have made my days at King's very pleasant.

A big thanks to my colleagues and staff in the Centre for Robotics Yohan, Agostino, Shan, Helge, Xiaojing, Junghwan, Thomas, Lukas, Ali, Sina, Jim Trotter, Prof. Lakmal Seneviratne, and Dr. Thrishantha Nanayakkara.

I would also like to thank my co-authors and collaborators Dr. Petar Kormushev, Dr. Véronique Perdereau, Dr. Guillaume Walck, Dr. Mohamed Abderrahim, Silvia Rodríguez Jiménez and Dr. Nicolas Burrus. My gratitude goes also to the people at Shadow Robot Company: Mark Addison, Toni Oliver and Rich Walker, for giving me the opportunity to work with their robot hand and for the experience in the “real world”.

My biggest gratitude goes also to my supervisors Prof. Kaspar Althoefer for his trust and support and to Dr. Hongbin Liu for his mentoring, his patience and most of all, his friendship.

Finally, I would like to thank my family for their love: Pai, Mãe, Avó, Nuno, Luis, Vânia and Ana Maria.

Contents

Declaration of Authorship	1
Abstract	2
Acknowledgements	3
Contents	4
List of Figures	7
List of Tables	11
Symbols and Definitions	12
1 Introduction	15
1.1 Scope and Motivation	15
1.2 Problem Presentation	17
1.3 Research Contributions	21
1.4 List of Publications	22
1.5 Thesis Structure	24
2 Background	25
2.1 Robot Grasping and Manipulation	26
2.2 Tactile Sensing	29
2.2.1 Intrinsic Tactile Sensing	31
2.2.2 Tactile sensing arrays	35
2.3 Sensor Fusion	38
2.4 Object Pose Estimation	40
2.4.1 Vision-based Object Pose Estimation	40
2.4.2 Tactile-based Pose Estimation	42
2.5 Conclusions	48
2.6 Mathematical Background	51

2.6.1	Rigid Body Motions	51
2.6.1.1	Coordinate Frames and Matrix Representation	51
2.6.1.2	Euler Angles	53
2.6.1.3	Quaternions	55
2.6.2	Optimisation	58
2.6.2.1	Introduction	58
2.6.2.2	Gradient-Based	59
	Gradient Descent	61
	Levenberg-Marquardt	61
2.6.2.3	Stochastic	62
2.6.2.4	Other Methods	65
2.6.3	k -d Trees	65
2.6.3.1	Definition	65
2.6.3.2	Construction	66
2.6.3.3	Searches	66
2.6.3.4	Computational Remarks	68
2.6.4	Principal Component Analysis	69
2.6.4.1	Definition	69
2.6.4.2	Computing the Principal Components	70
3	Pose Correction using Local Optimisation	72
3.1	Introduction	73
3.2	Methods	74
3.2.1	Algorithm	74
3.2.2	Distance-based optimisation	76
3.2.2.1	Objective Function	76
3.2.2.2	Simulation Results	77
3.2.3	Addition of Normal Force Information	80
3.2.3.1	Contact Normal	80
3.2.3.2	Objective function	82
3.2.3.3	Simulation Results	83
3.3	Results	84
3.3.1	Analysis of Simulation Results	84
3.3.2	Results on a Real System	89
3.3.2.1	System Overview	89
3.3.2.2	Using Distance Information	91
3.3.2.3	Using Distance and Normal Information	92
3.4	Discussion	96
4	Pose Estimation using Global Optimisation	98
4.1	Introduction	99
4.2	Methods	100
4.2.1	Algorithm Setup and Cost Function	100
4.2.2	Search Algorithm	101

4.2.3	Generation of the Initial Population	102
4.2.4	Re-sampling scheme	105
4.2.5	Noise addition	108
4.2.6	Minimisation of the objective function	110
4.2.7	Post processing of results	111
4.3	Results	112
4.3.1	Simulation Results	112
4.3.1.1	Pose correction	113
4.3.1.2	Global pose estimation	116
4.3.2	Results Using a Real System	118
4.3.2.1	Experimental Setup	118
4.3.2.2	Pose correction from vision	119
4.3.2.3	Global Pose Estimation – Hand Over and Place	120
4.4	Discussion	122
5	Pose Estimation from tactile arrays	124
5.1	Introduction	125
5.2	Methods	126
5.2.1	PCA on Tactile Data	126
5.2.2	Selection of Scaling Parameter	128
5.2.3	Computing the Eigenbasis	131
5.2.4	Matching tactile to 3D point cloud covariance	132
5.2.5	Object Pose Estimation From Descriptor	136
5.3	Results	138
5.3.1	System overview	138
5.3.2	Pose Estimation Results	139
5.4	Conclusions	147
6	Conclusions	149
6.1	Main Contributions	150
6.2	Applications	152
6.2.1	Object Identification	152
6.2.2	Grasp Stability	153
6.3	Discussion and Critique	154
6.4	Future Work	157
	Bibliography	159

List of Figures

1.1	Camera’s point of view, visualisation of the robot’s posture and vision-acquired object pose overlaid in yellow.	17
1.2	Thesis objective scheme – $f_{1\dots3}$ shows the robot finger positions, the blue object is the initial estimate and the red object is the resulting pose, matching the contact information	18
1.3	Thesis objective scheme – $f_{1\dots3}$ shows the robot finger positions, the blue object is the initial estimate and the red object is the resulting pose, matching the contact information	20
2.1	Grasp stability diagrams.	27
2.2	Intrinsic Tactile Sensing scheme – acting force in green, measured quantities in red, computed parameters in blue	32
2.3	Intrinsic Tactile Sensing diagram – a force is exerted in the xoy plane	32
2.4	Overview of the Intrinsic Tactile sensor	34
2.5	Intrinsic force-torque sensor with soft rubber layer	36
2.6	Rigid Transformation	52
2.7	Ambiguities in Euler Angle representation	55
2.8	Interpretation of quaternions as axis-angle	56
2.9	Function $f(x, y)$ and minimum point	58
2.10	Gradient Descent on a scalar function	60
2.11	Estimating the value of π through Monte Carlo simulation	64
2.12	k -d tree data structure principle	67
2.13	Principal Components of a Distribution. 1 st , 2 nd and 3 rd components in red, green and blue respectively	70
3.1	Problem overview – Finding a transformation \mathbf{x} that displaces the object from an initial estimate	73
3.2	Regions created in the object point cloud to minimise the computational effort	75
3.3	Results of Gradient Descent using simulated data – Object point cloud in black. Original (\bullet), displaced (\times) and corrected (\circ) finger tip locations.	78
3.4	Progress of the algorithm in a cubic object using Gradient Descent. Each color represents the distance between object at the estimated pose for a finger contact.	78

3.5	Results of Levenberg-Marquardt using simulated data – Object point cloud in black. Original (●) , displaced (×) and corrected (○) finger tip locations.	79
3.6	Progress of the algorithm for a cylindrical shaped object using Levenberg-Marquardt. Each color represents the distance between object at the estimated pose for a finger contact.	79
3.7	Rigid contact	81
3.8	Mesh triangle normal	81
3.9	Results of Levenberg-Marquardt with normal force information – Object point cloud in black. Original (●) , displaced (×) and corrected (○) finger tip locations.	83
3.10	Progress of the algorithm for a cylindrical shaped object using Levenberg-Marquardt. Each color represents the distance between object at the estimated pose for a finger contact	84
3.11	Simulation results – green represents the ground truth, gray the initial misplaced pose and yellow the resulting object pose.	85
3.12	Initial vs Final MDTS using Gradient Descent	88
3.13	Initial vs Final MATN using Gradient Descent	88
3.14	Initial vs Final RME using Gradient Descent	88
3.15	Initial vs Final MAE using Gradient Descent	88
3.16	Initial vs Final MDTS using Levenberg-Marquardt	88
3.17	Initial vs Final MATN using Levenberg-Marquardt	88
3.18	Initial vs Final RME using Levenberg-Marquardt	88
3.19	Initial vs Final MAE using Levenberg-Marquardt	88
3.20	Overview of the experimental setup of the multi-modal sensing system	89
3.21	Results using real data. Initial estimate in grey, solution in pink	91
3.22	Visualisation of a grasped object scene. The green point cloud represents the object in the pose detected by the vision system and the pink point cloud represents the object after its pose has been corrected using our approach	93
3.23	Ground truth measurement method	93
3.24	Experimental results: blue and green represent the components x and y . Rings plot the pose obtained by vision and lines the pose estimated by the proposed method. Red and cyan dots are recorded ground truth	95
3.25	Initial vs. Final error with large initial error when using Gradient Descent	97
3.26	Initial vs. Final error with large initial error when using Levenberg-Marquardt	97
4.1	Cost to Weight Function	106
4.2	Re-sampling scheme	107
4.3	Computation time to generate each thousand particles	108
4.4	Noise added to particles over algorithm iterations	109
4.5	Progress of the global optimisation algorithm – cost over iterations	110

4.6	Collision checker for valid poses. Blue/green: object point cloud in valid pose, red: invalid pose	111
4.7	Pose correction. Initial estimate in red, ground truth in orange and resulting estimated pose in blue	114
4.8	Histograms for initial and final errors on rotation and translation for pose correction	114
4.9	Mean errors after pose correction for different number of contacts and noise levels	115
4.10	Rate of success for pose correction for different number of contacts and noise level. A trial is considered successful if the error is under 1 cm and 15°.	115
4.11	Global pose estimation. Initial estimate in red, ground truth in green and result pose in orange, force normals are displayed as red arrows	117
4.12	Mean error in global pose estimation for different number of contacts and noise levels	118
4.13	Rate of success for global pose estimation different number of contacts and noise level. A trial is considered successful if the error is under 1 cm and 15°.	118
4.14	Pose correction result – Vision based tracking results in yellow before and after occlusions are created by the grasp. The pose corrected using the proposed method is displayed in purple	119
4.15	Pose correction results with different objects	120
4.16	Robot grasping a pencil, object model overlaid with point cloud	120
4.17	Result of estimation of a pencil’s pose	121
4.18	Hand over and place experiment	121
4.19	Particle Landscape	122
4.20	Initial vs. Final Error for global search	123
5.1	Principal Components of a tactile sensor frame	127
5.2	Finite Element Analysis	128
5.3	Effect of pressure scaling value α in the tactile profile	129
5.4	Cross-section of the pressure profile and largest principal component for different values of α	130
5.5	Pressure profile and main principal component for different indentation depths	131
5.6	Finger model, interpolated tactile data, principal components axes and object pointcloud patch	133
5.7	Evaluation of the matching. Active tactile elements in red, principal components in blue, red and yellow, and local object geometry in green	135
5.8	Comparison of tactile data with and without silicon layer	139
5.9	Pose estimation overlaid on a picture of the grasp. Blue pointcloud shows the result when minimising distance from the surface to the contacts. Red pointcloud shows the result when using the proposed method	141

5.10	Result with horizontal thermal bottle. Blue pointcloud shows the result when minimising only the distance from contacts to surface. Red pointcloud shows the result when using the proposed method .	142
5.11	Evaluation of proposed the cost function <i>versus</i> a benchmark based on distance alone. Clockwise from the top: grasping scenario, different poses evaluated, proposed cost function, distance based cost function. Note: the colors in the lower plots match the pose hypotheses on the upper right	143
5.12	Result of cost function <i>vs.</i> angle error	144
5.13	Result of a global search using the proposed descriptor (grasp (f) in Table 5.3)– red pointcloud: initial pose; green pointcloud: resultant pose.	146
5.14	Results of applying ICP to fit a line to a plane and two planes.	148
6.1	Object identification using the proposed method	153
6.2	Grasp quality according to the Grasp Wrench Space metric.	154
6.3	Grasp quality improvement. Blue – Improve quality Red – Reduce quality	154
6.4	Example of a situation that might cause the algorithm to fail. Parts of the object surface coincide almost perfectly in two different poses. Ground truth is shown in pink.	155
6.5	Proposed system for pose estimation.	156

List of Tables

2.1	Comparison of existing tactile pose estimation methods in the literature	48
3.1	Comparison of optimisation methods	87
5.1	PCA angle error for $\alpha = 9.66 \cdot 10^{-4}$	130
5.2	Comparison of optimisation techniques	138
5.3	Comparison of cost functions – mean error, error at minimum cost and cost <i>vs.</i> error correlation coefficient	145
6.1	Comparison of existing tactile pose estimation methods	151

Symbols and Definitions

\mathbf{x} : rigid transformation parameters

m : number of fingers touching the object

\vec{t} : 3-dimensional translation vector

\mathbf{q} : rotation quaternion

a.u. : arbitrary units

\vec{F} : contact force

$$\vec{F} = [f_x, f_y, f_z]^T$$

p_c : contact location

∇S : surface gradient

\hat{n} : normal unit vector

$p(a|b)$: conditional probability

\mathbf{R} : rotation matrix

\mathbf{T} : homogenous transformation matrix

$\mathbf{A}_B^{\mathbf{T}}$: homogenous transformation matrix from frame A to frame B

$\mathbf{J}_{\mathbf{G}}(\mathbf{x})$: Jacobian matrix of function $\mathbf{G}(\mathbf{x})$

$\tilde{\mathbf{J}}_{\mathbf{G}}(\mathbf{x})$: approximate Jacobian matrix of function $\mathbf{G}(\mathbf{x})$

\mathbf{A}^+ : pseudo-inverse of matrix \mathbf{A}

$\mathbf{C}_{\mathbf{M}}$: covariance matrix of \mathbf{M}

*Dedicado a
Manuel Bimbo*

Chapter 1

Introduction

1.1 Scope and Motivation

Robot grasping, and particularly the fine manipulation of an object by a robot, require very accurate sensing of the object's pose and acting forces. In fact, even for a human, tactile sensing is fundamental for performing tasks that require a great deal of accuracy. This was shown by Rothwell [1], who studied a man with neurological damage who, despite not having any motor problem and able to perform most tasks using only vision to control his movements, failed when asked to perform fine manipulation tasks such as fastening a button or using a pen to write. Similarly, a robot equipped with a camera can perform simple grasps but it must possess a very accurate sense of touch, that not only senses the contact location but also the direction of the force, in order to carry out precise manipulation tasks.

Among the essential information needed by a robot to be able to manipulate an object is knowledge of the object's pose (position and orientation), with respect to the robot hand or gripper. Typically, stereo cameras and RGB-D sensors are employed to provide this information. However, the pose estimate provided by vision might be inaccurate due to hardware limitations or bad calibration. Moreover, in a robot grasping and manipulation setting, as the robot reaches and grasps the

object, its body – arm, hand and fingers – will cover the object. In this situation, the target object is said to be occluded – it cannot be entirely seen by the vision system – which introduces added difficulties for the tracking of its pose.

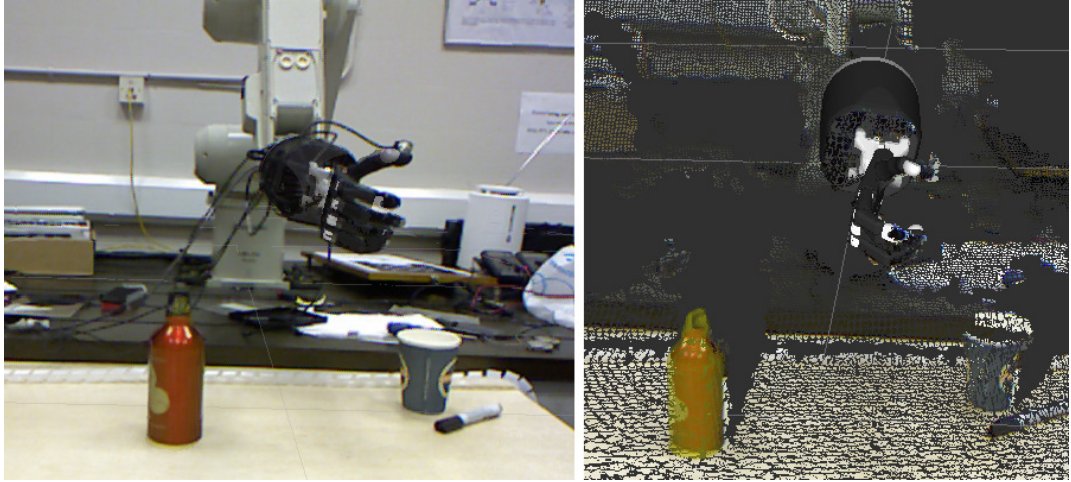
In summary, robot manipulation, *i.e.* the physical interaction between a robot and an object, requires accurate tracking of the object pose but at the same time creates occlusions which can severely compromise the accuracy and robustness of any vision based pose estimation approach. Figure 1.1 illustrates this problem.

Figure 1.1(a), shows, on the left, a grasping situation before the robot reaches to grasp the target object (a red thermal bottle). The object is sitting on the top of a table, clearly visible by the RGB-D camera from which this image was obtained. The vision system is able to accurately identify the object pose, which is shown as the yellow overlay on the right image.

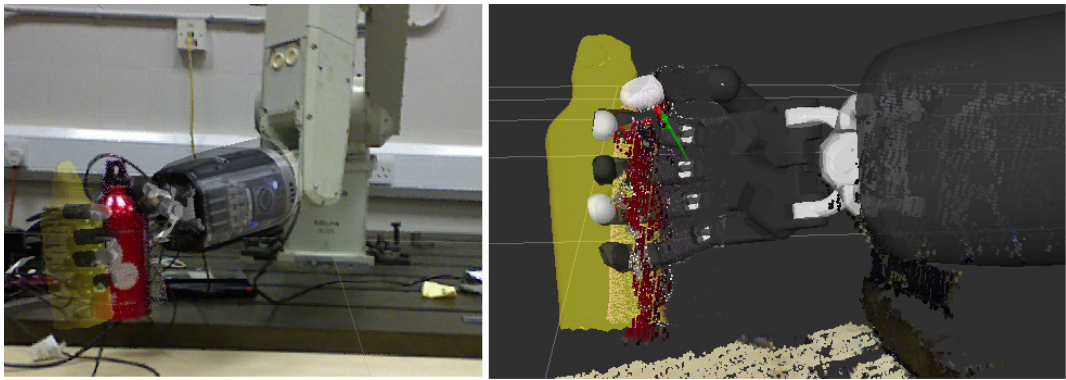
When the robot reaches and grasps the object and the robot fingers envelop the object, the vision system can only partially see the object and it does not easily distinguish points belonging to the robot body from those belonging to the object. This situation is shown in Figure 1.1(b) where the object is inside the robot hand but vision wrongly estimates it to be intersecting the robot fingers.

This thesis deals with this problem, prevalent in robot grasping and manipulation applications, by using the robot’s kinematics and sense of touch to improve the accuracy of the object’s pose estimation. To this end, the two most common tactile sensing modalities are considered: *intrinsic* tactile sensing, which is detailed in Section 2.2.1, and distributed tactile arrays.

Besides the combination of vision and tactile sensing, where tactile sensing is used to rectify a coarse pose obtained from vision, this thesis also presents how the same methodology can be employed to estimate a grasped object’s pose when no vision input is available. This is useful, not only because it provides redundancy in case of a camera malfunction, but also for situations where vision systems are unusable or very unreliable. Circumstances where this limitation is present include hazardous environments such as disaster scenarios where there may be fire



(a) Object pose estimation obtained from vision when the object is clearly visible on top of the table



(b) Effect of the occlusions created by the robot hand in pose estimation accuracy

FIGURE 1.1: Camera's point of view, visualisation of the robot's posture and vision-acquired object pose overlaid in yellow.

and smoke, underwater and complete darkness. This method can also be applied to the manipulation of transparent objects, which are very difficult to track by vision or RGB-D systems.

1.2 Problem Presentation

The object of this thesis is the development of methods that use a robot's sense of touch to improve its knowledge about the in-hand position and orientation of a grasped object.

This estimation of the object pose is done using the following methodology: given an object geometry, the current contact information and possibly a coarse pose estimate (commonly provided by vision), find a new estimate for the pose that matches that contact sensing data. Figure 1.2 schematically outlines the approach followed in this thesis. In this figure, $f_{1...3}$ represent a robot's three fingers touching the object. This contact generates sensory inputs which are encoded into what is referred to as the contact information. The blue object represents the initial pose, usually obtained from a computer vision system. This pose can be displaced from its real pose, as previously shown in Figure 1.1. In other situations, where there is no prior pose estimate available, the initial pose can be chosen to be anywhere in the proximity of the robot hand. Starting from this initial estimate, a new pose is calculated which places the object in a position that is coherent with the current contact information. In other words, the objective is to find the transformation \mathbf{T} , which satisfies the current tactile sensory inputs.

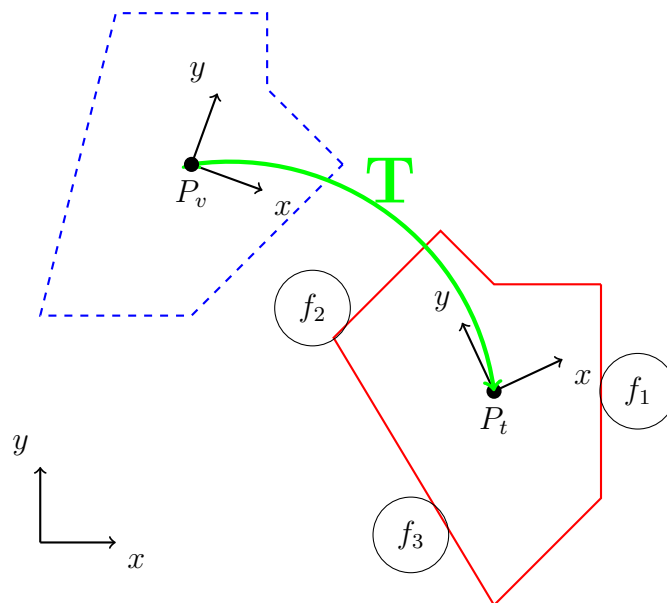


FIGURE 1.2: Thesis objective scheme – $f_{1...3}$ shows the robot finger positions, the blue object is the initial estimate and the red object is the resulting pose, matching the contact information

This problem is formulated as an optimisation problem requiring the following inputs: 1) the object's geometry, either known *a priori* and loaded from a database or acquired online before the grasp is initiated, 2) the current robot hand posture,

obtained from forward kinematics, \mathcal{B}) the contact information from at least two fingers touching the object.

An objective (or cost) function is devised according to the current tactile sensing inputs and the expected contact information if the object was at a candidate solution. This candidate solution is a pose parametrised by a translation vector \vec{t} and a rotation quaternion \mathbf{q} . These parameters \mathbf{x} are introduced in (1.1).

$$\mathbf{x} = [\mathbf{q}, \vec{t}]^T \tag{1.1}$$

$$\mathbf{x} = [q_w, q_x, q_y, q_z, t_x, t_y, t_z]^T$$

Thus, the object of this study is to firstly devise cost functions such that their minimisation leads to a correct estimation of the object's location, and secondly to design and implement optimisation methods that allow that minimum to be found.

In order to reduce computation time, the problem can be postulated as finding a transformation not on the object pointcloud that matches current contact information, but instead find a transformation on the contacts that matches the object's geometry. This allows that, at every function evaluation, the transformation is calculated only for m contacts, instead of transforming the object pointcloud, which is typically made of tens of thousands of points. This transformation, obtained through solving the optimisation problem, is then inverted and applied on the object.

It is important to point out that this estimation is not done continuously, but instead relies on a single measurement. This choice was made due to the high-dimensionality of the problem as well as the inherent difficulties in predicting the movement of an object under multiple external disturbances caused by the robot fingers. Thus, the methods presented in this thesis rely heavily on the usage and interpretation of rich contact information. Exploiting the capabilities of different

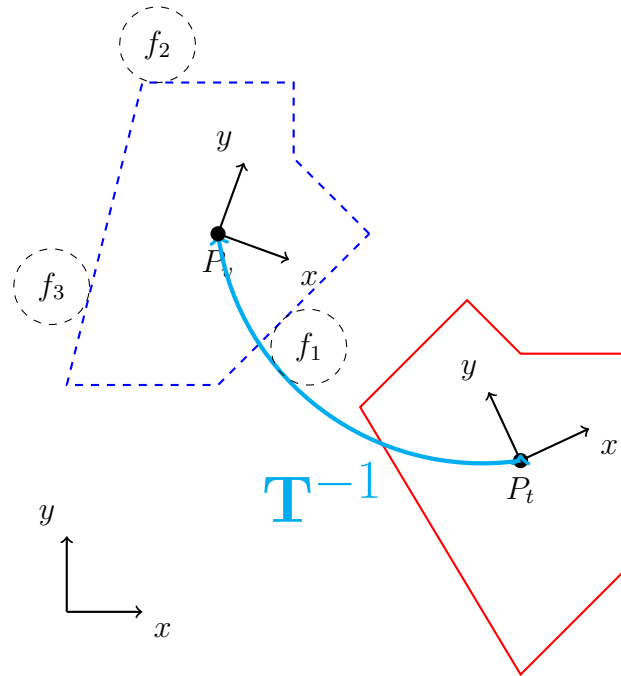


FIGURE 1.3: Thesis objective scheme – $f_{1...3}$ shows the robot finger positions, the blue object is the initial estimate and the red object is the resulting pose, matching the contact information

tactile sensing technologies allows local shape features to be captured. Optimisation algorithms are then used to find poses of the object such that the local features of each contact match the obtained contact information. This methodology differs from most of the existing implementations in literature, which often use limited contact information (*e.g.* contact location alone) and rely on exploration to estimate the object’s pose. Active exploration of a grasped object entails however a number of challenges. First, the aforementioned difficulty of predicting the movement of a grasped object when exploring it. Also, robot hands may not possess the needed dexterity to touch different parts of the object while keeping a stable grasp.

While continuous estimation of the object pose is possible under this approach, simply using the previous resultant pose as the initial pose estimate instead of using the input from vision, this work aims not to be a definitive solution to the real-time time tracking of a manipulated object. Instead, it can be seen as a building block to achieve that purpose, which can be combined, for example in a Bayesian Filter, with an object motion model which performs the *prediction* step,

essential in this type of recursive estimation [2].

Besides estimating the pose of the object, the methods presented in this thesis can also be used to identify an unknown object through the sense of touch, from a set of possible objects whose geometries are known *a priori*. Another application of the proposed approach is related to grasp quality metrics such as those based in the convex hull of the grasp wrench space, which require the knowledge of the object's centre of mass location. Therefore, assessing grasp quality also becomes possible after the accurate estimation of the object pose.

1.3 Research Contributions

The scientific contribution of this thesis to the body of knowledge in robotics and particularly in the field of sensing for robot grasping and manipulation can be summarised by the following achievements:

- Development of a method for object pose estimation in grasping and manipulation settings that rectifies an initial approximate pose using tactile sensing using local optimisation;
 - The description of this work is presented in chapter 3 and was presented initially at the *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*[3], and was further extended and presented at the *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* [4];
- Design of a global optimisation method that can estimate the pose of a grasped object with or without an approximate estimate;
 - The details of this work were published in *Autonomous Robots*, 39(1), 25–41, 2015 [5] and are presented in chapter 4
- Formulation of a descriptor for distributed tactile array data that can be used to match to local object geometry;

- This work is presented in chapter 5 and its results for object pose estimation were published in *IEEE Robotics and Automation Letters*, 1(1), 570–577, 2016 [6]

1.4 List of Publications

Journal Papers

- **Joao Bimbo**, S. Luo, K. Althoefer, and H. Liu, “In-hand object pose estimation using covariance-based tactile to geometry matching,” *IEEE Robotics and Automation Letters*, 1(1), 570–577, <http://doi.org/10.1109/LRA.2016.2517244>, 2016.³
- **Joao Bimbo**, P. Kormushev, K. Althoefer, and H. Liu, “Global estimation of an object’s pose using tactile sensing,” *Advanced Robotics*, 29(5), 363–374. <http://doi.org/10.1080/01691864.2014.1002531>, 2015.²
- H. Liu, K. C. Nguyen, V. Perdereau, **Joao Bimbo**, J. Back, M. Godden, L. D. Seneviratne, and K. Althoefer, “Finger contact sensing and the application in dexterous hand manipulation,” *Autonomous Robots*, 39(1), 25–41. <http://doi.org/10.1007/s10514-015-9425-4>, 2015.

Papers in Peer-Reviewed Conferences

- J. Back, **Joao Bimbo**, Y. Noh, L. Seneviratne, K. Althoefer, and H. Liu, “Control a contact sensing finger for surface haptic exploration,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2736–2741.

- **Joao Bimbo**, L. D. Seneviratne, K. Althoefer, and H. Liu, “Combining touch and vision for the estimation of an object’s pose during manipulation,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 4021–4026.¹
- **Joao Bimbo**, S. Rodriguez-Jimenez, H. Liu, X. Song, N. Burrus, L. D. Senerivatne, M. Abderrahim, and K. Althoefer, “Object pose estimation and tracking by fusing visual and tactile information,” in *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2012, pp. 65–70.¹
- H. Liu, X. Song, **Joao Bimbo**, K. Althoefer, and L. Senerivatne, “Intelligent fingertip sensing for contact information identification,” *Advances in Reconfigurable Mechanisms and Robots I*, pp. 599–608, 2012.

Presentations in Workshops

- **Joao Bimbo**, K. Althoefer, and H. Liu, “Object pose estimation using tactile to geometric covariance matching,” in *IROS Late Breaking Results Session* [Poster], 2015.³
- **Joao Bimbo** and H. Liu, “Soft fingers for robotic grasping,” in *Perceptions on Soft-based Contact workshop at IEEE CASE 2015* [Talk], 2015.
- J. Back, **Joao Bimbo**, M. Addison, U. Cupcic, G. Cassidy, R. Walker, L. D. Seneviratne, K. Althoefer, and H. Liu, “Finger surface following control through intrinsic contact sensing,” in *Autonomous Grasping and Manipulation: An Open Challenge at ICRA* [Poster], 2012.
- **Joao Bimbo**, S. Rodriguez-Jimenez, H. Liu, N. Burrus, L. D. Senerivatne, M. Abderrahim, and K. Althoefer, “Fusing visual and tactile sensing for manipulation of unknown objects,” in *Mobile Manipulation Workshop on Interactive Perception at ICRA 2013* [Poster], 2013.¹

- **Joao Bimbo**, H. Liu, L. D. Senerivatne, M. Abderrahim, and K. Althoefer, “Active perception of objects for robot grasping,” in *Closing the Action-Perception Loop Workshop at IROS 2012* [Presentation], 2012.¹
- **Joao Bimbo**, “Managing coordinate frames with ROS,” in *Handling ROS Introductory tutorial to ROS and its use for robot in-hand manipulation Workshop at IROS* [Workshop], 2012.

1.5 Thesis Structure

Having outlined the purpose of this thesis, the next chapter provides a literature review on grasping, tactile sensing and object pose estimation in the context of robot grasping. It also provides an overview of the mathematical background required to understand the concepts presented in this thesis.

Chapters 3, 4 and 5 contain the main body of this thesis and detail the research contributions detailed in Section 1.3. Each of these chapters begins with a short introductory summary of its contents and ends with some concluding remarks on the results obtained. The last chapter discusses the thesis’ findings and limitations, and also proposes future research directions.

¹ The contents of these articles is presented in chapter 3

² The contents of these articles is presented in chapter 4

³ The contents of these articles is presented in chapter 5

Chapter 2

Background

Chapter Summary

This chapter presents some introductory concepts and reviews the literature relevant for the understanding of this thesis. It contains a general overview of the field of robot grasping and manipulation and an outline of different tactile sensing approaches used in the field. A literature review on methods to estimate an object's pose from different sensing sources is also provided along with an introduction to pertinent mathematical concepts.

2.1 Robot Grasping and Manipulation

Man's dominion over Nature can be partially attributed to his ability to grasp and manipulate objects. It was through the crafting and handling of artifacts that human beings began shaping the world around them, and it should come as no surprise that a large part of the human motor cortex is dedicated to manipulation [7]. The importance of manipulation in the human brain also serves as an indication of the complexity that is associated with this task.

Providing a robot with a similar capability to grasp and manipulate objects in complex, unstructured environments, *i.e.* where there is significant uncertainty, has proven to be one of the biggest challenges in present-day robotics. To date, the ability to skillfully wield objects of different shapes, sizes and other physical properties at a human-like level of dexterity is yet to be attained by robots.

One of the first notable examples of a design of a robotic hand was the 1962 paper by Tomovic and Boni [8], who proposed a design and control scheme for an artificial limb that could be used both as a prosthetic or an autonomous hand. The work by Hanafusa and Asada [9] presents one of the earliest designs for an autonomous robot hand with compliant fingers with focus also on grasp planning and stability. Salisbury and Craig [10] introduced a tendon-driven sensorised robot hand, together with a control scheme and introduced the concept of a Grasp Matrix. An extended taxonomy of human grasps and their transferability to robots was presented by Cutkosky [11, 12] who, along with Mason [13] undertook pioneering work on grasping, particularly from the robot hand design and mechanical analysis point of view.

A formal definition of the necessary conditions for the stability of a grasp was presented by Fearing [14] that take into account the forces applied \vec{F} and contact locations \vec{r} of every i^{th} finger touching the object, as well as the friction coefficient μ . F_n and F_t denote the normal and tangential force respectively. These conditions are:

1. No net force or torque: $\sum_i \vec{F}_i = 0, \sum_i \vec{r}_i \times \vec{F}_i = 0$

2. No slippage – all forces within the friction cone: $\mu F_n > |F_t|$
3. Any arbitrary applied force can be resisted by increasing the grasping force.

This last criterion conveys, if extended to wrenches, what is commonly referred to as force-closure [15]. This concept laid the foundation for the computation of the finger positions in a grasp, *i.e.* grasp synthesis. Nguyen developed a method to compute force-closure grasps using the space spanned by the contact wrenches [16]. This approach was also followed Ferrari and Canny, who proposed a criterion for grasp quality based on the construction of the convex hull generated by the set of all possible wrenches [17]. Both the total volume and the radius of the largest hypersphere centered at the origin that can be contained inside this convex hull can be used to give an indication of grasp quality. Figure 2.1 outlines these concepts, where in Figure 2.1(a) three fingers are sufficient to maintain a force-closure grasp on the object O . Figure 2.1(b) shows the quality criterion according to Ferrari and Canny in 3-D, where the sphere is tangent to one of the planes generated by the convex hull. In a real scenario, a six dimensional hypersphere is used to accommodate both force and moment.

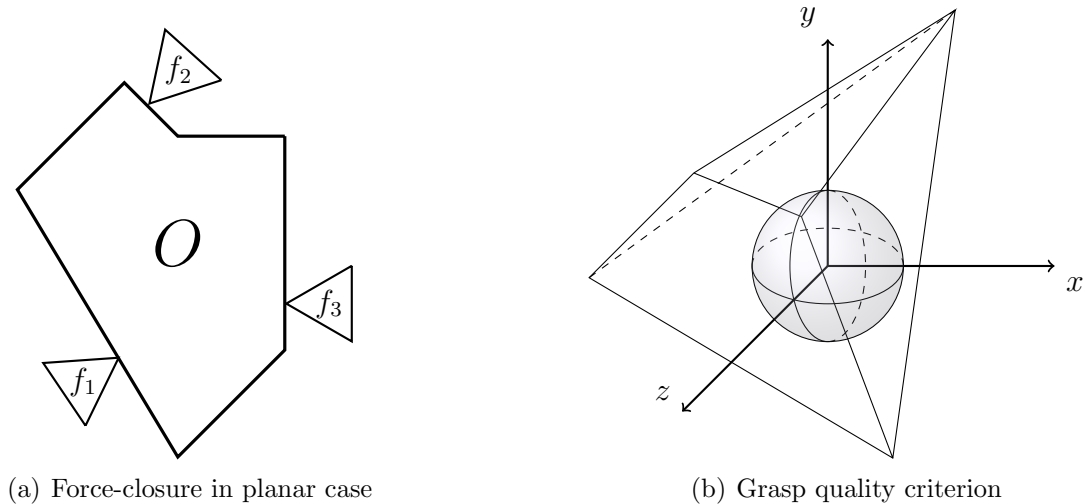


FIGURE 2.1: Grasp stability diagrams.

Given the large size of the Grasp Wrench Space (GWS) – the space of possible wrenches that can be applied on an object –, this requirement becomes very strict and computationally expensive. More recently, more relaxed criteria have been

put forward, such as the Object Wrench Space (*OWS*) [18] and the Task Wrench Space (*TWS*) [19]. Instead of testing that the current finger positions can oppose any arbitrary force, these concepts limit the possible space, respectively, to the set of wrenches that are possible to be applied on the grasped object, or that arise from performing the desired task. Other strategies relying on a similar methodology and concepts, but without specifically computing the convex hull have been proposed in [20] and [21], resulting in a decrease in the computation time required to synthesise force-closure grasps.

While analytical techniques were predominant in earlier research efforts, recent advances in artificial intelligence and machine learning have led to the development of alternative approaches to robot grasping and manipulation. Robots can learn how to grasp objects through human demonstration [22, 23], or with human aid [24]. These approaches rely on humans performing grasps, with the finger movements being learned by the robot and the knowledge transferred to the robot as motor skills. Due to the high dimensionality of the task, and to cope with the number of degrees-of-freedom (DOF) in the human – and possibly the robot – hand, a technique that encodes *grasp synergies* was proposed by Santello *et al.* [25]. Grasp synergies, also known as postural synergies and eigengrasps, are the directions of highest variance in the joint space obtained from applying Principal Component Analysis (PCA) [26] on the joint angles of the hand in a number of trial grasps. It is shown that two principal components account for 80% of the variance in grasps performed in 57 objects of different shapes and sizes. This dimensionality reduction allows the control of hand movement along the directions of these principal components, greatly decreasing the computational effort required to learn and reproduce grasps [27] and also allows their easier generalization [28, 29].

Besides kinaesthetic learning, reinforcement learning can also be used using data that is obtained from other sensing sources. A robot can learn to stably grasp an object by repeatedly grasping a number of objects and finding the sensory inputs which correspond to stable grasps [30]. Vision is another source of sensory inputs that can be used to choose grasping points which are likely to lead to

stable grasps [31], even when the object shape is not accurately known [32] or not known at all, finding only regions in the object surface where to place the robot fingers [33, 34].

Machine learning strategies have shown remarkable results in the field of robotics, endowing robots with the capability of performing complex behaviours, which are inherent in robot grasping and manipulation tasks. However, these approaches suffer from a number of drawbacks, chiefly the lack of transparency they provide to the user, as the inner workings of a learned strategy may not be meaningful to a human observer. The problem of overfitting and the difficulty to transfer these tasks to different settings, such as working with different hardware becomes difficult and usually requires re-training on the new robot platform. In the particular case of robot grasping, machine learning may not be able to cope with subtle changes in the task. For example, it would be difficult for a machine learning strategy to accommodate for changes if two similar objects possess slightly different physical properties, such as friction or weight, that would require completely different grasps. An approach that combines analytical computations with machine learning elements is likely to be the key for the success of future robot grasping and manipulation systems.

2.2 Tactile Sensing

Research in tactile sensing has developed hand-in-hand with research in grasping and manipulation. From the beginning, it was clear to researchers in the field that, similarly to humans, who struggle to perform complicated manipulation tasks when their sense of touch is impaired [1, 35, 36], a robot would also require an acute sense of touch in order to manipulate objects [8, 10]. This sensing modality is what allows humans (and robots) to detect, measure and characterise the physical interactions with themselves and the environment. We humans rely on the sense of touch for three types of activities: manipulation, exploration and response [37]. While the entirety of our bodies' surface is covered with mechanoreceptors (nerve

endings responding to mechanical stimuli), the glabrous skin in our hands and particularly our fingertips possess a much larger density of afferents [38]. Four type of afferents are present in our glabrous skin which are tasked with sensing different events [39]:

- FA-I (fast adapting type I) Meissner’s corpuscles are sensitive to high frequencies ($\sim 5 - 50$ Hz)
- SA-I (slow adapting type I) Merkel’s discs are sensitive to low frequencies (< 5 Hz) and static forces
- FA-II (fast adapting type II) Pacinian corpuscles sense transients and higher frequency vibrations ($\sim 40 - 400$ Hz)
- SA-II (slow adapting type II) Ruffini endings detect static forces and skin stretch

During manipulation tasks, these different afferents “fire” at different times and with different intensities, allowing us to experience the weight of the object, feel its texture, detect its slippage, sense its shape, etc.

Currently, no single tactile sensing technology exists that can provide the same richness of information we have available at our fingertips, with all these different properties and events being perceived simultaneously [40]. However, when dealing with only one single tactile perception task at a time, it is not uncommon for robots to match or even outperform humans. An example of this can be seen in any commercial force-torque sensor, which is likely to have a much higher sensitivity and resolution than our fingertips when quantifying interaction forces. Distinguishing between similar materials such as brass, aluminium and steel may be challenging for a human, but robots have no problem identifying these materials through their friction properties [41] or texture [42]. Incipient slip can be detected very rapidly by robots [43, 44] and slippage can be prevented altogether by predicting the force ratio at which break-away will occur [45].

