UNIVERSITÀ DEGLI STUDI
DI GENOVA

ISTITUTO ITALIANO
DI TECNOLOGIA

UNIVERSITY OF GENOVA

DOCTORAL THESIS

Sense, Think, Grasp: A study on visual and tactile information processing for autonomous manipulation

---

Author:
Giulia Vezzani

Supervisors:
Dr. Lorenzo Natale
Dr. Ugo Pattacini

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

Humanoid Sensing and Perception - iCub Facility
Istituto Italiano di Tecnologia (IIT)

March 7, 2019

Declaration of Authorship

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others.

Giulia Vezzani
March 7, 2019

Abstract

Giulia Vezzani

Sense, Think, Grasp:
A study on visual and tactile information processing for autonomous manipulation

Interacting with the environment using hands is one of the distinctive abilities of humans with respect to other species. This aptitude reflects on the crucial role played by objects' manipulation in the world that we have shaped for us. With a view of bringing robots outside industries for supporting people during everyday life, the ability of manipulating objects *autonomously* and in *unstructured environments* is therefore one of the basic skills they need. Autonomous manipulation is characterized by great complexity especially regarding the processing of sensors information to perceive the surrounding environment. Humans rely on vision for wide-ranging tridimensional information, priopriocеption for the awareness of the relative position of their own body in the space and the sense of touch for local information when physical interaction with objects happens. The study of autonomous manipulation in robotics aims at transferring similar perceptive skills to robots so that, combined with state of the art control techniques, they could be able to achieve similar performance in manipulating objects. The great complexity of this task makes autonomous manipulation one of the open problems in robotics that has been drawing increasingly the research attention in the latest years.

In this work of Thesis, we propose possible solutions to some key components of autonomous manipulation, focusing in particular on the perception problem and testing the developed approaches on the humanoid robotic platform iCub. When available, vision is the first source of information to be processed for inferring how to interact with objects. The *object modeling and grasping pipeline* based on superquadric functions we designed meets this need, since it reconstructs the object 3D model from partial point cloud and computes a suitable hand pose for grasping the object. Retrieving objects information with touch sensors only is a relevant skill that becomes crucial when vision is occluded, as happens for instance during physical interaction with the object. We addressed this problem with the design of a novel *tactile localization algorithm*, named *Memory Unscented Particle Filter*, capable of localizing and recognizing objects relying solely on 3D contact points collected on the object surface. Another key point of autonomous manipulation we report on in this Thesis work is bi-manual coordination. The execution of more advanced manipulation tasks in fact might require the use and coordination of two arms. Tool usage for instance often requires a proper in-hand object pose that can be obtained via dual-arm re-grasping. In pick-and-place tasks sometimes the initial and target position of the object do not belong to the same arm workspace, then requiring to use one hand for lifting the object and the other for locating it in the new position. At this regard, we implemented a *pipeline for executing the handover task*, i.e. the sequences of actions for autonomously passing an object from one robot hand on to the other.

The contributions described thus far address specific subproblems of the more complex task of autonomous manipulation. This actually differs from what humans do, in that humans develop their manipulation skills by learning through experience and trial-and-error strategy. A proper mathematical formulation for encoding this learning approach is given by Deep Reinforcement Learning, that has recently proved to be successful in many robotics applications. For this reason, in this Thesis we report also on the six month experience carried out at Berkeley Artificial Intelligence Research laboratory with the goal of studying *Deep Reinforcement Learning and its application to autonomous manipulation*.

Publications

This work has been carried out during my Ph.D. course in *Advanced and Humanoid Robotics* from November 2015 to November 2018. This three-year project resulted in the following publications (at the time of writing):

- G. Vezzani, N. Jamali, U. Pattacini, G. Battistelli, L. Chisci, and L. Natale, "A novel Bayesian filtering approach to tactile object recognition," *IEEE International Conference on Humanoid Robots*, pp. 256 - 263, 2016, Cancun.
- G. Vezzani, U. Pattacini, G. Battistelli, L. Chisci, and L. Natale, “Memory Unscented Particle Filter for 6-DOF tactile localization,” *IEEE Transaction on Robotics*, 33 (5), 1139 - 1155, 2017.
- G. Vezzani, U. Pattacini, and L. Natale, “A Grasping approach based on superquadric models,” *IEEE International Conference on Robotics and Automation*, pp. 1579 - 1586, 2017, Singapore.
- G. Vezzani, U. Pattacini, M.Regoli and L. Natale, “A novel pipeline for bi-manual handover task,” *Advanced Robotics*, 31 (23-24), 1267 - 1280, 2017.
- G. Vezzani, and L. Natale, “Real-time pipeline for object modeling and grasping pose selection via superquadric functions,” *Frontiers in Robotics and AI*, 4, 59, 2017.
- G. Vezzani, U. Pattacini and L. Natale "Improving superquadric modeling and grasping with prior on object shapes", *IEEE Internationl Conference on Robotics and Automation (ICRA)*, pp. 6875 - 6882, 2018, Brisbane.
- C. Fantacci, G. Vezzani, U. Pattacini, V. Tikhanoff and L. Natale "Markerless visual servoing on unknown objects for humanoid robot platforms", *IEEE Internationl Conference on Robotics and Automation (ICRA)*, pp. 3099 - 3106, 2018, Brisbane.

- A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations", *Robotics: Science and Systems (RSS)*, 2018, Pittsburgh.

This Thesis provides a structured discussion on top of these aforementioned papers in order to describe the overall contribution of the Ph.D. project. As such, some ideas and figures have already appeared in those publications.

Acknowledgements

First and foremost, I would like to thank Dr. Lorenzo Natale, for giving me this opportunity. I still remember my excitement and happiness when I was accepted for starting my PhD in humanoid robotics. Together with guiding me during my research, Lorenzo has also given me the freedom to try my own ideas and improve my knowledge and skills with wonderful experiences during the past three years.

A special thanks goes to Dr. Ugo Pattacini, that has been and still is a mentor to me. During the last three years, he has taught me with patience and constancy not only countless technical skills but also the proper attitude for working efficiently and effectively.

I am grateful to Prof. Pieter Abbeel, for giving me the opportunity to visit the Berkeley Artificial Intelligence Research lab at UC Berkeley.

I also would like to thank the friends who have always been there and those I met during my PhD. Thank you all for always being close and present in my life, regardless of the distance and the PhD busy schedule. In particular to my ex- and current lab mates: thanks for sharing the fun, anxieties and abroad experiences that made the PhD a wonderful time.

Last, but not least, many thanks go to my family and Claudio, for always believing in me and supporting all my choices in the first lines.

Contents

List of Figures

List of Tables

Preface

"Man grew the most intelligent among animals because of having his hands" is what the philosopher Anaxagoras[1] said. Nearly one century later, Aristotle argued against this claim: *"It is more reasonable to say that man received his hands because he is the most intelligent [. . .]. Giving a flute to someone who can play it is a better plan than teaching someone which already has a flute how to play it. Considering that nature always acts in the best way, we must conclude that man does not own his intelligence to his hands, but his hands to his intelligence. "* [1]

According to Aristotle then, we did not choose the best path towards autonomous robotic manipulation. We equipped robots with hands they did not know how to properly use and we are now struggling to teach them.

Maybe this is the reason why robotic manipulation is such a hard problem to solve...

(Actually, Anaxagoras' intuition has been later on confirmed by several findings of paleoanthropologists, showing that the mechanical dexterity of the human hand has been a major factor in allowing homo sapiens to develop a superior brain (a similar role played by the anatomical structure of the human larynx in relation with speech capabilities has been also recognized). We therefore have hope of solving autonomous robotic manipulation and this thesis makes sense (fortunately)).

[1](500?–428 BC).

Part I

Introduction

Chapter 1

The importance of perception and autonomous manipulation

"Could you please give me the big bottle of oil?"

This is a task that every human more than five years old can successfully accomplish just taking a little care. Making a robot able to autonomously perform the same actions would require some of the most advanced techniques available in robotics and, most likely, the final outcome would be strongly customized for solving this specific task and would fail if objects positions or vision condition change. This is just one example of the huge gap that still exists between humans' and robots' skills in autonomous manipulation.

Using hands for interacting with the environment has been one of the key features of human evolution. The ability of operating tools allowed us to shape the world into a place where the chances of surviving were higher at the beginning of our history, and then created opportunity to further improve the quality of our lives with the emergence of technologies and arts.

Aside from its great benefits and power, autonomous manipulation is characterized by great complexity. Together with advanced motor and coordination skills, humans strongly rely on their perception system while interacting with objects: vision provides wide-ranging tridimensional information about the surrounding environment; prioprioception makes aware of relative position of one's own body in the space and the sense of touch gives local information when physical interaction with objects happens. Even if in the lack of one of the mentioned perception skills the other two can compensate, only the combination of all of them leads to the best performance. When trying to grasp an object in the darkness for example, a rough knowledge of the arm and hand positions is given by proprioception and touch helps when first contacts with the objects are detected. However, it might happen that we hit the object and make it drop while blindly looking for it. Equally, striking a matchstick without the sense of touch is almost impossible and it requires several trials to learn how to accomplish the task without the tactile information[1].

Autonomous robotic manipulation shows the same complexity as the human counterpart but no equal dexterous and powerful skills have been reached yet, making it one of the most challenging open problem in nowadays robotic research. While state of the art control techniques allow robots to follow complex trajectories with high speed and precision, equally efficient and reliable methods for sensors information processing and decision making are still missing. In particular, the ability of modeling the surrounding environment and autonomously inferring how to interact with objects do represent the core skills that mark the boundary between *preprogrammed* and *autonomous* manipulation.

This work of Thesis addresses some of the key problems of autonomous manipulation such as *object modeling* and *localization, grasping* and *bi-manual coordination*. The vision and motivation that accompanied the Ph.D. research activity are shown in Section 1.1. Then, Section 1.2 summarizes the main contributions and outlines the Thesis structure.

[1]This experiment was conducted by Prof. Roland Johansson at the University of Umeå, Sweden. More information are available here: http://roboticmaterials.com/rm/why-making-robotic-manipulation-harder-than-it-is/.

1.1 Vision and motivation

Industrial manipulators are already in use in factories for many years by now. They can achieve great performance in terms of speed, precision and reliability, because advanced control techniques are flanked with very *accurate knowledge of the environment and the objects to manipulate* (Fig. 1.1). Such an information level is possible since industrial manipulators operate

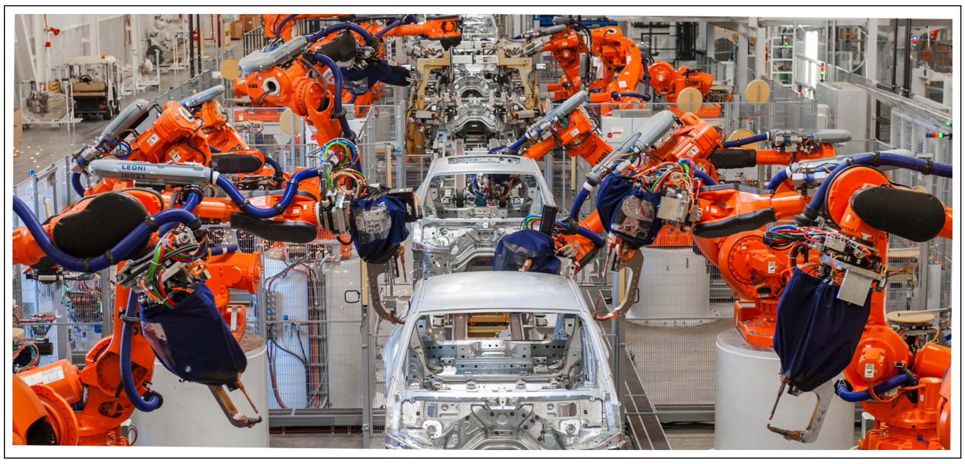


FIGURE 1.1: Industrial manipulators for car assembly.

in strongly structured environments. Though very effective, this restricts the operational domain of the such robots and requires engineering process of the environment. By removing such a constraint, production chains would be built more easily, but also, and more importantly, robots could be brought outside industries for supporting people in their everyday life. Robots ability to manipulate objects *autonomously* and in *unstructured environment* would be in fact crucial in several applications, such as cargo handling, people and object recovery after disasters, elder people assistance (Fig 1.2). Moving towards this new robots generation requires the design of novel hardware and cognitive skills. Taking inspiration from humans[2], the required features are the following:

1. A motor system reproducing the human upper body, in particular including two arms and hands with sufficient degrees of freedom (DOFs) and dexterous workspaces.

[2]Human-inspired robots are not necessary the optimal choice but taking inspiration from humans seems reasonable given their impressive dexterity in manipulation. However, a proper discussion about whether human-inspired robots are the best choice for autonomous manipulation does not fall within the scope of this work.

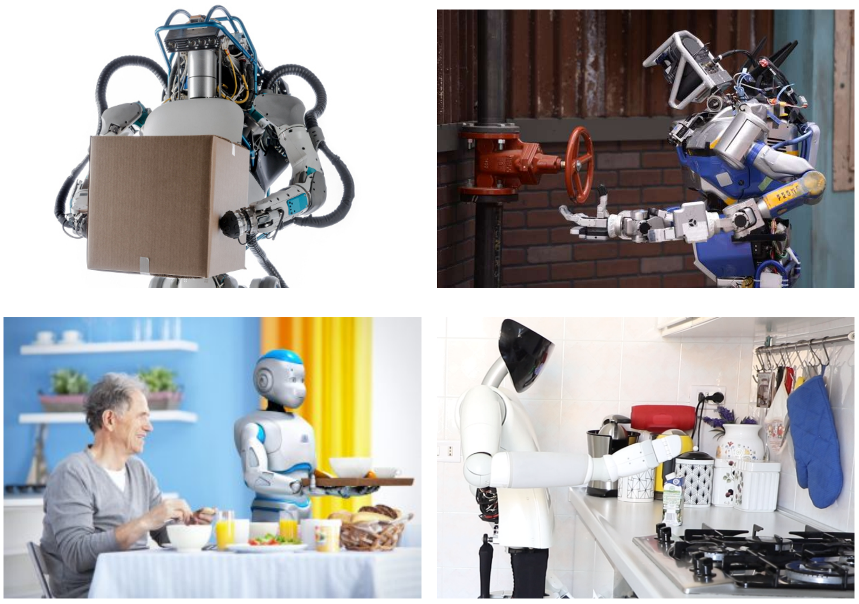


FIGURE 1.2: How autonomous robots could have beneficial effects on the society: some examples.

2. A proper sensor system including: cameras for acquiring images and depth information, tactile sensors on the robot hands and encoders for priopriocection.

3. A control system of the upper body for reaching desired poses in a reliable, safe and accurate way.

4. The ability of processing perceptive information generated by vision and tactile sensors.

5. Motor coordination with respect to the environment and the objects to manipulate and between the robot arms themselves.

The hardware design of several recent humanoid robots, such as the iCub [2] - the platform used for testing all the contributions presented in this Thesis work -, presents good, even still upgradeable, solutions for the fulfillment of the first three requirements in the aforementioned list. For this reason, the attention of this research activity focuses on the other two points, i.e. on the processing of the perceptive information and planning of proper movements for accomplishing manipulation tasks. The techniques derived from this research are still general and can be applied to any other humanoid robots satisfying points 1., 2. and 3. .

1.2 Contribution and outline

During the Ph.D. activity, our research focused on the following problems:

- **Object localization**: the problem of estimating the pose of a known object, using the information acquired from the robot perceptive system (vision and/or tactile sensors).
- **Bi-manual coordination**: the planning of proper actions to be executed with two arms for the accomplishment of a unique task.
- **Object modeling**: the reconstruction of an efficient mathematical representation of an unknown objects in terms of their shape, dimensions and pose, by using information provided by the robot sensors.
- **Object grasping**: the design of a suitable pose of the robot hand with respect to the object in order to enable grasps characterized by given properties (e.g. robustness, precision etc).

For each problem, we proposed specific solutions that together provide the main contributions of this Thesis:

- A *localization algorithm*, named *Memory Unscented Particle Filter*, capable of estimating the object poses using only 3D points collected on the object surface.
- A complete pipeline for the execution of *object bi-manual handovers* with the robot iCub. In other words, the robot is asked to pass one object from his hand on to the other.
- An *object modeling and grasping pose computation approach based on superquadric function*, that uses vision information to reconstruct the object model and compute a robot hand pose for grasping the object.

The solutions described thus far address specific key-parts of the general manipulation task. In particular, the solution of a complex and general problem such as manipulation is turned into the solution of many simpler sub-problems. This is the standard way to go in robotics, but it actually differs from what humans do. Humans develop their manipulation skills by *learning* them through experience and trial-and-error strategy. A mathematical formulation that encodes this approach is Reinforcement

Learning (RL) [3]. While previously applied mostly to game playing, recent works [4, 5, 6] show successful applications of Deep RL[3] to robotic manipulation. For this reason, six months of the Ph.D. activity presented in this work took place at Berkeley Artificial Intelligence Research laboratory at UC Berkeley with the main goal of studying Deep Reinforcement Learning and its application to autonomous manipulation.

This Thesis is organized as follows. A review of the state of the art regarding manipulation and more in particular the four problems we tackled is provided in Chapter 2, followed by a brief description of the iCub humanoid robot in Chapter 3. Chapters 4 and 5 respectively describe the Memory Unscented Particle Filter, including the mathematical derivation and experimental evaluation, and its application on a challenging tactile recognition task. In Chapter 6, we describe the entire pipeline implemented on the iCub robot for the execution of bi-manual handover tasks. Chapter 7 and 8 reports the proposed object modeling and grasping pose computation respectively for simple and more complex kinds of objects. In Chapter 9, the activity carried out while studying Deep Reinforcement Learning is detailed. Finally, Chapter 10 ends this Thesis work with concluding remarks and more general discussion.