Different taxonomies exist for classifying tactile sensors. The classification can be based on the technology used [46] – optical, conductive elastomer or silicon strain gauges –, sensor dimensionality [47] – zero, one or two dimensions –, sensing principle [48] – array sensors and force-torque sensors –, or the location of the sensor [49] – intrinsic and extrinsic tactile sensing. The two last classification arrangements – by sensing principle or by the location of the sensors – can be thought of being somewhat coincident and allow a parallel to be drawn between robot and human tactile sensing [49]. In this metaphor, intrinsic tactile sensing, which is obtained from force-torque sensors mounted within the body of the robot, corresponds to the *kinaesthetic* sensing of humans, while extrinsic tactile sensing is given by tactile arrays distributed over the robot surface and is analogous to human *cutaneous* sensing [50]. This classification seems the most appropriate for this overview of tactile sensing as it also determines the two different approaches for pose estimation presented in the next chapters.

2.2.1 Intrinsic Tactile Sensing

Intrinsic sensors are located within the mechanical structure of a robot [51], such as force-torque or joint angle sensors. Bicchi *et al.* introduced a sensing scheme to obtain tactile information from an intrinsic six axes force-torque sensor mounted at the robot fingertips, under a rigid parametrisable convex surface [52]. By assuming that the contact is approximately rigid, that there is only one contact surface acting on the fingertip and that the force is always compressive, one can obtain the following tactile information from this scheme [53]:

- contact centroid on the fingertip (a point always within the contact area) p_c
- magnitude of the normal component of the contact force \vec{F}_n
- magnitude and direction of the friction force \vec{F}_t
- torque generated around the contact normal direction $\vec{\tau}$

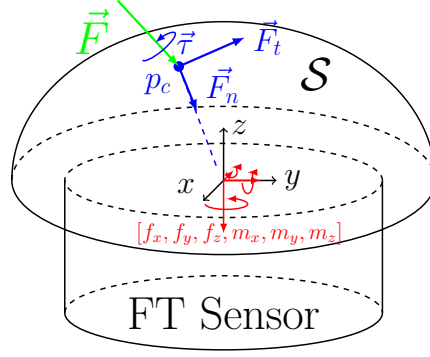


FIGURE 2.2: Intrinsic Tactile Sensing scheme – acting force in green, measured quantities in red, computed parameters in blue

Figure 2.2 shows a diagram of this tactile sensing approach. Given a contact force \vec{F} in green, the force-torque sensor mounted under the surface \mathcal{S} registers the forces f_x , f_y , f_z and the moments m_x , m_y , m_z . A force balance allows the computation of the line in space where the force is acting, and the contact point can be calculated when the surface equation is added.

For a better understanding, consider the situation in Figure 2.3, where a force is being exerted in the xy plane and thus only two forces f_x , f_y and one moment m_z are present. Measuring these three quantities and applying the force balance in Equation (2.1), the dashed line on the figure can be obtained. The intersection between this line and the surface equation identifies the contact location uniquely, since the surface is convex and only compressive forces are allowed.

$$\vec{M} = \vec{r} \times \vec{F} \iff m_z = r_x \cdot f_y - r_y \cdot f_x \quad (2.1)$$

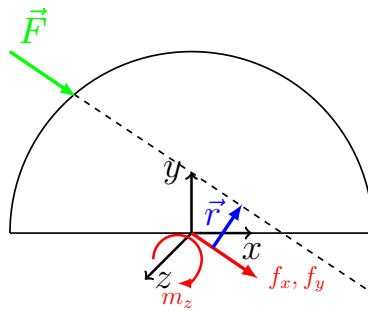


FIGURE 2.3: Intrinsic Tactile Sensing diagram – a force is exerted in the xy plane

In the general case, another unknown q is added, which is the torque around the surface normal, generated by friction forces. Thus, finding the contact point $p_c = [x, y, z]$ is done simply by solving the system of equations in Equation (2.2), which describes the force balance for a single contact and the surface equation:

$$\begin{cases} p_c \times \vec{F} + \vec{\tau} = \vec{M} \\ \mathcal{S}(x, y, z) = 0 \end{cases} \quad (2.2)$$

Separating these equations for each dimension and putting them in the relaxed form yields (2.3), which is the system of equations that needs to be solved to find the contact location $p_c = [x, y, z]$ and k which is proportionally related to $\vec{\tau}$.

$$\vec{H}(x, y, z) = \begin{bmatrix} k\nabla\mathcal{S}_x - f_y z + f_z y - m_x \\ k\nabla\mathcal{S}_y - f_z x + f_x z - m_y \\ k\nabla\mathcal{S}_z - f_x y + f_y x - m_z \\ \mathcal{S}(x, y, z) = 0 \end{bmatrix} = 0 \quad (2.3)$$

After finding the contact location p_c , it becomes trivial to find the normal and tangential components \vec{F}_n and \vec{F}_t . The normal vector \hat{n} , coinciding with the surface normal at point p_c , is obtained simply by the gradient of the surface equation \mathcal{S} at that point, normalised as shown in Equation (2.4)

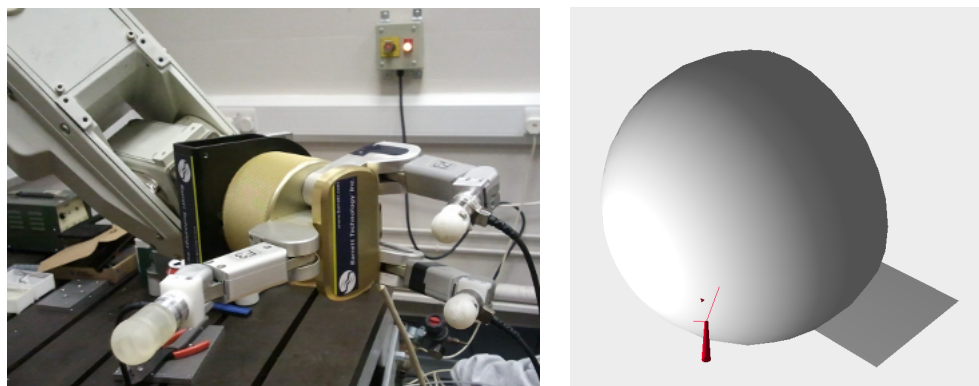
$$\hat{n} = \frac{\nabla\mathcal{S}(p_c)}{\|\nabla\mathcal{S}(p_c)\|} \quad (2.4)$$

The normal component of the contact force is obtained by projecting the total force onto the normal direction vector \hat{n} , while the tangential force is simply the rejection of this vector, as shown in Equation (2.5) and Equation (2.6)

$$\vec{F}_n = \frac{\vec{F} \cdot \hat{n}}{\hat{n} \cdot \hat{n}} \cdot \hat{n} \quad (2.5)$$

$$\vec{F}_t = \vec{F} - \vec{F}_n \quad (2.6)$$

Figure 2.4 shows an implementation of the Intrinsic Tactile sensing scheme, where Figure 2.4(a) shows the hardware implementation with hemispherical fingertips. Figure 2.4(b) shows a visualization tool depicting that hemispherical fingertip, with the total force vector as a red cone. The lines coming out of the tip of the cone are the normal and tangential components of the force, and the arrow on top of the cone is the magnitude of the local torque. This sensor was evaluated by Liu *et al.* [54], where the Levenberg-Marquardt algorithm [55, 56] was used to solve the system of equations in Equation (2.3), and presented an accuracy of 266 μm , running at frequencies over 800 Hz.



(a) Barrett Hand with Force-Torque sensors mounted on the fingertips (b) Intrinsic Tactile sensing visualisation

FIGURE 2.4: Overview of the Intrinsic Tactile sensor

This Intrinsic Tactile sensing strategy provides very accurate results in terms of contact location, as well as information on the normal and tangential force, which is crucial for manipulation tasks. It has successfully been used for surface identification [41], slippage detection and prevention [45, 57], surface following [58], etc. However, it is limited to a single contact point or surface, as the intrinsic sensor can only measure the net force acting on the sensing area and it also assumes

that the contacts are rigid. This latter limitation implies that the contact cannot generate a very large friction force, which is essential for manipulation. To address this restriction, Liu *et al.* devised and validated a method that can deal with small deformations on the fingertip [59], achieving an accuracy on the contact location of $320\ \mu\text{m}$ on a fingertip covered with a thin rubber layer which provides friction and compliance.

In summary, implementing an intrinsic tactile sensing scheme on the fingertips of a robot hand presents significant advantages for robot grasping. The rich information provided by this method has been demonstrated to enable or facilitate the following abilities:

- directly measure the magnitude and direction of external forces, enabling safe interaction with the environment
- obtain an accurate measurement of normal force which can be used for force control
- explore an object surface through the simultaneous control of normal and tangential forces [58]
- accurate contact location estimation – $266\ \mu\text{m}$ [60]
- slippage detection and prediction [57]
- surface material classification through its friction parameters [61]
- increase grasp robustness [53]
- assess grasp quality (force and form closure), from the contact locations and normals.

2.2.2 Tactile sensing arrays

Tactile sensors usually refer to sensors distributed over a robot's surface that detect and measure local contact information. This class of sensors are also termed



FIGURE 2.5: Intrinsic force-torque sensor with soft rubber layer

as extrinsic tactile sensors and are deployed to convey cutaneous information, mimicking the role of the skin in humans and other animals [49]. Tactile sensors can be used to measure a number of different physical quantities, mainly pressure, skin deformation and skin acceleration, or a combination of these [37]. These different modalities also mirror the previously mentioned afferents present in human glabrous skin (FA-I, SA-I, FA-II and SA-II).

The main requirements put forward in the literature for distributed tactile sensing technologies intended for robot grasping applications are [49, 62]:

- Spatial resolution around 1 mm;
- High force sensitivity and range of 1 to 1000 gram-force;
- Low hysteresis
- Frequency response up to 400 Hz.

Dynamic tactile sensing is typically carried out by embedding piezoelectric [44] or accelerometers [43] into a soft fingertip, detecting the vibrations associated with transient events such as contact, lift or incipient slip, which is characterised by stick-and-slip phenomena [63]. This approach of embedding accelerometers [64] or force-torque sensors [65] in a soft medium was also successfully employed to identify surface materials. Given the object of this thesis, the dynamic response

of tactile sensors is not as important as the measurement of distributed forces, pressures and skin deformation on a robot fingertip.

Different technologies have been studied in terms of sensing principle and manufacturing technology, with the main contributions to the design of tactile sensors coming from the fields of medical robotics and robot manipulation. The most widespread type of tactile sensor is the distributed pressure array [37]. These sensors are commonly distributed in a deformable matrix of elements and measure the pressure exerted in the normal direction in each element – also known as *taxels*. Common sensing principles of this class of sensors include piezoresistive, capacitive, optical and MEMS barometers. Covering a large part of a robot finger surface, these sensors can detect contact location and, after appropriate modelling and calibration, also estimate normal force and surface shape.

Distributed pressure array devices can already be considered a mature technology, with capacitive touch sensors existing in current touch screens and touchpads and FSR (Force Sensing Resistors) being used in devices like joysticks. For researchers, also “off-the-shelf” commercial solutions are available, such as PPS [66], Tekscan [67], Weiss sensors [68] and Takktile [69]. Other prominent examples of distributed tactile sensors exist, such as the ROBOSKIN modular sensor patches [70], the DEXMART robotic hand force/tactile sensor [71], the optical based TACTIP [72] and the biomimetic BioTac [73]. Other tactile sensors able to measure normal and tangential force were presented by D’Amore [74] and Yousef [75]. Extensive literature reviews have been carried out over the years, with the ones from Nicholls [47], Dahiya [49], Yousef [76] and Kappassov [62] standing out as very complete and thorough.

Representing and interpreting the data from distributed tactile sensors in an effective way is also an open challenge in tactile sensing. A common way of handling data from a tactile array is to treat the data as a “tactile picture”, and apply methods which are inspired by those commonly used in computer vision. In these methods, each tactile element is considered a pixel, different features are extracted from the tactile image [77] and descriptors such as SIFT are used to encode the

data [78]. A framework which maps sensing elements on the robots in a 2D *somatosensory* map has been proposed by Cannata [79], inspired by the human brain. Machine learning approaches to represent tactile data [80] also exist, with dimensionality reduction techniques such as PCA being applied, which treat every tactile element as a dimension [81–84] or take the spatial location of the elements into account [85].

Besides grasp control, which is present in nearly every study, tactile sensing has seen other uses in the field of robot grasping and manipulation. Tactile data has been used to detect events in a pick-and-place scenario, automatically triggering transitions between the different phases of this task (grasp, lift, replace, unload, etc.) [86]. The inverse dynamics of a humanoid robot were learned using a distributed tactile array that covered the robot’s arm. As the robot arm collided with the environment, tactile and force/torque data was used as an input to learn the robot’s joint torques, outperforming an analytical approach [87]. Tactile sensors have also been used to analyse surface shapes and textures [88] and for object recognition and pose estimation [89, 90]. A data-driven method was presented by Bekiroglu [30] that was able to predict grasp stability based on tactile data and without any analytical consideration such as the ones presented in Section 2.1.

2.3 Sensor Fusion

It is impossible or unfeasible for a robot to know the complete state of its surroundings when it is operating within an unstructured environment. By equipping a robot with sensors that provide different types of information, which may be independent, complementary or even overlapping, an intelligent system can increase its knowledge of the world around it [91]. Multiple sensing sources can grant a synergistic effect, with the data from one sensor serving as cues for the operation of other sensors, and also contribute to a more robust system, through the redundancy in the information which allows the fusing of data to reduce sensor error and noise, or persistence in case of malfunction [92].

In the field of robot grasping, the fusion of multiple sensing sources has been the object of extensive research. Besides tactile and force sensing, vision also plays an important role in grasping and manipulation, particularly during the pre-grasp phase, when the grasp is planned and the robot hand approaches the object [93]. This is similar to the approach taken by humans, who also use vision to plan the positioning of the fingers on an object, while relying mainly on their sense of touch during the grasp [94].

Peter K. Allen from Columbia University was one of the early researchers that combined vision, force and tactile sensing in the context of robot grasping [95, 96]. A robot hand was equipped with a tactile sensor suite and a real-time vision tracking module was added to the system, besides the native joint sensor and strain gauges which measure the load on the finger's outer link. In different experiments vision was used to detect contact location, measure joint angles, find the position of the fingers relative to the object, and drive a manipulation task through visual servoing of the fingers, using force and tactile sensing to ensure a tight grasp [97]. One of the first instances of autonomous grasping was presented by Kragic, Miller and Allen, who combined a real-time object tracking system with the GraspIt! [98] grasp planner. In this work, the object pose is first detected from vision, using the previously known object geometry [99]. The object and the robot hand models are used by the grasp planner to compute a stable grasp, according to the GWS criteria (see Section 2.1). After the grasp is finished, vision is again used to monitor the execution of the task and to servo the robot arm to reach a desired object pose.

In a work that reinforced the understanding of the importance of integrating these three sensing modalities (force, vision and tactile), Prats *et al.* [100] compared the effectiveness of different combinations of these sensors. Using a mobile manipulation platform consisting of a mobile robot, a 7 degree of freedom redundant arm, a robot hand with force/torque and tactile sensing, and an external camera, the robot was tasked with opening a sliding door. The task was carried out and a comparison was made between using force alone, using vision and force and combining vision, force and tactile sensing. When combining all three sensing modalities, tactile sensing was used to ensure that the door handle was aligned with the robot

hand, with this combination outperforming the other two cases, with the robot succeeding in opening the door in all experiments.

Vision and tactile sensing are also used in the work by Bekiroglu *et al.*, with vision being used to estimate the pose of the object and tactile data for assessment of grasp stability in a data-driven fashion [101].

An approach to determine the location of contacts in space and the interaction forces between a robot and an object, which combined different sensing modalities, was proposed by Felip *et al.* [102]. This framework discretised the 3D space into voxels, and contact hypotheses from different sensing sources – force-torque, tactile, visual, depth – were added to this hypotheses space. Each sensing modality has a measurement model, which attributes a value of likelihood for the readings. When hypotheses from multiple sources coincided at the same voxel, these were fused together using the combination of their likelihoods.

2.4 Object Pose Estimation

2.4.1 Vision-based Object Pose Estimation

One of the key factors in the success of a grasping task is the accurate estimation of an object's pose. When a robot operates in an unstructured environment, the position and orientation of the target objects in the environment are not known *a priori* and require their robust estimation [103]. Even small errors when estimating the location of the target object can lead to the wrong placement of the fingers on the object, create false assumptions on grasp stability and even compromise the success of a grasping task. This requirement is even more imperative during manipulation tasks that, by definition, aim to modify an object's position and orientation [104].

The most common means to obtain an object's pose in robot systems designed for robot grasping is through the use of computer vision – single, multiple (stereo) or

RGB-D cameras. The work by Kragic *et al.*, previously discussed in Section 2.3, details one of the first occurrences of using vision to estimate an object's pose in the context of robot grasping [105]. The same author further developed a method that used stereo and *foveal* cameras, which could focus the gaze of the robot on the desired object. The proposed system could segment, identify the object and estimate its pose [106].

Stereo vision is also featured on the head of the ARMAR humanoid robot, developed in the Karlsruhe Institute of Technology. This vision system relied on the combination of model and appearance-based methods [107]. Model based methods use the object's 3D geometric model while methods based on appearance use visual features such as color and texture [108].

It is also possible to estimate the object pose using the object's 3D model and a single camera image, by finding the 2D-3D point correspondences through minimisation [109]. In a work by Detry *et al.*, vision was used to estimate the pose of the object to be grasped, together with cues which might indicate affordance locations [110]. These affordances are geometrical patterns present in object parts that when gripped lead to stable grasps. Examples of affordances include the grip of a frying pan, the stem of a glass or the handle of a mug.

With the generalised availability of RGB-D cameras, providing synchronised color and depth information in real-time, computer vision approaches for robot grasping systems have seen remarkable developments. These systems use a structured light pattern projection, come at a low price and are extremely suited for robot grasping, which is typically done indoors, with the camera at a distance between 1 and 2 meters of the target object.

Burrus [111] proposed a method to recognise and estimate the pose of objects using this type of camera, furthermore reconstructing the object's geometry from the visible points. It is pointed out in this early work, as far as structured light RGB-D cameras have been used, that both object recognition and pose estimation greatly improve when using depth sensing.

Other object pose estimation and tracking methods have been developed in recent years, which take advantage of these new cameras [112–114], along with libraries such as the Point Cloud Library (PCL), that incorporate trackers as well as common tools to operate these depth images [115]. A very detailed survey of object tracking using traditional cameras was done by Yilmaz *et al.*, providing also an introduction to the topic [116].

2.4.2 Tactile-based Pose Estimation

As previously discussed in Section 1.1, vision alone may not ensure the required accuracy for pose estimation when the object is placed inside the robot hand. Figure 1.1 exemplified how a vision-based object tracker may provide an accurate estimate of the object pose when the item is sitting on top of a table, and a large part of it is seen by the camera. The occlusions generated by the robot body hide a significant part of the object, impairing the performance of the vision system. Besides, vision may not be suited for estimating the pose of objects that are transparent, or too small to be detected by vision. Some environments may also hinder, or even prevent the usage of vision systems, such as underwater [117] or hazardous scenarios with fire, smoke and debris [118]. The importance of redundancies and synergies created by the combination of different sensing sources, highlighted in Section 2.3, supply further demonstration of the advantages of using tactile sensing to estimate the object’s position and orientation [119].

The possibility of recognising and estimating the pose of an object using tactile sensing, with or without the support of a vision system, was present from the early days of research in manipulation. Gaston and Lozano-Pérez [120] introduced the concept of an “interpretation tree”, where an object’s identity, possible poses and relations between the object surfaces and contacts were laid out. The problem of recognising and locating the object was solved through taking measurements that pruned this interpretation tree and matched the contact points obtained with the edges of the tree. This search-tree approach was followed by Siegel *et al.* [121] to

find the pose of an object whose geometric model is known *a priori*, using only joint position and joint torque sensors. Contact locations were assigned to different object vertices where the distance between the contacts were consistent with the distance between vertices.

A robot system, consisting of a robot arm, a gripper with array sensors, and a camera, was used to recognise an object sitting on a surface, by creating a tree of possible objects based on feature vectors [122]. These feature vectors were generated using the image moments from the camera and the tactile array data, which was also treated as an image. The tree was traversed using the features in the camera image first, and then the tactile images. If the object was not yet discriminable, it would be moved into another position and the process restarted.

Tactile exploration was introduced around the same time, with a theoretical framework to recognise and localise objects using tactile sensing that used the same interpretation tree approach. This strategy used 2D planar objects and sequentially found paths for probing the object that necessarily discriminate between the possible objects and poses using the geometric intersection of these possibilities [123].

A method that fused vision and tactile sensing was presented by Honda *et al.*, which used a multi-fingered hand and a stereo camera to determine an object's pose [124]. An initial pose of a cylindrical object was obtained from vision, by projecting a light pattern onto the surface of the object to generate visual features. The contact locations between the fingers and the object were obtained from tactile sensing and forward kinematics and the object's pose was determined through the minimisation of the distance between the contact locations and the object surface.

An approach that matched finger contact locations to object facets using the DLR hand was presented by Haidacher and Hirzinger [125], where a description for each object was generated offline, that contained the relations between its facets. During contact, the tactile measurements from the robot fingers were used to search the database for consistent matches to the relationships between facets.

More recently, methods to locate a grasped object using tactile sensing have posed the question as a state estimation problem solved through Bayesian Filtering. This approach aims to continuously estimate the object state (its pose) through recursively updating the probability distribution over possible states, given a set of measurements. Bayesian Filtering requires the system to satisfy the Markov property, *i.e.* the current state x_t depends only on the previous state x_{t-1} and action u_t or, in other words, there is no memory of past actions u and measurements z . It also requires an update (or motion) model ($p(x_t|x_{t-1}, u_t)$, *i.e.* the probability of transitioning to a state given the previous state and action) and a measurement (or observation) model ($p(z_t|x_t)$, *i.e.* the probability of measuring z_t given that the world is in state x_t). The posterior belief ($bel(x_t)$) is recursively updated by first integrating the product of the previous belief and the motion model, and multiplying it by the measurement model. This Bayesian Filter method is shown in Equation (2.7) and (2.8), where η is a normalisation factor [2].

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx \quad (2.7)$$

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t) \quad (2.8)$$

Bayesian filtering, and in particular particle filters, where the posterior is represented by a set of samples drawn from the distribution instead of a parametric form (*e.g.* a Gaussian distribution), has been extensively and successfully used in estimation problems such as localising a mobile robot [126]. However, estimating the 6-DOF pose of an object (3 dimensions for position and 3 for orientation) with a particle filter becomes unfeasible due to the “curse of dimensionality” that exponentially increases the required number of particles used for estimation. As an example, if a 3-DOF problem such as mobile localisation (2 dimensional position and one orientation angle) takes 1 second to achieve a desired accuracy, the corresponding 6-DOF problem would take 1.5 years [127]. This presents a major difficulty for applying the standard particle filter to this problem, to which many alternatives have been suggested to simplify the computation.

Sensing data from a stereo camera, a force-torque sensor mounted at the wrist of a robot hand with joint encoders was fused in the measurement vector of an Extended Kalman Filter (EKF), that estimated the discrete contact modes [128]. These contact modes were the space of all possible combinations between fingers in contact and the surfaces of the object, which restricts the usage of this strategy to objects of simple geometry.

Corcoran and Platt [129] developed an approach based on Bayesian filtering which incorporates a measurement model of “negative” contacts, *i.e.* the likelihood of not touching the object in a region outside of the object surface. This was implemented along with a closed form approximation of the “positive” contacts measurement model, which assumed the likelihood of contacts to follow a Gaussian distribution.

The work by Petrovskaya and Khatib [127, 130] presents a very interesting approach to using Bayesian state estimation to the problem of object localisation. In this work, a probe mounted on a robot arm served as a single robot finger which explored the object and collected measurements of contact locations and normals. The proposed method, termed Scaling Series (SS), combines a special particle filter with an annealing technique that “heats up” the measurement model, making the distribution broader (noisier) and easier to sample from, requiring less particles. The main feature of this work is that in this particle filter-like approach, particles do not represent single hypotheses but rather a region in the search space. These particles increase granularity (shrink) with the progress of the algorithm, increasing the accuracy of the estimation, while also pruning regions with low probability. The combination of these features leads to the acquisition of accurate pose estimates while also allowing a trade-off between the desired accuracy and run time.

Starting from a pose estimate obtained by vision, a particle filter was implemented that used a collision detector and tactile sensing as measurement models and the grasp matrix as the update model [131]. As the robot contacts the object, the weight of each particle representing a pose is determined by whether the object’s

geometry does not penetrate any of the robot links' and the distance between the contacts and the object surface.

The discriminatory nature of tactile sensors, *i.e.* they either detect contact or the absence of it, results in a very “spiky” measurement model, which can lead to poor performance of particle filters for object pose estimation based on tactile sensing. This problem was tackled by Koval *et al.* [132] through combining both observations (contact and no-contact) into a measurement model and the introduction of a manifold particle filter. This particle filter takes advantage of the fact that, during contact between a robot and an object, all the possible object poses lie on the lower dimensional contact manifold. This manifold is built from the robot and object geometries and represents the boundary between the space of poses which penetrate the robot geometry and the poses which do not activate any tactile sensor. Assuming that the object is lying on a surface, the robot sweeps the surface to find the object and estimates its 3 DOF pose.

Exploring the object to increase the available information about the object pose and/or identity can also be included within a Bayesian Filtering approach. A strategy for exploring an object's edges using tactile information and, with this data, identify the object from a large number of possibilities stored in a database and estimate its pose was done for underwater applications [90]. This approach, termed BRICPPF (Batch RANSAC, ICP, Particle Filter), combined a batch RANSAC (RANdom SAMple Consensus) to match detected object features with the objects in the database, a particle filter that continuously estimated the the 7 DOF (object identity and 6 DOF pose), and Iterative Closest Point (ICP) matching to locally minimise the distance between the spatial tactile data and the object point cloud.

Object exploration was also implemented within a strategy that sequentially planned and executed arm trajectories, deviating from the shortest trajectory in order to gain information on the object pose that placed the robot hand in a stable grasping posture. While executing this reaching motion, if an observation occurs that was not expected if the object was at its most likely pose, the belief state is updated, the robot hand is pulled back and a new trajectory is planned [133]. Framing the

problem as a POMDP (Partially Observable Markov Decision Process), Hsiao *et al.* [134] defined the state as the 3 DOF object pose and used a Bayesian Filter for estimation. Actions were defined as WRT (World Relative Trajectories), which represent robot poses in the object frame and can be used to grasp the object, explore it or re-orient it. Actions are selected according to their look-ahead cost of executing given the current belief state.

A framework for grasping was presented by Laaksonen [135] which aims to achieve a high probability for a stable grasp by marginalising over the possible object poses. In this work, two probability distributions are required: the probability of a stable grasp S , given a grasp G and an object pose O : $P(S|G, O)$ and the measurement model $P(O|G, T)$, where T is the tactile data. These distributions were obtained through the collection of grasping data and using Gaussian Process Regression (GPR). The robot continually attempts to grasp the object at the point of maximum probability of a stable grasp and updates its knowledge on the object pose until the probability of a stable grasp is above a threshold value, wherein the grasp is executed and the object is lifted.

Vazquez *et al.* [136] compared the performance of in-hand object pose estimation using different sensor types, contact sensors, torque sensors, distributed arrays and force sensors, and applied different techniques, such as particle filters and ICP.

Table 2.1 summarises the assumptions and results (when present) of the most relevant articles described in this section. The accuracy and speed of each method is detailed along with its most important features. Namely, the ability to: *a*) fully estimate the object pose in six dimensions; *b*) deal with objects composed of thousands of polygons; *c*) compute the pose without any prior estimate; *d*) can estimate the pose from a single measurement or requires exploration of the object through a series of touches *e*) enable the identification of the object *f*) take into account the object movement caused by the contacts.

None of the reviewed articles completely fulfills the stated abilities and, most importantly, approaches which can deal with complex objects require the implementation of some exploration strategy. This limitation raises issues due to the

fact that, in real grasping settings, the extent to which the robot can explore the object is limited by the dexterity of the robot and the possibility of dropping the object. Furthermore, the object displacement caused by these multiple simultaneous contacts becomes difficult to predict.

The accuracy and computation time for each study where it is stated is also compared, with results in the range of a few millimetres and durations in the range of seconds. An exception to this are the results by Honda *et al.* [124], which were taken in a very controlled environment, where a vision system provided a very accurate pose estimate to begin with ($\approx 1\text{mm}$).

Reference	6-DOF	Complex Objects	Global Pose	No Explor.	Object Ident.	Moving Object	Accuracy [mm/°]	Speed ^a [s]
Aggarwal[90]	✓	✓	✓	✗	✓	✓	<5/-	–
Chalon[131]	✓	✗	✗	✗	✗	✓	$\approx 10/10$	–
Corcoran[129]	✗	✗	✗	✗	✗	✗	$\approx 3/8$	3
Hebert[128]	✓	✗	✗	✓	✗	✓	6/2	–
Honda[124]	✓	✗	✗	✓	✗	✓	0.5/2	0.033
Petrovskaya[127]	✓	✗	✓	✗	✗	✓	5/3	30
Koval[132]	✗	✗	✓	✗	✗	✓	<20/-	1
Laaksonen[135]	✗	✗	✗	✗	✗	✗	–	–
Zito[133]	✓	✓	✗	✗	✗	✗	–	200
Pezzementi[137]	✓	✓	✓	✗	✓	✗	4/ ≈ 30	–

^aRun times are merely indicative, as the experiments were run in different hardware

TABLE 2.1: Comparison of existing tactile pose estimation methods in the literature

2.5 Conclusions

In this chapter, the topics of robot grasping, tactile sensing and object pose estimation were introduced and a review of the relevant literature was provided.

The knowledge of the location of the object within the robot hand is essential for the manipulation of objects, as it allows the computation of grasp stability and quality, as well as the planning of finger placements and forces to manoeuvre the object. Tactile sensing plays a very important role in manipulation tasks, not only for the control of finger forces but because it can also aid the perception of the hand-object system state. Given the different nature of the information provided by different tactile sensors, the strategies to process and interpret this information need to be adapted to the existing hardware.

Using tactile sensing to estimate the position and orientation of an object is still very much an open problem in the field of robot grasping and manipulation. Most robot systems currently available employ vision alone for the determination of the object pose, but this approach presents some drawbacks during manipulation tasks, where the robot body occludes the target object and the visible parts of the object are not sufficient to accurately estimate its location. Different techniques to estimate the object pose using tactile sensing have been proposed, with different advantages and also with some disadvantages. Early analytical methods cannot deal with objects with arbitrary geometric complexity, and their computational cost becomes impractical for objects composed of thousands of planes, as may be the case in household objects. Recently, most strategies aim to continuously estimate the object pose relying on Bayesian Filters such as Particle Filters. This approach has been proven to be very reliable in localisation problems, particularly in the context of mobile robotics, and has naturally been extended to the field of in-hand object pose estimation with promising results. However, and unlike in the field of mobile robots, this approach has not yet provided a definitive solution to this problem and, while theoretically sound, it has not shown the level of robustness that is suited for real world robot situations. The hindrances presented by using Bayesian Filtering depend greatly on the form in which the problem is stated, which researchers have tried to tackle using different formulations. As with analytical methods, object complexity can present difficulties to this approach, and render the problem untractable, and research has focussed on simple geometries. Another issue arises with the multi-dimensional nature of this

problem, which requires a great deal of computational power. Thus, some solutions present assumptions to reduce the pose to only three dimensions. Particle starvations and the fact that contact sensing is extremely discriminative (*i.e.* if a candidate pose places the object one millimetre away from the contact sensor there is no contact and thus the measurement model should report no contact) are also added difficulties for the use of Particle Filters when compared to mobile robots, which use range sensors and can always report a distance measurement. Bayesian Filters also require and are highly dependent of the existence of a transition (or motion) model. These models predict the evolution of a particular state given the current state and the action performed in a probabilistic manner. While these models can be computed for systems with wheels, or a single kinematic body, the nature of robot manipulation with multiple fingers, inaccurate friction models and objects of unknown properties makes it particularly complicated to predict the relative motion of the object with respect to the robot. As such, Bayesian Filters use simplified motion models which do not accurately predict the motion of the object or assume the object is immobile altogether, which is unrealistic for the use in real world scenarios.

A simpler solution, that attempts to estimate the pose of objects using tactile sensing in combination with vision and on its own, that can deal with objects of arbitrary complexity and use only a “frame” of tactile data (instead of continuous estimation) and thus do not rely on motion models, is the objective of this thesis. It aims to find object pose(s) that are coherent with current tactile information by stating the problem as the minimisation of this incoherence and using optimisation techniques that can rectify a coarse object pose estimate provided by a vision system or without any prior information.

Different strategies that attempt to solve the problem in these different scenarios, where vision may or may not be available, and in the presence of different contact sensing modalities are explored in the next chapters, after a brief overview of required mathematical concepts.

2.6 Mathematical Background

2.6.1 Rigid Body Motions

2.6.1.1 Coordinate Frames and Matrix Representation

In 3D Euclidean space, proper rigid motions are transformations $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ which preserve distance between points and handedness. The set of all such transformations forms the special Euclidean group $\mathbf{SE}(3)$, which can be represented in matrix form by the 4×4 *homogenous* coordinate transform matrices containing the rotation and translation parts of the transformation, as shown in Equation (2.9). These matrices represent the pose of a coordinate system with respect to another, where \mathbf{R} is the 3×3 rotation matrix and \mathbf{t} is the three-dimensional translation vector [138].

$$\mathbf{T} = \left[\begin{array}{ccc|c} & & & \\ & \mathbf{R}^{3 \times 3} & & \mathbf{t}^{3 \times 1} \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (2.9)$$

In this representation, a 1 is appended to the coordinates of a point and a zero to the coordinates of a vector, to obtain vectors in \mathbb{R}^4 . \mathbf{R} is an orthogonal matrix of determinant 1 and contains the unit vector coordinates of the target Cartesian frame as its columns. An important aspect of any orthogonal matrix is the fact that its inverse equals its transpose, which represents the inverse rotation.

For a thorough description of rigid body motions, the books by Craig [139] and Murray [140] provide very complete analyses on the topic. In this chapter, it suffices to state the properties of transformations and present the different representations for rotations. Figure 2.6 shows a system with two coordinate frames

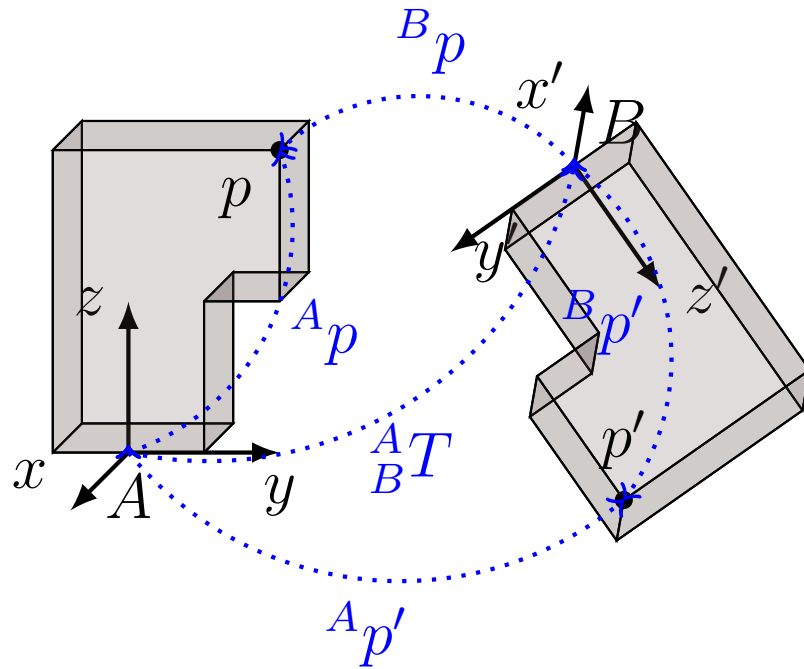


FIGURE 2.6: Rigid Transformation

with the same object represented in each. The same point p is shown for the two poses to highlight the following identities:

Let ${}^A_B\mathbf{T}$ be the transformation from frame A to frame B , represented by a homogeneous transform matrix and ${}^A p$ the coordinates of point p in frame A :

- The transformation from frame A to frame B is obtained through the inverse of ${}^A_B\mathbf{T}$

$${}^B_A\mathbf{T} = {}^A_B\mathbf{T}^{-1} = \left[\begin{array}{ccc|c} \mathbf{R}^T & & & -\mathbf{R}^T \vec{t} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (2.10)$$

- The coordinates of point p' in frame B coincide with the coordinates of ${}^A p$.

$${}^B p' \equiv {}^A p \quad (2.11)$$

- The coordinates of point p' in frame A are obtained through transformation ${}^A_B\mathbf{T}$ (left-multiplication) of point ${}^B p'$

$${}^A p' = {}^A_B \mathbf{T} {}^B p' \quad (2.12)$$

- The coordinates of point p in frame B are obtained through the transformation ${}^B_A\mathbf{T}$ (left-multiplication) of point ${}^A p$

$${}^B p = {}^B_A \mathbf{T}^{-1} {}^A p \quad (2.13)$$

- Transformations can be concatenated through matrix multiplication

$${}^A_D \mathbf{T} = {}^A_B \mathbf{T} {}^B_C \mathbf{T} {}^C_D \mathbf{T} \quad (2.14)$$

Rotation matrices contain 9 elements to represent the 3 DOF orientation of a coordinate frame with respect to another. Different formalisms exist to represent rotations in a more compact manner, with the most popular being Euler angles, axis-angle and quaternions.

2.6.1.2 Euler Angles

Euler angles describe rotations through three parameters (ϕ, θ, ψ) , sometimes referred to as *roll-pitch-yaw*, representing the rotation angle around each axis. In order to operate rotations using this convention, these parameters are used to construct three rotation matrices which are then used to perform the transformation. This construction is done as follows:

$$\mathbf{R}_x(\phi) = \text{Roll}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (2.15)$$

$$\mathbf{R}_y(\theta) = \text{Pitch}(\theta) = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.16)$$

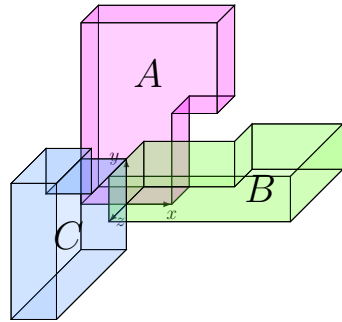
$$\mathbf{R}_z(\psi) = \text{Yaw}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.17)$$

While each rotation matrix is uniquely defined by its axis and angle, this notation carries a number of ambiguities. Since matrix multiplication is not a commutative operation, the order of rotations needs to be defined, as different rotation orders will result in different poses. Also, Euler angles can describe rotations with either extrinsic (fixed) or intrinsic (rotating) axes. This means that after one of the rotations has been applied, the following axis of rotation can be either from the initial or from this intermediate frame. When using intrinsic frames, the matrices should be multiplied in the rotation order to obtain the total transformation matrix. The equivalent rotation using extrinsic frames requires the order of rotations to be reversed. In other words, if using extrinsic frames and with the order of rotation $z-y-x$, or *yaw-pitch-roll*, the same rotation can be achieved with intrinsic frames if the order is reversed to $x-y-z$. In both cases, the resulting rotation matrix is the one in Equation (2.18).

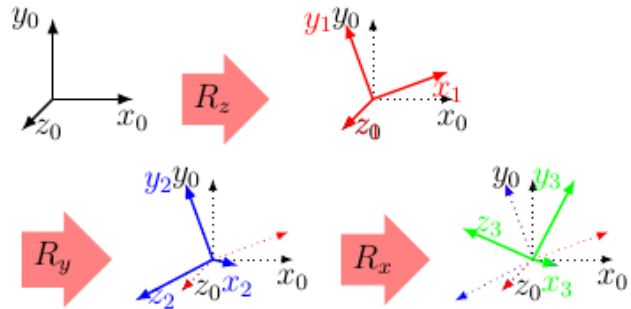
$$\mathbf{R} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \quad (2.18)$$

Figure 2.7(a) shows how transforming object A with angles $(\phi, \theta, \psi) = (90^\circ, 90^\circ, 0^\circ)$ can result in two different poses, if the rotation around x is performed first (case B), or last (case C). Figure 2.7(b) shows a sequence of rotations $z-y-x$ with intrinsic axes, where each rotation is performed using the rotated axes, *i.e.* the

second rotation is around y_1 , whereas if using extrinsic axis it would be around y_0 . A problem arising when using this representation is known as the “gimbal-lock”, which arises when the pitch $\theta = \pm \pi/2$. In this situation, the roll axis becomes parallel to the yaw axis, losing one degree of freedom and breaking the unique correspondence between a pose and a set of parameters (ϕ, θ, ψ) , which can lead to computational problems.



(a) Difference of rotation axes order



(b) Difference of fixed vs. rotating axes

FIGURE 2.7: Ambiguities in Euler Angle representation

2.6.1.3 Quaternions

Quaternions are a generalisation of complex numbers in 4 dimensions, which can be used to represent rotations. A quaternion is composed of a scalar, or real part $q_w \in \mathbb{R}$ and an imaginary, or vector part $\vec{q} = [q_x, q_y, q_z]$. Rotations can be represented by a quaternion of norm 1, where the quaternion $\mathbf{q}_I = [1, 0, 0, 0]$ is the identity rotation (a zero angle rotation).

Since every rotation or sequence of rotations can be represented as a single rotation of angle θ around an axis \vec{u} , quaternions can be interpreted as a rotation axis represented by its imaginary part and a rotation angle related to the real part, as shown in Figure 2.8. More precisely, $\vec{u} = \vec{q} / \sqrt{1 - q_w^2}$ and $\theta = 2 \cdot \cos^{-1}(q_w)$. The correspondence between the quaternion representation and its equivalent rotation matrix is shown in Equation (2.19).

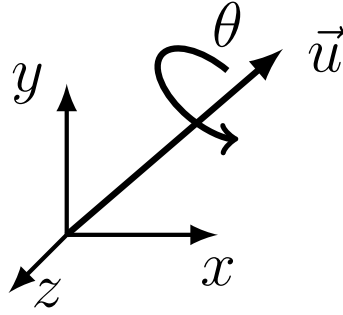


FIGURE 2.8: Interpretation of quaternions as axis-angle

$$\mathbf{T} = \begin{bmatrix} 1 - 2 \cdot (q_y^2 + q_z^2) & 2 \cdot (q_x \cdot q_y - q_z \cdot q_w) & 2 \cdot (q_x \cdot q_z + q_y \cdot q_w) \\ 2 \cdot (q_x \cdot q_y + q_z \cdot q_w) & 1 - 2 \cdot (q_x^2 + q_z^2) & 2 \cdot (q_y \cdot q_z - q_x \cdot q_w) \\ 2 \cdot (q_x \cdot q_z - q_y \cdot q_w) & 2 \cdot (q_y \cdot q_z + q_x \cdot q_w) & 1 - 2 \cdot (q_x^2 + q_y^2) \end{bmatrix} \quad (2.19)$$

Quaternions possess an algebra where the product is defined as the operation in (2.23). The operation of rotation defined by a quaternion and the inverse rotation are defined in (2.21) and (2.22), where \mathbf{q}^* denotes the conjugate operation, shown in (2.20). It can be seen that it not only requires fewer calculations when compared to rotating using a transformation matrix but also the computations involved to perform a rotation with quaternions are much easier for a computer to deal with, as there are no trigonometrical operations such as *sine* or *cosine*, improving the overall computational speed of a calculation.

The computational costs of using quaternions *vs.* homogenous matrices have been compared by Salamin [141] and Funda *et al.* [142], highlighting the advantages of quaternions for representing rotations, particularly when re-normalisation is often required. They also point out the more intuitive nature of quaternions to represent rotations according to the definition shown in Figure 2.8.

$$\mathbf{q}^* = [q_w, -q_x, -q_y, -q_z]^T \quad (2.20)$$

$$\vec{v}' = \mathbf{q} \begin{bmatrix} 0 \\ \vec{v} \end{bmatrix} \mathbf{q}^* \quad (2.21)$$

$$\vec{v} = \mathbf{q}^* \begin{bmatrix} 0 \\ \vec{v}' \end{bmatrix} \mathbf{q} \quad (2.22)$$

$$\mathbf{p} \cdot \mathbf{q} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w * q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w * q_z + p_x q_y - p_y q_x + p_z q_x \end{bmatrix} \quad (2.23)$$

While both Euler angles and quaternions present advantages and disadvantages, the quaternion representation was chosen in this thesis because, although it adds an extra variable (a quaternion consists of four parameters, while some other representations can be defined with only three), it presents advantages in terms of computational efficiency and suitability for optimisation methods. Ude proposed a method to improve iterative least squares optimisation when using unit quaternions [143]. This approach consists of mapping \mathbb{R}^4 quaternions into an S^3 sphere manifold and force the iteration steps to be taken along that manifold. A technical report by Wheeler [144] also discusses the application of quaternions in gradient-based searches, in particular for the problem of pose estimation, arriving also to the conclusion that it is more advantageous than other rotation representations. The possibility of reducing the rotation parameters to three by taking advantage of the fact that the quaternion must have a unit norm exists, and its application to nonlinear least squares problems has been investigated [145]. This parametrisation requires the initial computation of an \mathbb{R}^3 vector basis and provides the formulae to map these 3-vectors into unit quaternions.

2.6.2 Optimisation

2.6.2.1 Introduction

Optimisation is the branch of mathematics that deals with finding the best solution to a given problem. These problems are commonly stated in terms of finding the minimum (or maximum) of a function, or the arguments that minimise/maximise a given function, possibly subjected to a number of constraints. This function is typically called the “objective” function, or “cost” in case of a minimisation. Figure 2.9 shows an example of a function with two variables $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ stated in Equation (2.24).

$$f(x, y) = 0.2y^2 + 0.1 \sin(y + x^2) - e^{-(x+0.3)^2 - (y+0.1)^2} + 10 \quad (2.24)$$

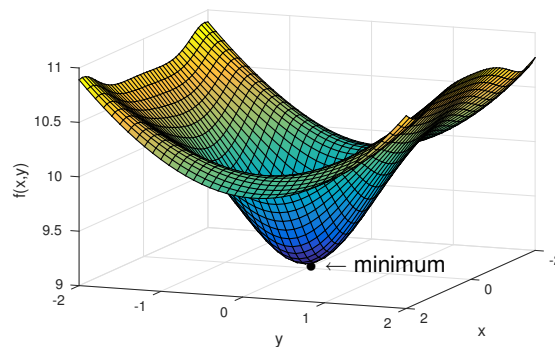


FIGURE 2.9: Function $f(x, y)$ and minimum point

The search for the (x, y) values at the minimum c , shown in Figure 2.9, is an optimisation problem, which can be stated as Equation (2.25), and has a solution at $c = (-0.2707, -0.1264)$.

$$c = \arg \min_{(x, y) \in [-2, 2]} f(x, y) \quad (2.25)$$

The subject of optimisation comprises a large part of the applied mathematics field, with usages in nearly every aspect of science and engineering. Depending on the nature of the problem different techniques can be employed, with sub-fields

including Linear, Stochastic, Quadratic, Integer, etc. Different optimisation approaches are presented in this section which are relevant to the following chapters. These approaches can be either local or global, deterministic or stochastic.

2.6.2.2 Gradient-Based

Gradient-based optimisation methods are iterative algorithms which, starting from an initial point in the search space, take steps in the direction of the negative of the gradient evaluated at the current point, until a local minimum is found. This approach is highly dependent on the starting point, as it will converge to the first local minimum it encounters.

Figure 2.10 shows an example of gradient descent on a scalar function. The function and its derivative are evaluated at x_1 and the next estimate x_2 is calculated according to the update rule in Equation (2.26). where λ is a learning rate that controls the size of each step. Care should be taken in choosing λ so that it is not too high, as it may “skip” over the minimum value, or too low as it will converge too slowly to the minimum. Green lines represent the slope of the line tangential to the function at point x_n .

$$x_{n+1} = x_n - \lambda \nabla f(x_n) \quad (2.26)$$

When dealing with objective/cost functions which are vector functions of several variables ($\mathbf{G} : \mathbb{R}^a \mapsto \mathbb{R}^b, a > 1, b > 1$), the equivalent to the gradient of a scalar function in this type of functions is the Jacobian Matrix. The Jacobian matrix contains the derivatives of each element of $\mathbf{G}(\mathbf{x})$ with respect to each variable. In other words, the Jacobian is the matrix where the rows are the gradients of each element of the vector function.

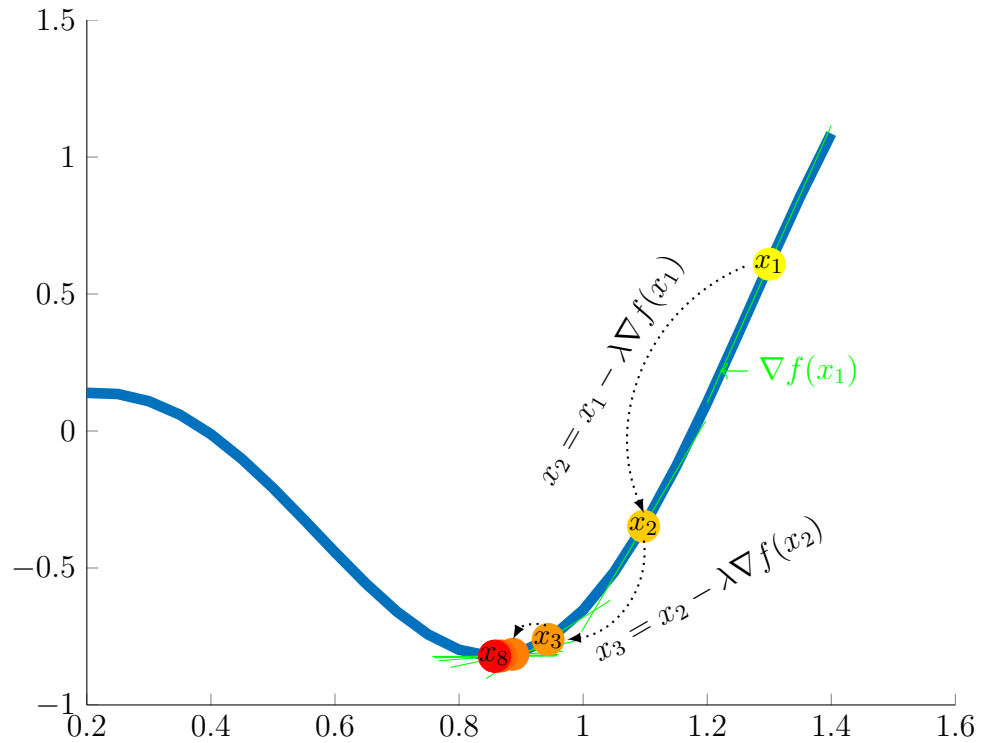


FIGURE 2.10: Gradient Descent on a scalar function

$$\mathbf{J}_g(\mathbf{x}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}) & \frac{\partial g_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial g_1}{\partial x_a}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_b}{\partial x_1}(\mathbf{x}) & \frac{\partial g_b}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial g_b}{\partial x_a}(\mathbf{x}) \end{bmatrix} \quad (2.27)$$

However, in some cases, the objective function is not differentiable and an approximate Jacobian needs to be calculated at every step. This can be done using the forward difference derivative [146], where the j -th column is calculated following (2.28), where h is a small number and \hat{e}_j is the unit vector in the direction of the j -th dimension.

$$\tilde{\mathbf{J}}_{\mathbf{G}}(\mathbf{x}) = (\nabla_h \mathbf{G})(\mathbf{x})_j = \frac{\mathbf{G}(\mathbf{x} + h\hat{e}_j) - \mathbf{G}(\mathbf{x})}{h} \quad (2.28)$$

Gradient Descent

The simplest example of gradient-based optimisation is Gradient Descent, which uses the update rule given in (2.29). At each iteration, the approximate Jacobian Matrix is calculated using the formula in (2.28) and a step is taken in the direction and proportion of the negative of the gradient $\mathbf{J}_{\mathbf{G}}(\mathbf{x}^{(i)})^T \mathbf{G}(\mathbf{x}^{(i)})$.

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \lambda(\mathbf{J}_{\mathbf{G}}(\mathbf{x}^{(i)})^T \mathbf{G}(\mathbf{x}^{(i)})) \quad (2.29)$$

Two conditions are evaluated to determine the convergence of the algorithm:

1. a solution is found: $\max(\mathbf{G}(\mathbf{x}^{(i)})) < \varepsilon_g$
2. a maximum number of iterations is reached: $i > \max_i$

The step size λ can be defined experimentally and will make the convergence faster or more accurate.

Levenberg-Marquardt

This method was developed in 1944 by Kenneth Levenberg and improved by Donald Marquardt in 1963 [55] and combines the advantages of the Gradient Descent and the Gauss-Newton methods, interpolating between the behaviour of these two methods. If, at the current iteration, the parameters are far from an optimal value, this method acts in a similar way to Gradient Descent. As it approaches a solution, its behaviour becomes closer to the Gauss-Newton method. The update rule for the Levenberg-Marquardt method, is shown in (2.30).

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - (\mathbf{J}^T \mathbf{J} + \lambda \cdot \text{diag}[\mathbf{J}^T \mathbf{J}])^+ (\mathbf{J}^T \mathbf{G}(\mathbf{x}^{(i)})) \quad (2.30)$$

This update rule is slightly modified from the standard Levenberg-Marquardt method that is used to solve least-square problems. Firstly, the Jacobian is approximated using forward differences due to the inability to differentiate the objective

function. Also, the approximate Hessian matrix calculated as $\mathbf{J}^T \mathbf{J}$ can become singular and, as such, not invertible. The alternative is to use the Moore-Penrose [147, 148] pseudo-inverse \mathbf{A}^+ , calculated using Singular Value Decomposition as shown in (2.31). Σ^+ is a diagonal matrix obtained by replacing each of the non-zero elements in the diagonal by its multiplicative inverse, leaving the zeros unchanged and \mathbf{U}^* is the conjugate transpose of \mathbf{U}

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^* \implies \mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^* \quad (2.31)$$

The learning parameter λ is set dynamically at each iteration following the rule: if the error decreases from the previous iteration, λ is reduced by a factor of ten, otherwise, if the error increases, the step is rejected and λ increased by the same factor. The stopping criteria to determine the convergence of the algorithm are the same as for Gradient Descent detailed in Section 2.6.2.2, with the additional criterion where the optimisation is stopped if the learning rate λ exceeds a large number: 10^{100} .

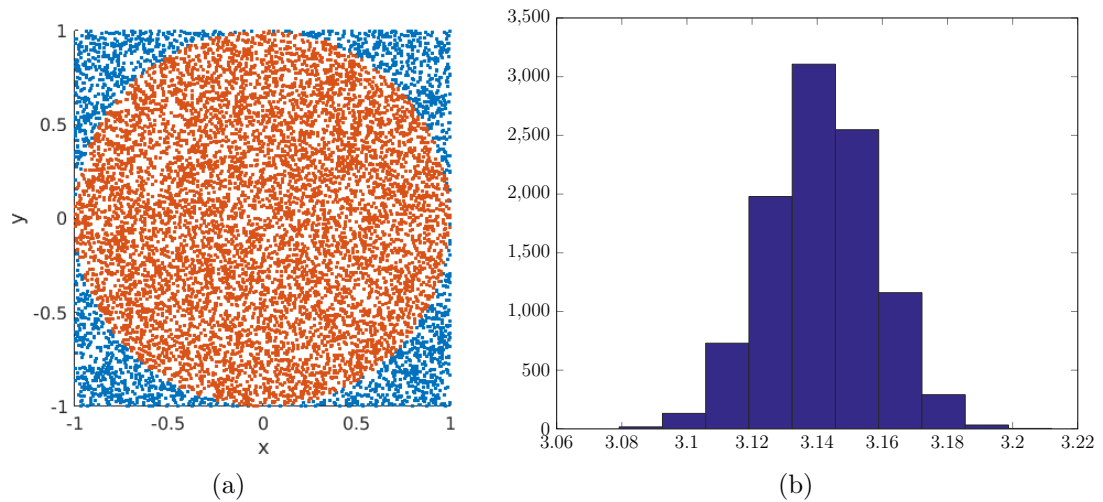
2.6.2.3 Stochastic

While the gradient-based methods previously presented are local (they converge to local minima), and deterministic (using the same parameters will always result in the same outcome), another class of methods exist which stochastically tries to find the minimum (or maximum) of a given function. Stochastic methods use random variables and heuristics to determine a function's global minimum, which is often difficult due to the existence of multiple local minima.

A simple method to increase the chances of finding the global minimum consists of using one of the previously mentioned gradient-based methods with multiple, random initial points. Although this approach is not guaranteed to find the global minimum, it can greatly increase the odds of finding it, depending on the number of searches and the distribution of initial points within the search space, presenting the added possibility of being done in parallel.

A class of algorithms known as Monte Carlo methods can be used to approximate a distribution, find the global minimum of a function or to extract underlying properties of a mathematical function. Monte Carlo methods rely on randomly drawing samples from a distribution, which can be combined with heuristics to determine the search direction in order to move towards the solution. This class of algorithms was first developed by Metropolis and Ulam [149] in the 1940's, and applied in mathematical physics problems, but have seen applications ranging from the simulation of stochastic processes, approximating the expected value of a probabilistic event and optimisation, where the objective is to estimate a set of parameters that minimise a devised objective function. This latter application has seen extensive application within the field of robotics, with notable examples of the usage of this class of algorithms in localisation problems in mobile robotics [2, 126, 150] and reinforcement learning, where optimal policies were found using Monte Carlo methods [151, 152]. A simple example that illustrates well this class of algorithms is its usage for the calculation of the constant π . A circle is inscribed within a square of known size and points lying inside the square are picked randomly. Counting the number of points that lie inside and outside the circle gives a ratio r that can be used to calculate π with very high accuracy. Figure 2.11(a) displays an example of one thousand 2-D points p sampled from a uniform distribution such that $p = (x, y), x, y \in [-1, 1]$. Points within the circle of radius 1 are colored in red and are chosen such that $\sqrt{x^2 + y^2} < 1$. The ratio between the points in red, which satisfy this radius condition, and the total points approximates the ratio between the area of the square $A_s = 1$ and the area of the circle $A_c = \pi \cdot r^2$. In the case plotted in Figure 2.11(a), the result of calculating π is $0.786 = \tilde{\pi} \cdot 0.5^2 \Leftrightarrow \tilde{\pi} = 3.144$. Performing this operation ten thousand times results in the histogram shown in Figure 2.11(b), with the average approximating the value of π .

The forerunning work of the application of Monte Carlo methods was presented by Metropolis *et al.* [153], where the equations of state describing the position of a set of particles that minimised the total energy of the system, according to thermodynamical laws. In this work, which was later generalized by Hastings [154],

FIGURE 2.11: Estimating the value of π through Monte Carlo simulation

an initial arrangement is chosen and the total energy is calculated. A new, random state is selected according to a proposal distribution and if the energy in this new state is lower this step is accepted and the process is repeated. If this new state has a higher energy, the transition to this new state is done with a probability related with the difference in energy the two states.

One notable adaptation of this formulation for the problem of finding a global minimum or maximum is Simulated Annealing (SA) [155]. This method makes an analogy to the process of increasing and decreasing the temperature of a metal, where this heating corresponds to an increased possibility of accepting steps to states that are worse than the current state. As the temperature becomes lower, the algorithm starts progressively rejecting more transitions to “worse” states.

Within Monte Carlo methods, a subclass of algorithms take inspiration from Darwin’s theory of natural selection and are referred to as Evolutionary or Genetic algorithms [156]. This approach starts by randomly generating a set of candidate solutions and uses the concept of the “survival of the fittest”, where fitness relates with the value of the function in that candidate and whether the problem is a maximisation or minimisation. Candidates with higher fitness will reproduce and generate a new set of candidate solutions – the *offspring*. These new solutions will inherit the features of the candidates which originated them, sometimes termed *chromosomes*, with modifications generated by different genetic operations

which are also based in biological processes. Typically, these modifications can be *crossovers*, where the offspring will inherit chromosomes from multiple parents, and have a set of parameters which is a combination of the chromosomes of its parents. Another operation is termed *mutations* and correspond to slight modifications of the parameters which occur randomly, with a given probability. The candidates with lower fitness will have a high probability of perishing without generating any offspring. As the algorithm progresses, candidates with high fitness will survive and reproduce, and generate new candidates with higher fitness, increasing the probability of converging to the function's global maximum.

2.6.2.4 Other Methods

Another class of methods exist which are deterministic and are able to find the global maximum without the need of calculating derivatives. One such method is DIRECT [157], which divides the search space into rectangles, and performs both local and global searches simultaneously and requires virtually no tuning of parameters.

Besides the methods approached in this section, other algorithms exist, such as particle swarm, which also belong to the class of Monte Carlo methods, interior point, Nelder-Mead, or Branch-and-Bound methods. Since these methods are not important for the understanding of this thesis, they are left out of this chapter.

2.6.3 *k*-d Trees

2.6.3.1 Definition

k-d trees are data structures designed to partition the space, organising *k*-dimensional points to enable quick range and nearest neighbour searches. This technique was introduced by Bentley *et al.* in the 1970's [158] and it consists of creating a binary tree, where each tree node represents a point. Every non-leaf node in the tree represents also a plane which splits the space in two parts. As one traverses the

tree, each layer relates to one of the dimensions of the points, cycling between them. Queries using this data structure can be efficiently performed through the immediate pruning of large areas of the tree.

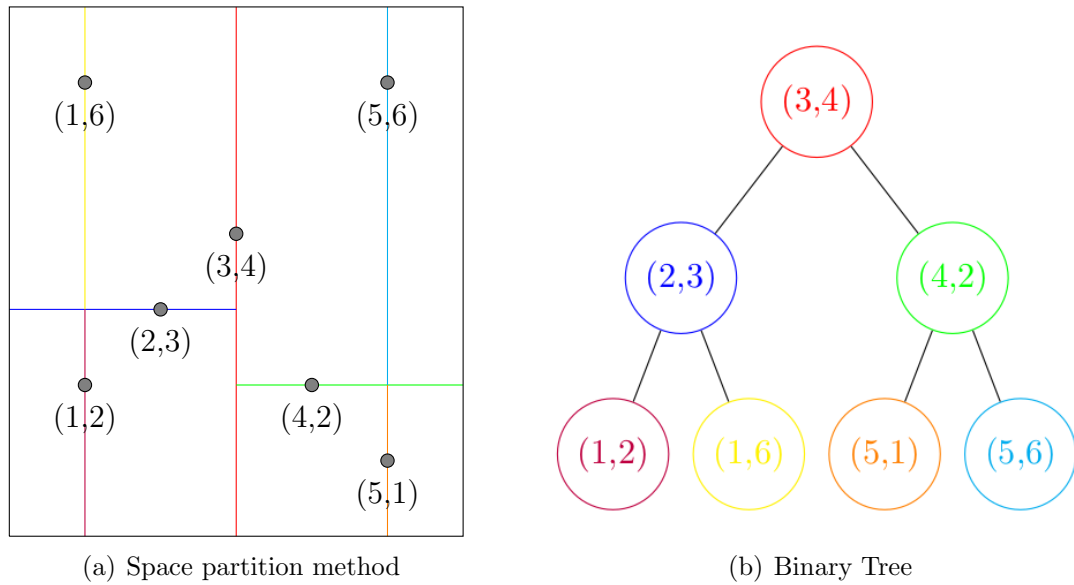
2.6.3.2 Construction

A k -d tree is built recursively, sequentially calculating the median of a group of points. Figure 2.12(a) shows the method to construct a two-dimensional tree, and the resulting tree is shown in Figure 2.12(b).

The root node is selected as the median point according to an arbitrary chosen dimension. Commonly, when the k -d tree represents points in space, the first dimension is x . After choosing this root node, a line/plane/hyperplane along the chosen dimension passing through the node separates all the child points in two sides. Points with the chosen dimension larger than the root node sit on the right side of the root with the others sitting on the left side of the root. In the example of Figure 2.12, the median point along the x dimension (point $(3, 4)$) is chosen as the root node, splitting the space along that dimension with the red line. The median of the points to the left of the red line along the y dimension is chosen (point $(2, 3)$) and added as a left child of the root. This sub-space is again divided along this median (blue line). The same is done to the points on the right of the red line, and this process is repeated recursively until all the points are included in the tree.

2.6.3.3 Searches

k -d trees are particularly suited for range and nearest neighbour searches. Range searches are queries that search for the points within a rectangular region, *i.e.* find the set of points $P_r = \{(x, y) | x_1 < x < x_2, y_1 < y < y_2\}$. Given a desired range, one can obtain all the points in a k -d tree that lie within that region using the following heuristics:

FIGURE 2.12: k -d tree data structure principle

1. Start at root node;
2. If the desired range rectangle lies on the left/below of the splitting line/-plane/hyperplane search only left child;
3. *idem* if range lies on the right/above side;
4. If the divider intersects the range rectangle search both children nodes.

Nearest neighbour queries can be used to either find the nearest point to query point p or find the n points closest to p . To find the nearest point to a query point, a similar algorithm to range searches is followed:

1. Start at root node, record current smallest distance;
2. If it is possible that a closer point is located within a region, keep this region as searchable.
3. Direct the search towards the query point, *i.e.* if query point is to the left/below of divider, proceed to the left child.
4. As the current closest point is updated, more regions can be pruned.

For the situation in Figure 2.12, for an example query point $p_n(0.5, 3.1)$, the search begins at point $(3, 4)$ and the smallest distance $d = 2.6571$ is saved. Since there are points on both sides of this divider that can be nearer to point p_n , both regions are kept, but the search continues on the left direction. Point $(2, 3)$ is at a distance of 1.5033, becoming the new nearest point, with both sides of this divider possibly containing nearer points. Next, given that $3.1 > 3$, the point to the right $(1, 6)$ is searched, with the distance being larger than the current minimum. However, points below the blue line passing through $(2, 3)$ can still be closer to p_n , so it is evaluated. The new minimum distance is then 1.2083, with no other region possibly containing nearer points. Thus, point $(1, 2)$ is the nearest neighbour of point p_n .

2.6.3.4 Computational Remarks

Creating a k -d tree is a $O(n \log(n))$ operation, with insertion, deletion being on average a $O(\log(n))$, with the worst case being $O(n)$. For a balanced tree, range searches are typically $O(R + \log(n))$, with the worst case at $O(R + \sqrt{n})$, where R is the number of points inside the range. Nearest neighbour searches are usually performed in $O(\log n)$, with the worst case being $O(n)$ [159]. Multiple implementations exist for the construction of k -d trees and the fast computation of nearest neighbours. In some situations, it is acceptable to obtain a neighbour that is not guaranteed to be the absolute closest point to the query. In these cases, Approximate Nearest Neighbor such as FLANN [115, 160] can be used, greatly improving the computation time and memory requirements of the search. Nearest neighbour searches using this k -d tree implementation are reported to achieve a speed-up of 1000-fold compared to a linear search in a database of 100 thousand elements [161].

2.6.4 Principal Component Analysis

2.6.4.1 Definition

Principal Component Analysis (PCA) is a statistical technique that operates in a set of multidimensional data, finding the directions of highest variance which are orthogonal between them. PCA takes a set of n -dimensional data that might be correlated and finds up to n principal components, each one being an n -dimension vector. This technique was first introduced by Karl Pearson in 1901 [162] and has seen extensive application in many fields of science and data analysis.

An important feature of PCA is that the relative magnitudes among the principal components translate into the percentage of variance that is “explained” by each principal component. Thus, when dealing with some multidimensional problems, selecting only a small subset of the larger principal components can enable the problem to become much more tractable while still accounting for the majority of variance in the original data [26]

Figure 2.13 shows an example of the three principal components in a 3 dimensional distribution, represented by the red, green and blue arrow. Most of the variance (72%) is present in the direction of the red arrow and a further 21% is explained by the second largest principal component, shown in green.

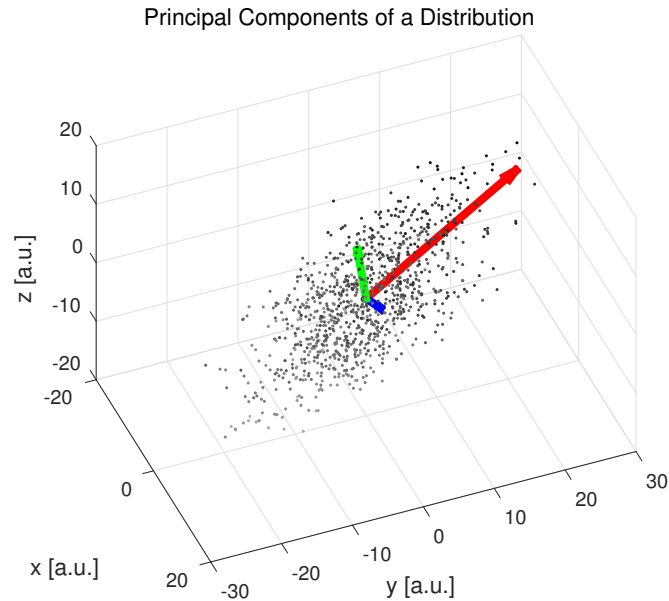


FIGURE 2.13: Principal Components of a Distribution. 1st, 2nd and 3rd components in red, green and blue respectively

Besides dimensionality reduction, PCA can be a powerful tool for representing data in a compact and useful way, which is also easily computed.

2.6.4.2 Computing the Principal Components

The principal components can be obtained through the computation of the eigenvectors and eigenvalues of the covariance matrix of a set of data. Given a set \mathbf{p} of m data points, each point being n -dimensional, the covariance matrix is obtained by first subtracting the mean from the original data as in Equation (2.32), and then through the operation in Equation (2.33).

$$\mathbf{M} = \left[(\mathbf{p}_1 - \bar{\mathbf{p}}_1) \quad , \quad (\mathbf{p}_2 - \bar{\mathbf{p}}_2) \quad , \quad \dots \quad , \quad (\mathbf{p}_n - \bar{\mathbf{p}}_n) \right]^T \quad (2.32)$$

$$\mathbf{C}_M = \frac{1}{m-1} \mathbf{M} \mathbf{M}^T \quad (2.33)$$

Since covariance matrices are, by definition, symmetric (*i.e.* $C_{i,j} = C_{j,i}$) it is possible to decompose the matrix using the method described in the LAPACK users'

guide [163] for Symmetric Eigenproblems, shown in Equation (2.34). The symmetric nature of the covariance matrix results in its eigenvectors being orthogonal to each others, as is required in Principal Component Analysis.

$$\mathbf{C}_M = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T \quad (2.34)$$

The columns of matrix \mathbf{E} are the unit norm eigenvectors of the covariance matrix \mathbf{C}_M , presented in (2.35)

$$\mathbf{E} = \begin{bmatrix} \vec{e}_1 & , & \vec{e}_2 & , & \vec{e}_3 \end{bmatrix} \quad (2.35)$$

The eigenvalues of \mathbf{C}_M are the elements of the diagonal matrix $\mathbf{\Lambda}$. To obtain the Principal Components, the eigenvectors are sorted and scaled according to their respective eigenvalue. The relative values of each eigenvalue translates into the percentage of variance that is “explained” by the direction of its eigenvector.

Chapter 3

Pose Correction using Local Optimisation

Chapter Summary

This chapter frames the problem presented in chapter 1: estimating an object's pose (position and orientation) using tactile sensing, as a local optimisation problem. It presents and compares two gradient-based optimisation methods to correct a coarse pose estimate obtained from vision. The importance of rich contact information to increase the accuracy of the results is also investigated.

3.1 Introduction

The goal of this chapter can be summarised as finding an object pose that is coherent with the current tactile data. This is achieved through the minimisation of a defined cost function, given the current contact data obtained from an *intrinsic* tactile sensing scheme. This sensing approach, previously presented in Section 2.2.1, is leveraged to obtain rich contact information, including contact normal. The addition of this normal information is investigated and the results compared with a minimisation that takes into account only the contact locations.

Starting from a coarse pose obtained from vision, the objective is to find a transformation that improves the pose estimate given current tactile data, as shown in Figure 3.1. This pose consists of a rotation quaternion and a translation vector, as shown in Equation (3.1).

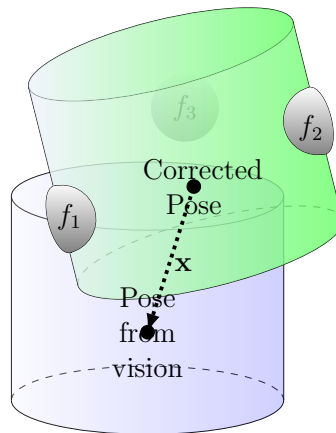


FIGURE 3.1: Problem overview – Finding a transformation \mathbf{x} that displaces the object from an initial estimate

$$\mathbf{x} = [\mathbf{q}, \vec{t}]^T \quad (3.1)$$

$$\mathbf{x} = [q_w, q_x, q_y, q_z, t_x, t_y, t_z]^T$$

The problem of finding a set of parameters \mathbf{x} that minimises a given objective function (also known as cost function), is typically referred to as an optimisation problem. As presented in Section 2.6.2, a prominent class of such optimisation

techniques are gradient-based methods. In this chapter, this class of optimisation algorithms was used to find the parameters that satisfy the current tactile measurements.

Given that an object's geometrical model usually contains thousands of points, applying a transformation on the object would require all these points to be transformed into a new pose. In order to facilitate the computation and reduce the time to evaluate the cost function, the transformation defined by \mathbf{x} is a transformation on the contacts to match the initial pose. When the result is obtained, this transformation is inverted and applied to the object.

To evaluate the performance of the methods, simulation experiments were carried out in MATLAB [164] to accurately test the methods against a ground truth and draw comparisons between the different approaches. In this chapter, two optimisation algorithms were implemented, tested and compared. Besides, a study was carried out on the advantages of adding surface normal information to the cost function.

The two gradient-based methods presented in Section 2.6.2 were implemented: Gradient Descent (also known as Steepest Descent), and Levenberg-Marquardt. These two algorithms were tested and their performance was compared in terms of accuracy and speed of convergence. The initial point that is provided to the algorithm is an initial coarse estimate of the parameters that, in this particular case of estimating a grasped object's pose, is typically obtained from a vision system.

3.2 Methods

3.2.1 Algorithm

The first step of the method consists of describing the object and all the contact locations into a common reference frame. Next, regions in the object surface are

selected as possible contact locations on the object. Each region consists of the surface points within a neighbourhood of each contact at the object's current pose. This step aims at reducing the computation time of the algorithm by assuming that the contact locations on the object at its real pose are within a distance ε_d of the fingertip. In other words, it is assumed that the initial estimate from vision will provide an approximate initial pose and, as such, it is not useful to test poses where a finger is touching the farthestmost parts of the object when at its initial pose estimate.

Thus, this initial procedure creates regions in the object where each finger is predicted to lie, so that the algorithm only iterates over these regions instead of going through the whole object. This distance is dynamically set so that each finger iterates over a minimum number of points in the object pointcloud O . For example, finger 1 will iterate over the set $S^{(1)}$, which contains all the points $s_1^{(1)}, s_2^{(1)} \dots s_n^{(1)}$ with distances to the finger contact location $f^{(1)}$ within the neighbourhood ε_d .

$$S^{(k)} = \{s_i^{(k)} \in O : \|s_i^{(k)} - f^{(k)}\| \leq \varepsilon_d\}, \quad k \in \{1 \dots m\} \quad (3.2)$$

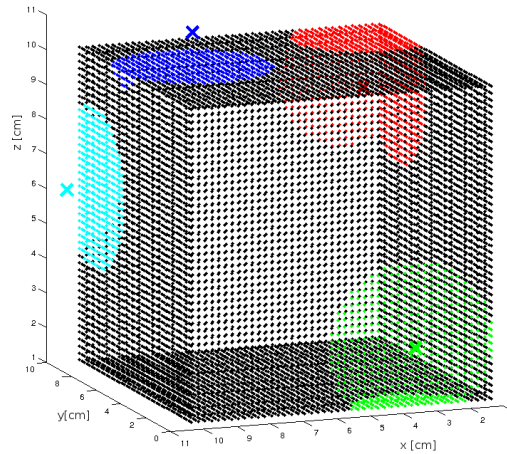


FIGURE 3.2: Regions created in the object point cloud to minimise the computational effort

This step is described by equation (3.2) and shown in Fig. 3.2. In this case, there are four fingers touching the object ($m = 4$) and the neighbourhood ε_d is set to 15 mm.

Figure 3.2 shows a simulated cubic object point cloud O plotted in black with four fingers contacting the object surface. Each contact location is plotted as a cross, and the selected neighbourhood regions $S^{(m)}$ in their respective colour. It can be seen, that in this pose, the contacts do not sit on the object surface. This is particularly noticeable on the blue contact, where there is a significant distance between the contact and the object surface.

3.2.2 Distance-based optimisation

3.2.2.1 Objective Function

The simplest approach is to find a pose which minimises the distance between the object surface and the contact locations. As such, the problem is formulated as an optimisation problem where the objective is to find a set of parameters \mathbf{x} that define a transformation that minimises an objective function $\mathbf{G}(\mathbf{x})$.

As explained in Section 3.1, the parameters \mathbf{x} describe a transformation on the contact locations. The objective function $\mathbf{G}(\mathbf{x})$ to be minimised is then the set of squared distances from each contact location to the nearest point $s_i^{(m)}$ in its respective region $S^{(m)}$, as described in equation (3.3). The operation $\mathbf{q}f^{(2)}\mathbf{q}^*$ is the rotation operation when using quaternions, described in Section 2.6.1.3.

$$\mathbf{G}(\mathbf{x}) = \begin{cases} \min_i (\|(\mathbf{q}f^{(1)}\mathbf{q}^* + \vec{t}) - s_i^{(1)}\|^2) \\ \min_i (\|(\mathbf{q}f^{(2)}\mathbf{q}^* + \vec{t}) - s_i^{(2)}\|^2) \\ \dots \\ \min_i (\|(\mathbf{q}f^{(m)}\mathbf{q}^* + \vec{t}) - s_i^{(m)}\|^2) \end{cases} \quad (3.3)$$

3.2.2.2 Simulation Results

In order to effectively compare the two optimisation algorithms, artificial data was used, so that the object models were “ideal” and the “ground truth” on the real contact locations was accurately known. Four points selected from the object’s surface were displaced using an arbitrary rotation and translation, which was used as the initial estimate for the pose correction. The resultant transformation should then be the inverse transformation from this arbitrary displacement.

Two objects, one cubic and one cylindrical, were tested and the results are shown in Figures 3.3 to 3.6, where the ground truth contact locations are plotted as a filled circle (\bullet), the displaced locations which serve as the initial estimate for the algorithm (mimicking what a vision system might output) are represented by a cross (\times) and the final position of the optimisation is represented by a ring (\circ).

Gradient Descent

The results from applying gradient descent are pictured in Figure 3.3. It can be seen that the algorithm converges to a pose that approximately matches the object surface, although the final contact locations are not the initial ones. This is even clearer if the object is revolute, such as a cylinder, where there may be infinite solutions that minimise the distance to the object, as can be seen in Figure 3.3(b).

The progress of the algorithm for the cube-shaped object is shown in 3.4, for the first 500 iterations, showing the convergence of the algorithm. The final distance from contact locations to the object surface in this case was around 46% of the initial distance.

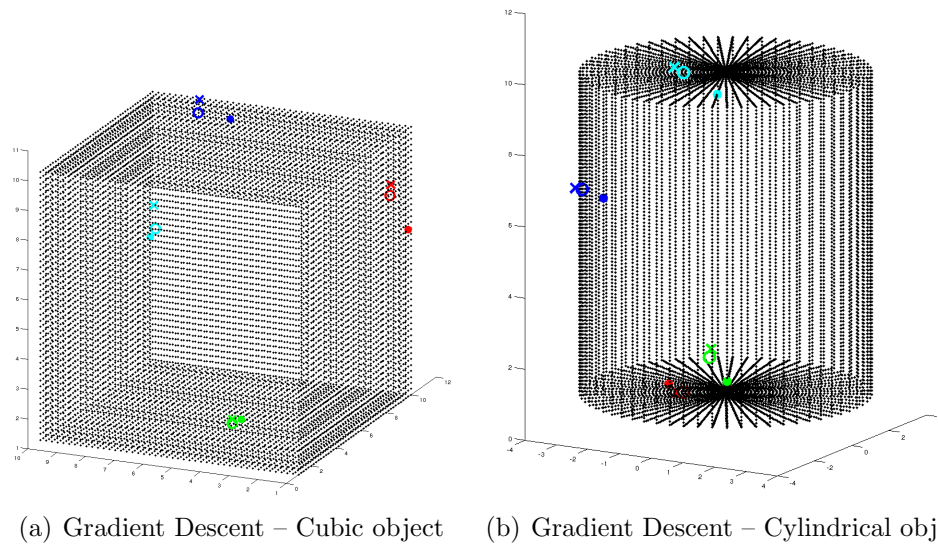


FIGURE 3.3: Results of Gradient Descent using simulated data – Object point cloud in black. Original (\bullet), displaced (\times) and corrected (\circ) finger tip locations.