[3]Deep Reinforcement Learning is the combination of Reinforcement Learning together with deep networks. Deep networks are extremely helpful when, for example, the algorithm is asked to learn from raw inputs, such as vision, without any hand-crafted features or domain heuristics.

Chapter 2

What is the stage of autonomous manipulation?

Autonomous manipulation requires the solution of different subproblems, through the combination of techniques belonging to various areas of robotics and the fusion of information provided by different sensors. Considering the intrinsic multidisciplinarity nature of the problem at hand so as the contributions of this Thesis work, we analyze the current stage of autonomous manipulation by focusing on the following topics: *tactile object localization and recognition* (Section 2.1), *bi-manual coordination* (Section 2.2), *object modeling from vision* (Section 2.3) and *autonomous grasping* (Section 2.4).

We complete the overview of the state of the art in autonomous manipulation by reporting on the recent progress of *Deep Reinforcement Learning* in dexterous manipulation and the related challenges (Section 2.5).

2.1 Tactile object localization and recognition

The interest in tactile sensing and perception has considerably increased over the last decade [7], often flanking or even replacing vision information during object manipulation, localization and recognition [8, 9, 10, 11, 12]. This extensive usage of the sense of touch in robotics is put forward in several findings in human physiology that testify how humans jointly exploit vision and touch in order to accomplish manipulation tasks and how humans are even able to explore objects by means of tactile perception solely [13, 14]. Certainly, the surge of interest on this topic is also encouraged by the recent advances in tactile technology [15, 16, 17, 18, 19, 20] that have made it possible to build tactile systems that are reliable enough and can be deployed on real robots at a reasonable price [21, 22, 23]. Tactile sensors can be categorized in multiple manners, such as according to their sensing principles, fabrication methods or the body parts they are analogous to [7, 24]. The main sensing principles exploited in the current technologies are resistance, capacitance, piezoelectricity, optic component, or magnetics. Another possible classification regards the output nature of touch sensors. The major focus is often to measure the contact force and location over a certain area. Some other sensors, such as the ones exploited in this Thesis work [21], retrieve instead the pressure exerted on the sensor when contact is detected. Very recently, a novel vision-based optical tactile sensor, named GelSight [25, 26, 27], has been developed and commercialized. Unlike the traditional tactile sensors, GelSight basically measures geometry, with very high spatial resolution. The sensor has a contact surface of soft elastomer, and by a conventional camera it captures the deformations of the elastomer when the sensor is pressed against an object. The contact force and slip can be inferred from the sensor deformation itself.

Tactile sensors play a fundamental role in autonomous manipulation since they provide useful information when contacts with the objects are detected. Vision is usually the major source of measurements for perceiving the surrounding environment but it might be occluded while interacting with the object. In such a scenario, the ability of *localizing* and *recognizing* objects by means of solely tactile measurements is crucial for the execution of manipulation tasks. Hereafter, we briefly review the state of the art in *tactile object localization* and *recognition*.

Tactile object localization

The first contributions on tactile object localization (1980s) tackled the problem by using mostly iterative optimization methods and focused on finding a single solution best fitting the set of available measurements [28, 29, 30]. Since these methods tend to be trapped in local minima, *low initial uncertainty* is assumed so as to ensure that the optimization algorithm is initialized near the solution. In order to avoid local minima, the algorithm can be executed multiple times from different starting points.

Over the last years, Bayesian methods have been playing an important role in tactile localization [31, 32, 33, 34]. In particular, these methods are capable of working with noisy sensors, inaccurate models, moving objects and can give information on where to sense next during tactile exploration. Thus, they can be used not only to localize the object, but also to provide useful information for collecting measurements and for real exploration.

Since the localization problem is intrinsically of multimodal nature (i.e. the probability density exhibits multiple peaks), nonlinear Kalman filtering techniques (such as the extended or unscented Kalman Filter) cannot be satisfactorily used. In this respect, the Bayesian framework (e.g. particle filtering) is more appropriate, since it intrinsically handles multimodal distributions. On the other hand, its main drawback is represented by the computational complexity, which grows exponentially with the number of independent variables (DOFs) and polynomially with the size of the initial region of uncertainty. For example, recalling that the localization of an object involves 6 DOFs, a particle filter should be configured to run with a number of particles in the order of 10^6, which might entail an unaffordable computational burden for real-time operation. In fact, most of the existing work is characterized by assumptions limiting the number of DOFs and the size of initial uncertainty.

In this respect, the first known work traces back to 2001 and is due to Gadeyne et al. [31], who performed 3-DOF localization of a box with initial uncertainty of 300 *mm* in position and 360 degrees in orientation. Measurements were taken by a force-controlled robot and a sampled measurement model, stored in a lookup table and constructed off-line, was used. In 2005, Chhatpar et al. [32] used particle filters to achieve 3-DOF localization with 20 *mm* of initial uncertainty in peg-in-hole assembly tasks. The exploited measurement model was obtained by sampling the object in advance.

An interesting approach to tactile localization makes use of the *Scaling Series* method [34, 35] developed by Petrovskaya et al., by which 6-DOF localization has been achieved with large initial uncertainty of 400 *mm* in position and 360 degrees in orientation. This method, which combines Bayesian Monte Carlo and annealing techniques, exploits measurements of contact points and surface normals. It performs multiple iterations over the data, gradually scaling precision from low to high. For each iteration, the number of particles is automatically selected on the basis of the complexity of the annealed posterior. As it will be shown in Chapter 4 through specific experiments, the Scaling Series algorithm is however affected by low reliability, that can be attributed to the automatic process of particle generation. Particularly, a rough parameter tuning can easily lead to the generation of an insufficient number of particles or, on the contrary, to their exponential growth, thus precluding the final convergence of the algorithm.

In 2010, Corcoran et al. [33] used an annealed particle filter to estimate a 4-DOF pose and radius of cylindrical objects. The initial uncertainty was of 250 mm in position and unrestricted in orientation. The measurement model proposed in [34] was extended by exploiting the concept of "negative information". To this end, a set of "no-contact measurements" is defined to account for regions explored by the robot where it is known or it can be inferred that the object cannot be located, since no contacts are perceived.

In 2013, Chalon et al. [36] presented another particle filter method including information on both object and finger movements. A recent work [37] combines global optimization methods with the Monte Carlo approach in order to provide a batch solution to the global localization problem, either improving the estimate of the object pose obtained by vision or globally estimating pose when vision is not available.

In [38], Koval et al. propose a novel particle filter, named Manifold Particle Filter. This algorithm samples particles directly from a contact manifold guaranteeing the non-penetration constraint. In fact, the real pose of the object in contact with the robot hand belongs to a lower-dimensional manifold determined by non-penetration constraints between the object and the hand itself. If this kind of constraint is not incorporated, the sampled particles might correspond to configurations in which the manipulator and the object are overlapping or separated. The main limitation of this approach is however given by the fact that it relies on explicit analytic and sampled-base representations of the contact manifold. This kind of representation fits well to low-dimensional domains but do not scale properly to more complex scenarios. For this reason in [39], the same authors addressed the problem by proposing an implicit representation of the contact manifold to apply the Manifold Particle Filter to six or more dimensional state spaces.

In 2017, we proposed the Memory Unscented Particle Filter [40] which combines an Unscented Particle Filter with a windowing based memory strategy to estimate the 6D pose of a stationary object using 3D tactile contact points - and no contact normals - and starting from an initial uncertainty of 400 mm in position and 360 degrees in orientation. This approach is one of the main contributions of the Thesis and is presented in Chapter 4.

Some recent works [41, 42, 43] have focused on the particular problem of tactile in-hand object pose estimation, i.e. the estimation of an object pose when hold in the robot hand. In particular, [43] makes use of a particle filter for processing tactile information and of haptic rendering models of the robot hand. The particle filter at first estimates the grasp pose of the object using the touch measurements. Then, hypotheses of grasp poses are evaluated by comparing tactile measurements and expected tactile information from CAD-based haptic renderings.

Another interesting application, which the attention in the latest years has been turned on, is visuo-tactile localization, i.e. the exploitation and fusion of tactile and vision information with the final goal of estimating the object pose [10, 27, 44]. A detail review of the state of the art of those approaches is hower out of the scope of this work.

Tactile object recognition

Tactile object recognition refers to the problem of discriminating an object with respect to others considering some of its properties measurable with touch sensors. Several features can be taken into account for the solution of the recogntion problem such as the object shape, surface texture or curvature.

Different methods have been proposed in the literature in order to solve tactile object recognition. They can be classified depending on the type of information they use and the object features they recover, namely, material and shape properties. Some researchers have focused on identifying material properties [45, 46, 47]. Decherchi et al. use multiple techniques to classify object materials with tactile data [46]. Liu et al. [47] apply a dynamic friction model to determine physical properties of surfaces while a robotic finger slides along the object with different speeds.

To recognize object shapes, a viable approach is to recover local geometry from each contact point, i.e., surface normal and curvature. By using a cylindrical tactile sensor, Fearing et al. propose a nonlinear, model-based inversion to recover contact surface curvatures [48]. Contact location point-clouds have also been used to reconstruct object shapes with computer graphic techniques [49, 50, 51, 52]. Allen et al. fit points from tactile sensors readings to superquadric surfaces to reconstruct unknown shapes [51]. A similar approach, proposed by Charlebois [53], uses tensor B-spline surfaces instead of superquadratic surfaces. Through these methods, arbitrary object shapes can be identified by estimating surface curvatures. In [54], we perform tactile recogniton by using contact location point-clouds but without reconstructing the object shape or estimating surface normals. We cast tactile recognition into a localization problem wherein multiple models are fit to the available measurements and objects are recognized by selecting the model that minimizes the localization error. More details about this approach are provided in Chapter 5.

Another solution to recognizing object shapes is to use machine learning techniques on the output of tactile sensor arrays. In this case, object features are extracted from the tactile data and/or haptic measurements. A classifier is then trained to predict the shapes of novel objects [18, 55, 56, 57, 58, 59, 60, 61, 62, 63].

The recent development of new sensors, such as GelSight, leads to sensitivity and resolution exceeding that of the human fingertips. This opens the possibility of measuring and recognizing highly detailed surface textures and shapes. The GelSight sensor, when pressed against a surface, delivers a height map. This can be treated as an image, and processed using the tools of visual texture analysis. An example of Gelsight application on texture recognition is shown in [64], where a material can be correctly categorized among a database of 40 classes of tactile textures corresponding to different materials. Other kinds of tactile sensors are also mounted on robot hands for extracting tactile images and applying computer vision technique to process such an information for object recognition [65, 66].

2.2 Bi-manual coordination

Humans take advantage of both their arms for executing plenty of manipulation tasks. Bringing bi-manual coordination skills to humanoid robots, themselves equipped with two arms, is then fundamental within the view of replicating the human manipulation abilities on a robotic platform.

Some examples of manipulation tasks where bi-manual coordination provides the solution are:

- Moving towards a desired position objects too heavy or big to be grasped with a single hand.
- Pick-and-place scenarios where the initial and target position of the object do not belong to the same arm workspace.
- Re-grasping objects whose in-hand pose is not suitable for performing the required task.

In the literature, bi-manual coordination has been addressed focusing on all the manipulation tasks listed above. One of the earliest contributions on bi-manual coordination is [67], in which the authors address the problem of planning the path of two cooperating robot arms to carry an object from an initial configuration to a goal configuration amidst obstacles. The paper compares three 2D planning techniques with different arms – 2-DOFs and 3-DOFs – in different scenarios (without and with obstacles). These results were later extended to 3D planning [68, 69, 70]. A more recent work [71] implements multi-arm handover for object movements towards a final position using the motion planning framework proposed in [70]. In [72], Kromer et al. address bi-manual coordination in a Reinforcement Learning setting. Their method exploits the phase structure in which tasks can be split in order to learn manipulation skills more efficiently. Starting with human demonstrations, the robot learns a probabilistic model of the phases and the phase transitions. Then model-based Reinforcement Learning is used to create motor primitives that well generalize to new situations and tasks. Another work where bi-manual coordination is cast into the learning framework is [73] where the authors combine a dynamical systems formulation of the demonstrated trajectories and a task- parameterized probabilistic model to extract the relevant features of the demonstrated skill.

Another application of bi-manual coordination is object re-grasping [74, 75]. In [74], bi-manual re-grasping is formulated as an optimization problem, where the objective is to minimize execution time. The optimization problem is supplemented with image processing and a uni-manual grasping algorithm based on machine learning that jointly identifies two good grasping points on the object and the proper orientations for each end-effector. The optimization algorithm exploits this data by finding the proper re-grasp location and orientation to minimize the execution time. The work presented in [75] instead provides an interesting study on when one- or dual-arm re-grasp is to be performed, according to the object properties. Dual-arm re-grasp is more flexible and versatile but if the two hands grasps overlap on the object, an higher success rate is provided by single-arm re-grasp.

Dual-arm re-grasp is a specific case of the more general handover task, i.e. passing the object from one hand on to the other. The handover can be performed for executing pick-and-place tasks when the initial and target position of the object do not belong to the workspace of the same arm [76]. Other works as [54, 77, 78] provide solutions to the general handover problem. In [77], Gasparri et al. cast the handover into a robust optimization problem focusing on the choice of optimal stiffness to accomplish the handover by minimizing the forces involved. Both [54, 78] rank a priori generated poses for the execution of the handover. A detailed description of [54] is provided in Chapter 6, as one of the main contributions of this Thesis work.

2.3 Object modeling from vision

The ability of reconstructing in real-time 3D information about the scene and, particularly, the object to be manipulated is central to autonomous manipulation. Raw 3D information can be obtained differently according to the vision system the robot is equipped with, such as stereo or RGB-D cameras.

Several works in manipulation [79, 80, 81, 82] exploit only partial point clouds collected from a single view of the object to infer how to approach and manipulate the object. However, ignoring a 3D representation of the occluded portions of the object might lead to failures during the execution of the task. Reconstructing instead a full 3D model, capable therefore to represent also the occluded portions, provides much powerful information on the object shape and volume.

Object modeling has been studied since the 1980s mostly in computational geometry and computer graphics as surface reconstruction, i.e. the problem concerned with recreating a surface from scattered data points sampled from an unknown object. Several methods have been developed in last decades [83, 84, 85, 86, 87, 88] mostly returning mesh models as output of the modeling process.

In the last years, a great interest for object modeling arose also in robotics due to the growing availability of depth or stereo cameras [89, 90, 91, 92]. Several recent works reconstruct object mesh models from partial 3D informations using Deep Learning techniques [89, 93, 94, 95, 96, 97]. In [89], for instance, the authors perform shape completion through the use of a 3D convolutional neural network (CNN), trained on a large dataset of mesh models. At runtime, a 2.5D point cloud captured from a single point of view is fed into the CNN that returns a full 3D model of the object. This way, the occluded portion of the object is inferred by the network from the training set, thus providing more realistic models than just using symmetries or minimum volume closures. Another popular kind of object model used in most recent works consists of voxel-based models [98, 99]. They represent in fact the object volume and shape as a binary occupancy grid and therefore provide a representation suitable for being used with CNN, that are increasingly popular in 3D applications.

A different type of 3D model introduced in computer graphics in 1981 by A.H. Barr is the superquadric model [100, 101], a generalization of quadrics that has been well studied in graphics and computer vision [102]. Superquadrics and extensions such as hyperquadrics [103] and deformable superquadrics [101] are a convenient representation for a large class of both convex and non-convex objects. The most popular method to determine superquadric parameters for fitting partial or full object point clouds was proposed by Solina in 1990 [102]. Recently, several works have focused on speeding up computation [90, 104] and refining the model by extending it to approximate complex shapes with a set of superquadrics [105, 106]. In addition to object approximation, superquadrics have been used for object detection [107], object segmentation [108, 109], collision detection [110, 111] and grasping [90, 112, 113, 114, 115, 116]. The great advantage of superquadrics with respect to mesh or voxel-based models consists of the small number of parameters to be memorized and their closed-form mathematical representation. Nevertheless, the new Deep Learning methods show much lower computation test time, since most of the effort is done during the offline training. In this respect, some preliminary ideas on how to speed up the superquadric computation making use of the Deep Learning framework have been proposed in [117].

2.4 Autonomous grasping

The grasping problem consists of computing a feasible pose of the robot hand, which allows grabbing the object under consideration. While great performance can be achieved if the shape and position of the object are accurately provided, autonomous grasping of unknown objects or whose pose is uncertain is still an open problem. Although the problem has been addressed since the late 1980s [118], recently the robotic community has shown an increasing interest in autonomous grasping [119, 120, 121]. Diverse methodologies have been explored addressing various goals still belonging to the field of grasping. In this respect, the broad field of autonomous grasping can be divided into more specific areas according to several criteria. For instance, grasp actions can be divided into *power* and *precision* grasps [122]. Power grasp involves large areas of contact between the hand and the object, without the adjustment of the fingers after contact [123]. On the contrary, precision grasp provides sensitivity and dexterity, since in this case the object is held with the tips of the fingers [124]. In precision grasp tasks, the hand touches the object at small contact points, therefore the study of grasp stability plays an important role.

Another classification criterion [120] considers how the robot hand pose for approaching the object is computed, grouping the methodologies in *analytic* and *empirical*. The former formulates the grasping problem only in terms of force-closure and form-closure, looking for specific conditions on the contact wrenches that ensure a certain hand configuration to firmly hold any object. These approaches usually assume that contact point locations were given without explicitly relating the hand configuration to the object geometry. Empirical or *data-driven* approaches instead mimic human grasping in selecting a grasp that best conforms to task requirements and the target object geometry. They often rely on sampling grasp candidates for an object and ranking them according to a specific metric.