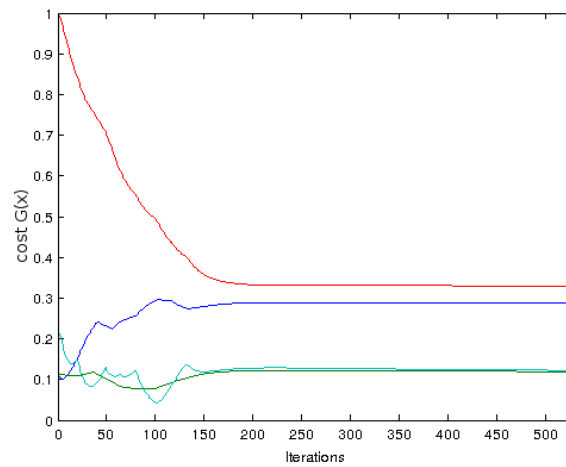
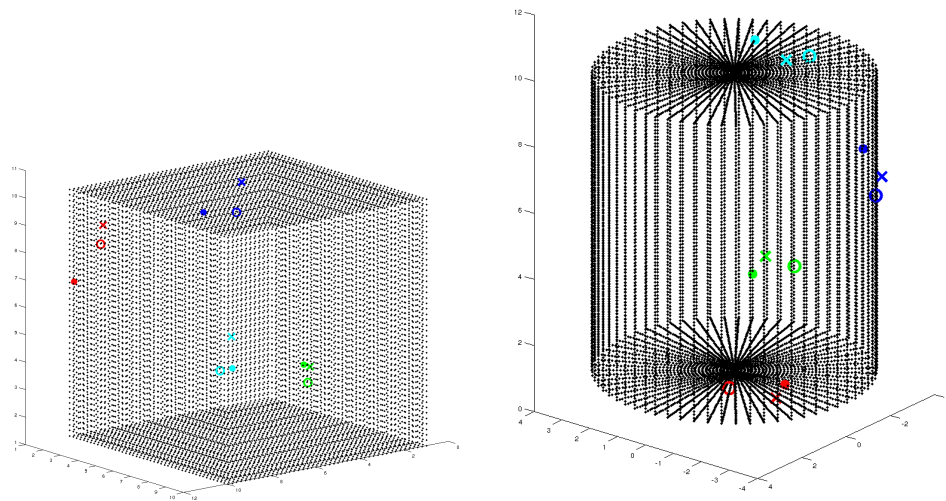


FIGURE 3.4: Progress of the algorithm in a cubic object using Gradient Descent. Each color represents the distance between object at the estimated pose for a finger contact.

Levenberg-Marquardt

The same input was used to evaluate the Levenberg-Marquardt algorithm, with the results being plotted in 3.5. The accuracy of the results using this method is superior to those produced by Gradient Descent.

The progress of the algorithm is shown in 3.6, where it can be seen that the algorithm successfully converged to a solution. The error was reduced by more than



(a) Levenberg-Marquardt – Cubic object (b) Levenberg-Marquardt – Cylindrical object

FIGURE 3.5: Results of Levenberg-Marquardt using simulated data – Object point cloud in black. Original (\bullet), displaced (\times) and corrected (\circ) finger tip locations.

90%, requiring also less iterations than the previously described Gradient Descent. These results demonstrate what was intuitively expected: that the Levenberg-Marquardt outperformed Gradient Descent in both accuracy and speed.

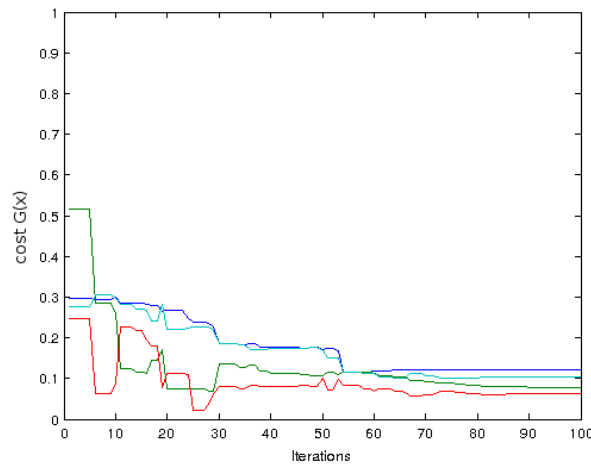


FIGURE 3.6: Progress of the algorithm for a cylindrical shaped object using Levenberg-Marquardt. Each color represents the distance between object at the estimated pose for a finger contact.

3.2.3 Addition of Normal Force Information

3.2.3.1 Contact Normal

The results presented in the previous section show that the estimation of the pose of a grasped object can be improved by minimising the distance between the contact locations, obtained through forward kinematics and contact sensing. However, when attempting to determine the correct pose of an object, other contact information can be useful. Understanding which facet of the object is being touched by the robot finger is of the utmost importance when grasping an object, as it has fundamental implications on grasp stability, as previously shown in Section 2.1. Simply minimising distance between the contact location and the object's surface can give an incorrect result if, for example, the contact is close to a vertex or an edge of the object. An incorrect estimation in this case may lead to wrong assumptions on the stability of the grasp and result in the failure of a grasping or manipulation task. When minimising distance only, this aspect is overlooked and there is the possibility that the resulting pose estimate, however accurate in terms of distance, can yield an inadequate pose. In Figure 3.5(a), the contact plotted in cyan is an example of that situation, where the resulting contact location is near the correct point but not lying in the same facet of the cube. Assessing grasp stability under these two situations (real and estimated), would render very different results.

This problem can be overcome by making use of the rich contact information provided by intrinsic tactile sensing, which by estimating the contact locations, is able to also determine the normal and tangential components of the interaction force. Having this normal force direction information and taking advantage of the fact that, if the contact is assumed to be approximately rigid, the normal force direction coincides with the normal direction of both the object surface and the fingertip, one can estimate the pose of the object by trying to match the contact locations with the object surface and also the surface normal with the normal force direction.

Figure 3.7 shows a diagram where an object is touching the sensor’s hull, generating force and moment. Through the intrinsic tactile sensing scheme presented in Section 2.2.1, the contact location p_c and the normal force are calculated. The direction of the normal force \hat{n} is also the perpendicular direction to the sensor hull’s surface and to the object surface at that contact point. This direction is commonly referred to as the contact normal.

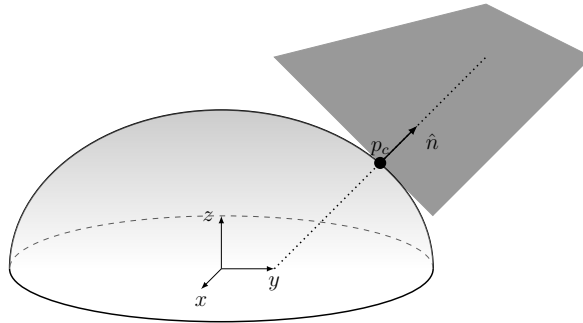


FIGURE 3.7: Rigid contact

The object’s geometrical model is described by a polygonal mesh, consisting of a list of vertices and triangles. The computation of the surface normal of each triangle of the object’s polygonal mesh is done through the cross product of the vectors defined by the triangle vertices, as shown in Figure 3.8 and Equation (3.4).

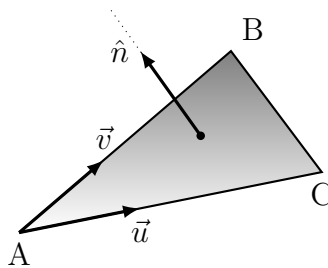


FIGURE 3.8: Mesh triangle normal

$$\hat{n} = \frac{\vec{u} \times \vec{v}}{\|\vec{u} \times \vec{v}\|} \quad (3.4)$$

Thus, we can reformulate the problem to try to find a pose that satisfies both these measurements – contact location and normal direction – to obtain not only a more accurate estimate of the object’s pose but also an estimate that is more meaningful in terms of assessing grasp stability.

3.2.3.2 Objective function

The objective function can then be reformulated to make use of the normal information. This desired function should converge to solutions that parametrise a transform such that not only the distance from the contact locations to the object surface, but also the angle between the measured normal force direction on the finger tip sensor and the direction perpendicular to the object's surface at that point are minimised.

This objective function (3.5) was defined for each contact as the sum of a component related with the distance between finger contact locations and the object's surface plus a component related with the angle between the sensed normal direction and the object's surface normal direction at that contact point.

$$\mathbf{G}(\mathbf{x}) = \begin{cases} \min(\|(\mathbf{q}^{f(1)}\mathbf{q}^* + \vec{t}) - s_i^{(1)}\|^2 + w_n|1 - \langle \mathbf{q}\hat{u}^{(1)}\mathbf{q}^*, \hat{n}_i \rangle|) \\ \min(\|(\mathbf{q}^{f(2)}\mathbf{q}^* + \vec{t}) - s_i^{(2)}\|^2 + w_n|1 - \langle \mathbf{q}\hat{u}^{(2)}\mathbf{q}^*, \hat{n}_i \rangle|) \\ \dots \\ \min(\|(\mathbf{q}^{f(m)}\mathbf{q}^* + \vec{t}) - s_i^{(m)}\|^2 + w_n|1 - \langle \mathbf{q}\hat{u}^{(m)}\mathbf{q}^*, \hat{n}_i \rangle|) \end{cases} \quad (3.5)$$

The first component is, as before, the minimum squared distance $\|\cdot\|^2$ between the contact locations f transformed by the rotation and translation parameters \mathbf{q} and \vec{t} and an object point $s_i^{(k)}$, belonging to the set $S^{(k)}$, where $k \in \{1 \dots m\}$. The component related with the angle is calculated using the inner product $\langle \cdot, \cdot \rangle$ of the contact normal force unit vector \hat{u} rotated by \mathbf{q} and the surface normal at point $s_i^{(m)}$, denoted \hat{n}_i . The symbol $|a|$ denotes the absolute value. This angle component is bounded between 0 and 2, and approaches zero as the angle becomes smaller.

These two components are mediated by a weighting factor w_n , which is related to the prior information of the object model. w_n is tuned according the requirements of the real system and how accurate we know the object model to be. By giving

a large value to w_n , the algorithm will try to adjust the orientation of the object to fit the normals more than it will try to minimise the distance.

3.2.3.3 Simulation Results

The same approach was taken to test the performance of the algorithm using the new objective function that takes into account distance and normal force information. Figure 3.9 shows the results of the optimisation. It can be seen in Figure 3.9(a) that the algorithm converges to a pose where the estimated contacts accurately sit on the true contact locations and that the contact plotted in cyan sits at the correct facet of the cube, even if the initial pose was closer to an incorrect facet. The cylindrical object in Figure 3.9(b) shows a higher distance between contacts which is caused by the revolute nature of the object. In this situation, infinite solutions are equally correct and the method converged to one of these solutions.

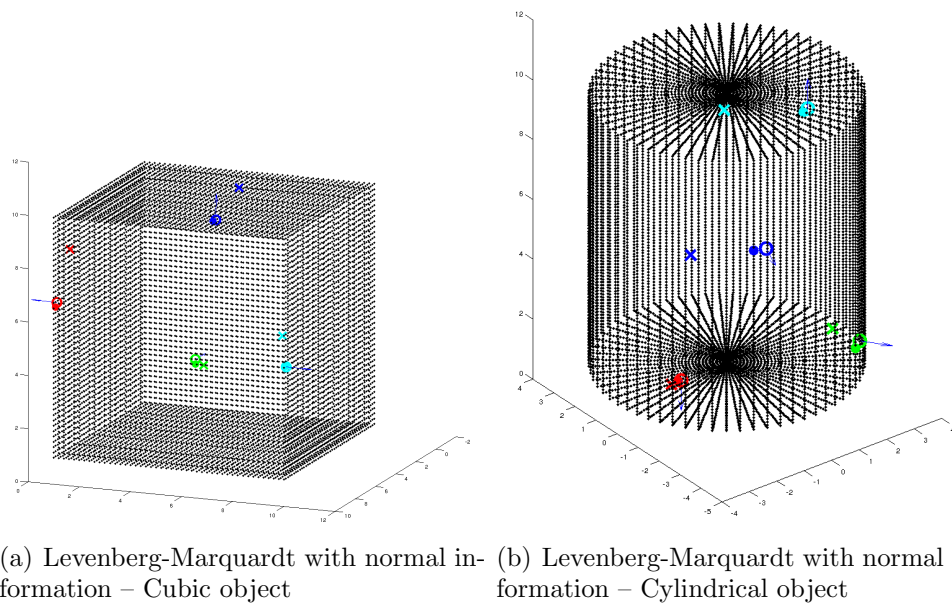


FIGURE 3.9: Results of Levenberg-Marquardt with normal force information – Object point cloud in black. Original (\bullet), displaced (\times) and corrected (\circ) finger tip locations.

The progress of the algorithm for the cylindrical object situation is shown in Figure 3.10. It should be noted that the objective function has changed from the previous

experiments and the cost plotted in the vertical axis is not comparable to the costs shown in Section 3.2.2.2.

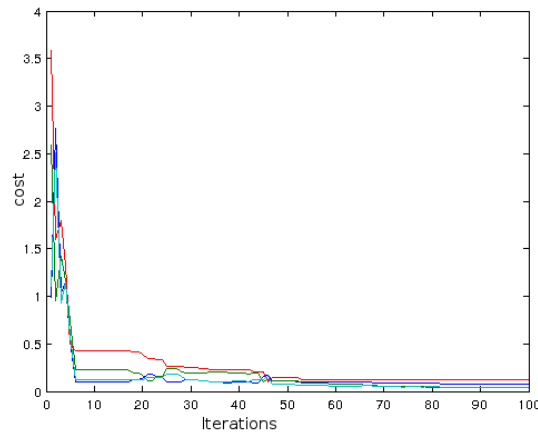


FIGURE 3.10: Progress of the algorithm for a cylindrical shaped object using Levenberg-Marquardt. Each color represents the distance between object at the estimated pose for a finger contact

These results show significant improvement when compared to the sole minimisation of distance. The final output of the algorithm should be a transformation applied on the object that matches the contact locations given by the kinematics of the robot and the tactile sensors. Thus, the resulting transformation that was obtained by transforming the contacts to match the object needs to be inverted and applied to the object. Figure 3.11, shows the final result of the algorithm. The grey point cloud shows the initial pose of the object, the green point cloud represents the ground truth and the yellow point cloud shows the transformed pose using the inverse of the resulting parameters from the optimisation. The resulting pose of the object coincides almost perfectly with the ground truth.

3.3 Results

3.3.1 Analysis of Simulation Results

The previous section shows that Levenberg-Marquardt outperforms Gradient Descent and that the addition of contact normal information greatly improves the accuracy of the result. In order to validate this statement, the algorithms were

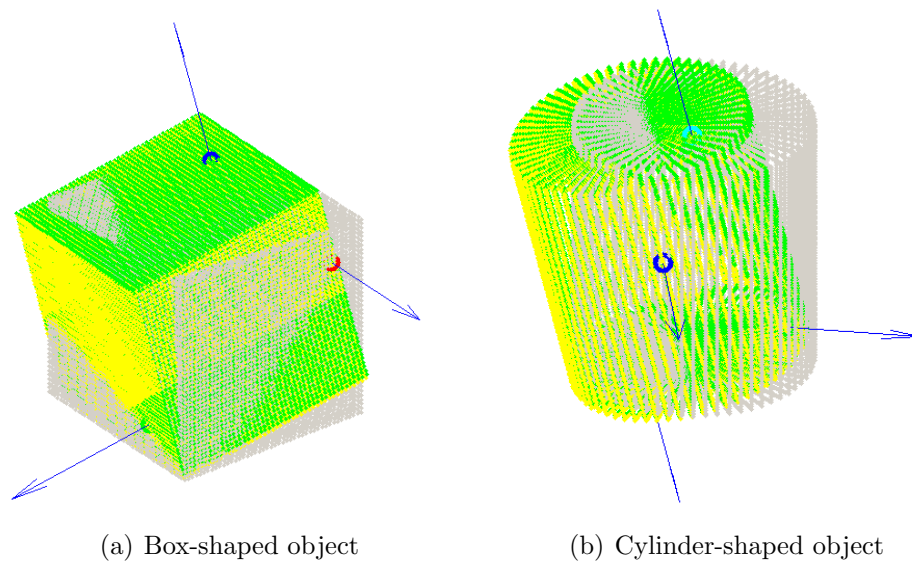


FIGURE 3.11: Simulation results – green represents the ground truth, gray the initial misplaced pose and yellow the resulting object pose.

run 100 times with different weights w_n attributed to the normals information. $w_n = 0$ corresponds to the situation where only distance is minimised.

Different criteria were used to compare results, since there will typically exist multiple poses which correctly describe the pose of the object. This is mostly due to the symmetric nature of the objects. To extensively characterise the performance of the algorithms, the chosen criteria belong to two classes: the convergence of the algorithm (poses that match contact information) and correctness with respect to the ground truth.

Table 3.1 summarises the obtained results for each iterative method for similar accuracies.

- **MDTS** stands for mean distance to surface and is the average distance between the contact locations and the object's surface.
- **MATN** stands for mean angle to normal and it is the angle between the measured contact normal and the surface normal at the resultant contact location.
- **RME** is the real mean distance error – the average distance from the resulting contact locations to the selected initial points.

- **MAE** is the mean angle error, which is the average angle between the surface normal at the ground truth contact locations and the resultant angle at the measured locations.

While the first two criteria characterise the convergence of the optimisation algorithm, the latter two concern the difference between the found solution and the ground truth. In a real situation, we do not know which point on the object the robot is touching, so the algorithm tries to find a transform where each finger is matching a point in the region chosen in equation (3.2). In this case of simulated data, the points were selected *a priori*, so this ground truth is known. This difference is fundamental for the understanding of this problem, as large values of **RME** and **MAE** do not necessarily mean that the result is wrong. This is particularly clear in the case of the cylindrical object, where the algorithm might minimise the distance better but the result may be further away from the “real” solution – the initial chosen points. This has to do with the fact that a cylinder is a revolute object and will likely have infinite solutions that minimise the objective function similarly. This is an inherent feature of this problem and, although it can not be solved, it should not compromise the ability of a robot to find better grasping points. From Table 3.1 we can also see that, when compared to the Gradient Descent (GD), the Levenberg-Marquardt (LM) method is able to achieve better results in fewer iterations and shorter time. It can also be seen from the same table that the inclusion of the information on the normals reduces the real error significantly, without significantly increasing the computation time. Since the cost function is changed when adding the information on the surface normals, the desired accuracy was changed accordingly for those experiments.

In order to evaluate the merits of this approach, one can also compare the initial *versus* the final error. This way, it is possible to evaluate whether the proposed method consistently improved the estimate of the object pose. Figures 3.12 to 3.19 compare initial and final error for both algorithms for the box-shaped object using all four evaluation criteria. Blue dots show an improvement on the initial estimate and the red crosses depict trials where the final pose estimate has larger error than

TABLE 3.1: Comparison of optimisation methods

Method	w_n	Shape	Its.	MDTS	MATN	RME	MAE	Speed(s)
Initial	–	Cube	–	0.696	16.715°	1.534	9.015°	–
		Cylinder	–	0.265	8.515°	0.772	9.148°	–
Gradient Descent	0	Cube	198.3	0.166	10.177°	0.589	4.802°	6.791
		Cylinder	200	0.148	6.181°	0.574	7.070°	7.506
	20	Cube	139.5	0.200	3.706°	0.399	2.832°	4.553
		Cylinder	156.2	0.110	3.228°	0.426	4.995°	4.877
Initial	–	Cube	–	0.672	16.820°	1.465	8.867°	–
		Cylinder	–	0.259	8.555°	0.759	9.062°	–
Levenberg- Marquardt	0	Cube	145.9	0.103	11.519°	0.765	6.847°	5.00
		Cylinder	97.0	0.072	7.984°	0.633	7.481°	4.336
	20	Cube	94.2	0.113	4.619°	0.572	4.638°	3.54
		Cylinder	18.4	0.083	2.670°	0.397	4.925°	0.813

the initial pose. It can be seen that in the vast majority of the cases ($> 95\%$), the method improves the initial pose estimate regardless of the criterion chosen, when the initial error is below 15° and 1.4 centimetres.

Both algorithms successfully minimise the distance to the object and consistently improve an initial pose estimate. As expected, Levenberg-Marquardt performs faster and achieves a better minimisation than Gradient Descent. Another aspect that becomes clear from Figures 3.12 to 3.19 is that the quality of the result is highly dependent on the initial estimate. This is expected when using gradient based optimisation, where the solution is usually the first local minimum found starting from the initial estimate.

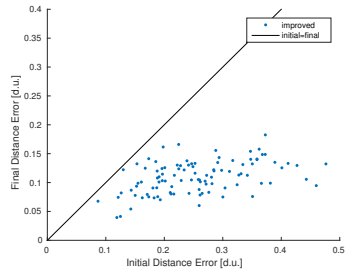


FIGURE 3.12: Initial vs Final MDTs using Gradient Descent

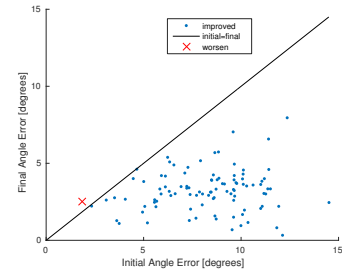


FIGURE 3.13: Initial vs Final MATN using Gradient Descent

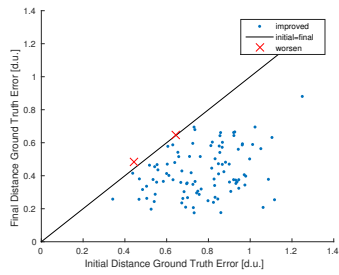


FIGURE 3.14: Initial vs Final RME using Gradient Descent

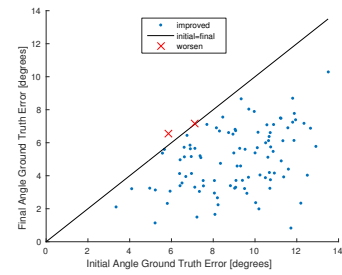


FIGURE 3.15: Initial vs Final MAE using Gradient Descent

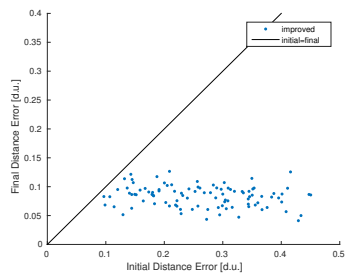


FIGURE 3.16: Initial vs Final MDTs using Levenberg-Marquardt

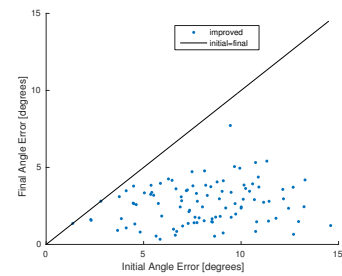


FIGURE 3.17: Initial vs Final MATN using Levenberg-Marquardt

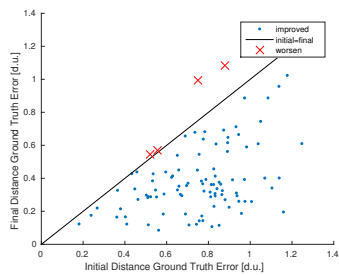


FIGURE 3.18: Initial vs Final RME using Levenberg-Marquardt

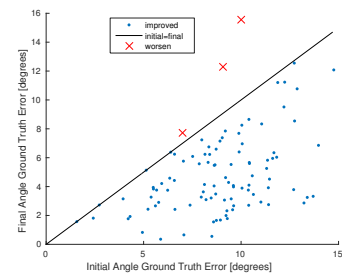


FIGURE 3.19: Initial vs Final MAE using Levenberg-Marquardt

Figures 3.12 to 3.19: Results for box-shaped object.

3.3.2 Results on a Real System

3.3.2.1 System Overview

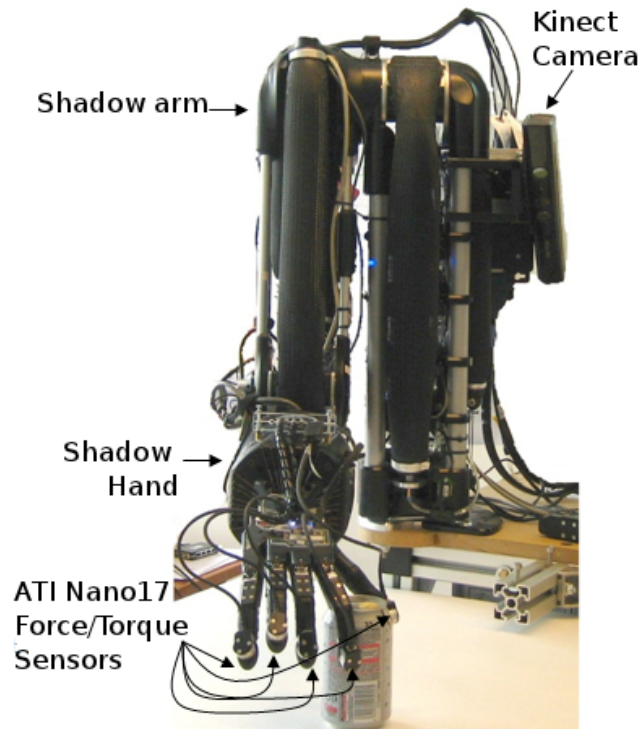


FIGURE 3.20: Overview of the experimental setup of the multi-modal sensing system

The proposed method was implemented in a real system using C++, under the ROS platform [165]. The system is shown in Figure 3.20 and comprised:

- A 4 Degree of Freedom (DoF) Shadow robot arm.
- A 24-DoF Shadow hand with ATI Nano17 6-axis force/torque sensors instrumented on the fingertips.
- A Microsoft Kinect camera mounted on the left side of the robotic arm and oriented to get a top view of the objects lying on a table.

The tactile sensing strategy used the approach described in Section 2.2.1, providing contact location, normal and tangential components of the contact force and the

torque around the contact normal. The details of the tactile sensing strategy were presented by Liu *et al.* [59]. The chosen common reference frame was not an external fixed frame but the palm frame. This allowed that the object pose could be tracked during a stable grasp, as the object will stay stationary with respect to the palm, even if the robot arm is moving the object around the environment.

The object point cloud can be obtained either from a database, containing accurate mesh representations of the geometry of a number of known objects or online, using the object reconstruction method presented by Burrus and Rodriguez-Jimenez [111, 166]. This visual tracking tool provided the initial estimate of the object's location. A model of the robot was used to obtain the contact locations, through forward kinematics. This information was fed into the pose estimation algorithm, summarised in Algorithm 3.1, which used the Levenberg-Marquardt minimisation algorithm, and the results are presented in this section.

Algorithm 3.1 Pose correction

Input: Object point cloud and number of fingers touching the object ≥ 2 .

for all fingers in hand **do**

if finger is in contact **then**

 Transform contact point ($f^{(m)}$) and contact normal $\hat{u}^{(m)}$ to palm coordinate frame

end if

end for

for all $f^{(m)}$ **do**

for all points p_i in object **do**

$j \leftarrow 0$

while $j < 50$ **do** ▷ If there are at least 50 points in neighbourhood ε_d

if $\|p_i - f^{(m)}\| \leq \varepsilon_d$ **then**

$s_j^{(m)} = p_i$

$j \leftarrow j + 1$

end if

end while

end for

end for

Minimise $\mathbf{G}(\mathbf{x})$ in Equation (3.5) using the update rule in Equation (2.30).

if minimisation is successful **then**

 Invert transformation defined by x_i

 Apply transformation to object

end if

Given the difficulty to benchmark the results against an accurate ground truth, the results were evaluated qualitatively or using indirect methods. Two methods to benchmark the results were used, both proving to be unable to successfully obtain accurate ground truth values. Fiducial markers did not provide the degree of accuracy required to correctly evaluate the algorithm. The usage of magnetic trackers was also explored, but their accuracy dropped significantly when in close distance to magnetic materials, particularly because of the presence of magnets in the Hall-effect sensors used in the joint encoders of the Shadow hand.

3.3.2.2 Using Distance Information

Figure 3.21 shows the result of a minimisation that used distance only. The grey object shows the initial pose estimate of the object, as obtained from the vision system. The geometry of the object was obtained online, and as such, is not a perfect model of the real object. It can be seen that this pose is displaced from the fingertips which were touching the object, with a mean distance between the contact locations and the object surface of 4.9 cm. The algorithm was run and converged to a pose that matched the tactile sensing data, with a mean distance between contact locations and object surface of 5 mm. Regarding computational performance, the algorithm took 350 ms to converge to this solution.



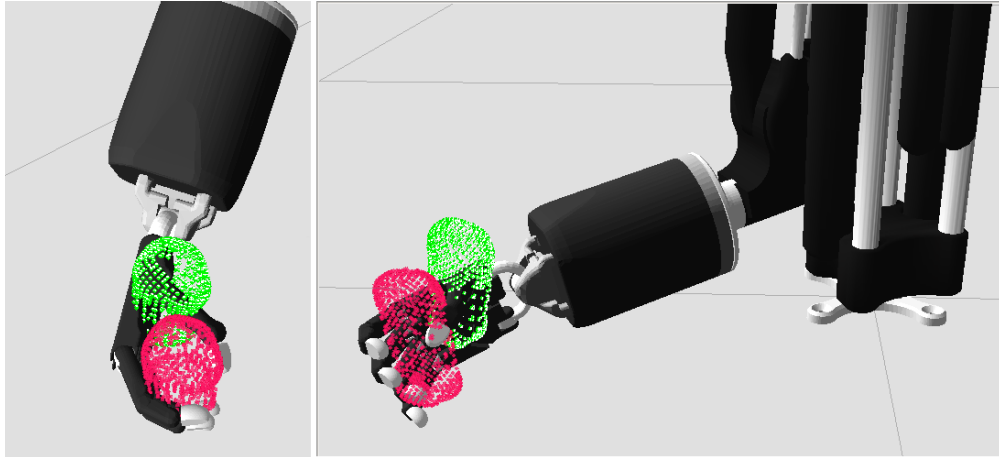
FIGURE 3.21: Results using real data. Initial estimate in grey, solution in pink

3.3.2.3 Using Distance and Normal Information

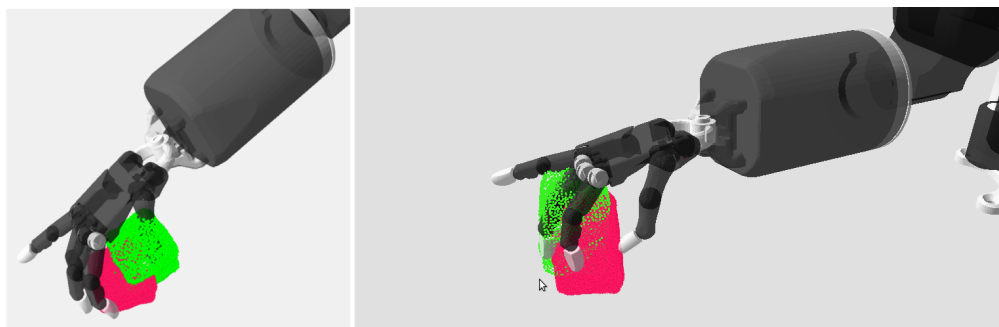
Figure 3.22 shows the results of pose correction using both distance and normals information for different objects and grasps. The robot model shows the robot at its current posture and the point clouds show the object at the location estimated by vision and after the pose correction using tactile information. The green point clouds show the object at the location estimated by vision. This point cloud is clearly away from the contact locations and is the result of the vision not being able to segment the object from the robot body. Pink point clouds show the object at the pose estimated by the method detailed in Section 3.2.3, using the Levenberg-Marquardt optimization algorithm. Qualitatively, it is clear that the accuracy of the estimate of the object's pose is greatly increased by using this method, and that the object sits correctly within the robot hand, coherently matching the tactile information.

Due to the mentioned difficulty in having accurate and continuous ground truth, validation was done manually using millimetre paper. The procedure consisted of attaching a transparency to the object base, with squares marked on it, as shown in 3.23. The object's base centre coincided with the point p_0 , at the intersection of the four squares drawn on the transparency. By moving the object close to the millimeter paper and measuring the locations of square corner points p_1 and p_2 , the actual location and orientation of the object with respect to the robot base can be determined, using Equation (3.6).

$$p_0 = p_2 + \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (p_1 - p_2) \quad (3.6)$$

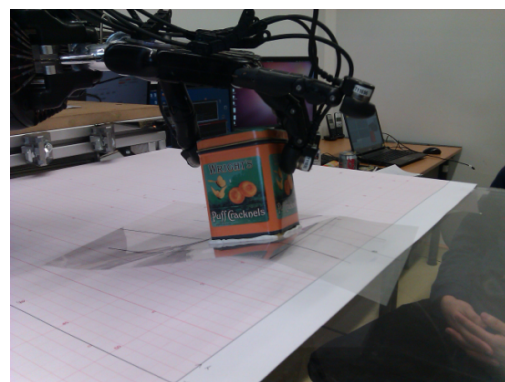
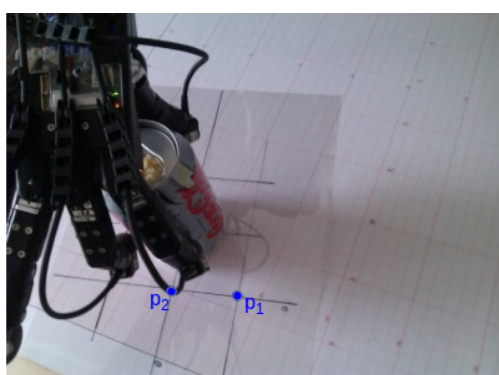


(a) Large diameter grasp on a soda can



(b) Tripod grasp on a cuboidal tea box

FIGURE 3.22: Visualisation of a grasped object scene. The green point cloud represents the object in the pose detected by the vision system and the pink point cloud represents the object after its pose has been corrected using our approach

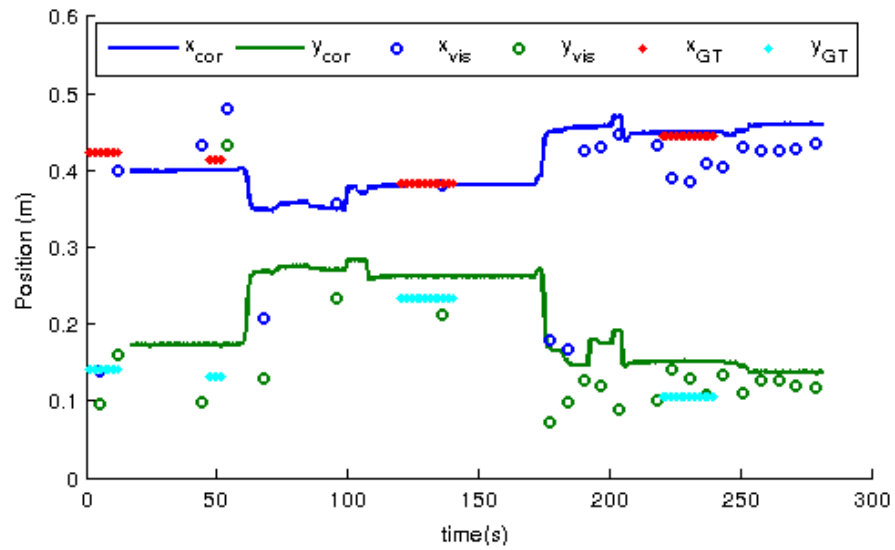


(a) Soda can glued to marked transparency (b) Box-shaped object (Tea box) being grasped

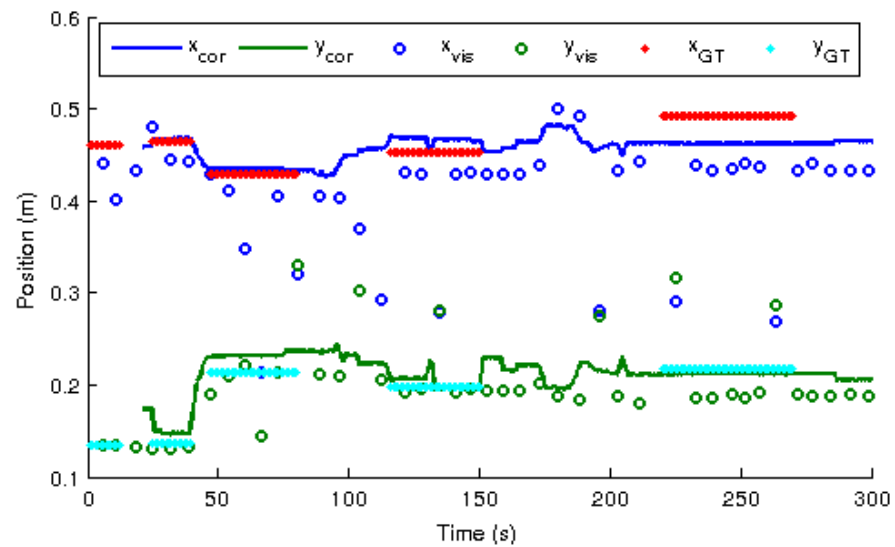
FIGURE 3.23: Ground truth measurement method

The object was grasped and moved between four locations, with the vision tracker continuously estimating the pose of the object and the proposed pose correction method correcting that estimate. Figures 3.24(a) and 3.24(b) show the x and y position of the centre of the object's base according to the different estimates. Blue and green denote the x and y dimensions respectively, with the ring shapes plotting the estimates from vision and the lines plotting the corrected object's base position after the pose estimation. Ground truth at the recorded locations are plotted as dots in red and cyan for x and y . The fact that the corrected estimate is continuous arises from the fact that the object pose is expressed in the palm coordinate frame and the assumption that the grasp is stable and, as such, there is no relative movement between the object and the hand.

The mean distance error was taken at the points that the ground truth was known, and was reduced from 8.58 cm and 8.02 cm to 2.66 cm and 2.0 cm for the can and the tea box respectively. This accuracy criterion corresponds to **RME** used in Section 3.3.1. The running time of the algorithm was, on average, 0.171 seconds, with an average 91.2 iterations.



(a) Results for the cylindrical can



(b) Results for the cuboid box.

FIGURE 3.24: Experimental results: blue and green represent the components x and y . Rings plot the pose obtained by vision and lines the pose estimated by the proposed method. Red and cyan dots are recorded ground truth

3.4 Discussion

During a manipulation task, the tracking of the grasped object cannot rely solely on 3D vision. Occlusions created by the robot hand and fingers preclude the accurate estimation of the object's pose.

This chapter presented a method that, given an approximate estimate of the object's position and orientation, use contact sensing to correct this estimate through a gradient-based optimisation. These methods start from the coarse pose obtained by 3D vision and aim to find a pose that minimises a cost function that depends on the matching between contact information and the object geometric information.

This chapter demonstrated that using the *intrinsic* tactile sensing scheme, one can use normal force information to increase the accuracy of the pose estimate. Besides, the Levenberg-Marquardt algorithm was shown to perform better than a simple gradient descent in terms of accuracy and speed.

However, these local optimisation methods present an important drawback, since they are heavily dependent on the starting point – in this case the initial estimate provided by vision. If the initial estimate is too far off the object's real pose or if the object's geometry presents a high degree of complexity, the algorithm will converge to a local minimum, failing to find the object's most likely pose, which would be the cost function's global minimum. Figures 3.25 and 3.26 show an evaluation of initial vs. final error, starting from a large range of initial errors. It can be seen that both Gradient Descent and Levenberg-Marquardt fail to converge to an accurate estimate when the initial error is large.

In order to overcome this problem, a global search method is presented in the next chapter.