Until about twenty years ago, most popular robotic grasping approaches belonged to the analytical class, as reviewed in [119]. Data-driven approaches started to become popular with the availability of the simulator GraspIt! [125]. Several works have been developed [112, 126, 127] using this or analogous simulators and being characterized in how grasp candidates were sampled and ranked. More recently, some studies [128, 129] showed that techniques just evaluated in simulation do not provide good predictors for grasp success in the real world. This motivated several researchers [130, 131, 132] to let the robot learn how to grasp by experience gathered during grasps execution. The problem of transferring from simulated environments to the real world is then removed at the cost of extremely time-consuming collection of examples. Then, a crucial point becomes how to generalize the collected experience to novel objects so to contain the amount of required data. A great improvement in this respect has been provided by Saxena et al. [133] who trained simple logistic regressors on large amounts of synthetic labeled data to predict good grasping points in a monocular image. The authors demonstrated their method in a household scenario in which a robot emptied a dishwasher. Several other works were subsequently proposed addressing the problem of inferring discriminative object features for grasping [134, 135].

The availability of affordable and accurate depth sensing devices starting from the 2010 [136, 137] encouraged grasping research to rely increasingly more on 3D data. Processing depth map or point clouds is nowadays the starting point for the majority of approaches proposed for the grasping problem [90, 113, 123, 124, 138].

A further incentive for the study of grasping has been provided by the Amazon picking challeng [139], whose goal is the development of robotics hardware and software able to identify objects, grasp and move them from place to place. Since 2015 this challenge has been giving rise to different approaches [140, 141, 142] sharing the same scenario: two-fingered grippers are used to grasp known and novel objects in the clutter.

The methods mentioned thus far place more weight on the object representation and the processing of perceptive data to retrieve grasps from some knowledge base or sample and rank grasps by comparison to existing experience. As a result, a convenient way to group data-driven approaches [121] is based on the prior knowledge assumed on the query object: if it is *known, familiar* or *unknown*. The trend of most recent works is in particular to focus on *familiar* or *unkwnon* objects, often generalizing from the available knowledge on how to grasp *known objects*.

One approach towards generating grasp hypotheses for unknown objects is to approximate objects with shape primitives. Marton et al. [91] show how grasp selection can be performed exploiting symmetry by fitting a curve to a cross section of the point cloud of an object. For grasp planning, the reconstructed object is imported in a simulator. Grasp candidates are generated through randomization of grasp parameters on which then the force-closure criteria is evaluated. Rao et al. [143] use segmentation, especially relying on depth information. A supervised localization method is then employed to select graspable segments and plans a grasping strategy, after shape completion from the partial 3D information. Bohg et al. [144] propose a related approach that reconstructs full object shape assuming planar symmetry. Recently, superquadrics functions have become a popular alternative to point clouds or mesh models for representing novel objects. As already mentioned in Section 2.3, they provide a compact mathematical formulation and their precision in modeling the object shape and volume occupancy has been proven to be suitable to grasping tasks. Some works [112, 113, 114] reconstruct the object model using a single or a set of superquadrics and then rely on grasp candidates generators (such as GraspIt!) to select the grasp candidate. In [90] instead, we exploit superquadrics not only for estimating the object 3D model but also for representing the volume graspable by the hand. A single proper grasp candidate is computed by overlapping the superquadric representing the volume graspable by the robot hand onto the object superquadrics while meeting some orientation and obstacle avoidance constraints (e.g. avoidance of the support on which the object is located). We will provide extensive details on this approach in Chapter 7. Other ideas about how to use the superquadric model to compute directly grasp candidates are shown in [115] and [116]. Makhal et al. in [115] design the grasping pose by maximizing force balance and stability and taking advantage of dimension and surface curvature information obtained from the object superquadric parameters. In [116], the grasp candidates are located in proximity of the superquadric cardinal points of the upper part of the object (for avoiding the support on which the object is located), taking into account proper constraints on orientations. The poses are then ranked according to their reachability and the matching between the object and hand dimensions.

Another approach to deal with unknown objects consists of extracting from 2D or 3D visual information those features able to encode some properties of the object relevant to the grasp. Several heuristics and patterns have been proposed to extract grasp candidates from low-level [79, 80, 138, 145] or more global shape information [146, 147]. Very recently, the most common technique used to extract grasping features from 3D data is Deep Learning [81, 82, 148, 149]. A relevant work in this respect is given by the DexNet project [150, 151, 152] including a growing synthetic dataset of million point clouds, grasps, and analytic grasp metrics generated from thousands of 3D models and a model that rapidly predicts the probability of success of grasps from depth images, where grasps are specified as the gripper planar position, angle and depth. Throughout their papers, Mahler et al. have extended the Dexnet dataset and improved the model predicting grasp success, being able to provide very high success rate on novel and adversarial objects. The large number of samples of the Dexnet dataset is linked to a crucial limitation of Deep Learning techniques: the need of a huge amount of labeled data to enable generalization. Collecting such a number of samples in the real world is not time-wise affordable. The alternative is to generate the data in simulation environments. However, as already mentioned, grasp predictors trained only on simulated data are very likely to behave poorly in the real world. New common strategies for facing this issue are *domain randomization* [153, 154] and *adaptation* [155, 156]. The former consists in training the grasp predictor in simulation using tons of data randomly generated and synthetically labeled and randomizing over different parameters of the system (e.g. image noise and physical parameters etc.). The latter is a particular case of Transfer Learning that utilizes labeled data in one or more relevant source domains to execute new tasks in a target domain. In grasping applications the source domains are generated and used in simulation and real-world is the target domain.

In the last years, also the Deep Reinforcement Learning framework has been used for addressing the grasping problem. More details about the latest results in this respect are collected in the following Section.

For the sake of comparison, Table 2.1 summarizes the state of the art of autonomous grasping by focusing on data-driven methods for power grasp of unknown objects, including also Deep RL methods that will be described in the following Section.

TABLE 2.1: State of the art comparison among data-driven approaches for power grasp of unknown objects.

Reference	Methodology	Distinctive feature	Input	Robot hand
[123]	pose ranking	geometry-based	3D partial point cloud	multi-fingered
[112]	pose ranking	simulator-based	reconstructed superquadrics	multi-fingered
[126]	pose ranking	geometry-based	reconstructed shapes	multi-fingered
[127]	supervised learn.	SVM	reconstructed superquadrics	multi-fingered
[130]	supervised learn.	automatic data collection	RGB images	gripper
[132]	supervised learn.	predict pose reliability	RGB images	multi-fingered
[133]	supervised learn.	synthetic dataset	RGB images	gripper
[134]	supervised learn.	predict pose stability	RGB-D images	multi-fingered
[90]	optimization	robot agnostic	reconstructed superquadrics	multi-fingered
[138]	pose ranking	local features	3D point cloud	gripper
[91]	pose ranking	simulated grasp planner	reconstructed mesh model	multi-fingered
[143]	supervised learn.	fill missing depth data	reconstructed shape	multi-fingered
[144]	pose ranking	simulated grasp planner	reconstructed mesh model	multi-fingered
[115]	geometry-based	mirror partial point cloud	reconstructed superquadrics	gripper
[116]	pose ranking	geometry-based	reconstructed superquadrics	multi-fingered
[79]	pose ranking	local features	RGB-D images + tactile data	gripper
[80]	pose ranking	local features	RGB-D images	gripper
[145]	local features	active exploration	RGB images	gripper
[146]	pose ranking	global features	RGB images	gripper
[147]	pose ranking	global features	RGB-D images	multi-fingered
[150]	supervised learn.	synthetic dataset	RGB-D images	gripper
[151]	supervised learn.	synthetic dataset	RGB-D images	gripper
[152]	supervised learn.	synthetic dataset	RGB-D images	gripper + suction
[153]	supervised learn.	domain randomization	RGB-D images	gripper
[154]	supervised learn.	domain randomization	RGB-D images	gripper
[155]	supervised learn.	domain adaptation	RGB images	gripper
[156]	supervised learn.	domain adaptation	RGB images	gripper
[4]	supervised learn.	spatial softmax CNN	RGB images	gripper
[5]	deep RL	trained on real-robot	RGB images	gripper
[6]	deep RL	trained on real-robot	RGB images	gripper
[157]	deep RL	trained on real-robot	RGB images	gripper
[158]	deep RL	trained on real-robot	RGB images	gripper

2.5 Deep Reinforcement Learning for autonomous manipulation

One goal of artificial intelligence is to provide fully autonomous agents interacting with the environment and learning optimal behaviors through trial and error. Reinforcement Learning (RL) provides a suitable mathematical framework for experience-driven autonomous learning [159] and has been shown to be successful in several tasks in the past [160, 161, 162] although in the context of low-dimensional domains. RL is in fact affected by the lack of scalability due to its memory requirements and computational complexity. As happened in many areas of machine learning, such as computer vision, speech recognition and language translation, Deep Learning had had a significant impact on RL, improving considerably the state of the art and defining the so-called field of Deep Reinforcement Learning. The main reason of Deep Learning success is that deep neural networks are able to learn compact low-dimensional representations, i.e. *features*, of high dimensional data, such as images. This way, RL has scaled to decision-making problems that were previously intractable, due to their high-dimensional state and action spaces. One example of the successful stories of Deep RL is the development of an algorithm able to play a set of Atari 2600 video games with super-human performance, learning directly from image pixels [163]. Another example is the hybrid DeepRL system, AlphaGo, that defeated a human world champion in Go [164] and brought to another level the AI revolution started two decades earlier with DeepBlue [165] and Watson DeepQA [166] that won respectively chess and quiz competitions against human players. In particular, the novelty of AlphaGo consisted of the usage of neural networks trained with supervised and reinforcement learning together with a traditional heuristic search algorithm.