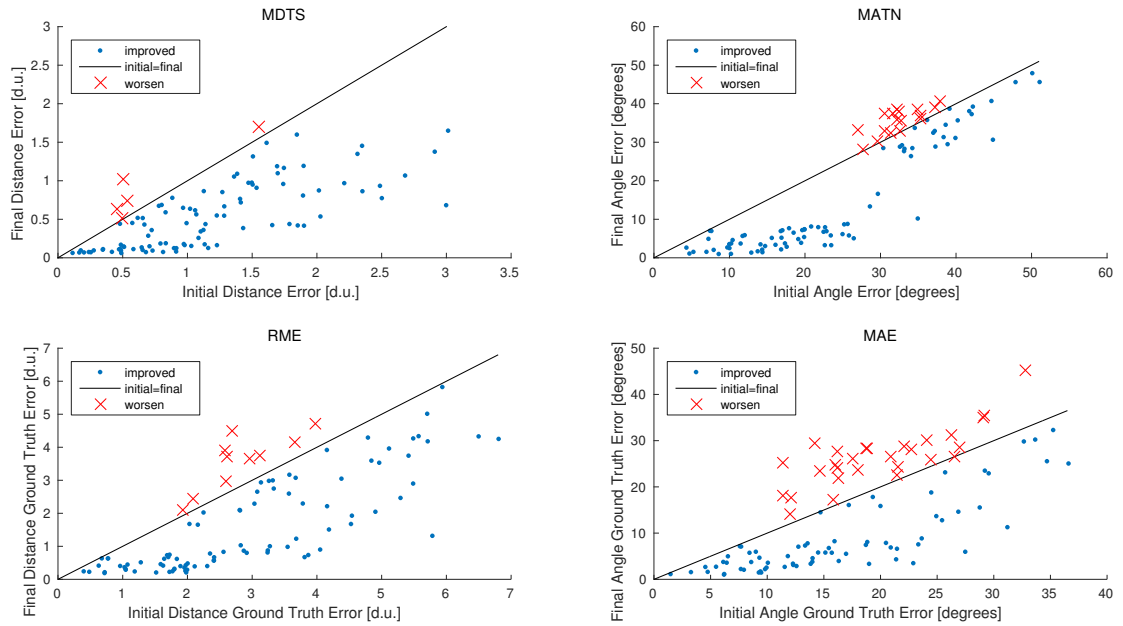


FIGURE 3.25: Initial vs. Final error with large initial error when using Gradient Descent

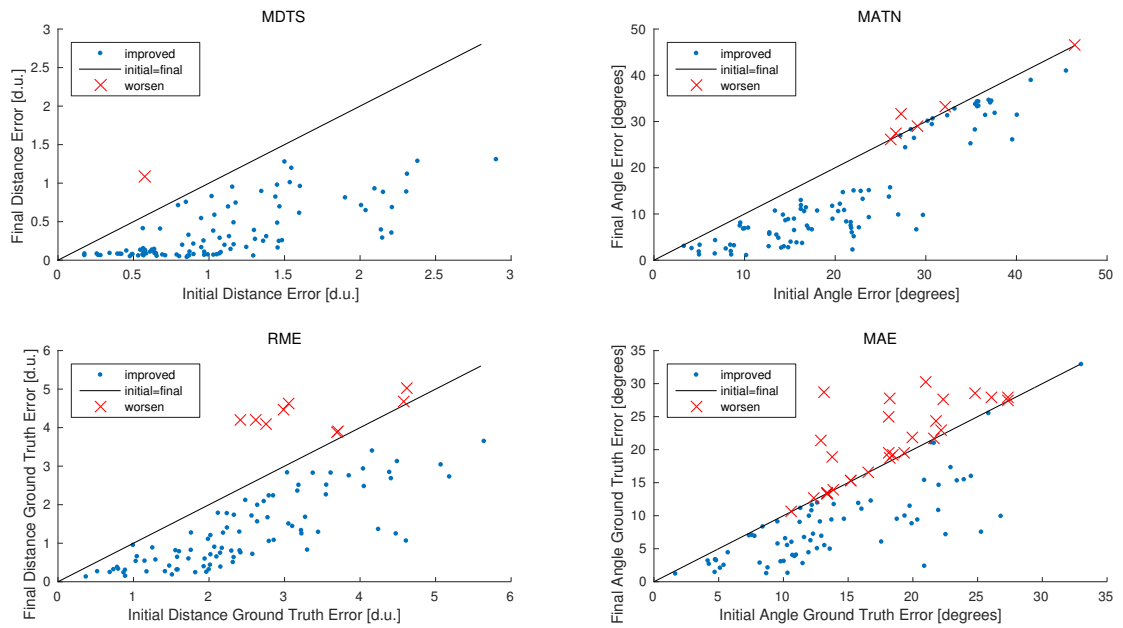


FIGURE 3.26: Initial vs. Final error with large initial error when using Levenberg-Marquardt

Chapter 4

Pose Estimation using Global Optimisation

Chapter Summary

This chapter presents a global optimisation algorithm to estimate the pose of a grasped object. It aims at correcting a coarse estimate given from vision or estimating the pose without any prior estimate. This algorithm is a stochastic Monte Carlo optimisation method, and the details of its implementation are outlined, along with results in simulation and in a real system.

4.1 Introduction

As discussed in Section 3.4, local optimisation approaches fail to converge to the correct or a satisfiable result if the initial estimate is too far off the real pose of the object. This is due to the existence of local minima, where gradient-based optimization methods tend to get trapped in. Besides, local methods are also unsuitable for objects with complex geometries. If an object is composed of hundreds or thousands of planes, vertices and edges, it becomes increasingly difficult to find the object pose based on a limited number of contact points. Furthermore, there are situations where vision information may be very unreliable or not available at all. These include environments with reduced visibility, such as underwater, disaster scenarios with smoke and debris, or complete darkness (in case of using cameras) or fire (in case of infra-red sensors) [118]. These hazardous settings demand that a system is able to stably grasp and manipulate objects without any visual cues.

In this section, a method to estimate the object's pose using a purpose-made global optimisation algorithm is presented. This method presents several advantages when compared to gradient-based optimisation, namely:

1. it avoids local optima
2. it is able to obtain the object's pose without an initial estimate
3. it is capable of estimating the pose of objects with a high degree of complexity

This method may be used under two different scopes:

- Together with a vision-based object tracking system, correcting an initial estimate obtained from this sensing modality. This is achieved by setting a reduced search space, allowing a very fast rectification of the object pose, even for very complex shaped objects.
- Without any initial knowledge of the object's pose, searching the whole space of positions and orientations around the robot hand, finding putative poses

and ranking these poses according to how well they fit the tactile sensory information.

The proposed algorithm belongs to the class of Monte Carlo stochastic optimisation methods, previously presented in Section 2.6.2.3. In particular, it is an evolutionary algorithm, also inspired by Simulated Annealing methods, where the system “temperature” is related to the size of the jumps allowed inside the search space. It also bears resemblances to the method proposed by Kormushev *et al.* [151, 152], which was used in the context of reinforcement learning. This chapter presents an overview of the method, along with the details of its implementation. Simulation results are presented and analysed along with results on a real robot and possible applications of the method.

4.2 Methods

4.2.1 Algorithm Setup and Cost Function

The objective function to be minimised is similar to Equation (3.5) introduced in Section 3.2.3.2. It takes into account both the distance between contact locations $f^{(m)}$ and object surface S and the angle between the surface \hat{n} and the contact normals $\hat{u}^{(m)}$. While in gradient search methods it is valid to use a set of equations (one for each contact), the re-sampling step of the global optimisation algorithm requires the objective function to output a single positive scalar value. This is obtained through the addition of the individual costs for each contact. Since the distance component is always a positive value and the inner product of two unit vectors is always in the range of $[-1, 1]$, the objective function in Equation (4.1) is guaranteed to return a positive value.

$$G(\mathbf{x}) = \sum_{m=1}^k \min_{s^{(i)} \in S} (\|(\mathbf{q}f^{(m)}\mathbf{q}^* + \vec{t}) - s^{(i)}\| + w_n(1 - \langle \mathbf{q}\hat{u}^{(m)}\mathbf{q}^*, \hat{n}^{(i)} \rangle)) \quad (4.1)$$

The first step of the algorithm consists of pre-processing the object in order to allow faster computation of the cost function. This is essential for the method’s performance, as the cost function in (4.1) is evaluated thousands of times. This pre-processing stage consists of initially taking the polygon mesh of the object and compute the unit normal vector for each triangle according to Figure 3.8 and Equation (3.4). Then, the object pointcloud is used to construct a k -d tree, as previously detailed in Section 2.6.3 using the PCL kdtree FLANN implementation [115, 160].

4.2.2 Search Algorithm

In order to find the set of parameters \mathbf{x} that minimises the objective function in Equation (4.1), a global optimisation algorithm was implemented. The method proposed in this chapter can be classified as belonging to the class of Evolutionary Algorithms, where the search for the set of parameters that minimise the objective function is done through the sequential evaluation of this function with random parameters. The resulting cost of each evaluation translates into the fitness of these parameters. Each set of random parameters can be understood as a “guess” that is being made, with lower values of the cost function corresponding to poses that better explain the current sensing data. A guess and its associated cost are paired and are henceforth in this thesis referred to as a *particle*. By re-sampling these “guesses” according to their fitness and applying small modifications to the parameters, the method converges to locations in the search space where the fitness is higher, hence more likely to be the solution of the optimisation problem. Using the terminology presented in 2.6.2.3, these particles are the candidate solutions, with the parameters being the chromosomes. Due to the nature of the problem, only mutations are allowed to occur, and these are termed *noise*.

4.2.3 Generation of the Initial Population

Before the re-sampling scheme begins, an initial population needs to be created. This initial population is created according to the desired application, as introduced in Section 4.1. These can be either a pose rectification from an initial estimate provided by a vision tracking system, or a pose estimation without any prior knowledge of the object's pose.

One of the main differences between these two usages of the global pose estimation method lies in this initial phase. For the pose correction method, the initial population is generated to lie within a limited search space around the initial estimate, while for the global pose estimation without an initial “guess”, the population must cover the whole space of positions and orientations around the robot palm.

Since this generation of new particles is heavily based on the generation of pseudo-random numbers, functions to generate two types of random numbers were implemented:

- Uniform pseudo-random numbers were generated using the C++ `rand()` function, which generates a random integer from a uniform distribution. Dividing this integer by the maximum random integer constant `RAND_MAX`, one obtains a uniform floating point random number in the range of $[0, 1]$.
- Normally distributed pseudo-random numbers were generated using the Box-Muller transform [167].

The Box-Muller transform generates a pair of normally distributed random numbers from a pair of uniform random floating point numbers in the range of $[-1, 1]$. This transform is explained in Equations (4.2) and (4.3), where two random numbers $\{z_0, z_1\}$ are generated such that their squared sum is smaller than 1. These two numbers are then used for the generation of the pair $\{r_1, r_2\}$:

$$\begin{cases} z_0 = 2 \cdot u_1 - 1 \\ z_1 = 2 \cdot u_2 - 1 \end{cases}, s = z_0^2 + z_1^2 < 1 \quad (4.2)$$

$$\begin{cases} r = \sqrt{-2 \cdot \frac{\log(s)}{s}} \\ r_1 = z_0 \cdot r \\ r_2 = z_1 \cdot r \end{cases} \quad (4.3)$$

Different combinations of population generation were tested with both uniform and normally distributed (or Gaussian) random numbers, with the best results for each intended application being obtained using the following configurations:

Pose Correction

The initial population for correcting the pose requires only the creation of seven normally distributed random numbers ($r_{\{1,\dots,7\}}$) through the Box-Muller transform, shown in Equations (4.2) and (4.3). Setting the search spaces for rotation ss_r and translation ss_t , the values of the parameters are calculated according to Equations (4.4) and (4.5), where the quaternion is normalised after being set. This guarantees that the population of initial transformation parameters generated are normally distributed around the initial estimate.

$$q_w = 1 - r_1 \cdot ss_r$$

$$q_{\{x,y,z\}} = r_{\{2,3,4\}} \cdot ss_r \quad (4.4)$$

$$\mathbf{q} = \mathbf{q} / \|\mathbf{q}\|$$

$$t_{\{x,y,z\}} = r_{\{5,6,7\}} \cdot ss_t \quad (4.5)$$

Global Pose Estimation

For global pose estimation, the initial population needs to be carefully constructed to make sure the whole search space is evenly covered. The first 30% of this initial population is generated so that there are quaternions containing rotations of 90° and 180° . This is done by setting at most two elements of the quaternion as ones and the others as zero. After normalisation, these quaternions will represent “straight” orientations (“upside down”, “right side up”, etc.). This is done in order to have members of the initial population in these orientations which typically everyday objects tend to be in.

The next 70% of the initial population is generated to make sure the search space is evenly covered. The rotation quaternions are created through the method presented by Marsaglia [168], guaranteeing that the orientation search space is uniformly covered. This method, used to uniformly choose points from the surface of an hypersphere, requires the generation of four uniform pseudo-random numbers in the range of $[-1, 1]$. Equations (4.6) to (4.8) show the computation of each random quaternion.

$$\begin{cases} x_1 = 2 \cdot u_1 - 1 \\ y_1 = 2 \cdot u_2 - 1 \end{cases}, s_1 = x_1^2 + y_1^2 < 1 \quad (4.6)$$

$$\begin{cases} x_2 = 2 \cdot u_3 - 1 \\ x_2 = 2 \cdot u_4 - 1 \end{cases}, s_2 = x_2^2 + y_2^2 < 1$$

$$R = \sqrt{\frac{1 - s_1}{s_2}} \quad (4.7)$$

$$\mathbf{q} = [x_1, y_1, R \cdot x_2, R \cdot y_2] \quad (4.8)$$

The position vector is obtained through the generation of three uniform pseudo-random numbers in the range $[-ss_t, ss_t]$.

4.2.4 Re-sampling scheme

Monte Carlo optimisation and evolutionary methods depend heavily on the design of good heuristic to re-sample the candidates. This procedure consists of, given a population of sets of parameters and their associated cost with respect to a given cost function, replicate the individuals according to their fitness/cost. Given that the objective is to minimise a cost function, the function in Equation (4.9) was devised, which translates the cost $G(\mathbf{x})$, computed by the objective function in Equation (4.1) into its *weight* $W(\mathbf{x})$. This weight W relates inversely to the cost and is then a measure of probability that this particle is re-sampled.

$$W(\mathbf{x}) = \left(1 + \frac{1}{1 + G(\mathbf{x})}\right)^{p_p} \quad (4.9)$$

The parameter p_p sets the relative importance between estimates and can be adjusted according to the desired application. The relationship between the cost $G(\mathbf{x})$ and weight $W(\mathbf{x})$ and how it is affected by different values of p_p is plotted in Figure 4.1. Increasing p_p increases the weight of the best particles relative to others with higher cost, augmenting the probability that only the best particles are re-sampled. This means that the search is more “aggressive” and is able to converge quicker to a solution, trading off the possibility of finding multiple solutions. This is suitable for pose rectification, as we know in advance that the real pose of the object should be in the vicinity of the initial estimate.

A lower value of p_p allows for slower convergence and a broader search, avoiding convergence to a local minimum. This is desirable in the global search, where there is no initial guess of the object’s pose and is essential to thoroughly explore the whole search space.

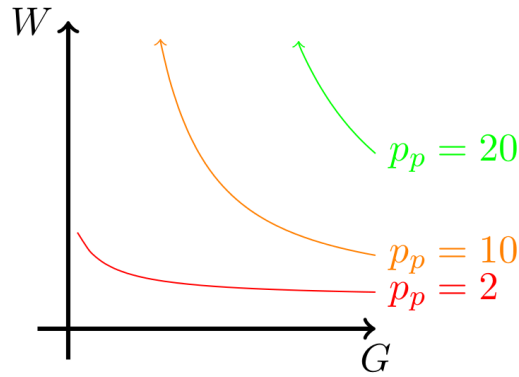


FIGURE 4.1: Cost to Weight Function

The re-sampling scheme was implemented with careful consideration of computational performance. The implementation followed a roulette wheel scheme, where the size of each particle in the roulette wheel is equal to its weight. Each time a particle is generated, its weight is saved into an array and added to an accumulated sum σ_W . To generate a new particle, a uniform pseudo-random number $r_n \in [0, \sigma_W]$ is created. The particle to be replicated \mathbf{x}_d will be the one where, in the n -length array of calculated weights, $\sum_{k=d}^n W(\mathbf{x}_k) > r_n$. This approach differs from the typically used formula $\sum_{k=0}^d W(\mathbf{x}_k) > r_n$. This means that the linear search for the particle to be replicated starts adding the weights from the end of the array, taking advantage of the fact that, with the progression of the algorithm, particles with higher weights (lower cost), will be located mainly at the end of the array. The number of additions required by starting from the end of the array is then much smaller than if the addition started from the beginning.

Figure 4.2 shows a diagram of what the weights array might look like, with the weight of each element being represented by its width in the bar. The next particle to be sampled sits somewhere in the middle of the array, at the location pointed by the blue arrow, meaning that $r_n \approx \sigma_W/2$. Starting from the end of the array will require far fewer additions to reach that location than if one would start from the beginning of the array. This design choice allows a much faster resampling while maintaining the conditions for fitness proportionate selection.

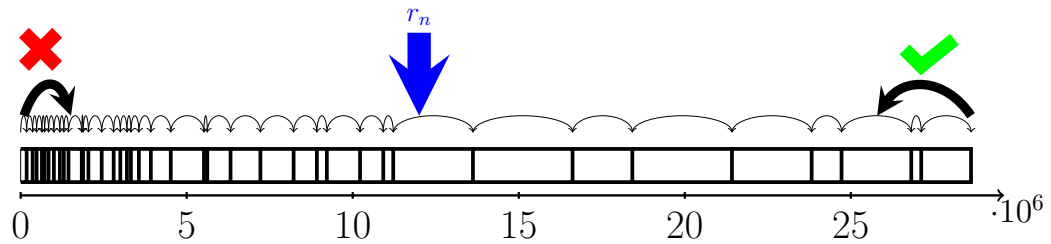


FIGURE 4.2: Re-sampling scheme

Differently from typical implementations of this type, for example in genetic algorithms, every particle is maintained throughout the duration of the search, instead of replacing older particles with new ones. This choice was made due to both practical and computational reasons. Replacement of particles could lead to *particle deprivation* [2] and the loss of diversity in the population. This means that the algorithm might converge too quickly to one of the solutions, and reach a situation where all the particles are very similar to each other. Keeping all the previous particles maintains a diverse population and retains the probability, however low, that worse particles can be resampled and converge to other solutions.

Computationally, this design does not require the weights array and its accumulated sum to be entirely recalculated at each iteration. Only one addition is performed and one insertion in the array. There is, however a computational trade-off in this choice, as the memory requirements increase, since the system must store the whole set of previous particles and their weights. Given the memory capacity available in modern computers, this requirement does not present a major setback, as the memory requirements to store this information is commonly in the range of a few megabytes.

The columns in Figure 4.3 show the time required to generate each thousand particles. In this type of algorithms, the time to resample would normally increase linearly as the number of particles that the algorithm samples from grows. By using this resampling strategy, on the contrary, the time to generate new particles slightly decreases with the progress of the algorithm. The reason for this nearly constant duration is that, with the progress of the algorithm, better and better

particles start being located at the end of the array, enabling their quick resampling. The choice of p_p also affects the performance of this step, as a higher value and a more “aggressive” search means that the algorithm will run faster, as the best particles quickly stack up at the end of the population array.

The example in Figure 4.3 was taken with a high value of p_p , with the resampling and modification of the particles (described in the next section) from a large population is done even faster than the generation of an initial population.

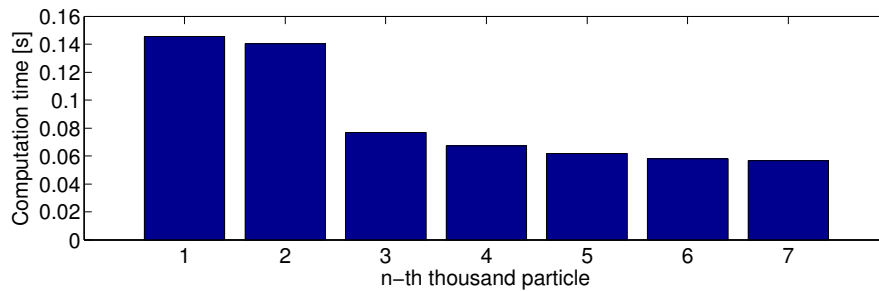


FIGURE 4.3: Computation time to generate each thousand particles

4.2.5 Noise addition

Another essential step of the algorithm is the slight modification of each resampled particle. This change is referred to in the literature as *noise*, *perturbation*, *variation* or, in the context of genetic algorithms, *mutations*. These changes in the particles are what allows the searched to be carried out locally, in the vicinity of the resampled particles.

This noise addition was done by producing two Gaussian pseudo-random values with standard deviations that decrease throughout the progress of the algorithm. These random values were created using the Box-Muller transform, already introduced in Equations (4.2) and (4.3), which conveniently creates two normally distributed random numbers with zero expected value and unit variance. Multiplying this random number by a scalar σ_0 , results in a pseudo-random number with standard deviation of σ_0 , while keeping the mean at zero. This property arises from the linearity of the expected value, as shown in Equations (4.10) and (4.11)

. This is in turn multiplied by the size of the desired search space to ensure that the search does not tend to go outside of the limits defined by the search space.

$$E[a \cdot X] = a \cdot E[X] , \sigma_X = \sqrt{E[(X - E[X])^2]} \quad (4.10)$$

$$\sigma_{aX} = \sqrt{E[(a \cdot X - E[a \cdot X])^2]} \implies \sigma_{aX} = a \cdot \sqrt{E[(X - E[X])^2]} \quad (4.11)$$

$$\therefore \sigma_{aX} = a \cdot \sigma_X$$

One of these random values is added to the orientation quaternion and the other to the translation vector. As mentioned, the variance of this added noise decreases with the progress of the algorithm, starting from an initial value of σ_0 and tending to zero as it approaches the end of the runtime. So the noise added to the j^{th} particle is shown in Equation (4.12) and an example is plotted in Figure 4.4 This makes the search more coarse in the beginning of the runtime, increasing granularity as the iterations approach their maximum number n_p . The search then becomes finer with the progress of the algorithm, with the rate at which the standard deviation of noise decreases depending on the design variable p_n .

$$\sigma_j = \sigma_0 \left(1 - \frac{j}{n_p}\right)^{p_n} \quad (4.12)$$

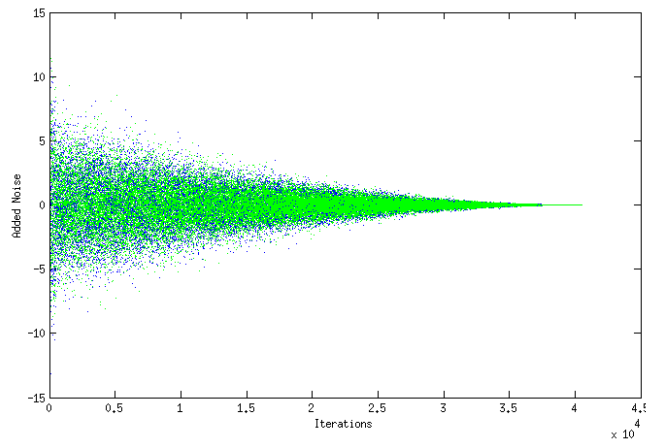


FIGURE 4.4: Noise added to particles over algorithm iterations

4.2.6 Minimisation of the objective function

Two stopping criteria were implemented: a maximum number of iterations n_p was reached or the discovery of an accurate estimate. Since the output of the cost function $G(\mathbf{x})$ does not provide enough information on its own about the quality of the estimate – it depends on w_n and the number of contacts k – a “confidence” indicator was put in place. It was calculated at every 1000 iterations for the current best estimate and, if the confidence was above a desired value, the algorithm stopped and a solution was assumed to be found. This confidence indicator can be thought of as the inverse of the average error over the k contacts, cancelling also the effect of w_n and scaled to a range of $[0, 100]$. It is computed through Equation (4.13).

$$C(\mathbf{x}) = 100 / \left(1 + 100 \left(\frac{G(\mathbf{x})}{k \cdot (1 + w_n)} \right) \right) \quad (4.13)$$

The evolution of particle cost is plotted in Figure 4.5, with the cost for each particle in the vertical axis (in log scale) plotted in blue, the average cost of the last 200 particles is plotted in red and the minimum cost found in green. It can be seen that the search converges to particles with lower cost, stopping after 15000 iterations. After the search is finished, the estimate with lower cost is saved, along with the last 1% of the generated particles.

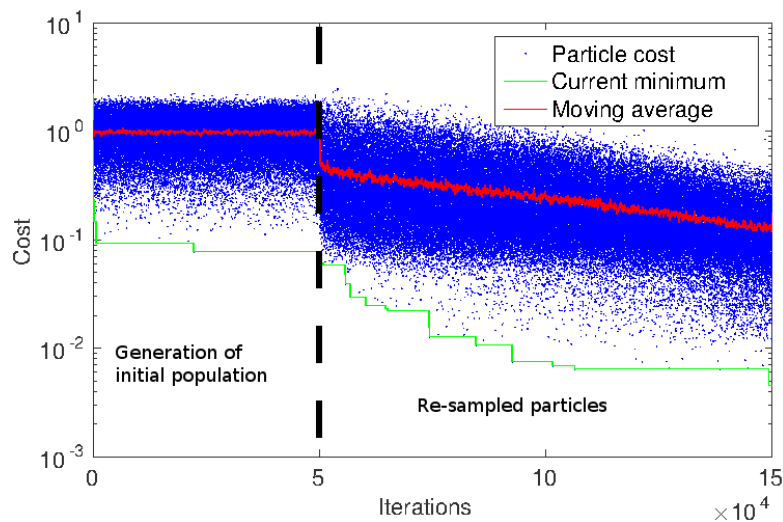


FIGURE 4.5: Progress of the global optimisation algorithm – cost over iterations

4.2.7 Post processing of results

When estimating the pose of an object without having an initial estimate, multiple solutions can be found simultaneously. This is due to symmetries in the object or from having few fingers touching the object. Therefore, a number of possible poses are kept for evaluation. This group of solutions is created by taking the poses obtained in the last 1% of the generated particles and comparing them to each other. This is done simply by finding the Euclidean distance between the orientation quaternions $\|\mathbf{q}_1 - \mathbf{q}_2\|$ and between the position vectors $\|\vec{t}_1 - \vec{t}_2\|$. If a solution is deemed sufficiently different from every existing accepted solution it is added as to the group. Then, a simple but fast collision detection method was implemented, which evaluated each of the solutions in the group. This step was made as simple as possible, for computational reasons, and required only that the object point cloud in that pose do not have any of its points within a vicinity of a number of points inside the robot (palm, knuckles, etc.). If a pose violates this condition it is discarded. A diagram of this collision detector is shown in Figure 4.6,

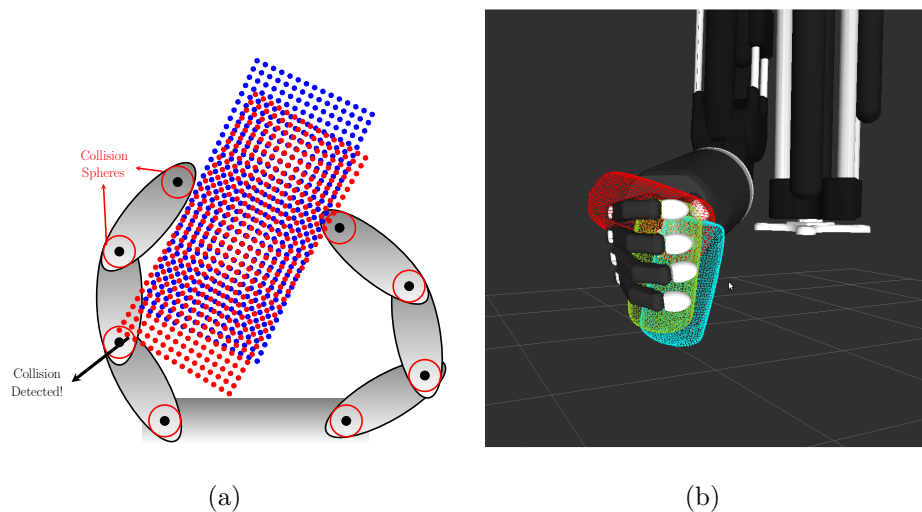


FIGURE 4.6: Collision checker for valid poses. Blue/green: object point cloud in valid pose, red: invalid pose

Finally, a Levenberg-Marquardt optimisation was performed on the best solution to further improve this estimate, similarly to what is described in Section 3.2. This

step ensures that the output of the algorithm is a minimum in that region.

4.3 Results

4.3.1 Simulation Results

This section presents the results obtained in a simulated environment. The simulator used is the Gazebo multi-robot simulator [169, 170], which allows us to obtain of contact information similar to that which is available when using the intrinsic contact sensing scheme – contact location and normal force direction. The decision to carry out the experiments in a simulated environment was made because it allows accurate quantitative validation of the results, given the direct availability of ground truth values.

The simulated robot was made to securely grasp two objects and two types of experiments were carried out:

- The first experiment took the object’s ground truth location and modified that pose with a small rotation and translation from that pose. This step simulated what is obtained in situations of reduced visibility of the object, as is the case during a robot grasp, where the robot hand occludes the object, deteriorating the performance of vision-based object tracking systems. In this experiment a very complex shaped object was used.
- The second experiment used a simpler shaped object and no approximate estimate of the object pose was provided to the algorithm.

On both experiments the data from the simulated tactile sensors was modified with increasing Gaussian noise on both the contact location and the normals and different number of fingers were made to grasp the object. This allowed a thorough evaluation of the method, outlining its strenghts and limitations, despite the amount of added noise being far greater than what is typically present in a real

system, where both the joint encoders for the forward kinematics and the force-torque sensors used for tactile sensing present a high degree of accuracy. Each trial represents a one-shot estimate, and does not rely on previous estimates of the object pose. This choice was done to evaluate the performance of the method on its own, although in real applications the system could use the pose obtained in the previous run as an initial estimate.

4.3.1.1 Pose correction

As mentioned earlier, the first scenario starts from a coarse estimate of the object's pose, obtained by applying a small transformation to the ground truth location. In this scenario, a reduced search space is set – an angle smaller than 45° and a maximum translation of 5 cm. Also, the search was tuned to a more aggressive convergence, setting the design parameter p_p in Equation (4.9) to a high value. An object of very complex geometry was used which, in principle, would pose difficulties to the accuracy of the algorithm, as it creates a number of local minima even within a small search space. The chosen object was a small 3D printed statue of the poet Sappho^a, containing thousands of vertices.

One result of a pose correction experiment is shown in Figure 4.7, where the object in its ground truth location is shown in orange, the initial estimate is in red and the result of the pose correction is shown in purple. It can be seen that, despite the high complexity of the object, the method correctly and accurately identifies the object pose, with the estimate overlapping the ground truth almost perfectly.

The results of running the pose correction algorithm is shown in Figure 4.8, with the upper row containing an histogram of the errors in the initial estimates provided to the algorithm for translation and rotation and the bottom row with the histograms of the final translation and rotation error after applying the pose correction method.

^a The bust of the poet Sappho was kindly provided by Artec3D – www.artec3d.com

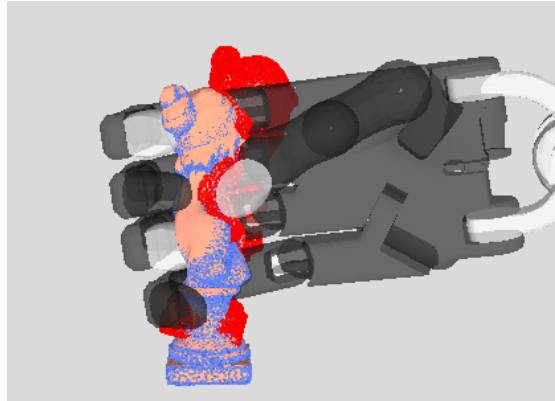


FIGURE 4.7: Pose correction. Initial estimate in red, ground truth in orange and resulting estimated pose in blue

Five fingers were touching the object and the sensor data was adulterated by adding random Gaussian noise to the contact location and normal, with means of 0.9 mm and 5° respectively, which is higher but within the same order of magnitude as the errors present in a real system using *intrinsic* tactile sensing.

The initial estimates have an average distance to the ground truth of 33.2 mm and an initial rotation angular error of 16° , which is comparable to the accuracies typically encountered in vision tracking systems using RGBD cameras when the object is being occluded. After applying the pose correction method the location and orientation errors were reduced to an average of 4.05 mm and 5.0° , with standard deviations of 2.8 mm and 2.19° . The average run time to obtain a solution was 0.64 seconds on a Intel[®] Core i7, allowing that the object is tracked at 1.5 Hz.

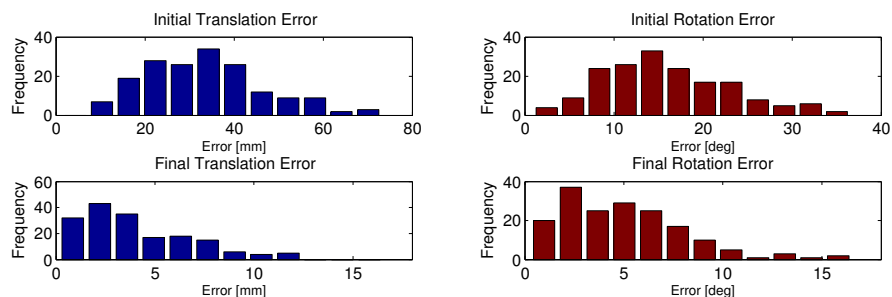


FIGURE 4.8: Histograms for initial and final errors on rotation and translation for pose correction

Further evaluation of the algorithm's performance was made by increasing the level of sensor noise and by grasping the object with three, four and five fingers. An arbitrary threshold of 1 cm for position and 15° for orientation was set, in order to consider a result successful if its error falls inside both these thresholds. The position error is calculated as the distance between the origins of the object's frame of reference, which in the case of the statue lies at its base and the orientation error is the total angle between orientations, calculated according to the formula $\theta = 2\cos^{-1}(q_w)$, which takes into account the angular error around every axis.

Figure 4.9 shows an histogram with the mean errors in position and orientation dependent on the number of fingers touching the object and the level of noise injected in the sensor data. Figure 4.10 shows the success rate when using the previously mentioned criteria of position error below 1 cm and orientation error below 15 degrees.

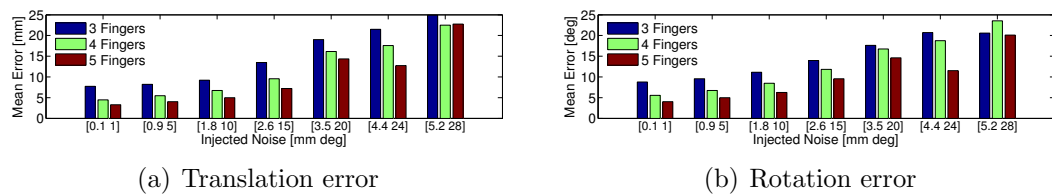


FIGURE 4.9: Mean errors after pose correction for different number of contacts and noise levels

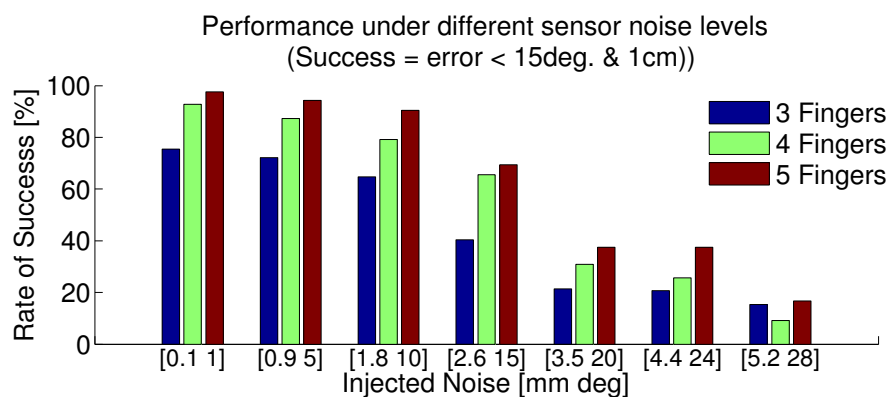


FIGURE 4.10: Rate of success for pose correction for different number of contacts and noise level. A trial is considered successful if the error is under 1 cm and 15° .

2978 tests were carried out, with the results showing that the success of the algorithm is dependent on the number of fingers touching the object and obtains a

correct estimate of the object’s pose over 80% of the time as long as the sensor noise is below 1.8 mm and 10° and four fingers are touching the object.

4.3.1.2 Global pose estimation

If an initial estimate is not available, the pose of a grasped object can still be estimated, relying only on the tactile sensing and proprioceptive data. This method can be useful in situations where tracking an object using vision is unfeasible. Such cases arise in environments with reduced visibility, such as disaster scenarios, where smoke, debris or fire render the information of cameras, RGBD and laser range finders unusable [118, 171]. An everyday example of another situation where the estimation of an object’s pose using vision is unreliable is when dealing with transparent objects. In this experiment, a wine glass was used which would prove difficult to accurately track by a vision system.

The formulation of the problem is alike the problem of pose correction from an initial estimate, with the object being placed arbitrarily in the region of the robot palm. Since there is no approximate knowledge of the object’s pose, the search space is augmented to fill all the possible orientations and positions around the robot palm. The design parameter p_p , in Equation (4.9) was also tuned to a lower value, to ensure the search is not too “aggressive” and does not converge too quickly to a solution which may not be the global minimum, but instead searches the whole space thoroughly.

Figure 4.11 shows the result of one experiment, where the arbitrary initial location of the object is shown in red, the ground truth is shown in green and the result of pose estimation is shown in orange. In this trial the estimate of the object pose accurately overlaps the real location of the object even if the initial location was placed deliberately far away from the robot hand, further validating the ability of the algorithm to converge to a satisfactory solution. It should be noted that in this grasp posture the small finger touches the glass stem, which was done in order to avoid that “upside down” poses yield similar costs and thus induce erroneous results.

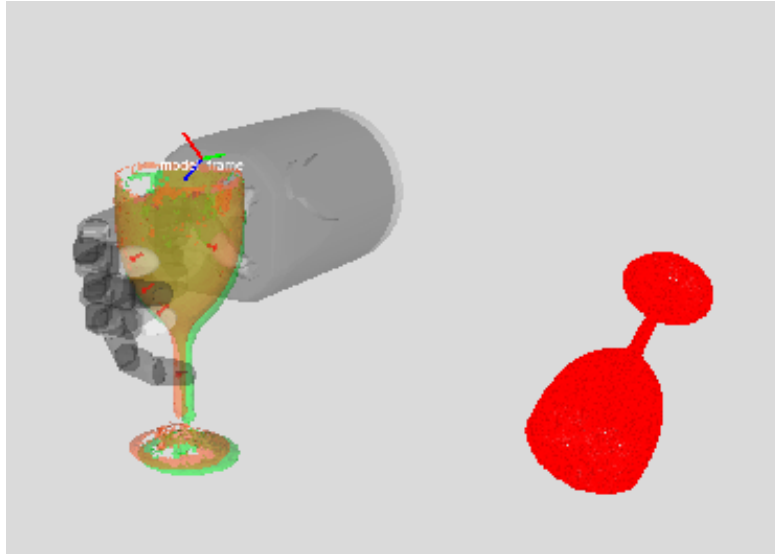


FIGURE 4.11: Global pose estimation. Initial estimate in red, ground truth in green and result pose in orange, force normals are displayed as red arrows

The method was evaluated, as before, injecting noise in the sensing data and touching the object with three, four and five fingers. In this case the accuracy of the results depends very heavily on the number of fingers touching the object. This is expected, as given the symmetries existing on the object a variety of poses that coherently match the tactile sensing data are present if there is a small number of contacts.

When five fingers are touching the object, the mean absolute error was 7.1 mm for position and 3.72° for orientation. In this case, the angle error discarded the error around the vertical axis, given that the object is revolute around this axis. The average duration of the algorithm was 63 seconds.

The previously defined criteria to evaluate a trial as successful was applied and yielded a success rate over 75% when using at least four fingers and a sensor noise below 0.9 mm and 5° , which is commensurate with the error on the real system. The number of tests carried out was 1329.

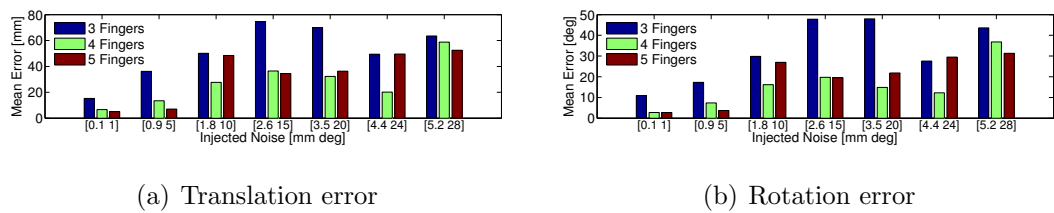


FIGURE 4.12: Mean error in global pose estimation for different number of contacts and noise levels

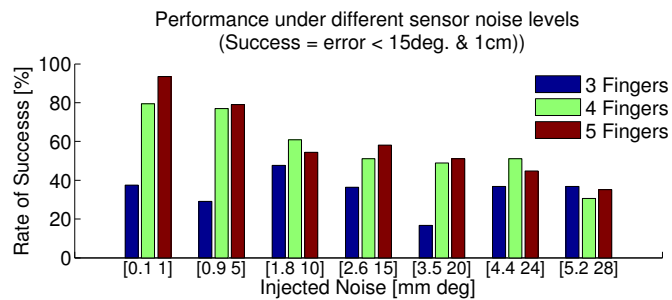


FIGURE 4.13: Rate of success for global pose estimation different number of contacts and noise level. A trial is considered successful if the error is under 1 cm and 15° .

4.3.2 Results Using a Real System

4.3.2.1 Experimental Setup

The experimental setup consisted of a Mitsubishi RV6-SL industrial robot manipulator equipped with a Shadow Dextrous Hand [172] and a Microsoft Kinect RGBD camera [173]. Force-torque sensors were mounted on three of the robot hand fingertips with the scheme detailed in Section 2.2.1. Vision tracking was done through the implementation of the Particle Filter point cloud tracker available with the PCL (Point Cloud Library) [115], which uses the depth information from the Kinect sensor.

In this section, quantitative results are not provided due to the difficulty of acquiring an accurate ground-truth benchmark. Since the available electromagnetic tracking systems' performance deteriorated significantly when in the vicinity of the robot, due to the metal parts and magnets on the Hall effect sensors. Fiducial

markers were also tested but did not provide the desired accuracy. Nevertheless, the pictures presented in this section should provide sufficient information to qualitatively evaluate the performance of the method and validate the results previously achieved in the simulated environment.

4.3.2.2 Pose correction from vision

The first set of experiments validate the method for the pose correction setting. Figure 4.14 provides an example where the object is accurately tracked when it is sitting on top of a table, where it is accurately tracked by the vision system. On the left side of the picture, before the object is grasped, the yellow object model almost perfectly overlays the red thermal bottle point cloud obtained by the Kinect sensor. As soon as the the object is being grasped by the robot hand, seen on the right side, the accuracy of this estimate decreases significantly. The result of pose correction is shown in purple, where it can be seen corresponds to the real pose of the object shown by the partial point cloud of the red object.

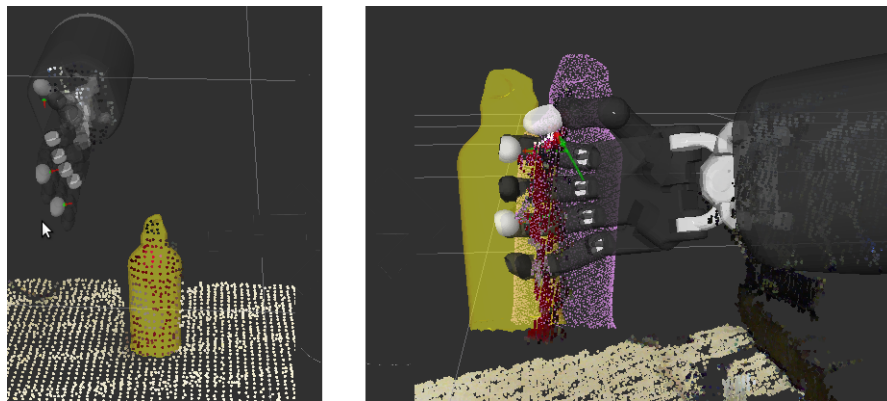


FIGURE 4.14: Pose correction result – Vision based tracking results in yellow before and after occlusions are created by the grasp. The pose corrected using the proposed method is displayed in purple

Figure 4.15 shows other examples of the pose correction results, displaying also the sensed force as green arrows and the sensed contact normal as red arrows.

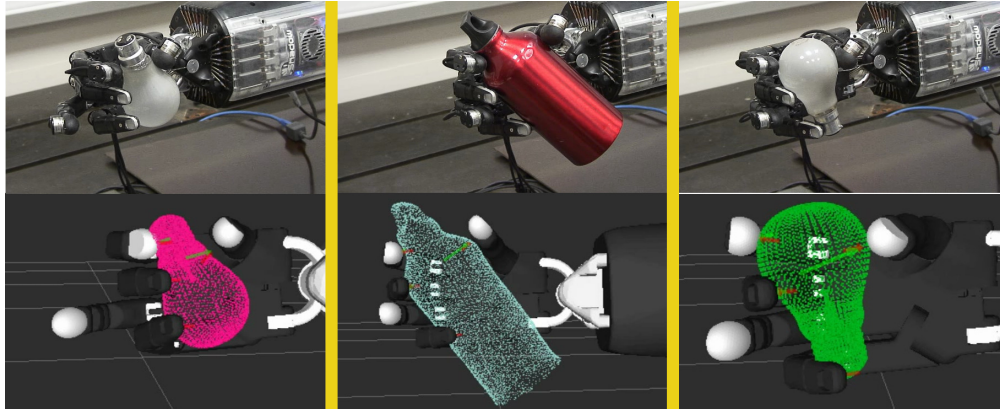


FIGURE 4.15: Pose correction results with different objects

4.3.2.3 Global Pose Estimation – Hand Over and Place

In an experiment that aims to illustrate a possible application of the pose estimation method, a robot receives a small object handed over by a human, placing it in a predefined location. In this experiment, it is not possible to obtain any initial estimate of the object pose, since before it is being grasped it is held by the human collaborator and is not visible by the vision system. Besides, the small size of the object poses another problem for the vision system, as the resolution of the RGBD camera is not enough to allow the detection of the object. Figure 4.16 shows, on the left, the robot hand grasping the blue pencil and, on the right, the sensed point cloud obtained by the 3-D camera overlaid with the robot model. It can be seen that the object is not clearly visible by the camera, with very few points belonging to the pencil being detected by the depth sensor, rendering it impossible to be tracked by the vision system.

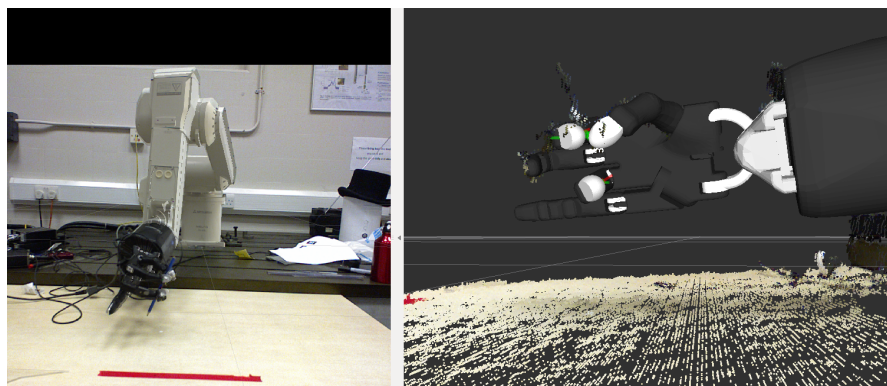


FIGURE 4.16: Robot grasping a pencil, object model overlaid with point cloud

Given this shortcoming of the vision system, the estimation of the object pose must rely solely on the tactile and proprioceptive data. The objective of the task was to place the pencil inside a narrow hole (1.5 cm radius) in a box, requiring the estimate to be very accurate, with errors over 1.5 cm or 15° jeopardising the successful completion of the task.

Figure 4.17 shows the result of the pose estimation method, with the interaction forces shown in green, contact normals shown in red and the object model at the estimated pose in pink. The experiment is shown in Figure 4.18, where a pencil is handed to the robot by a human collaborator, the pose is estimated and the object successfully inserted through a narrow hole in a box.

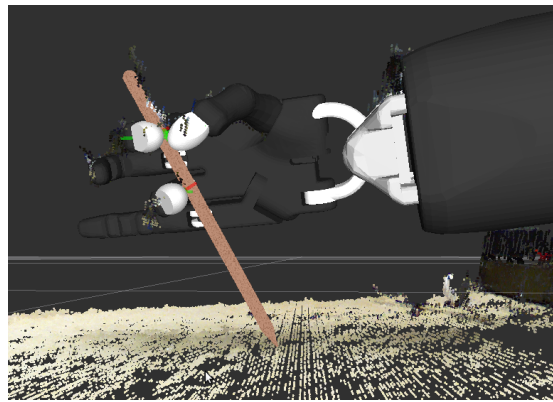


FIGURE 4.17: Result of estimation of a pencil's pose

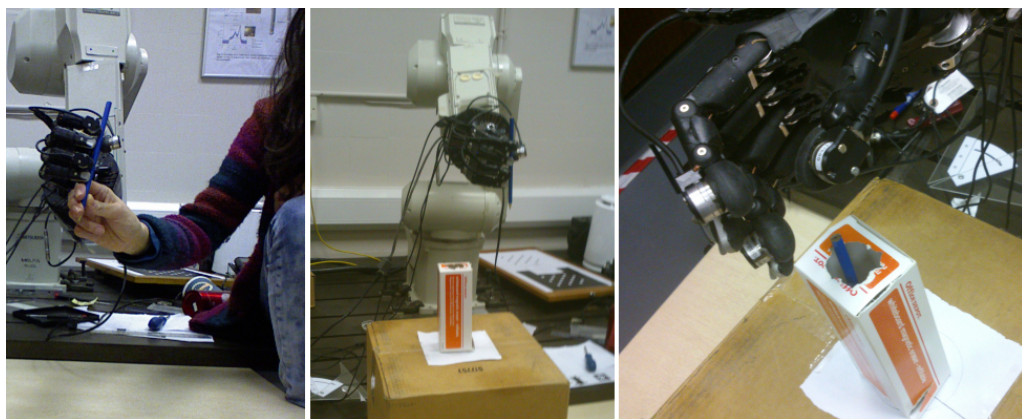


FIGURE 4.18: Hand over and place experiment

4.4 Discussion

Performing a global search to minimise the cost function parametrised by the object geometry and current tactile information proved to significantly outperform gradient based methods. Besides surmounting most of local methods' shortcomings, it enables the estimation of an object's pose even without any vision based tracking system.

Figure 4.19 shows the optimisation landscape for a pose estimation problem. This figure shows, in the same scale, the cost of each parameter in \mathbf{x} as a colored dot, with the lowest cost for that parameter shown in a solid line. The particle with minimum cost found ($G(\mathbf{x}) = 0.3$) for this problem was:

$$\mathbf{x} = \begin{bmatrix} -0.58 & 0.4 & 0.027 & -0.72 & -0.065 & 0.028 & -0.132 \end{bmatrix} \quad (4.14)$$

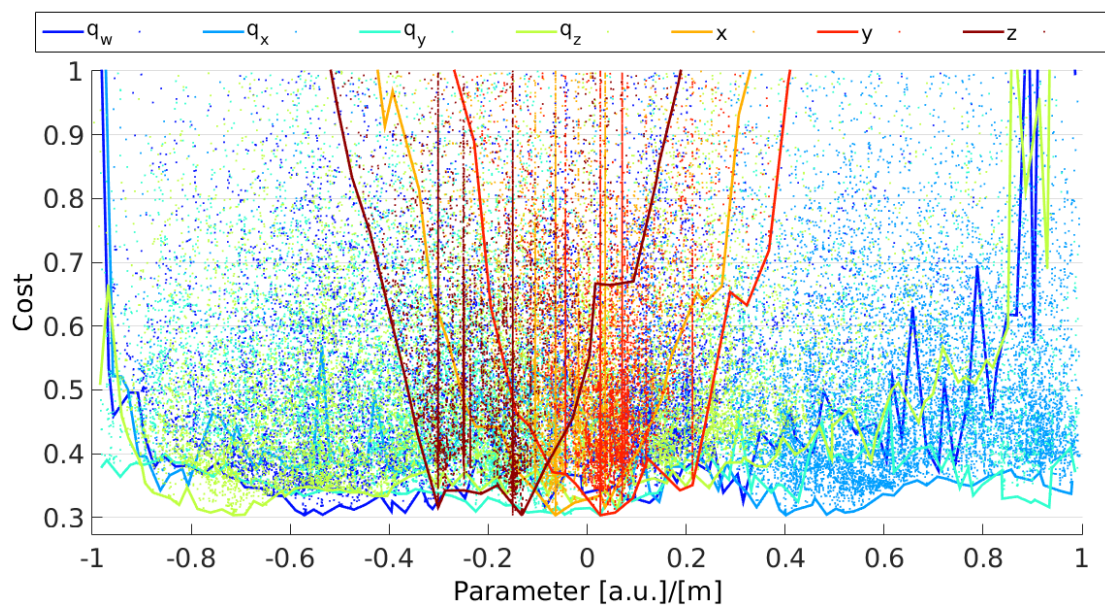


FIGURE 4.19: Particle Landscape

The “ruggedness” of this landscape is very noticeable, which indicates that it is a difficult problem to find its global minimum [174]. Despite this, the proposed method showed a high rate of convergence to a satisfiable solution. Figure 4.20

shows the results for pose correction, when comparing initial and final error for both rotation and translation. In this picture, points to the right side of the green line represent improvements on the estimate error. It can be seen that the algorithm successfully improved the object pose estimate, with only four instances where the angle error was higher than the initial error. It should be pointed out that these cases arise due to small initial angle errors and that the translation error was improved in all trials.

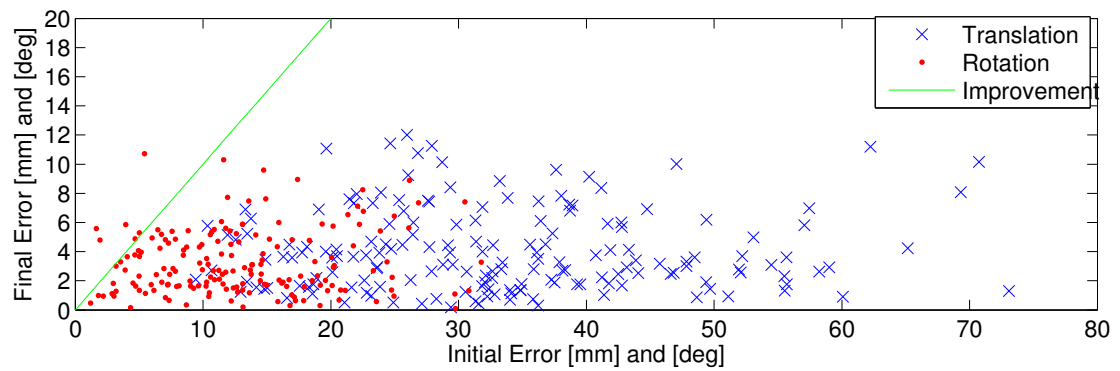


FIGURE 4.20: Initial vs. Final Error for global search

Chapter 5

Pose Estimation from tactile arrays

Chapter Summary

This chapter presents a strategy to represent data from a distributed pressure array sensor in a compact but meaningful way, which enables the detection of a matching geometric shape. Using this descriptor, which is based on Principal Component Analysis (PCA), a grasped object's position and orientation can be estimated by finding object poses where the geometry of local patches of the object surface is consistent with the information provided by the tactile sensors.

5.1 Introduction

Pressure array sensors are a pervasive technology in robotics, being the most widely used sensor type in modern robot hands [37]. These sensors provide information on the distribution of forces on its surface, enabling the estimation of contact force magnitude and location. Force direction and contact normal, however, cannot be measured by present commercial sensors. Furthermore, the data obtained by these distributed sensors is not easily interpretable by a machine, and the sensor capabilities go often unexplored. This chapter contains a descriptor to encode tactile data from distributed tactile arrays, which can be used to match the geometric shape of the surface in contact with the sensor. The descriptor relies on Principal Component Analysis (PCA) to find the directions of highest variance in the tactile data.

The presented technique is based on previous work by Liu *et al.* [85], who used a similar descriptor to classify the local contact shape (*e.g.* flat, edge, vertex, sphere, etc.). This approach is further extended in this chapter to enable the quantification of the coherence between tactile and an hypothesised local shape. Through optimisation it is possible to find object poses that present a high degree of matching in all the sensing areas of a robot hand.

An introduction to PCA was given in Section 2.6.4 and its application to tactile data is detailed in Section 5.2.1 along with the rationale of this data descriptor. Section 5.2.5 demonstrates the application of this method to the problem of pose estimation and Sections 5.3 and 5.4 present the results and conclusions of this approach.

5.2 Methods

5.2.1 PCA on Tactile Data

The application of Principal Component Analysis follows a similar approach to the one presented by Liu *et al.* [85]. The input data is an $m \times 3$ matrix containing the spatial position of each active tactile element on the sensor plane as the first two dimensions and the measured pressure, scaled by a factor α , as the third dimension, as shown in Equation (5.1). $(\bar{x}, \bar{y}, \bar{p})$ are the mean values and the parameter α scales the pressure values to be commensurate with the other two dimensions.

$$\mathbf{M} = \begin{bmatrix} (x_1 - \bar{x}) & , & (y_1 - \bar{y}) & , & \alpha(p_1 - \bar{p}) \\ (x_2 - \bar{x}) & , & (y_2 - \bar{y}) & , & \alpha(p_2 - \bar{p}) \\ \vdots & , & \vdots & , & \vdots \\ (x_m - \bar{x}) & , & (y_m - \bar{y}) & , & \alpha(p_m - \bar{p}) \end{bmatrix}^T \quad (5.1)$$

The covariance matrix \mathbf{C}_M of the above input tactile data can be decomposed into the eigenvector and eigenvalue matrices as shown in Equations (5.2) and (5.3).

$$\text{cov}(\mathbf{M}) = \mathbf{C}_M = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T \quad (5.2)$$

$$\mathbf{C}_M = \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \cdot \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{bmatrix}^T \quad (5.3)$$

Figure 5.1 shows an interpolated reading of a tactile array, overlaid with the calculated principal components as red green and blue lines. It can be seen that there is a larger variance in the direction of the red line, and also significant variance in the direction of the green line.

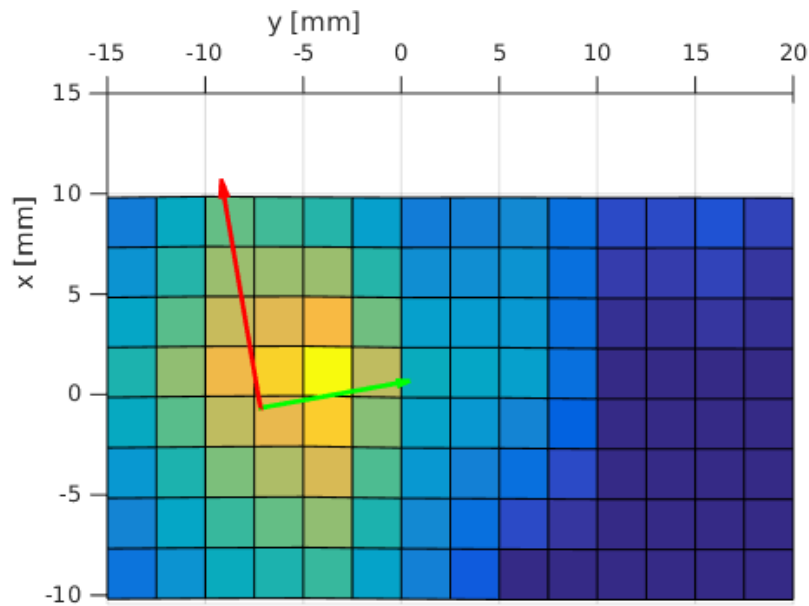


FIGURE 5.1: Principal Components of a tactile sensor frame

Describing the tactile information using these principal components and outputting only three vectors provides a compact description of the entire tactile frame. This descriptor has been already employed successfully to identify the shape of a touched surface by looking at the resultant eigenvalues [85]. In this chapter, the direction of the eigenvectors is also taken into account to assess the match between tactile data and the local geometry of a contact. The variances of a matching geometric shape will have a magnitude similar to the eigenvalues in the directions given by their respective eigenvectors. From this measurement model, the pose of a grasped object can be found such that there is a high degree of match in every tactile sensor.

5.2.2 Selection of Scaling Parameter

While the x and y dimensions are the same for both tactile and geometrical data, the pressure values need to be scaled to correspond with the spatial dimension corresponding to the sensor's normal axis. In order to accomplish that, a study using Finite Element Analysis (FEA) was carried out in Abaqus [175] to find the relationship between pressure data and the local geometry of an object .

Different shapes were pressed against a simulated tactile sensor, with angles of 0° , 2° , 5° , and 7° to the sensing plane, as shown in Figure 5.2. The sensing area was modelled as an hyperelastic, isotropic material, with density $\rho = 990 \text{ Kg/m}^3$, using the Arruda-Boyce model [176] with parameters $\mu = 10000$ and $\lambda_m = 1.1$.

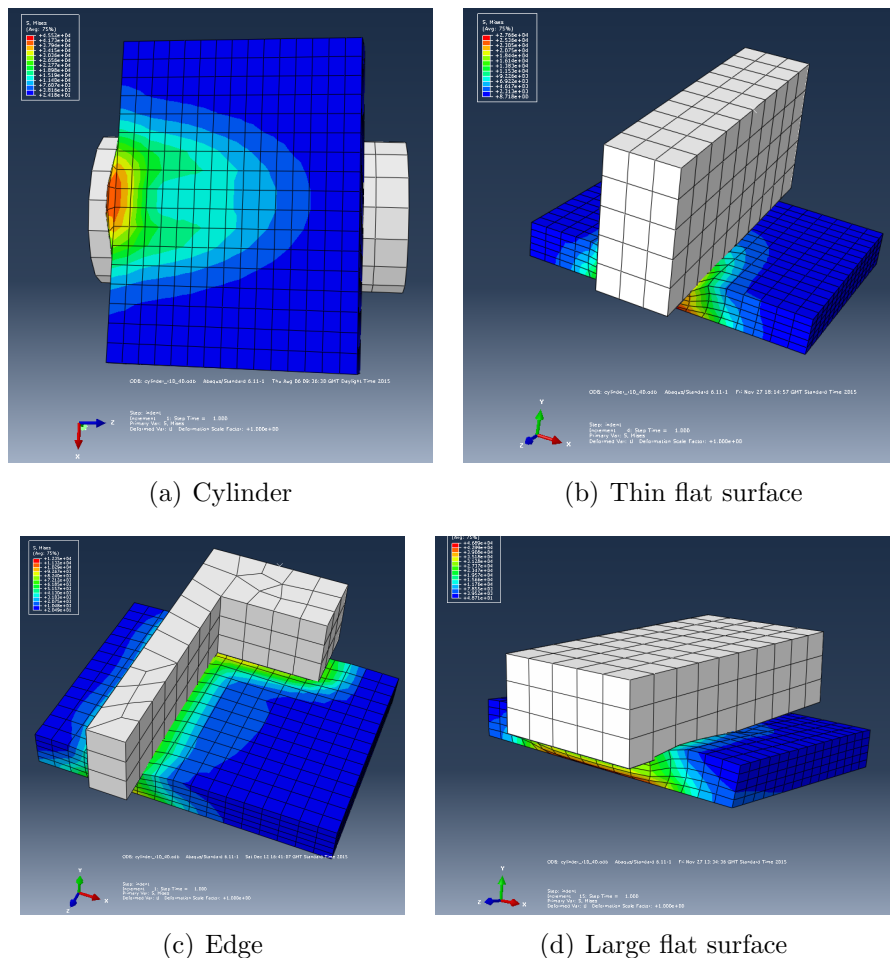


FIGURE 5.2: Finite Element Analysis

The Finite Element Analysis tests were done by Shan Luo

The objective of this study is to find a parameter α that scales the pressure values so that one of the principal components is aligned with the slope of the object geometry. Figure 5.3 shows the effect of the scaling parameter when a cylinder is pressed against the sensing area. In each picture, the principal components are shown as blue, red and yellow arrows.

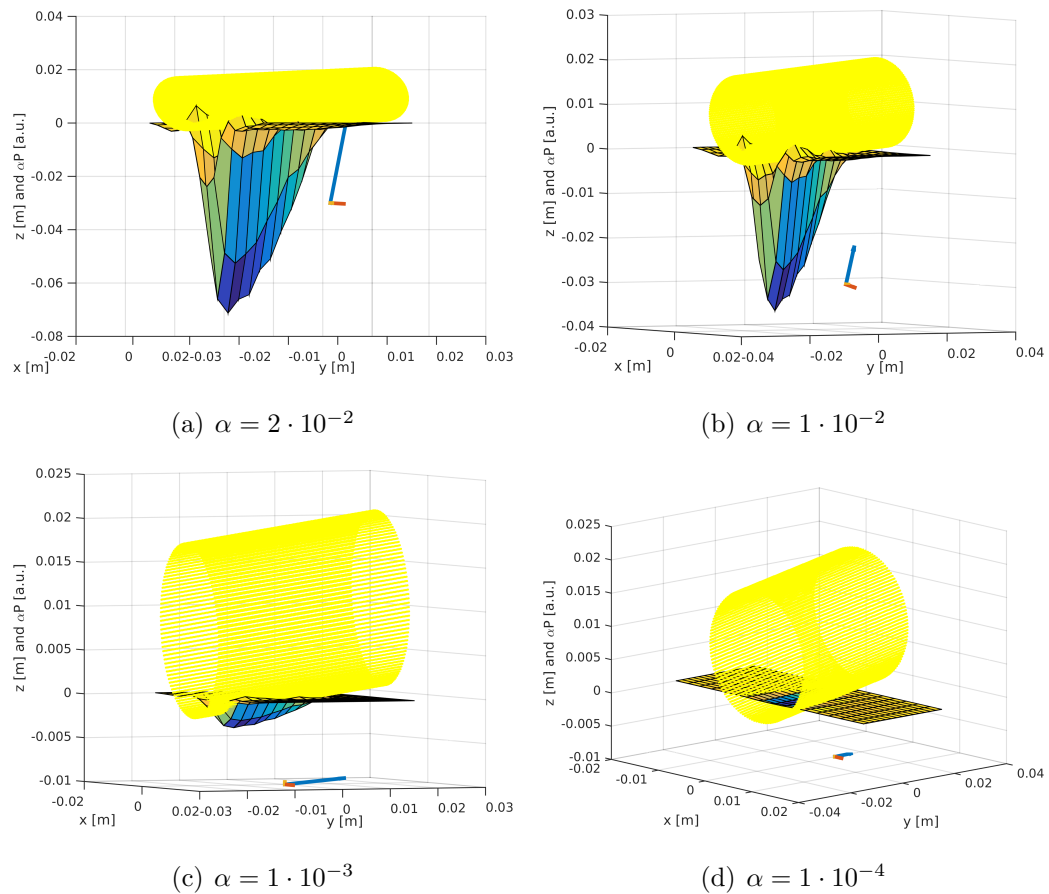


FIGURE 5.3: Effect of pressure scaling value α in the tactile profile

From the pictures above, it can be seen that the value of α that better approximates the slope of the object is the one in Figure 5.3(c), where $\alpha = 1 \cdot 10^{-3}$. Figure 5.4 shows a cross-section of the pressure profile (as dots) for the flat surface at 5° (Figure 5.3(d)). The arrows indicate the direction of the main principal component in this 2-D case for different values of α . A good scaling value will have this principal component parallel to the object cross-section shown in black. The selected value of α was $\alpha = -9.677 \cdot 10^{-4}$, which, in this case yielded an angle error of 0.14° .

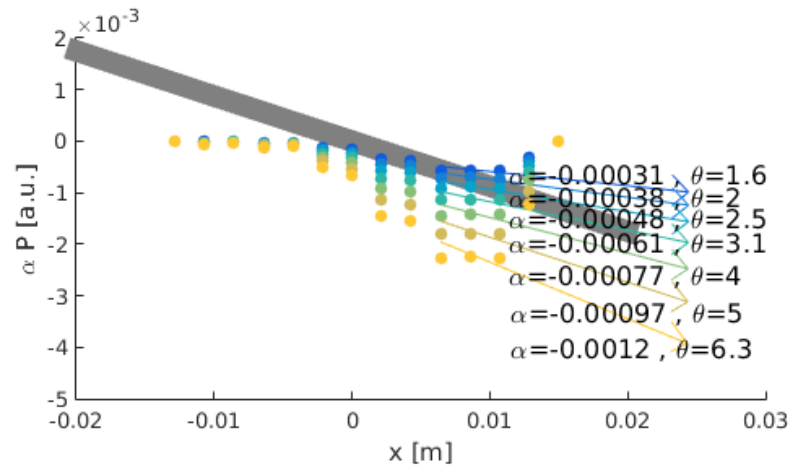


FIGURE 5.4: Cross-section of the pressure profile and largest principal component for different values of α

Table 5.1 summarises the error for the different tested shapes and contact angles. It can be seen that the error is kept always under 1° .

TABLE 5.1: PCA angle error for $\alpha = 9.66 \cdot 10^{-4}$

Shape	0°	2°	5°	7°
(a) Cylinder	0	0.9	0.14	0.42
(b) L-shape	0.19	0.50	0.81	0.68
(c) Flat (wide)	0	0.4	0.07	0.5
(d) Flat (thin)	0	0.07	0	0.62

This relationship does not show any significant dependence on the contact force. Figure 5.5 shows the pressure profile for the shape in Figure 5.2(d) pressed at the same angle, with different indentation depths. As the object is pressed harder, the number of active tactile elements increases, but the slope of its principal component is approximately constant.

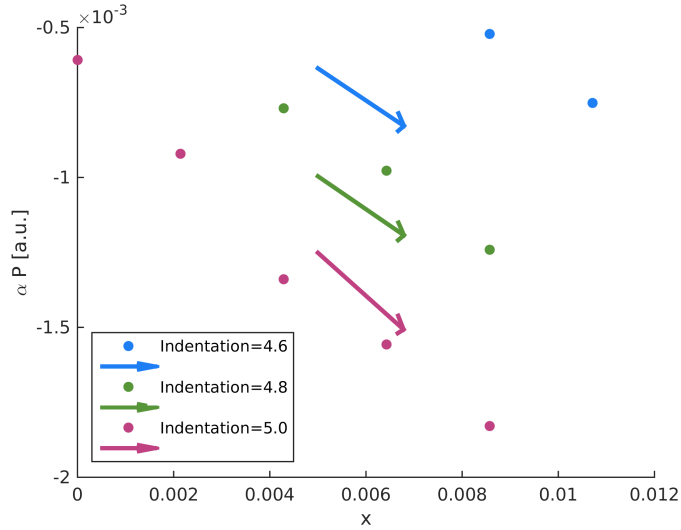


FIGURE 5.5: Pressure profile and main principal component for different indentation depths

This relationship between pressure and indentation depth is dependent of the tactile sensors used, the material properties of the soft layer above the sensor and the shape of the local contact. However, in practice, an approximate value is sufficient to obtain good results, since the variances in other directions are much larger than along this normal direction.

5.2.3 Computing the Eigenbasis

As previously discussed in Section 2.6.4, the symmetry of the covariance matrix leads to its eigenvectors being pair-wise orthogonal. As such, these vectors define a basis that can be interpreted as a Cartesian coordinate system. This coordinate system defined by the unit norm eigenvectors is commonly known as the *eigenbasis*.

Given that \mathbf{E} is an orthogonal matrix (*i.e.* its columns are orthogonal and have unit norm), then $\mathbf{E}^T \mathbf{E} = \mathbf{I}$, thus $\mathbf{E}^{-1} = \mathbf{E}^T$. If we define \mathbf{M}' as the set of points \mathbf{M} described in its eigenbasis, matrix \mathbf{E} can be used as the rotation matrix that performs this transformation.

$$\mathbf{M}' = \mathbf{E}^T \cdot \mathbf{M} \quad (5.4)$$

The covariance matrix of these points when transformed into its eigenbasis \mathbf{M}' results in the equality in Equation (5.5) (again, note that $\mathbf{E}^T \mathbf{E} = \mathbf{I}$):

$$\begin{aligned}
 \mathbf{C}_{\mathbf{M}'} &= \frac{1}{n-1} \mathbf{M}' \mathbf{M}'^T \\
 &= \frac{1}{n-1} \mathbf{E}^T \mathbf{M} \mathbf{M}^T \mathbf{E} \\
 &= \mathbf{E}^T \mathbf{C}_{\mathbf{M}} \mathbf{E} \\
 &= \mathbf{E}^T \mathbf{E} \mathbf{\Lambda} \mathbf{E}^T \mathbf{E} \\
 \mathbf{C}_{\mathbf{M}'} &= \mathbf{\Lambda}
 \end{aligned} \tag{5.5}$$

Equation (5.5) demonstrate that the covariance matrix of a set of data points transformed into its eigenbasis is the diagonal matrix containing the eigenvalues in its main diagonal. This is in agreement with the fact that PCA is an orthogonal transformation that converts a set of correlated data into a new space where the components are uncorrelated.

5.2.4 Matching tactile to 3D point cloud covariance

The central idea of this descriptor is that when the sensor is in contact with a geometric shape, the variances in the distribution of pressures in the tactile array should resemble the variances on the geometry of the touched shape. In other words, the eigenbasis of the tactile data covariance matrix should form a “good” coordinate system for the points in the geometry of the local patch of the object surface that is in contact with the sensor. Figure 5.6 shows a robot finger equipped with a pressure sensor array. The interpolated tactile data is overlaid in the figure, with its principal components shown in red, green and blue arrows. The pressure data is scaled down by the parameter α in Equation (5.1). The local object geometry is shown as the green point cloud and is the lateral strip of a cylindrical object. Intuitively, it can be seen that the directions of highest variance in this local patch should follow the same directions as in the tactile data.

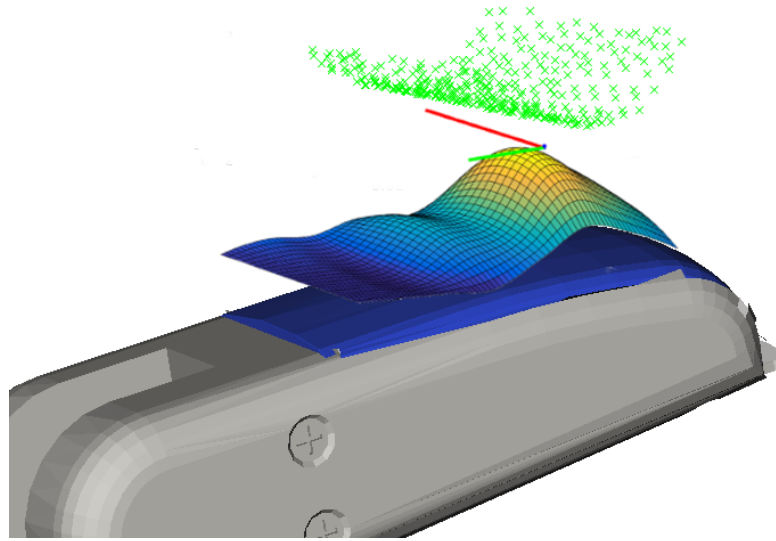


FIGURE 5.6: Finger model, interpolated tactile data, principal components axes and object pointcloud patch

In order to test the coherence between tactile and geometric data the principal components of the tactile data are calculated and the eigenbasis is constructed according to the method described in Section 5.2.3. Then, the patch of the object's point cloud that sits within the sensing region of the tactile sensor is cropped and transformed using that eigenbasis, as detailed in Equation (5.4). The covariance matrix of this set of points is calculated and, following (5.5), it should match matrix Λ , presenting variances in the main diagonal similar to the eigenvalues of the tactile data covariance. Also, the covariance values in the off-diagonal should be small. In summary, if \mathbf{E} is the orthogonal matrix with the eigenvectors of \mathbf{C}_M , then the covariance of the points in the local geometry $s \in \mathbf{S}$, transformed into the the eigenbasis \mathbf{E} should be similar to Λ .

$$\mathbf{S} = \begin{bmatrix} s_{1x} & s_{1y} & s_{1z} \\ & \vdots & \\ s_{nx} & s_{ny} & s_{nz} \end{bmatrix} \quad (5.6)$$

$$\mathbf{C}_{S'} = \text{cov}(\mathbf{E}^T \mathbf{S}) \approx \Lambda \quad (5.7)$$

This similarity can be exploited to gauge the coherence between a frame of tactile data and a patch of an object's surface geometry, represented by a 3D point cloud, by using the following approach:

Subtracting the eigenvalues from the covariance matrix of the surface pointcloud in the eigenbasis $\mathbf{C}_{S'}$. The resulting matrix can be interpreted as follows:

- Positive elements in the main diagonal – variances in the surface point cloud which are not explained by the tactile data
- Negative elements in the main diagonal – variances in the tactile data not present in the object point cloud
- Off-diagonal elements – covariances not explained by the principal components.

The Frobenius norm of this matrix (the square root of the sum of the squared matrix elements) can be used to obtain a cost function that takes into consideration all three sources of inconsistency. The equation that assesses the quality of the match between tactile and geometric data is then Equation (5.8)

$$q = \|(\mathbf{C}_{S'} - \mathbf{\Lambda})\|_F = \sqrt{\sum_{i,j} (\mathbf{C}_{S'} - \mathbf{\Lambda})_{ij}^2} \quad (5.8)$$

A numerical example of the application of this method is presented in Figure 5.7. The active elements of a tactile array are plotted in red, its principal components in blue, red and yellow and the point cloud of a patch of a cylinder in green.

The eigenvalues of the covariance matrix of the tactile data were calculated to be $\lambda_1 = 34.56 \cdot 10^{-6}$, $\lambda_2 = 7.86 \cdot 10^{-6}$, $\lambda_3 = 0.76 \cdot 10^{-6}$. The object point cloud was transformed into the tactile eigenbasis and the covariance matrix found on the object point cloud in the contact area is shown in Equation (5.9)

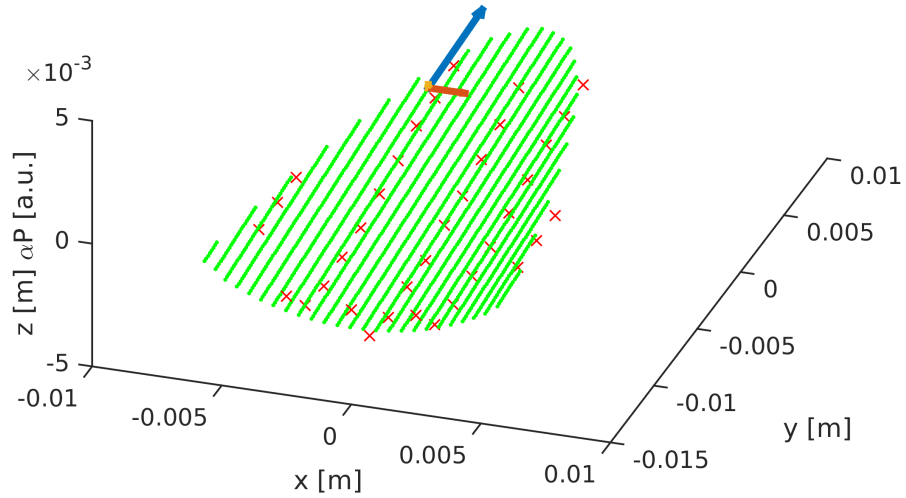


FIGURE 5.7: Evaluation of the matching. Active tactile elements in red, principal components in blue, red and yellow, and local object geometry in green

$$\mathbf{C}_{S'} = \begin{bmatrix} 34.15 & -0.01 & 0.26 \\ -0.01 & 7.55 & -0.00 \\ 0.26 & -0.00 & 0.15 \end{bmatrix} \cdot 10^{-6} \quad (5.9)$$

This result shows a good correspondence between tactile and geometric data with low values in the covariance matrix off-diagonal elements and variances similar to the tactile data eigenvalues in the main diagonal. The cost, calculated through Equation (5.8), is calculated in Equation (5.10) and shows a good degree of matching between tactile and geometrical data.

$$q = \left\| \begin{bmatrix} 0.41 & 0.01 & -0.26 \\ 0.01 & 0.31 & 0 \\ -0.26 & 0 & 0.61 \end{bmatrix} \right\|_F \cdot 10^{-6} = 0.87846 \cdot 10^{-6} \quad (5.10)$$

5.2.5 Object Pose Estimation From Descriptor

A straightforward application of the detailed descriptor is the estimation of a grasped object's pose. The fact that the cost function presented in Equation (5.8) outputs a positive scalar conveniently enables its direct usage in an optimisation routine. The goal is then to find a set of parameters \mathbf{x} , as defined in Equation (1.1), which define an object pose (position and orientation), where the patches of the object point cloud in the vicinity of each contact minimise the cost q . A description of the method is presented below and summarised in Algorithm 5.2.