Deep RL has been applied to a variety of different fields, including also robotics in the very last years. One of the most studied problems is autonomous manipulation [4, 5, 6, 157, 158, 167]. In the popular work described in [6], the authors propose a learning-based approach to learn hand-eye coordination for robotic grasping from monocular images. They train a large CNN to predict the probability that the motion of the gripper will achieve a successful grasps, using only monocular camera images independent of camera calibration or the current robot pose. For this reason the network is required to learn also the spatial relationship between the gripper and the objects in the scene. The approach was tested on real robots thanks to the collection of large-scale datasets, using up to 14 manipulators and two months of grasp attempts. In [4] Levine et al. propose a method to learn CNN policies that map from raw images to torques at robot motors. The approach is tested on a range of real-world manipulation tasks such as screwing a cap onto a bottle and placing a coat hanger on a clothes rack. Another work where DeepRL is used to learn policies directly from raw pixels is presented in [5]. This method uses a deep spatial autoencoder to acquire a set of feature points that describe the environment for the manipulation task to be solved, such as object position, and learns a motion skill with these feature using RL. This approach is shown with the PR2 robot on task including pushing toy blocks, picking up items with a spatula and hanging a loop of rope of a hook at various positions. Although outstanding, these works still show some limitations. In [6], a huge amount of data is required to be collected on the real robot for achieving good performances and the motions necessary for performing the tasks addressed in [5] are limited to simple actions.

An interesting extension in this respect is the use of multi-fingered manipulators for the accomplishment of dexterous manipulation tasks that cannot be executed with a single gripper, such as in-hand manipulation, complex grasping and tool use. These tasks turn out to be very challenging due to the high dimensional observation and action spaces involved and the difficulties in defining proper reward functions for guiding the agent during the training. Solving tasks with such a level of difficulty often require to face one of the greatest difficulties in RL: the *exploration versus exploitation problem*, i.e. the problem of choosing between non-optimal actions in order to explore the state space and exploiting the optimal action in order to make useful progress. One of the simplest exploration strategy, typical used in off-policy algorithms such as DQN [163], is the ϵ-greedy exploration policy that chooses a random action with probability $\epsilon \in [0, 1]$ and the optimal action otherwise. By decreasing ϵ over time, the agent progresses towards exploitation. As the task becomes more complex or temporally extended however, this kind of exploration strategies becomes less effective. Several exploration strategies have been proposed in the last years comprising different criteria used for encouraging exploration. In [168, 169], the exploration is based on intrinsic motivation. During the training, the agent learns also a model of the system and an exploration bonus is assigned when novel states with respect to the trained model are encountered. Novel states are identified as those states that create a stronger disagreement with the model trained until that moment. Another group of exploration algorithms are count-based methods that directly count the number of times a certain state has been visited to guide the agent towards states less visited. Obviously, such an approach is infeasible in continuous state space. For this reason, some works such as [170] extend count-based exploration approaches to non-tabular (continous) RL using density models to derive a pseudo-count of the visited states. Another approach to encourage exploration consists of injecting noise to the agent's parameters, leading to richer set of agent behaviors during training [171, 172].

These exploration strategies are task agnostic in that they aim at providing good exploration without exploiting any specific information of the task itself. More recently instead, the exploration problem has been cast into *meta-learning* (or learning to learn), the field of machine learning whose goal is to learn strategies for fast adaptation by using prior tasks [173]. An example of application of meta-learning for the exploration problem is shown in [174], where a novel algorithm is presented to learn exploration strategies from prior experience.

Alternatively, it is possible to get around the exploration problem by providing task demonstrations for guiding and speeding up the training [175, 176, 177, 178, 179]. The work presented in [180] shows how the proper incorporation of human demonstrations into RL methods allows reducing the number of samples required for training an agent to solve complex dexterous manipulation tasks with a multi-fingered hand in simulation. Even if verification on a real robot is still to be verified, the training time thus far obtained is compatible with real-world applications. More details about this work are presented in Chapter 9.

Chapter 3

The iCub humanoid robot and its key components for manipulation

In this Chapter, a brief overview of the iCub humanoid robot is provided. More specifically, the description will cover only those components that are relevant for the manipulation problem (see Fig. 3.1).

The iCub [2] is an open-source robotic platform developed for robotics research. It has the appearance of a child and its design is human-inspired. As mentioned in Section 1.1, the iCub is a proper platform for studying the manipulation problem as it is provided with the following components:

- a human-like upper body including two arms and multi-fingered hands (Section 3.1);
- a proper sensor system including two cameras, tactile sensors and encoders for joint angle sensing (Section 3.2);
- a Cartesian controller for the upper body (Section 3.3).

Section 3.4 ends the platform description introducing the software framework of the iCub.

3.1 iCub upper body

The total number of DOFs of the iCub for the complete body is 53. Focusing on the upper body, the (41) DOFs are distributed as follows:

- 6 in the head: 3 for the neck, providing full movements and 3 for the cameras, to support both tilt, pan and vergence[1] behaviors.

[1]The vergence is the simultaneous movement of both eyes in opposite directions to obtain or maintain single binocular vision.

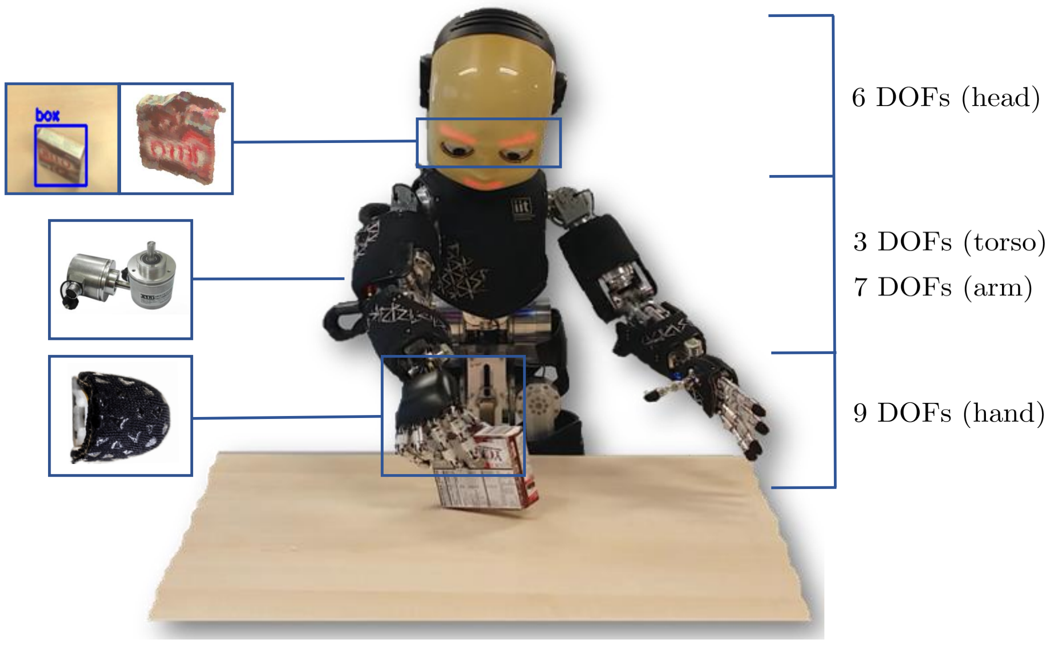


FIGURE 3.1: iCub hardware and perception key components for manipulation. On the left: the sensor system. Stereo vision provides RGB and depth information; encoders are used for proprioception through joints sensing; tactile sensors located on the fingertips detect contacts with the object to manipulate. On the right: how the DOFs are distributed in the upper body.