First, Principal Component Analysis is performed on the tactile array data, obtaining the eigenvalues and eigenvectors of its covariance matrix. A k -d tree is constructed from the object point cloud, allowing faster nearest neighbour searches [160]. A candidate pose is tested for its distance d_l from the sensor contact centroid to the object surface and rejected if the distance is above a threshold of $\epsilon = 30\text{mm}$. If there are surface points in the vicinity of contact, these points are transformed into the local sensor coordinate frame and cropped according to the sensor shape and inside a small height. The coordinate frame of this patch of points is then changed into the eigenbasis, centered on the contact centroid $c = [\bar{x}, \bar{y}, \alpha\bar{p}]$, using the transpose of matrix \mathbf{E}_H , shown in Equation (5.11). The cost is obtained by applying the cost function in Equation (5.8) and the algorithm is summarised in Algorithm 5.2. A distance element is added to the cost from the mean value of \mathbf{S}' , given the fact that the origin of the coordinate frame \mathbf{E}_H is at the contact centroid. After transforming the points into this *eigenbasis*, a good match will also result $\bar{\mathbf{s}}' \approx (0, 0, 0)$, so the norm of the mean value is added to the cost to ensure a small distance between the tactile and geometric centroids. To improve the optimisation procedure, three design parameters h_1 , h_2 and h_3 are added to the algorithm. h_1 penalises poses that do not contain any points in the vicinity of the sensor, h_2 penalises poses where the object penetrates the finger geometry and h_3 is a multiplier to add importance to the weight in Equation (5.8).

$$\mathbf{E}_H = \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.11)$$

Algorithm 5.2 Pose estimation using covariance matching

Input: Pose $\{\mathbf{x}\}$, object pointcloud $\{O\}$, PCA $\{\mathbf{E}_H, \lambda\}$

Output: $\{h_1, h_2, h_3, \epsilon\}$

- 1: Transform contacts using rotation, translation \mathbf{x}
- 2: **for all** k contacts **do**
- 3: Find distance d_l between O and c
- 4: **if** $d_l \leq \epsilon$ **then**
- 5: Get local patch of points $l \subset O$
- 6: Transform points l into tactile sensor frame
- 7: Collect points s within the sensing area ($s \subset l$)
- 8: **if** $\#s \geq 3$ **then**
- 9: **for all** $s_i \in S$ **do**
- 10: $r \leftarrow 0$
- 11: Transform points s into eigenbasis \mathbf{E}_H
- 12: **if** $s_{iz} < 0$ **then**
- 13: $r = r + (-s_z \cdot h_2)$
- 14: **end if**
- 15: **end for**
- 16: Compute covariance matrix $C_{S'}$ of s'
- 17: $G_k = \|(C_{S'} - \Lambda)\|_F \cdot h_3 + \bar{s}' + r$
- 18: **else**
- 19: $G_k = h_1 \cdot d_l + \|\Lambda\|_F \cdot h_3$
- 20: **end if**
- 21: **else**
- 22: $G_k = h_1 \cdot d_l \|\Lambda\|_F \cdot h_3$
- 23: **end if**
- 24: **end for**
- 25: **return** $\sum_k G_k$

As in the previous chapter, the pose of the object was estimated considering two different circumstances:

- One setting assumes the knowledge of the grasped object pose is approximately known, having obtained a coarse estimate from a vision tracking system.

- The other situation assumes that no initial estimate of the pose of the object is known.

Different optimisation methods were tested under these two conditions. In the first situation, the search was performed with a reduced search space, with an orientation angle below 20° and translation below 1 cm. The second setting searched all possible orientations and a position vector below 20 cm from the robot base. Given the possibility of multiple local minima, only global optimisation methods were used. Different global optimisation methods were tested, namely Simulated Annealing [177], DIRECT [157, 178], and the method previously developed by the authors [5], detailed in chapter 4. Table 5.2 compares the performance of these different optimisation methods, when using the same input data, for both conditions – with and without an initial estimate.

TABLE 5.2: Comparison of optimisation techniques

		Method in [5]	DIRECT	Simulated Annealing
PCA based	Average Cost	0.1343	0.1058	0.1308
	Average Duration	16.4901	10.7876	38.6335
distance based	Average Cost	1.2877	1.1102	1.1513
	Average Duration	20.7059	9.2987	30.7758

In this comparison, the DIRECT method shows better accuracy as well as improved computational performance and, as such, it was selected for validation of the pose estimation method presented in this chapter.

5.3 Results

5.3.1 System overview

This method was implemented in a real system, using a Barrett Hand™ BH-280 [179], which contains pressure sensor arrays with 24 tactile elements on each

of the fingers and on the palm. The software interface with the robot as well as the algorithms were implemented in ROS [165], using the Eigen linear algebra library [180] for calculations involving matrices.

The tactile sensors were covered with an Ecoflex[®] 00-10 soft silicone [181] to allow a better distribution of the forces over the tactile elements. A comparison of the distribution of contact pressures with and without using a soft silicone layer is shown in Figure 5.8. For very similar grasps, it can be seen that, when using the soft silicon layers, more tactile elements are active (the yellow and orange values in the upper left corner), obtaining a much more detailed pressure distribution.

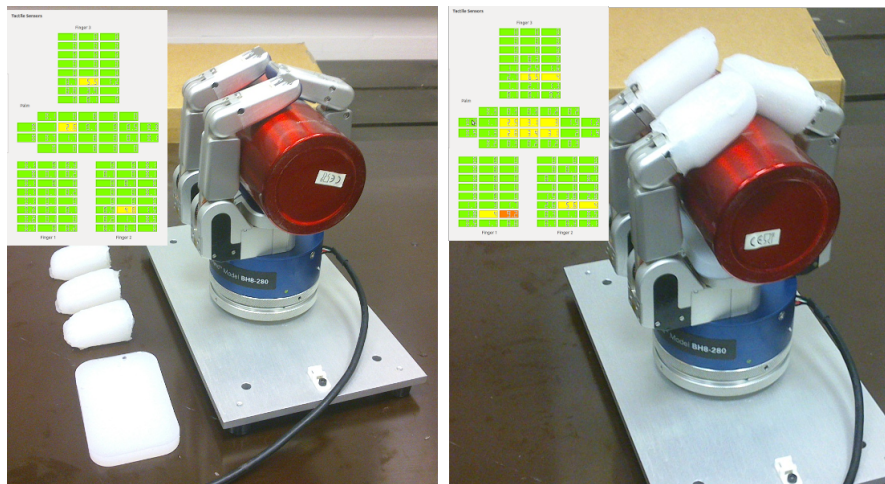


FIGURE 5.8: Comparison of tactile data with and without silicon layer

5.3.2 Pose Estimation Results

The object pose was estimated by having the robot hand grasp an object and perform the minimisation using DIRECT, which was the method which yielded the best results, as seen in Table 5.2. The cost function used for pose estimation is shown in Algorithm 5.2, and is the sum of the costs for each tactile element. This cost function took into account both the the covariance matching shown in (5.8) and the distance between the contact and the object point cloud centroids.

The evaluation of the method was made against a benchmark method where the cost function only took distance into account. This is an approach that resembles

the commonly employed Iterative Closest Point (ICP), while overcoming ICP's main drawback of getting trapped in local minima through the use of a global optimisation method. The algorithm for distance minimisation resembles what is typically used in this context [3, 90, 102, 124, 127, 133, 136]. This cost function outputs the average squared distance between each tactile element and the nearest point in the object surface and is detailed in Algorithm 5.3.

Algorithm 5.3 Pose estimation using distance

Input: Pose $\{\mathbf{x}\}$, object pointcloud $\{S\}$, contacts $\{C^{(k)}\}$

- 1: Transform contacts using rotation, translation \mathbf{x}
- 2: **for all** k contacts **do**
- 3: **for all** i elements **do**
- 4: Find squared distance d_i between S and $C_i^{(k)}$
- 5: $n \leftarrow n + 1$
- 6: **end for**
- 7: $G_k = \sum_i d_i/n$
- 8: **end for**
- 9: **return** $\sum_k G_k$

An initial qualitative evaluation was done using two different objects. In the first experiment the robot was made to hold a box with clearly distinguishable features – edges, vertices, rounded surfaces. The fingers were placed so that the tactile sensors touched these different features. This grasp is shown in Figure 5.9(a), with the features highlighted and labeled in red.

The result after applying this method is shown in Figure 5.9(b), with the picture overlaid with the robot 3D model and the resulting pose using both the ICP-type method and the covariance matching cost function approach. The result of the method matches the tea box seen in the background, better than the benchmark. It should also be pointed out that a correct estimate is obtained despite the significant error seen in the right finger first joint angle. This error is mitigated by the fact that the method can identify that the finger is touching a planar surface, increasing the cost for estimates that do not match this geometry.

The red point cloud, which shows the point cloud of the object at the pose obtained by using the covariance matching method matches the real location of the

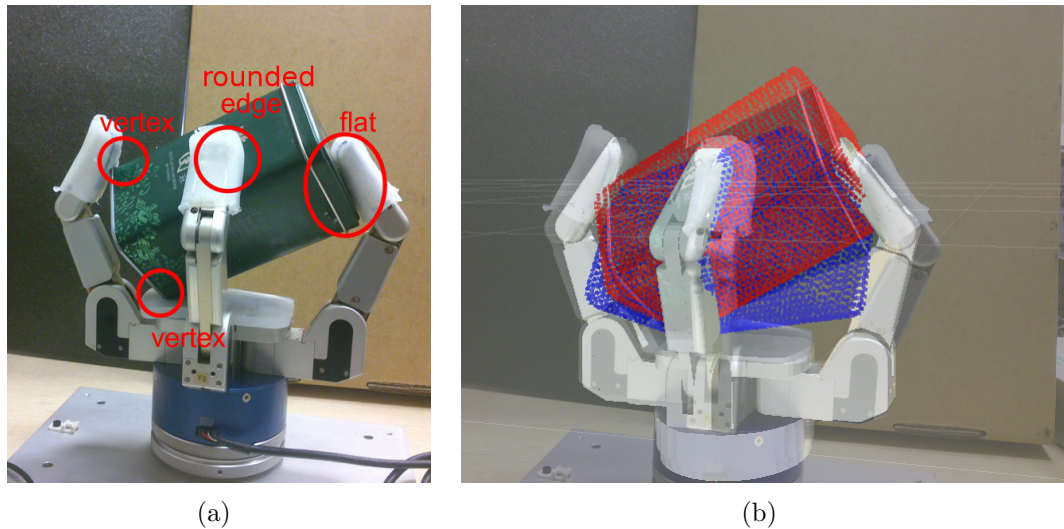


FIGURE 5.9: Pose estimation overlaid on a picture of the grasp. Blue pointcloud shows the result when minimising distance from the surface to the contacts. Red pointcloud shows the result when using the proposed method

object much closer than when minimising only distance. In this example, one explanation for the mismatch obtained the ICP approach may be the small error in the kinematics is present, and can be seen in the proximal angle of the right side finger. While this small mistake in the kinematics compromises the performance of the ICP-type approach, the covariance based matching method still yields an acceptable result. This is due to the local shape information that is contained in the covariance approach, allowing for this error, as it tries to match that finger with a planar surface that satisfies the covariances measured in the tactile sensor frame.

To further highlight the advantages of using this approach, a scenario was constructed that was predictably difficult to be tackled with the typical approaches that use distance information only. A cylindrical thermal bottle was grasped with the fingers being placed in a posture that was problematic to obtain the object pose from. The contacts were positioned at opposite sides of the cylinder, with the finger contacts being placed close together, with the palm contact at the other side of the cylinder. The results are shown in Figure 5.10.

When using information on distance only, it is difficult to disambiguate between poses around the vertical axis. This is because the distances between contact

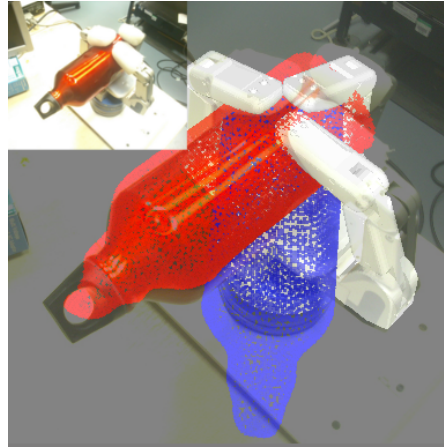


FIGURE 5.10: Result with horizontal thermal bottle. Blue pointcloud shows the result when minimising only the distance from contacts to surface. Red pointcloud shows the result when using the proposed method

centroid and object surface are very similar for very different poses around this axis. Using the covariance matching approach, however, allows to discriminate between these poses, as the largest principal component on the two fingers grasping the side of the bottle are aligned with the cylinder's main axis, resulting in a pose that aligns the bottle axis with those largest principal components.

Also, the method using principal components has an advantage over algorithms like ICP in that it penalises points in the object model not present in the measurement, *i.e.* surface points lying the sensing area where sensing elements are not active. Figure 5.11 shows the evaluation of the cost function for different poses, with angles around the vertical axis in the range of $[-57^\circ, 57^\circ]$. The cost was evaluated using both cost functions in Algorithms 5.2 and 5.3, with the colours of the point cloud in the upper right figure corresponding to the points in the plots below. While the distance only cost function found the minimum around 20° , the proposed covariance based cost function successfully predicted the minimum to be around 0° .

Quantitative evaluation was done using four different objects and six grasps. Obtaining an accurate ground-truth proved problematic, as electromagnetic trackers are not reliable when metal and magnets are present in the surroundings. As such, fiducial markers were used, and the pose was finely adjusted manually, since the accuracy provided by the fiducial markers was insufficient for the evaluation

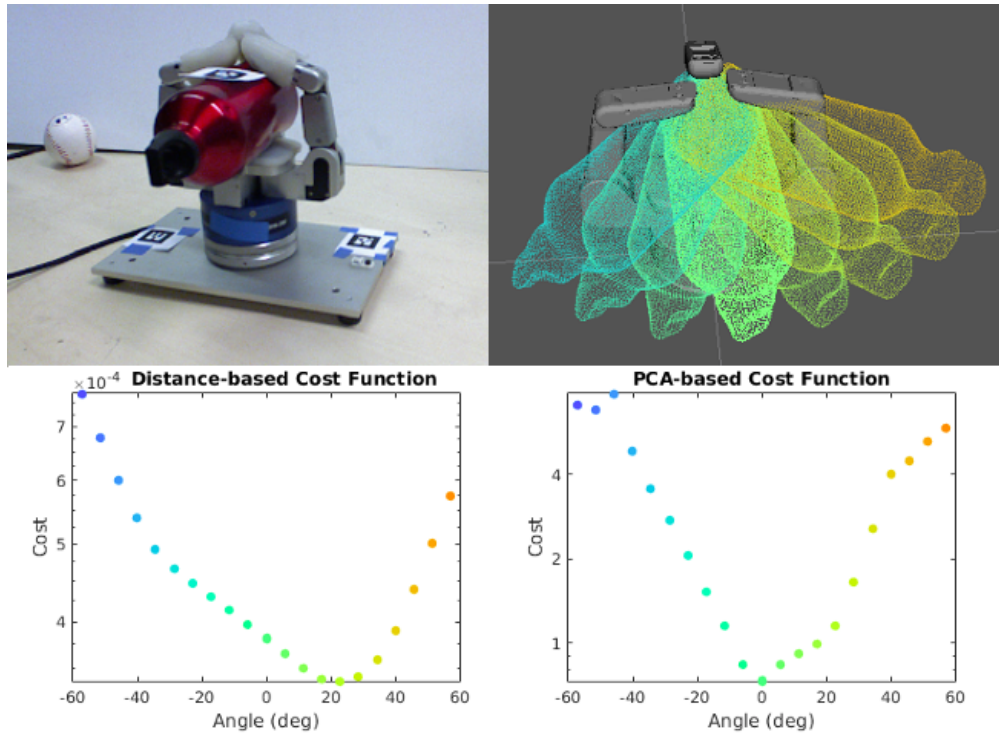


FIGURE 5.11: Evaluation of proposed the cost function *versus* a benchmark based on distance alone. Clockwise from the top: grasping scenario, different poses evaluated, proposed cost function, distance based cost function. Note: the colors in the lower plots match the pose hypotheses on the upper right

of the method. The ground truth was displaced by a random rotation between 0° and 20° around each axis and a translation between 0 and 10 mm along each direction $[x, y, z]$. This enabled to start the pose estimation from an approximate pose, emulating the situation where vision provides a coarse pose estimate to be corrected using tactile sensing. To evaluate the covariance-based matching against the distance-only optimisation benchmark, three criteria were used:

- The mean angle and displacement between the minimisation results and ground truth.
- The angle and displacement error at the lowest cost found in all trials
- The Spearman's ρ correlation coefficient between cost and error.

The mean angle provides a good indicator for the success of the pose estimation method. However, it is very dependent of the optimisation method used, as the optimisation may converge to a local minima, which does not correspond to the

ground-truth. The Spearman's ρ criterion, on the other hand, allows the assessment of the descriptor independently of the optimisation method. By finding the correlation between cost and error, the suitability of the descriptor becomes more evident, as a good descriptor will have stronger correlation between lower costs and higher accuracy in the results.

Spearman's ρ is a measure for the dependence between two variables [182]. It outputs a number between -1 and 1 , quantifying the monotonicity between two variables X and Y . In other words, if lower errors correspond to lower costs, the Spearman's ρ will be close to 1 . The calculation of the Spearman's ρ is done by ranking the n values of X and Y into rank numbers x_i, y_i and then applying the formula in Equation (5.12).

$$\rho = 1 - \frac{6 \sum (x_i - y_i)^2}{n(n^2 - 1)} \quad (5.12)$$

Figure 5.12 shows the resultant cost *vs.* the error for each of the 100 trials in one grasp. A linear fit is added for easier visualisation. In this experiment, the proposed descriptor shown in Fig. 5.12(b), displays a stronger correlation with the angular error than the distance-based cost function in Fig. 5.12(a). This stronger correlation demonstrates the advantages of using the proposed descriptor in optimisation methods or as the measurement model, for example in recursive Bayesian filtering methods.

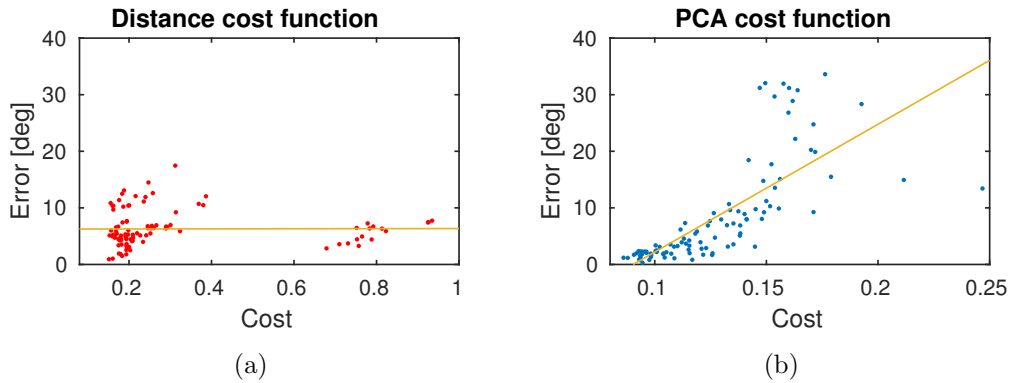
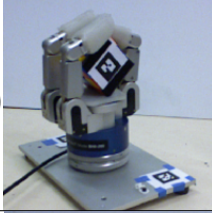
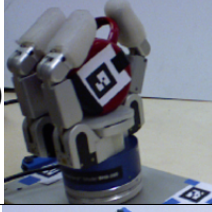
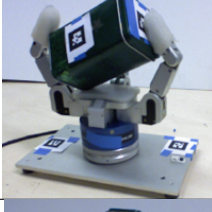
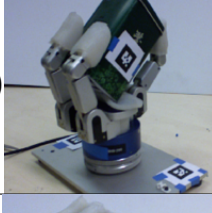
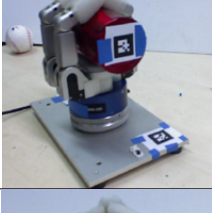
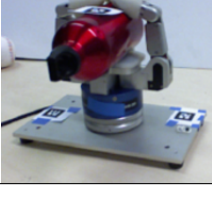


FIGURE 5.12: Result of cost function *vs.* angle error

The results of the experiments for the different objects and grasps are shown in Table 5.3, where the descriptor with best results is highlighted in bold.

TABLE 5.3: Comparison of cost functions – mean error, error at minimum cost and cost *vs.* error correlation coefficient

	Mean Error		Minimum Cost		Correlation Coefficient	
	PCA	distance	PCA	distance	PCA	distance
a) 	4.73° 7.8 mm	23.7° 7.5 mm	1.24° 0.5 mm	25.34° 7.1 mm	0.70 0.57	0.41 0.31
b) 	18.8° 8.5 mm	22.9° 16.2mm	20.2° 15.4mm	24.7° 26.4 mm	0.42 -0.03	0.07 -0.04
c) 	11.61° 5.5 mm	10.90° 4.1 mm	10.53° 4.6 mm	11.86° 3.7 mm	0.71 0.02	0.31 -0.38
d) 	8.30° 11.2 mm	8.70° 5.6 mm	2.74° 5.1 mm	6.60 4.3 mm	0.56 0.71	0.52 0.4
e) 	8.45° 12.8mm	6.288° 14.6mm	1.19° 1.0 mm	0.92° 11.5 mm	0.63 0.88	0.10 0.22
f) 	10.3° 7.8 mm	33.3° 22 mm	6.01° 3.7 mm	23.0° 20.7 mm	0.80 0.56	-0.01 0.74

Similarly to the approach presented in chapter 4, the global pose of the object can be estimated, even when an initial estimate is not available. This is done by increasing the search space to all possible orientations and positions around the robot hand and increasing the maximum number of iterations. Given the size of the search space, it can take up to a minute to reach the global minimum. Also, due to object symmetry, multiple global minima can be present and the optimisation may converge to one that is not the global minimum. Figure 5.13 shows this global search for the grasp (f) in Table 5.3, with the arbitrarily selected initial estimate shown in red and the result of pose estimation in green. These two results are local minima found by the optimisation algorithm. These two poses, when evaluated using the cost function that depends only on distance, yield very similar costs (0.0177 and 0.0170). When using the covariance-based cost function presented in this chapter, it is possible to disambiguate between them, as the correct pose results in a much lower cost (0.254 and 0.103 respectively).

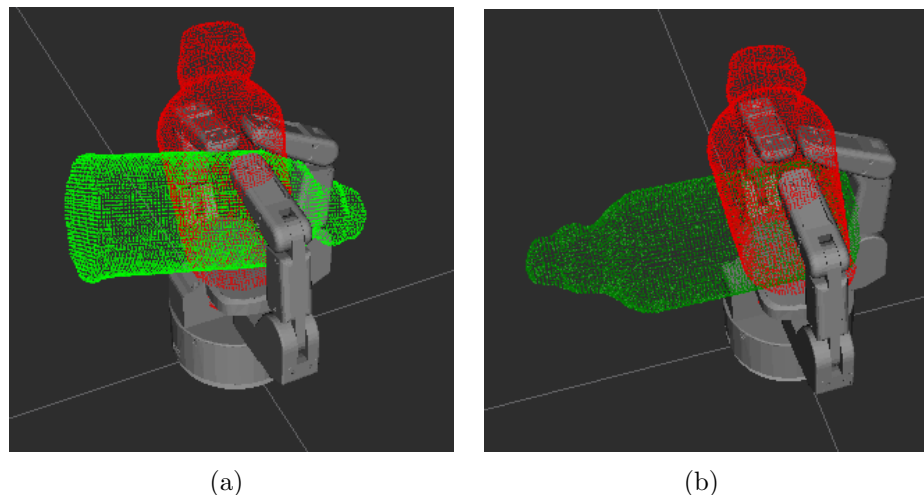


FIGURE 5.13: Result of a global search using the proposed descriptor (grasp (f) in Table 5.3) – red pointcloud: initial pose; green pointcloud: resultant pose.

The computational performance was very similar for both algorithms, with the average duration for pose correction being 9.06 seconds for the covariance-based algorithm and 9.68 seconds for the distance minimisation. The duration for the worst case timed during the experiments was similar for both algorithms, with the proposed cost function being between 1.4 times slower to 4 times faster, with an average duration of each evaluation of $500\mu s$. The discrepancies in the duration

of the proposed algorithm are due to the resolution of the object point cloud, since the calculations take longer when using point clouds with higher number of points, whereas the benchmark algorithm performs a fixed number of distance queries. However, it should be noted that the algorithm in Algorithm 5.2 can potentially run much faster, as it relies on matrix operations. Besides, the kd -tree implementation used in these experiments does not allow for range queries. Using this type of query would speed-up significantly the selection of points within the sensing region (*i.e.* steps 5 to 7 in Algorithm 5.2), which is one of the main bottlenecks in the performance of the algorithm.

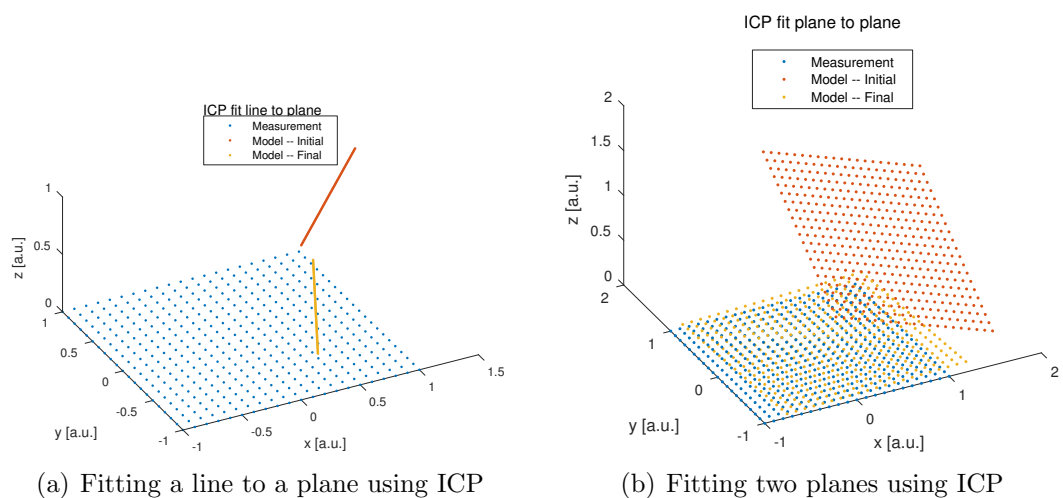
5.4 Conclusions

This chapter presented a new descriptor to interpret data from a distributed tactile array, which was applied to the problem of estimating the pose of a grasped object. The approach is based on Principal Component Analysis, a method that obtains a set of vectors in the direction of highest variance in the data, orthogonal between them. While this approach has been extensively used in many branches of science, and robotics in particular, the common usage of PCA is with the aim of reducing the dimensionality of multivariate data, with many notable examples of using PCA to process tactile data [77, 89, 183, 184]. This method, on the other hand, follows the approach by Liu *et al.* [85], preserving all the dimensions and is instead used to match tactile and geometric data.

The results of pose estimation are presented both qualitatively and quantitatively, showing higher accuracy than current standard techniques for this purpose, which merely rely on the distance between finger contacts and object surface. One paradigmatic example for this approach is the Iterative Closest Point [185, 186], which was used for benchmark. When comparing both approaches, ICP presents faster convergence but has a number of disadvantages, such as the fact that it converges to a local minimum. Another problem when using ICP, is that, for example, a sharp edge can be confused for a plane that contains that edge, as the

distance between active sensor elements and one point on the point cloud can be very small.

Figure 5.14 shows an example where a ICP was performed to fit a line to plane and a plane to another plane. This situation tries to emulate the active elements in a tactile sensor (blue markers), and a patch of the geometric shape of an object. Since PCA merely tries to minimise the distance between the points, a line will fit just as well as a plane to another plane. In the line to plane fitting the resulting cost of the algorithm was found to be lower (0.0408) than the error of fitting a plane to another plane (0.0747). The proposed method, on the other hand, penalises both cases where points in the measurement are not present in the object model and also where surface points inside the sensing area are not present in the measurement. This observation underlines some of the problems of using approaches similar to ICP and highlights the advantages of using the covariance based method to match geometric and tactile data.



(a) Fitting a line to a plane using ICP

(b) Fitting two planes using ICP

FIGURE 5.14: Results of applying ICP to fit a line to a plane and two planes.

Chapter 6

Conclusions

Chapter Summary

This chapter presents the final remarks to this thesis. It reviews and comments the main findings of this work, comparing it to other results in the literature. It also contains a critique of the proposed method and the limitations of the work. Suggestions and preliminary results for possible applications of the methods are proposed, along with possible future research directions.

6.1 Main Contributions

The main outcome of this thesis is the development of methods to estimate the pose of a grasped object. These methods are based on a single instance of tactile data and rely on optimisation to find a suitable pose. Two types of contact sensing data were used: intrinsic tactile sensing and distributed tactile arrays. The different information that is obtained from each of these sensing methods required the development of cost functions that exploited that information.

For intrinsic tactile sensing, the contact locations and normals were used, taking advantage of the fact that, for rigid contacts, the normal component of the contact force coincides with the normal of the object surface. For distributed tactile arrays, a descriptor that encoded the variances on the tactile data was created. Assuming that the local geometry of the object at the contact location will follow similar variances, a measure of this coherence allowed to find object poses which, for each contact, matched the tactile information to the local geometry of the object at that pose.

The proposed methods can be applied in two scenarios. If a coarse pose is available from vision, the pose can be quickly rectified to match tactile data. Also, when no initial pose is available, the global pose can be estimated by searching the whole space of rotations and translations around the robot hand.

Table 6.1 compares the literature results with the results from each results chapter in this thesis. While the accuracy and speed of the results are comparable to the existing literature, a noticeable feature of the proposed approach is its ability to estimate the 6-DOF pose of objects of arbitrary geometrical complexity, while requiring no exploration of the object. Preliminary results on the proposed methods' ability to identify the grasped object and assess grasp quality are presented in Section 6.2. A discussion and critical analysis on the methods' limitations is presented in Section 6.3, with a number of possible improvements to the method being suggested in Section 6.4.

Reference	6-DOF	Complex Objects	Global Pose	No Explor.	Object Ident.	Moving Object	Accuracy [mm/°]	Speed ^a [s]
Aggarwal[90]	✓	✓	✓	✗	✓	✓	<5/-	-
Chalon[131]	✓	✗	✗	✗	✗	✓	≈10/10	-
Corcoran[129]	✗	✗	✗	✗	✗	✗	≈3/8	3
Hebert[128]	✓	✗	✗	✓	✗	✓	6/2	-
Honda[124]	✓	✗	✗	✓	✗	✓	0.5/2	0.033
Petrovskaya[127]	✓	✗	✓	✗	✗	✓	5/3	30
Koval[132]	✗	✗	✓	✗	✗	✓	<20/-	1
Laaksonen[135]	✗	✗	✗	✗	✗	✗	-	-
Zito[133]	✓	✓	✗	✗	✗	✗	-	200
Pezzementi[137]	✓	✓	✓	✗	✓	✗	4/≈30	-
Chapter 3	✓	✓	✗	✓	✗	✗	20/ -	0.17
Chapter 4 (local)	✓	✓	✗	✓	✗	✗	4/5	0.64
Chapter 4 (global)	✓	✓	✓	✓	✓ ^b	✗	7/3	63
Chapter 5 (local)	✓	✓	✗	✓	✗	✗	7.6/10.4	9.7
Chapter 5 (global)	✓	✓	✓	✓	✓ ^b	✗	- / -	60

^aRun times are merely indicative, as the experiments were run in different hardware

^bPreliminary results

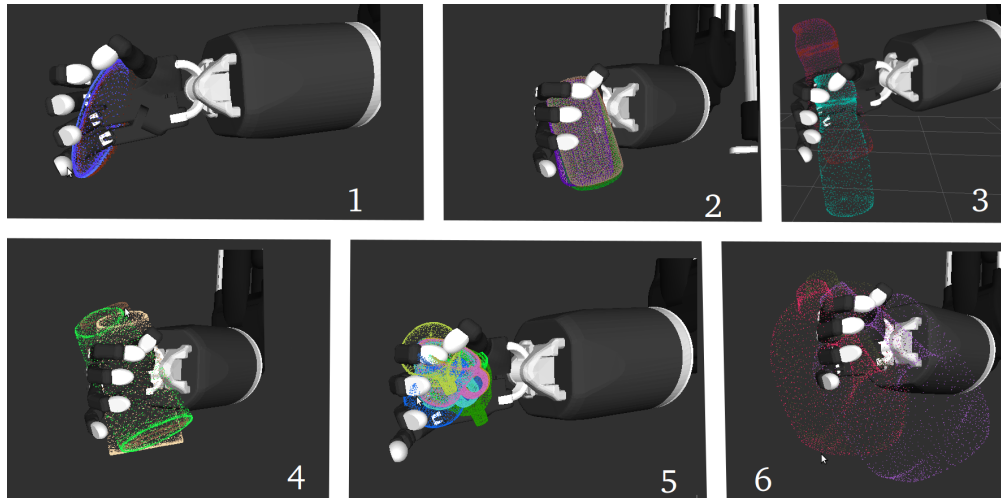
TABLE 6.1: Comparison of existing tactile pose estimation methods

6.2 Applications

6.2.1 Object Identification

One possible application for the global pose estimation algorithms is to identify the grasped object from within a number of possible objects in a database. This can be achieved through running multiple instances of the pose estimation algorithm in parallel. The final costs for each object can be ranked, with the lowest cost corresponding to the most likely object. Figure 6.1 shows preliminary results obtained from running the algorithm, using the grasp in Figure 6.1(b). The best poses for each possible objects are shown in Figure 6.1(a), and the likelihood is calculated according to the inverse of the cost: $l_i = \frac{1/G_i}{\sum_k 1/G_k}$. It can be seen that the object is correctly identified (object 2), with the glass (object 4) resulting also in a low cost, due to its similar shape and size.

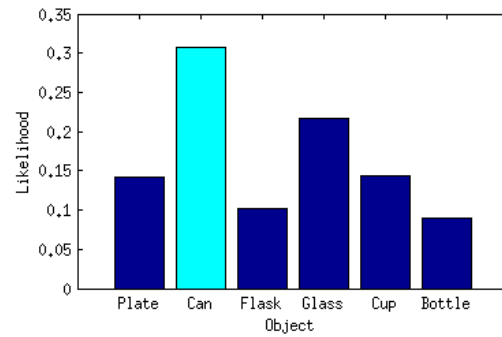
These results, while still preliminary, show a possible application of the method, but would profit greatly from the implementation of an exploratory strategy to disambiguate between objects with similar geometry.



(a)



(b)



(c)

FIGURE 6.1: Object identification using the proposed method

6.2.2 Grasp Stability

Understanding the pose of a grasped object is also of critical importance to assess the stability of a grasp. In order to apply the grasp stability criteria presented in Section 2.1, the location of the object's centre of mass must be accurately known. Figure 6.2 shows a Shadow robot hand grasping a light bulb. Contact forces are shown as green arrows and the normal directions are shown as red arrows. The convex hull, according to the Grasp Wrench Space, previously presented in Section 2.1, is shown in Figure 2.1(b), where green represents the forces and red represent the torques that this grasp can resist.

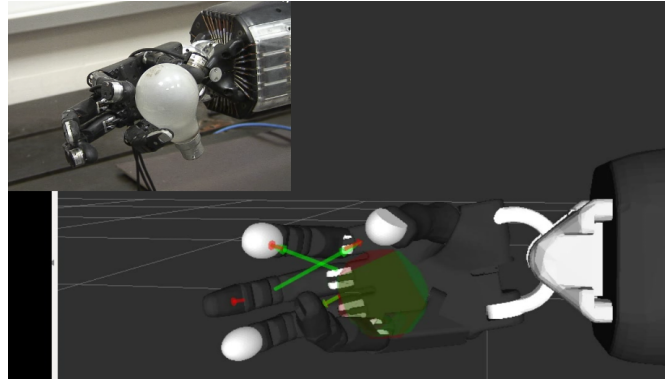


FIGURE 6.2: Grasp quality according to the Grasp Wrench Space metric.

This knowledge of the object pose can also allow the improvement of a grasp, by choosing points in the object where the placement of a finger would improve grasp quality. Figure 6.3 shows a robot hand grasping a glass. Points in the object are shown in a scale from red to blue, where quality would improve or decrease if a finger was moved to that position.

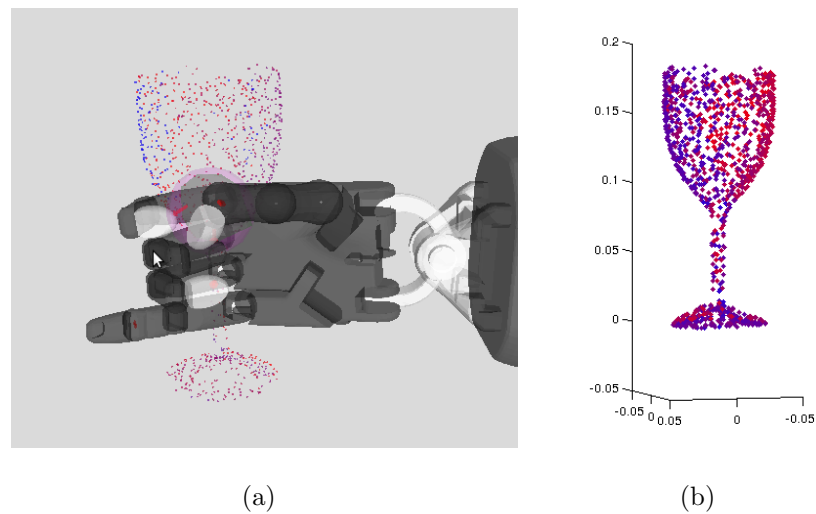


FIGURE 6.3: Grasp quality improvement. Blue – Improve quality Red – Reduce quality

6.3 Discussion and Critique

While the proposed methods can quickly and accurately estimate the pose of a grasped object when an initial estimate is available, globally estimating the pose

of the object without any prior estimate suffers from a number of issues.

Firstly, the computation time of the global pose estimate is excessively high. Secondly, ambiguities in the object pose can arise due, for example, to symmetries in the object. Also, as the geometric complexity of the object increases, more object poses are likely to satisfy current sensing data, in particular when few fingers are touching the object. Figure 6.4 present one such case, where two poses (blue and green) result in similar costs (blue and green). The true pose is shown in pink.

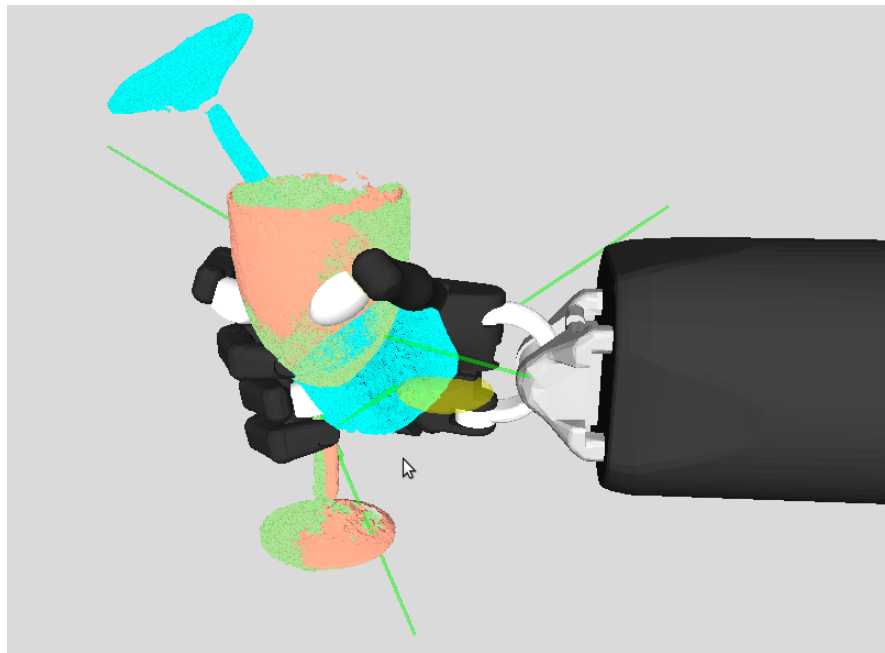


FIGURE 6.4: Example of a situation that might cause the algorithm to fail. Parts of the object surface coincide almost perfectly in two different poses. Ground truth is shown in pink.

Without an exploratory strategy, it is impossible to discriminate between these two poses. However, finger exploration requires the manipulation of the object, which will necessarily change the object pose. Pose changes due to small finger movements can be accommodated using this strategy, by running the minimisation continuously, using the previous estimation as the new initial estimate, or using prediction models such as the grasp matrix. Realistic dynamic manipulation situations, however, require an accurate and computationally fast update model, which can predict the state transitions from each possible pose.

Furthermore, while exploring an object to estimate its pose, it is essential to take into consideration the effects of a possible finger action in the stability of a grasp. This duality between exploration of the object and its stability is key to solving the problem of object pose estimation and the larger problem of object manipulation.

As mentioned in the Introduction of this thesis, this work does not presume to be a definitive solution to the problem of object pose estimation, but can be part of a larger system which comprises sensing, prediction, and decision-making. Figure 6.5 presents a possible system architecture that can estimate the pose of the object and take manipulative actions while keeping track of the object pose. This architecture can rely on Bayesian inference to sequentially estimate the pose of the object, given the available sensing data and a prediction on the object movement given a finger movement action.