- 7 for each arm: 3 for the shoulder, 2 for the elbow and 2 for the wrist.
- 9 for each hand: 3 for the thumb, 2 for the index, 2 for the middle finger, 1 for the ring and little fingers and 1 for finger abduction[2]. Consequently, each hand has three independent fingers whereas the fourth and fifth are used for additional stability. Fingers are tendon-driven, with most of the motors located in the forearm. Tendon driven robots (TDR) are robots whose limbs mimic biological musculoskeletal systems, using plastic straps. Such robots are claimed to move in a "more natural" way than traditional robots that use rigid metal or plastic limbs controlled by geared actuators.
- 3 in the waist. A 2 DOF waist/torso would be enough for effective crawling but a 3 DOF waist was incorporated to support manipulation. A 3 DOF waist provides increased range and flexibility of motion for the upper body resulting in a larger workspace for manipulation.

3.2 Perception system

Concerning perception, iCub is equipped different kinds of sensors, including digital cameras (one for each eye), gyroscopes and accelerometers, microphones, encoders, force/torque and tactile sensors. Hereafter, we focus only on those that are mostly relevant for manipulation tasks.

3.2.1 Vision

The RGB cameras mounted are used to perform stereo vision, i.e. the extraction of 3D information from digital images that contain two views of the same scene.

The 3D spatial information of the scene can be obtained by estimating the relative pose of one of the two cameras with respect to the other. This information is coded in the so-called extrinsic parameters of the stereo vision system. Dealing with moving eyes requires the estimation of the extrinsic parameters each time the robot eyes change their relative configuration. This is done on the iCub with a complete Structure From Motion

[2]In physiology, adbuction is the movement which separates a limb or other part from the axis.

pipeline [181]. The extrinsic parameters allows then the rectification of the images and the computation of the disparity map. Finally, the disparity map, the intrinsic parameters of the camera and the forward kinematics of the iCub eyes are used to compute the 3D coordinates of a point in the image expressed with respect to the root frame of the robot.

3.2.2 Tactile sensors

The iCub is equipped with tactile sensors based on capacitive pressure system and covering almost its entire body. The sensors on its arms, torso and legs are mostly used for force estimation and compliant control (see Paragraph 3.3). The tactile sensors on the palm[3] and, in particular, the fingertips of the hands are instead relevant for manipulation. The fingertip structure is the following, outlined in Fig. 3.2.

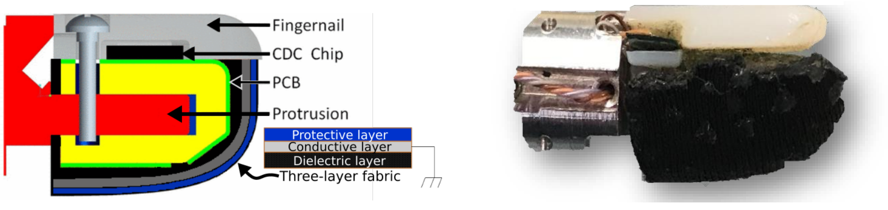


FIGURE 3.2: Cross-section of the fingertip. Yellow: inner support. Green: flexible PCB. Black, grey and blue: the composite three-layer fabric, respectively the dielectric, conductive and protective layers. This structure increases the robustness and repeatability of the fingertip.

A flexible PCB is wrapped around the inner support and hosts 12 sensors called taxels. The PCB also hosts a chip that converts capacitance readings to digital data. A plastic surface of 1mm is used as a mechanical interface to the external environment. It has an inner part that adapts to the shape of the PCB and an external part on which a three-layer fabric is glued. The fabric includes a dieletric layer, a conductive layer, connected to the ground, and a protective textile layer (the black material visible in Fig. 3.3). The conductive lycra is connected to ground. This assembly effectively forms a capacitive pressure sensor.

[3]Due to the hand design, the tactile sensors on the palm are rarely activated during manipulation tasks. The limited hand opening in fact usually prevent the manipulated object to be in contact with the robot palm. For this reason, sometimes we refer to the tactile sensors on the fingertips as the tactile sensors of the hand.

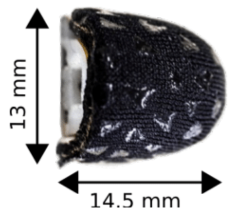


FIGURE 3.3: The protective textile layer on the fingertip.

3.2.3 Proprioception

Proprioceptive inputs in the iCub simply consist of angular position measurements in every joint. For most joints, they are provided by absolute 12bit angular encoders. The joint angles of the hand are sensed using a custom-designed Hall-effect-magnet pair. [182]

3.3 The Cartesian controller

The iCub is provided with a *cartesian controller* [183] for the arms and a *gaze controller* [184] for the head and the vision system. They provide an interface that exploits an inverse kinematics solver in order to control the arms and the head directly in the operational space, by querying 3D points instead of configurations at the joint level and generating trajectories with human-like minimum-jerk velocity profiles [185].

The robot joints can be then controlled with different control modes, including also impedance mode. This mode allows controlling the joint position and its compliance. In particular, both the equilibrium position of a virtual spring and its stiffness/damping are controlled. By tuning the stiffness parameters, the robot joint can behave like a hard or soft spring, while maintaining control on the desired joint position. The torque applied on the robot joints are estimated by combining the force-torque sensors available on the iCub shoulders and the tactile sensors on its upper body. The latter allow better estimating the application point of the torque. The impedance control mode implements a safe way to control robots operating in unstructured environments and interacting with humans.

3.4 Yarp

All the software that runs on the iCub is written using YARP [186]. YARP (Yet Another Robot Platform) is an open-source software framework that supports distributed computation and is compatible with multiple operating systems (Windows, Linux and Mac OS). YARP facilitates the reuse of code by decoupling the user code from the specific hardware (using special device drivers called *PolyDrivers*) and operating system (thanks to OS wrappers). It also enables the development of modular software architectures thanks to an intuitive inter-process communication mechanism based on the concept of Port. A YARP module open ports in order to communicate with other modules and send/receive commands and data. Ports are extremely versatile as they support several types of data (vectors, matrices, images, sounds, point clouds, etc.) and several protocols (e.g. TCP, UDP and many others). YARP also provides several libraries for mathematical computations (vectors, matrices, matrix inversion and singular value decomposition, etc.). Basic image processing is also possible thanks to the integration with the computer vision library OpenCV.

Part II

A novel tactile object localization algorithm: the Memory Unscented Particle Filter

Chapter 4

Memory Unscented Particle Filter for 6-DOF Tactile Object Localization

Accurate object perception is a necessary requirement for the execution of manipulation tasks with autonomous robots in real-world environments. This makes the advances in robotic manipulation and perception strongly related to each other. In particular, the development of new sensors and inference algorithms enhance the robot ability to deal with uncertainties and reduce the cost required to engineer the environment in which the robot will operate. Recently, a great interest arose regarding the use of tactile sensors for manipulation tasks [7, 8, 9, 10, 11, 12]. While the use of vision has been thoroughly investigated [187], recent advances in tactile technology have made it possible to build tactile systems that are reliable enough and can be deployed on real robots at a reasonable price [21, 22, 23]. This recent improvement of tactile sensors is surely one of the reasons for a surge of interest on this topic [15, 16, 17, 19, 20]. But tactile sensing is also fundamental whenever vision is unavailable or imprecise, for example due to occlusions and/or bad lighting conditions. In addition, findings in human physiology testify how humans jointly exploit vision and touch in order to accomplish manipulation tasks and how humans are even able to explore objects by means of tactile perception solely [13], [14].

Due to technological limitations, most tactile systems have low resolution and rarely provide other than estimation of the force normal to the surface. Object localization using tactile feedback is, therefore, challenging and requires the development of filtering techniques that allow appropriate fusion of multiple measurements, taking into account the presence of noise and the real-time requirements of the task.

In this Chapter, we present a novel algorithm, named *Memory Unscented Particle Filter (MUPF)* [40], designed for addressing global 6-DOF tactile object localization. This algorithm relies on the *Unscented Particle Filter* (UPF) [188] and exploits only the measured position of the contact points obtained from the tactile sensors on the robot. Other works in the literature instead take advantage of other measurements such as the surface normal at the contact points [34, 189, 190] or a 6-dimensional vector including force and torque [31]. The proposed solution is inherently recursive in that the measurements are sequentially processed in real time as they become available, and the algorithm can provide the object's pose estimate after the processing of each measurement. We take into account a recursive approach for several reasons: the algorithm can provide the object's pose estimate after the processing of each measurement, and not only at the final measurement acquisition time as with a batch procedure like the one in [34, 35]; it is compliant with active exploration techniques where the robot decides, at each time t, where to sense next on the basis of the current object's pose estimate; it can allow stopping the object localization procedure at a given time t whenever a suitable stopping criterion is satisfied.

The Chapter is organized as follows. Section 4.1 is a brief introduction to nonlinear filtering techniques useful for the subsequent theoretical developments. Section 4.2 provides a mathematical (Bayesian) formulation of the tactile localization problem. Section 4.3 presents the novel *Memory Unscented Particle Filter* (MUPF) approach to 6-DOF tactile localization. Section 4.4 demonstrate the effectiveness of the proposed approach by means of simulation and experimental tests on the iCub humanoid robot. Finally, Sections 4.5 ends the Chapter with concluding remarks, applications and perspectives for future work.