The yellow rectangle highlights the parts of this system that are the object of this thesis. The output of the algorithms presented in this thesis can be a pool of hypothesis poses, which can be further refined through Bayesian filtering.

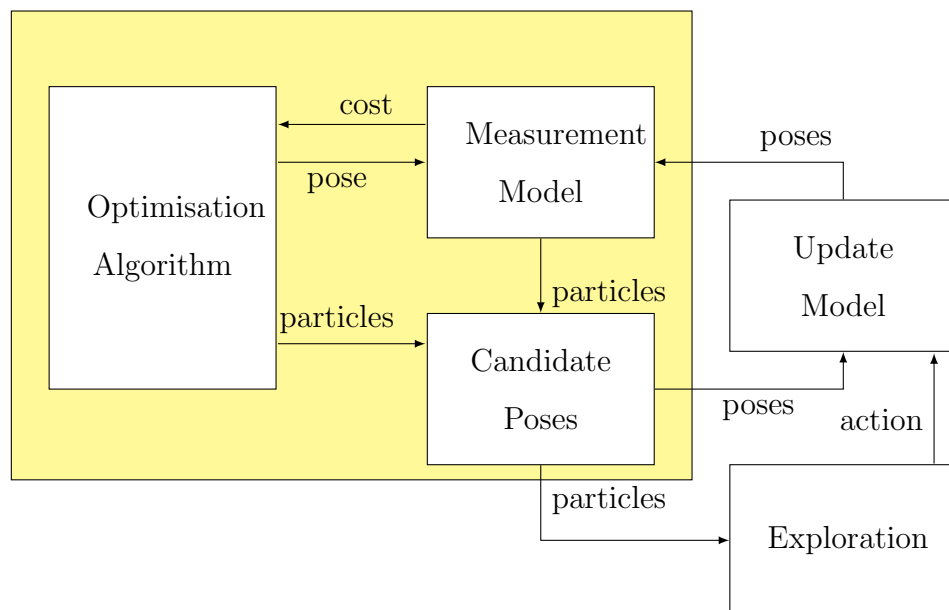


FIGURE 6.5: Proposed system for pose estimation.

An exploration strategy that can balance between the acquisition of information and the maintaining of grasp stability is still an open problem. While there is

significant work in planning trajectories that can maximise the information acquisition [133, 187, 188], considerations regarding grasp stability have not yet been included in these strategies. Also, creating an update model, *i.e.* a model that, given a starting object pose and an action, can predict the movement of the object is also of the utmost importance for robot manipulation and the estimation of the pose. Analytic models, such as the grasp matrix, simulation-based [189, 190], quasi-static analysis [191] or machine learning based methods [192] have been developed, but are yet to become commonplace within the robot grasping and manipulation community.

In summary, a system that is able to accurately track the pose of a grasped object while it is being manipulated is yet to be accomplished. A possible architecture for such a system would require the different modules shown in Figure 6.5 to be integrated together and carry out the estimation using a Bayesian framework.

6.4 Future Work

The previous section proposed an overview of a system architecture that can estimate the pose of a grasped object through Bayesian recursive estimation. Also, the applications presented in Section 6.2 present some possible applications that can be further developed into more mature algorithms. This section suggests practical improvements to the methods developed in this thesis, focussing mostly on the performance of the algorithms.

First, a parametrisation of quaternions that could reduce the search space to 6 dimensions, taking advantage of the fact that the quaternion must have unit norm should be further investigated. While the strategies presented by Ude [143] and Schmidt [145] were tested, the computational advantages were not evident. However, finding an efficient implementation of this type of parametrisation can bring about great improvements in the computational cost of the algorithm. The number of dimensions can also be further reduced by exploiting the symmetries of

the object. For example, if the object is a sphere, the search space can be reduced to the 3 translational dimensions.

Parallelisation of the search algorithm in Section 4.2 can also greatly decrease the computational time to search for solutions. To implement this solution a great deal of care must be given to avoid race conditions and also ensure the correct re-sampling of particles. A possible solution for this would be to re-sample and evaluate in batches instead of sequentially, adding the weights and the particles for the whole batch simultaneously.

For the descriptor presented in chapter 5, a k -d tree implementation that allowed range searches would greatly reduce the computational time to evaluate the cost function. Finally, this representation of tactile data could be used in other contexts, such as tactile exploration and grasp stability analysis.

Bibliography

- [1] J. C. Rothwell, M. M. Traub, B. L. Day, J. A. Obeso, P. K. Thomas, and C. D. Marsden. MANUAL MOTOR PERFORMANCE IN A DEAFFERENTED MAN. *Brain*, 105(3):515–542, 1982. ISSN 0006-8950. .
- [2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 45. The MIT Press, Sep 2002. ISBN 0262201623. .
- [3] Joao Bimbo, Silvia Rodriguez-Jimenez, Hongbin Liu, Xiaojing Song, Nicolas Burrus, Lakmal D. Seneviratne, Mohamed Abderrahim, Kaspar Althoefer, Lakmal D. Senerivatne, Mohamed Abderrahim, and Kaspar Althoefer. Object pose estimation and tracking by fusing visual and tactile information. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 65–70. IEEE, IEEE, Sep 2012. ISBN 9781467325110. .
- [4] Joao Bimbo, Lakmal D. Seneviratne, Kaspar Althoefer, Hongbin Liu, and Hongbin Liu. Combining touch and vision for the estimation of an object’s pose during manipulation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 4021–4026. IEEE, IEEE, Nov 2013. ISBN 978-1-4673-6358-7. .
- [5] Joao Bimbo, Petar Kormushev, Kaspar Althoefer, and Hongbin Liu. Global estimation of an object’s pose using tactile sensing. *Advanced Robotics*, 29(5):363–374, Mar 2015. ISSN 0169-1864. .

-
- [6] J. Bimbo, S. Luo, K. Althoefer, and H. Liu. In-hand object pose estimation using covariance-based tactile to geometry matching. *IEEE Robotics and Automation Letters*, 1(1):570–577, Jan 2016. ISSN 2377-3766. .
- [7] A.M. Okamura, N. Smaby, and Mark R. Cutkosky. An overview of dexterous manipulation. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 1, 2000. ISSN 1050-4729. .
- [8] R. Tomovic and G. Boni. An adaptive artificial hand. *IRE Transactions on Automatic Control*, 7(3), 1962. ISSN 0096-199X. .
- [9] Hideo Hanafusa and Haruhiko Asada. Stable prehension by a robot hand with elastic fingers. In *International Symposium on Industrial Robots*, pages 361–368, Tokyo, 1977.
- [10] J. Kenneth Salisbury and J. J. Craig. Articulated Hands: Force Control and Kinematic Issues. *The International Journal of Robotics Research*, 1(1): 4–17, Mar 1982. ISSN 0278-3649. .
- [11] M. Cutkosky and P. Wright. Modeling manufacturing grips and correlations with the design of robotic hands. *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, 3:1533–1539, 1986. .
- [12] Mark R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989. ISSN 1042296X. .
- [13] Matthew T Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, USA, 2001. ISBN 1101200529205. .
- [14] R.S. Fearing. Simplified Grasping and Manipulation with Dexterous Robot Hands. *IEEE Journal on Robotics and Automation*, 2(4):188–195, Dec 1984. ISSN 0882-4967. .

-
- [15] Antonio Bicchi. On the Closure Properties of Robotic Grasping. *The International Journal of Robotics Research*, 14(4):319–334, Aug 1995. ISSN 0278-3649. .
- [16] V.-D. Van-duc Nguyen. Constructing Force- Closure Grasps. *The International Journal of Robotics Research*, 7(3):3–16, Jun 1988. ISSN 0278-3649. .
- [17] Carlo Ferrari and John Canny. Planning optimal grasps. *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295, 1992. .
- [18] Nancy S. Pollard. Synthesizing grasps from generalized prototypes. In *Proceedings of IEEE International Conference on Robotics and Automation*, number April, pages 2124–2130, 1996. ISBN 0-7803-2988-0. .
- [19] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev. On the efficient computation of independent contact regions for force closure grasps. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 586–591, 2010. ISSN 2153-0858. .
- [20] G I Boutselis, C P Bechlioulis, M V Liarokapis, and K J Kyriakopoulos. Task specific robust grasping for multifingered robot hands. *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 858–863, 2014. ISSN 21530866. .
- [21] Shuo Liu and Stefano Carpin. A Fast Algorithm for Grasp Quality Evaluation Using the Object Wrench Space. In *IEEE International Conference on Automation Science and Engineering*, Goteborg, 2015. IEEE.
- [22] S. Ekvall and Danica Kragic. Interactive grasp learning based on human demonstration. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 4, 2004. ISSN 1050-4729. .

- [23] Robert Krug and Dimitar Dimitrov. Representing movement primitives as implicit dynamical systems learned from multiple demonstrations. *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–8, 2013. .
- [24] Eric L. Sauser, Brenna D. Argall, Giorgio Metta, and Aude G. Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 60(1):55–71, 2012. ISSN 09218890. .
- [25] M Santello, M Flanders, and J F Soechting. Postural hand synergies for tool use. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(23):10105–10115, Dec 1998. ISSN 0270-6474. .
- [26] I.T. Jolliffe. *Principal Component Analysis*. Springer Science & Business Media, 2002. ISBN 0387954422.
- [27] Alexandre Bernardino, Marco Henriques, Norman Hendrich, and Jianwei Zhang. Precision grasp synergies for dexterous robotic hands. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 62–67. IEEE, Dec 2013. ISBN 978-1-4799-2744-9. .
- [28] Giuseppe Cotugno, Vishawanathan Mohan, Kaspar Althoefer, and Thirishantha Nanayakkara. Simplifying Grasping Complexity through Generalization of Kinaesthetically Learned Synergies. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5345–5351. IEEE, May 2014. ISBN 9781479936847. .
- [29] Marek S. Kopicki, Renaud Detry, Maxime Adjigble, Rustam Stolkin, Ales Leonardis, and Jeremy L. Wyatt. One shot learning and generation of dexterous grasps for novel objects. *International Journal of Robotics Research*, Sep 2015. ISSN 0278-3649. .
- [30] Yasemin Bekiroglu, Janne Laaksonen, Jimmy Alison Jorgensen, Ville Kyrki, and Danica Kragic. Assessing Grasp Stability Based on Learning and Haptic Data. *IEEE Transactions on Robotics*, 27(3):616–629, Jun 2011. ISSN 1552-3098. .

- [31] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2470–2476. IEEE, 1999. ISBN 0-7803-2988-0. .
- [32] Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 2015. ISSN 09218890. .
- [33] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic Grasping of Novel Objects using Vision. *The International Journal of Robotics Research*, 27(2):157–173, Feb 2008. ISSN 0278-3649. .
- [34] Quoc V. Le, David Kamm, Arda F. Kara, and Andrew Y. Ng. Learning to grasp objects with multiple contact points. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5062–5069, 2010. ISSN 10504729. .
- [35] Catherine L. Reed, Richard J. Caselli, and Martha J. Farah. Tactile agnosia. Underlying impairment and implications for normal tactile object recognition. *Brain*, 119(3):875–888, Jun 1996. ISSN 00068950. .
- [36] Roberta L. Klatzky, Susan J. Lederman, and V a Metzger. Identifying objects by touch: an "expert system". *Perception & psychophysics*, 37(4):299–302, Apr 1985. ISSN 0031-5117.
- [37] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer Berlin Heidelberg, Berlin, Heidelberg, Nov 2008. ISBN 978-3-540-23957-4. .
- [38] Roland S. Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature reviews. Neuroscience*, 10(5):345–359, May 2009. ISSN 1471-003X. .
- [39] Zhongkui Wang, Lijuan Wang, Van Anh Ho, Shigehiro Morikawa, and Shinichi Hirai. A 3-D nonhomogeneous FE model of human fingertip based

- on MRI measurements. *IEEE Transactions on Instrumentation and Measurement*, 61(12):3147–3157, 2012. ISSN 00189456. .
- [40] Robert D. Howe. Tactile sensing and control of robotic manipulation. *Advanced Robotics*, 8(3):245–261, Jan 1993. ISSN 0169-1864. .
- [41] Hongbin Liu, Xiaojing Song, Joao Bimbo, Lakmal D. Seneviratne, and Kaspar Althoefer. Surface material recognition through haptic exploration using an intelligent contact sensing finger. In *IEEE International Conference on Intelligent Robots and Systems*, pages 52–57. IEEE, 2012. ISBN 9781467317375. .
- [42] Damith Suresh Chathuranga, Van Anh Ho, and Shinichi Hirai. A biomimetic fingertip that detects force and vibration modalities and its application to surface identification. *2012 IEEE International Conference on Robotics and Biomimetics, ROBOT 2012 - Conference Digest*, pages 575–581, 2012. .
- [43] Mark R. Cutkosky and J M Hyde. Manipulation Control with Dynamic Tactile Sensing. *International Symposium on Robotics Research*, 8:245–261, 1993.
- [44] J.S. Son, E.a. Monteverde, and Robert D. Howe. A tactile sensor for localizing transient events in manipulation. *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994. ISSN 10504729. .
- [45] Xiaojing Song, Hongbin Liu, Kaspar Althoefer, Thrishantha Nanayakkara, and Lakmal D. Seneviratne. Efficient break-away friction ratio and slip prediction based on haptic surface exploration. *IEEE Transactions on Robotics*, 30(1):203–219, 2014. ISSN 15523098. .
- [46] P Dario and D De Rossi. Tactile sensors and the gripping challenge. *IEEE Spectrum*, 22(5):46–53, 1985. ISSN 00189235.

- [47] H. R. Nicholls and M. H. Lee. A Survey of Robot Tactile Sensing Technology. *The International Journal of Robotics Research*, 8(October):3–30, Jun 1989. ISSN 0278-3649. .
- [48] Robert D. Howe, N. Popp, P. Akella, I. Kao, and Mark R. Cutkosky. Grasping, manipulation, and control with tactile sensing. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1258–1263. IEEE Comput. Soc. Press, 1990. ISBN 0-8186-9061-5. .
- [49] Ravinder S. Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile sensing-from humans to humanoids. *IEEE Transactions on Robotics*, 26(1):1–20, Feb 2010. ISSN 15523098. .
- [50] Jack M Loomis and Susan J Lederman. Tactual perception. *Handbook of perception and human performances*, 2(2):2, 1986. ISSN 0004-8402. .
- [51] Ravinder S. Dahiya and Maurizio Valle. *Robotic Tactile Sensing*. Springer Netherlands, Dordrecht, 2013. ISBN 978-94-007-0578-4. .
- [52] Antonio Bicchi, J. Kenneth Salisbury, and David L. Brock. Contact Sensing from Force Measurements. *The International Journal of Robotics Research*, 12(3):249–262, Jun 1993. ISSN 0278-3649. .
- [53] Antonio Bicchi, J. Kenneth Salisbury, and P. Dario. Augmentation of grasp robustness using intrinsic tactile sensing. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 302–307. IEEE Comput. Soc. Press, 1989. ISBN 0-8186-1938-4. .
- [54] Hongbin Liu, Xiaojing Song, Joao Bimbo, and Kaspar Althoefer. Intelligent Fingertip Sensing for Contact Information Identification. In *Proc. of the Second ASME/IEEE International Conference on Reconfigurable Mechanisms and Robots (ReMAR 2012)*, 2012.
- [55] K Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

- [56] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Non-linear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, Jun 1963. ISSN 0368-4245. .
- [57] Xiaojing Song, Hongbin Liu, Joao Bimbo, Kaspar Althoefer, and Lakmal Senerivatne. A Novel Dynamic Slip Prediction and Compensation Approach Based on Haptic Surface Exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [58] Junghwan Back, Yohan Noh, Lakmal Seneviratne, Kaspar Althoefer, Hongbin Liu, Joao Bimbo, Yohan Noh, Lakmal Seneviratne, Kaspar Althoefer, and Hongbin Liu. Control a contact sensing finger for surface haptic exploration. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2736–2741. IEEE, May 2014. ISBN 978-1-4799-3685-4. .
- [59] Hongbin Liu, Kien Cuong Nguyen, Véronique Perdereau, Joao Bimbo, Junghwan Back, Matthew Godden, Lakmal D. Seneviratne, and Kaspar Althoefer. Finger contact sensing and the application in dexterous hand manipulation. *Autonomous Robots*, 39(1):25–41, Jan 2015. ISSN 0929-5593. .
- [60] Hongbin Liu, Xiaojing Song, Joao Bimbo, Kaspar Althoefer, and Lakmal Senerivatne. Intelligent fingertip sensing for contact information identification. pages 599–608, 2012.
- [61] Xiaojing Song, Hongbin Liu, Joao Bimbo, Kaspar Althoefer, and Lakmal Senerivatne. Object surface classification based on friction properties for intelligent robotic hands. In *Proc. of World Automation Congress (WAC), 2012*, 2012.
- [62] Zhanat Kappasov, Juan-Antonio Corrales, and Véronique Perdereau. Tactile sensing in dexterous robot hands — Review. *Robotics and Autonomous Systems*, 74:195–220, Dec 2015. ISSN 09218890. .

- [63] Van Anh Ho and Shinichi Hirai. Understanding slip perception of soft fingertips by modeling and simulating stick-slip phenomenon. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.
- [64] Damith Suresh Chathuranga, Zhongkui Wang, and Shinichi Hirai. An anthropomorphic tactile sensor system with its applications in dexterous manipulations. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1085–1090. IEEE, Jun 2015. ISBN 978-1-4799-8728-3. .
- [65] Van Anh Ho, Dzung Viet Dao, Susumu Sugiyama, and Shinichi Hirai. Development and Analysis of a Sliding Tactile Soft Fingertip Embedded With a Microforce/Moment Sensor. *IEEE Transactions on Robotics*, 27(3):411–424, 2011. ISSN 1552-3098. .
- [66] PPS. <http://www.pressureprofile.com/>.
- [67] Inc Tekscan. URL: <http://www.tekscan.com/>.
- [68] Weiss Robotics. URL: <http://www.weiss-robotics.de>.
- [69] Takktile. URL: <http://www.takktile.com/>.
- [70] Aude Billard, Annalisa Bonfiglio, Giorgio Cannata, Piero Cosseddu, Torbjorn Dahl, Kerstin Dautenhahn, Fulvio Mastrogiovanni, Giorgio Metta, Lorenzo Natale, Ben Robins, and Others. The ROBOSKIN Project: Challenges and Results. *Romansy 19–Robot Design, Dynamics and Control*, (October 2015):351–358, 2013.
- [71] G. De Maria, Ciro Natale, and Salvatore Pirozzi. Tactile data modeling and interpretation for stable grasping and manipulation. *Robotics and Autonomous Systems*, 61(9):1008–1020, 2013. ISSN 09218890. .
- [72] Craig Chorley, Chris Melhuish, Tony Pipe, and Jonathan Rossiter. Development of a Tactile Sensor Based on Biologically Inspired Edge Encoding. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6, 2009. ISBN 978-1-4244-4855-5.

- [73] SynTouch. URL: <http://www.syntouchllc.com>.
- [74] Alberto D'Amore, Giuseppe De Maria, Luigi Grassia, Ciro Natale, Salvatore Pirozzi, and Alberto D Amore. Silicone-rubber-based tactile sensors for the measurement of normal and tangential components of the contact force. *Journal of Applied Polymer Science*, 122(2011):3758–3770, 2011. .
- [75] H. Yousef, J. P. Nikolovski, and E. Martincic. Flexible 3D force tactile sensor for artificial skin for anthropomorphic robotic hand. *Procedia Engineering*, 25:128–131, 2011. ISSN 18777058. .
- [76] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics - A review. *Sensors and Actuators, A: Physical*, 167(2):171–187, 2011. ISSN 09244247. .
- [77] Hongbin Liu, Juan Greco, Xiaojing Song, Joao Bimbo, Lakmal Senerivatne, Kaspar Althoefer, and Lakmal Seneviratne. Tactile Image based Contact Shape Recognition using Neural Network. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 138–143. IEEE, Sep 2012. ISBN 978-1-4673-2512-7. .
- [78] Shan Luo, Wenxuan Mou, Kaspar Althoefer, and Hongbin Liu. Novel Tactile-SIFT Descriptor for Object Shape Recognition. *IEEE Sensors Journal*, 15(9):5001–5009, Sep 2015. ISSN 1530-437X. .
- [79] Giorgio Cannata, Simone Denei, and Fulvio Mastrogiovanni. A framework for representing interaction tasks based on tactile data. In *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pages 698–703. IEEE, Sep 2010. ISBN 9781424479917. .
- [80] Marianna Madry, Liefeng Bo, Danica Kragic, and Dieter Fox. ST-HMP: Unsupervised Spatio-Temporal feature learning for tactile data. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269. IEEE, May 2014. ISBN 978-1-4799-3685-4. .

- [81] G. Heidemann and M. Schopfer. Dynamic tactile sensing for object identification. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 813–818. IEEE, 2004. ISBN 0-7803-8232-3. .
- [82] D. Goger, Nicolas Gorges, and H. Worn. Tactile sensing for an anthropomorphic robotic hand: Hardware and signal processing. In *2009 IEEE International Conference on Robotics and Automation*, pages 895–901. IEEE, May 2009. ISBN 978-1-4244-2788-8. .
- [83] Zachary Pezzementi, Erion Plaku, Caitlin Reyda, and Gregory D. Hager. Tactile-object recognition from appearance information. *IEEE Transactions on Robotics*, 27(3):473–487, Jun 2011. ISSN 15523098. .
- [84] Yevgen Chebotar, Oliver Kroemer, and Jan Peters. Learning robot tactile sensing for object manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3368–3375. IEEE, Sep 2014. ISBN 978-1-4799-6934-0. .
- [85] Hongbin Liu, Xiaojing Song, Thrishantha Nanayakkara, Lakmal D. Seneviratne, and Kaspar Althoefer. A computationally fast algorithm for local contact shape and pose classification using a tactile array sensor. *IEEE International Conference on Robotics and Automation*, pages 1410–1415, May 2012. ISSN 10504729. .
- [86] Joseph M. Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katherine J. Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079, 2011. ISSN 15523098. .
- [87] Roberto Calandra, Serena Ivaldi, Marc Peter Deisenroth, Elmar Rueckert, and Jan Peters. Learning inverse dynamics models with contacts. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3186–3191. IEEE, May 2015. ISBN 978-1-4799-6923-4. .
- [88] Benjamin Winstone, Gareth Griffiths, Tony Pipe, Chris Melhuish, and Jonathon Rossiter. TACTIP - Tactile Fingertip Device, Texture Analysis

- through Optical Tracking of Skin Features. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8064 LNAI, pages 323–334. 2013. ISBN 9783642398018. .
- [89] Zachary Pezzementi, Caitlin Reyda, and Gregory D. Hager. Object mapping, recognition, and localization from tactile geometry. In *IEEE International Conference on Robotics and Automation*, pages 5942–5948. IEEE, May 2011. ISBN 9781612843865. .
- [90] Achint Aggarwal and Frank Kirchner. Object recognition and localization: The role of tactile sensors. *Sensors (Switzerland)*, 14(2):3227–3266, Jan 2014. ISSN 14248220. .
- [91] Ren C. Luo. Guest Editorial. *IEEE Transactions on Industrial Electronics*, 43(3), 1996.
- [92] Ren C. Luo, Ying Chih Chou Ying Chih Chou, and O. Chen. Multisensor Fusion and Integration: Algorithms, Applications, and Future Research Directions. In *2007 International Conference on Mechatronics and Automation*, pages 1986–1991. IEEE, Aug 2007. ISBN 978-1-4244-0828-3. .
- [93] J.S. Son, Robert D. Howe, J. Wang, and Gregory D. Hager. Preliminary results on grasping with vision and touch. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, volume 3, pages 1068–1075, 1996. ISBN 0-7803-3213-X. .
- [94] P Jenmalm and R S Johansson. Visual and Somatosensory Information About Object Shape Control Manipulative Fingertip Forces. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, 17(11):4486–4499, 1997. ISSN 0270-6474.
- [95] Peter K. Allen and R Bajcsy. *Robotic Object Recognition Using Vision and Touch*, volume 34 of *The Kluwer International Series in Engineering and Computer Science*. Springer US, Boston, MA, 1987. ISBN 978-1-4612-9196-1. .

- [96] Peter K. Allen, Andrew T Miller, P.Y. Oh, Brian S Leibowitz, E Peter, Allen Andrew, T Miller Paul, and Y Oh Brian. Using tactile and visual sensing with a robotic hand. In *Proc. of 1997 IEEE International Conference on Robotics and Automation*, volume 1, pages 676–681. IEEE, 1997. ISBN 0-7803-3612-7. .
- [97] Peter K. Allen, Andrew T Miller, Paul Y Oh, and Brian S Leibowitz. Integration of Vision , Force and Tactile Sensing for Grasping. *International Journal of Intelligent Machines*, 4:129–149, 1999.
- [98] A.T. Miller and Peter K. Allen. Graspit!: A Versatile Simulator for Robotic Grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. ISSN 1070-9932. .
- [99] Danica Kragic, Andrew T Miller, and Peter K. Allen. Realtime tracking meets online grasp planning. *Proceedings - IEEE International Conference on Robotics and Automation*, 3:2460–2465, 2001. ISSN 10504729. .
- [100] Mario Prats, Pedro J. Sanz, and Angel P. Del Pobil. Vision-tactile-force integration and robot physical interaction. In *2009 IEEE International Conference on Robotics and Automation*, pages 3975–3980, May 2009. ISBN 978-1-4244-2788-8. .
- [101] Yasemin Bekiroglu, Renaud Detry, and Danica Kragic. Learning tactile characterizations of object- and pose-specific grasps. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1554–1560. IEEE, Sep 2011. ISBN 9781612844541. .
- [102] Javier Felip, Antonio Morales, and Tamim Asfour. Multi-sensor and prediction fusion for contact detection and localization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 601–607. IEEE, Nov 2014. ISBN 978-1-4799-7174-9. .

- [103] Dov Katz, Jacqueline Kenney, and Oliver Brock. How can robots succeed in unstructured environments? In *Workshop on Robot Manipulation: Intelligence in Human Environments at Robotics: Science and Systems*, Zurich, 2008.
- [104] Katharina Hertkorn, Maximo a. Roa, and Christoph Borst. Planning in-hand object manipulation with multifingered hands considering task constraints. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 617–624, 2013. ISSN 10504729. .
- [105] Danica Kragic, Andrew T Miller, and Peter K. Allen. Realtime tracking meets on line grasping planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2460–2465, 2001. ISBN 0780364759. .
- [106] Danica Kragic, Mårten Björkman, Henrik I. Christensen, and Jan-Olof Eklundh. Vision for robotic object manipulation in domestic settings. *Robotics and Autonomous Systems*, 52(1):85–100, Jul 2005. ISSN 09218890. .
- [107] Pedram Azad, Tamim Asfour, and Ruediger Dillmann. Stereo-based 6D object localization for grasping with humanoid robot systems. In *IEEE International Conference on Intelligent Robots and Systems*, pages 919–924. IEEE, Oct 2007. ISBN 1424409128. .
- [108] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, Mar 2009. ISSN 10773142. .
- [109] Chia-Ming Cheng, Hsiao-Wei Chen, Tung-Ying Lee, Shang-Hong Lai, and Ya-Hui Tsai. Robust 3D object pose estimation from a single 2D image. *2011 Visual Communications and Image Processing (VCIP)*, pages 1–4, 2011. .
- [110] Renaud Detry, E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater. Learning object-specific grasp affordance densities. *2009 IEEE 8th International Conference on Development and Learning, ICDL 2009*, 2(1):1–17, 2009. ISSN 2080-9778. .

- [111] Nicolas Burrus, M Abderrahim, Jorge Garcia, and Luis Moreno. Object Reconstruction and Recognition leveraging an RGB-D camera. In *Proceedings of the 12th IAPR Conference on Machine Vision Applications*, volume c, pages 3–6, 2011. ISBN 9784901122115.
- [112] Anders Glent Buch, Dirk Kraft, Joni-kristian Kamarainen, Henrik Gordon Petersen, and Norbert Kruger. Pose estimation using local structure-specific shape and appearance context. In IEEE, editor, *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2080—2087. IEEE, 2013.
- [113] Rui Pimentel de Figueiredo, Plinio Moreno, and Alexandre Bernardino. Efficient pose estimation of rotationally symmetric objects. *Neurocomputing*, 150(2):126–135, Feb 2015. ISSN 09252312. .
- [114] Eric Brachmann, Alexander Krull, Frank Michel, and Stefan Gumhold. Learning 6D Object Pose Estimation using 3D Object Coordinates. *Eccv*, pages 1–16, 2014. .
- [115] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, May 2011. ISBN 978-1-61284-386-5. .
- [116] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking. *ACM Computing Surveys*, 38(4):13–es, 2006. ISSN 03600300. .
- [117] Achint Aggarwal and Peter Kampmann. Tactile sensors based object recognition and 6D pose estimation. In *International Conference on Intelligent Robotics and Applications*, pages 406–416, 2012. .
- [118] Joseph W. Starr and B. Y. Lattimer. Evaluation of Navigation Sensors in Fire Smoke Environments. *Fire Technology*, 50(6):1–23, Aug 2013. ISSN 00152684. .
- [119] S. Shekhar, Oussama Khatib, and M. Shimojo. Sensor fusion and object localization. *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, 3:1623– 1628, 1986. .

- [120] P C Gaston and T Lozano-Perez. Tactile recognition and localization using object models: the case of polyhedra on a plane. *IEEE transactions on pattern analysis and machine intelligence*, 6(3):257–266, 1984. ISSN 0162-8828. .
- [121] D.M. Siegel. Finding the pose of an object in a hand. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, (April):406–411, 1991. .
- [122] Ren Luo and Wen-Hsiang Tsai. Object recognition using tactile image array sensors. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1248–1253. Institute of Electrical and Electronics Engineers, 1986. .
- [123] J. Schneiter. An objective tactile sensing strategy for object recognition and localization. *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, pages 1262–1267, 1986. .
- [124] Kyuhei Honda, Tsutomu Hasegawa, Toshihiro Kiriki, and Takeshi Matsuoka. Real-time Pose Estimation of an Object Manipulated by Multi-fingered Hand Using 3D Stereo Vision and Tactile Sensing. In *Proceedings of the 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, volume 3, pages 1814–1819, 1998. ISBN 0780344650. .
- [125] S Haidacher and G Hirzinger. Estimating Finger Contact Location and Object Pose from Contact Measurements in 3-D Grasping Institute for Robotics and Mechatronics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1805–1810, 2003. ISBN 0780377362. .
- [126] Sebastian Thrun, Dieter Fox, Wolfram Burgard, Frank Dellaert, Sebastian Thrun, Dieter Fox, Dieter Fox, Wolfram Burgard, Wolfram Burgard, Frank Dellaert, and Frank Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, May 2001. ISSN 00043702. .

- [127] Anna Petrovskaya and Oussama Khatib. Global localization of objects via touch. *IEEE Transactions on Robotics*, 27(3):569–585, 2011. ISSN 15523098. .
- [128] Paul Hebert, Nicolas Hudson, Jeremy Ma, and Joel Burdick. Fusion of stereo vision, force-torque, and joint sensors for estimation of in-hand object location. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5935–5941, 2011. ISBN 9781612843865. .
- [129] Craig Corcoran and Robert Platt. A measurement model for tracking hand-object state during dexterous manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4302–4308, 2010. ISBN 978-1-4244-5038-1. .
- [130] Anna Petrovskaya, Oussama Khatib, Sebastian Thrun, and Andrew Y Ng. Touch Based Perception for Object Manipulation. In *Robotics Science and Systems, Robot Manipulation Workshop*, pages 2–7, 2007.
- [131] Maxime Chalon, Jens Reinecke, and Martin Pfanne. Online in-hand object localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2977–2984, Tokyo, 2013. ISBN 9781467363570.
- [132] Michael C. Koval, Mehmet R. Dogar, Nancy S. Pollard, and Siddhartha S. Srinivasa. Pose estimation for contact manipulation with manifold particle filters. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4541–4548, Tokyo, 2013. ISBN 9781467363587. .
- [133] Claudio Zito, Marek S. Kopicki, Rustam Stolkin, Christoph Borst, Florian Schmidt, Maximo a. Roa, and Jeremy L. Wyatt. Sequential trajectory replanning with tactile information gain for dexterous grasping under object-pose uncertainty. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4013–4020, Tokyo, 2013. ISBN 9781467363587. .

- [134] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Robust grasping under object pose uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 31, pages 253–268, Jul 2011. .
- [135] Jonna Laaksonen, Ekaterina Nikandrova, and Ville Kyrki. Probabilistic sensor-based grasping. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2019–2026. IEEE, Oct 2012. ISBN 978-1-4673-1736-8. .
- [136] Andres S. Vazquez, Raul Fernandez, Antonio Lopez, Enrique Valero, Ismael Payo, and Antonio Adan. In-hand object localization: Simple vs. complex tactile sensors. In *Proc. of IEEE SENSORS*, pages 1710–1713. IEEE, Nov 2014. ISBN 978-1-4799-0162-3. .
- [137] Zachary Pezzementi and Gregory D. Hager. Tactile Object Recognition and Localization Using Spatially-Varying Appearance. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.
- [138] José-Luis Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Sep 2010.
- [139] John J Craig. *Introduction to Robotics Mechanics and Control*. Prentice Hall, 3 edition, 2005. ISBN 0201182408. .
- [140] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*, volume 29. 1994. ISBN 9780849379819. .
- [141] Eugene Salamin. Application of quaternions to computation with rotations. Technical report, Stanford University Artificial Intelligence Laboratory, 1995.
- [142] Janez Funda, Russell H. R.H. Taylor, and Richard P. R.P. Paul. On homogeneous transforms, quaternions, and computational efficiency. *IEEE*

- Transactions on Robotics and Automation*, 6(3):382–388, Jun 1990. ISSN 1042296X. .
- [143] Ales Ude. Nonlinear least squares optimisation of unit quaternion functions for pose estimation from corresponding features. *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, 1(August):425–427, 1998. ISSN 1051-4651. .
- [144] Mark D. Wheeler and Katsushi Ikeuchi. Iterative Estimation of Rotation and Translation using the Quaternion. Technical report, 1995.
- [145] Jochen Schmidt, Heinrich Niemann, and Fair Parametrizations. Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization. *VISION, MODELING, AND VISUALIZATION 2001*, 1 (Informatik 5):399 – 406, 2001.
- [146] C. T. Kelley. *Iterative methods for optimization*. Number 18. 1999. ISBN 0898714338. .
- [147] R Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(03):406–413, Jul 1955. ISSN 1469-8064.
- [148] Adi Ben-Israel and Thomas Nall Eden Greville. *Generalized Inverses: Theory and Applications*. Springer, 2003. ISBN 0387002936.
- [149] Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. ISSN 0162-1459. .
- [150] Sebastian Thrun, D Fox, and W Burgard. Monte carlo localization with mixture proposal distribution. *AAAI/IAAI*, 2000.
- [151] Petar Kormushev and Darwin G. Caldwell. Simultaneous discovery of multiple alternative optimal policies by reinforcement learning. In *IS'2012 - 2012 6th IEEE International Conference Intelligent Systems, Proceedings*, pages 202–207. IEEE, Sep 2012. ISBN 9781467327824. .

-
- [152] Petar Kormushev and Dg Caldwell. Direct policy search reinforcement learning based on particle filtering. In ... *on Reinforcement Learning*, number 2010, pages 1–13, 2012.
- [153] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. ISSN 00219606. .
- [154] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 0006-3444. .
- [155] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. ISSN 0036-8075. .
- [156] James C. Spall. Stochastic Optimization. In *Handbook of Computational Statistics*, pages 173–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 3-540-40464-3. .
- [157] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.
- [158] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. ISSN 00010782. .
- [159] R Sedgewick and K Wayne. *Algorithms*. Pearson Education, 2011. ISBN 9780132762564.
- [160] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, Sep 1977. ISSN 00983500. .
- [161] Marius Muja and David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer*

- Vision Theory and Applications (VISAPP '09)*, pages 1–10, 2009. ISSN 00301299. .
- [162] K Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
- [163] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide: Third Edition*. 1999. ISBN 0898714478.
- [164] The Mathworks Inc. MATLAB, 2014.
- [165] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System, 2009. URL: <http://www.ros.org>.
- [166] Silvia Rodriguez-Jimenez, Nicolas Burrus, and Mohamed Abderrahim. 3D Object Reconstruction with a Single RGB-Depth Image. In *Proc. of International Conference on Computer Vision Theory and Applications (VISAPP 2013)*, 2013.
- [167] G. E. P. Box and Mervin E. Muller. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958. ISSN 0003-4851. .
- [168] George Marsaglia. Choosing a Point from the Surface of a Sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, 1972. ISSN 0003-4851. .
- [169] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3, 2004. .
- [170] Open Source Robotics Foundation. Gazebo Simulator. URL: <http://gazebosim.org/>.
- [171] J. Pascoal, L. Marques, and A.T. de Almeida. Assessment of Laser Range Finders in risky environments. In *2008 IEEE/RSJ International Conference*

- on Intelligent Robots and Systems*, pages 3533–3538. IEEE, Sep 2008. ISBN 978-1-4244-2057-5. .
- [172] Shadow Robot Company. The Shadow Dextrous Hand. URL: <http://www.shadow.org.uk>.
- [173] Microsoft Corporation. Kinect sensor. URL: <https://www.microsoft.com/en-us/kinectforwindows/>.
- [174] Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio J. Nebro. *Why is optimization difficult?*, volume 193. 2009. ISBN 9783642002663. .
- [175] Dassault Systemes. Abaqus. URL: <http://www.3ds.com>.
- [176] Ellen M Arruda and Mary C Boyce. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *Journal of the Mechanics and Physics of Solids*, 41(2):389–412, 1993.
- [177] Shawn Waldon and Everett Carter. SIMANN: Simulated Annealing library. URL: <https://github.com/CISMM/SimulatedAnnealing>.
- [178] D. E. Finkel. DIRECT optimization algorithm user guide. *Center for Research in Scientific Computation, North Carolina State University*, 2:1–14, 2003.
- [179] Barrett Technology Inc. Barrett Hand BH8-280. URL: <http://www.barrett.com>.
- [180] Gael Guennebaud, Benoit Jacob, and Others. Eigen v3. <http://eigen.tuxfamily.org>, 2010. URL: <http://eigen.tuxfamily.org>.
- [181] Smooth-on. Ecoflex. URL: <http://www.smooth-on.com>.
- [182] C. Spearman. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1):72, Jan 1904. ISSN 00029556. .

- [183] Tadeo Corradi, Peter Hall, and Pejman Iravani. Tactile features: recognising touch sensations with a novel and inexpensive tactile sensor. In Chris Mistry, Michael and Leonardis, Aleš and Witkowski, Mark and Melhuish, editor, *Advances in Autonomous Robotics Systems*, pages 163–172. Springer, 2014.
- [184] Yi-Hung Liu, Yu-Tsung Hsiao, Wei-Teng Cheng, Yan-Chen Liu, and Jui-Yiao Su. Low-Resolution Tactile Image Recognition for Automated Robotic Assembly Using Kernel PCA-Based Feature Fusion and Multiple Kernel Learning-Based Support Vector Machine. *Mathematical Problems in Engineering*, 2014:1–11, 2014. ISSN 1024-123X. .
- [185] Y. Zhang and W.A. Gruver. Definition and force distribution of power grasps. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1373–1378. IEEE, 1995. ISBN 0-7803-1965-6. .
- [186] Hans Martin Kjer and Jakob Wilm. *Evaluation of surface registration algorithms for PET motion correction Bachelor thesis*. PhD thesis, DTU, 2010.
- [187] Lauren M. Miller and Todd D. Murphey. Trajectory optimization for continuous ergodic exploration. In *2013 American Control Conference*, pages 4196–4201. IEEE, Jun 2013. ISBN 978-1-4799-0178-4. .
- [188] Yonatan Silverman, Lauren M. Miller, Malcolm A Maciver, and Todd Murphey. Optimal Planning for Information Acquisition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5974–5980, Tokyo, 2013. ISBN 9781467363570.
- [189] Svetoslav Kolev and Emanuel Todorov. Physically consistent state estimation and system identification for contacts. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1036–1043. IEEE, Nov 2015. ISBN 978-1-4799-6885-5. .
- [190] D. J. Duff, J. Wyatt, and R. Stolkin. Motion estimation using physical simulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1511–1517, May 2010. .

-
- [191] M. R. Dogar and S. S. Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2123–2130, Oct 2010. .
- [192] Marek S. Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Mörwald, and Jeremy Wyatt. Learning to predict how rigid objects behave under simple manipulation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5722–5729, 2011. ISSN 10504729. .