4.1 Mathematical background

Hereafter, *tactile localization* is cast into the Bayesian framework and addressed as a nonlinear multimodal filtering problem. Recall that filtering is the problem of recursively estimating the state $x_t \in \mathbb{R}^n$ of a dynamical system while acquiring and processing noisy observations on-line. Specifically, from a Bayesian viewpoint, the goal of the filtering problem is to recursively compute the following conditional PDFs

$$\begin{aligned} p_{t|t}(\boldsymbol{x}) &= p(\boldsymbol{x}_t = \boldsymbol{x}|\boldsymbol{y}^t) \\ p_{t+1|t}(\boldsymbol{x}) &= p(\boldsymbol{x}_{t+1} = \boldsymbol{x}|\boldsymbol{y}^t), \end{aligned} \tag{4.1}$$

given the noisy observations $\boldsymbol{y}^t = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$ with $\boldsymbol{y}_t \in \mathbb{R}^p$.

The solution of the filtering problem is given by the Bayesian recursion, starting from the initial prior $p_{1|0}(\cdot)$ and consisting of two functional equations, i.e. the following Bayes and respectively Chapman-Kolmogorov equations:

$$p_{t|t}(\boldsymbol{x}) = \frac{\ell_t(\boldsymbol{y}_t|\boldsymbol{x})p_{t|t-1}(\boldsymbol{x})}{\int \ell_t(\boldsymbol{y}_t|\boldsymbol{\xi})p_{t|t-1}(\boldsymbol{\xi})d\boldsymbol{\xi}} \tag{4.2}$$

$$p_{t+1|t}(\boldsymbol{x}) = \int \varphi_{t+1|t}(\boldsymbol{x}|\boldsymbol{\xi})p_{t|t}(\boldsymbol{\xi})d\boldsymbol{\xi}, \tag{4.3}$$

where $\varphi_{t+1|t}(\boldsymbol{x}|\boldsymbol{\xi})$ is the *Markov transition density* representing the conditional probability that the state at time $t+1$ will take value $\boldsymbol{x}$ given that the state at time t is equal to $\boldsymbol{\xi}$, and $\ell_t(\boldsymbol{y}|\boldsymbol{x})$ is the *measurement likelihood function* denoting the probability that the measurement at time t will take value $\boldsymbol{y}$ given that the state is equal to $\boldsymbol{x}$.

However, in many practical applications, such as navigation, tracking and localization, the transition and likelihood models are usually affected by nonlinearities and/or non-Gaussian noise distributions, thus precluding analytical solutions of (4.2) and (4.3). In these cases, one must invariably resort to some approximation technique. Most of the existing approximation techniques can be divided in two families: Kalman-filtering-like approaches, and sequential Monte Carlo methods. The algorithms belonging to the former family (like the *Extended Kalman filter* and the *Unscented Kalman filter (UKF)* [191]-[192]) propagate only the first- and second-order moments (i.e., mean and covariance) of the posterior state distribution. Such methods are usually characterized by a low computational cost, but are not appropriate for multimodal distributions like the one arising from the tactile localization problem. On the other hand, sequential Monte Carlo methods, also known as *particle filters* [193], can deal with arbitrary nonlinearities and distributions and supply a complete representation of the posterior state distributions.

Particle filtering techniques stem from the idea of approximating the posterior density function $p_{t|t}(\boldsymbol{x})$ by means of a finite set of weighted samples (particles) as

$$\hat{p}_{t|t}(\boldsymbol{x}) \approx \sum_{i=1}^{N} \tilde{w}_t^{(i)} \, \delta(\boldsymbol{x} - \boldsymbol{x}_{t|t}^{(i)}), \tag{4.4}$$

where $\delta(\cdot)$ is the Dirac delta function, $\boldsymbol{x}_{t|t}^{(i)}$ is the position of the i-th particle and $\tilde{w}_t^{(i)}$ its normalized importance weight. In this way, the evaluation of the integrals that are necessary for application of the Bayesian filtering equations (4.2) and (4.3) is performed via the Monte Carlo numerical integration method, i.e., by transforming the integrals into discrete sums.

In principle, the particle approximation (4.4) can be computed by drawing a set of independent and identically distributed samples $\{\boldsymbol{x}_{t|t}^{(i)}, i = 1, \ldots, N\}$ from the posterior $p_{t|t}(\boldsymbol{x})$. While such a solution is not feasible because $p_{t|t}(\boldsymbol{x})$ is not known, the difficulty can be circumvented by sampling each particle i from a known, easy-to-sample, *proposal distribution* $q^{(i)}(\boldsymbol{x}_t|\boldsymbol{y}^t)$, and then compute the normalized importance weights as

$$w_t^{(i)} = \tilde{w}_{t-1}^{(i)} \frac{\ell_t(\boldsymbol{y}_t|\boldsymbol{x}_{t|t}^{(i)}) \; \varphi_{t|t-1}(\boldsymbol{x}_{t|t}^{(i)}|\boldsymbol{x}_{t|t-1}^{(i)})}{q^{(i)}(\boldsymbol{x}_{t|t}^{(i)}|\boldsymbol{y}^t)}, \tag{4.5}$$

$$\tilde{w}_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)} . \tag{4.6}$$

In fact, by comparing (4.5) with (4.2) and (4.3), it is an easy matter to see that the resulting particle-based description approximates the true posterior $p_{t|t}(\boldsymbol{x})$ at time t.

4.1.1 The Unscented Particle Filter

The main drawback of particle filtering techniques is that, unless special care is taken, the number N of particles needed to make the approximation (4.4) sufficiently accurate can increase exponentially with the dimension n of the vector to be estimated (since it is required to sample in a subset of $\mathbb{R}^n$). In this respect, a critical point of particle filtering is how to choose the proposal distribution $q^{(i)}(\boldsymbol{x}_t|\boldsymbol{y}^t)$ so as to approximate the posterior reasonably well with a moderate number of particles. Among the most effective variations, there is the *unscented particle filter* (UPF) which exploits the UKF in the proposal distribution to improve performance [188]. In the following part of this section, an outline of the UPF algorithm is provided.

The UPF propagates a set of extended particles $\mathcal{P}_t = \{\mathcal{P}_t^{(1)}, \ldots, \mathcal{P}_t^{(N)}\}$, each one comprising a weight $\tilde{w}_t^{(i)}$, a mean $\boldsymbol{x}_{t|t}^{(i)}$, and a covariance $P_{t|t}^{(i)}$, i.e.,

$$\mathcal{P}_t^{(i)} = \{\tilde{w}_t^{(i)}, \boldsymbol{x}_{t|t}^{(i)}, P_{t|t}^{(i)}\}\,.$$

Given the set of particles at time $t-1$, the UKF prediction and correction steps are applied to each particle mean and covariance so as to move the particle towards the measurements. Then, for each i, a new particle is sampled using $\mathcal{N}(\boldsymbol{x}_t; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)})$ as proposal distribution where $\bar{\boldsymbol{x}}_t^{(i)}$ is the updated mean after the correction step, $P_{t|t}^{(i)}$ is the updated covariance, and $\mathcal{N}(\boldsymbol{x}; \bar{\boldsymbol{x}}, P)$ denotes the normal distribution with mean $\bar{\boldsymbol{x}}$ and covariance P, thus achieving a more dense sampling in the most relevant areas of the search space.

In order to apply the UKF to each particle, it is necessary to assume that the Markov transition density $\varphi_{t+1|t}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$ and measurement likelihood function $\ell_t(\boldsymbol{y}_t|\boldsymbol{x}_t)$ are generated by a state transition and, respectively, measurement equation, so that the time evolution of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ can be described by the discrete-time dynamical system

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_t(\boldsymbol{x}_t, \boldsymbol{\omega}_t) \tag{4.7}$$

$$\boldsymbol{y}_t = \boldsymbol{h}_t(\boldsymbol{x}_t, \boldsymbol{\nu}_t). \tag{4.8}$$

Notice that in system (4.7)-(4.8) the probabilistic nature of the model is captured by the *process disturbance* $\boldsymbol{\omega}_t$ and *measurement noise* $\boldsymbol{\nu}_t$, which are supposed to be sequences of independent random variables with known probability density functions.

The UKF does not directly approximate the nonlinear process and observation models, but exploits the nonlinear models, approximating the distribution of the state. This is made possible by means of the *scaled unscented transformation (SUT)* [194], which is a tool for computing the statistics of a random variable undergoing a nonlinear transformation. Specifically, the state distribution is specified using a minimal set of deterministically chosen sample points. Such sample points exactly provide the true mean and covariance of such a variable and, when propagated through the nonlinear transformation, they approximate the posterior mean and covariance accurately to the *2nd order* for any nonlinearity. For the reader's convenience, a brief review of the SUT is provided hereafter.

Let $\boldsymbol{x} \in \mathbb{R}^{n_x}$ be a random variable, with mean $\bar{\boldsymbol{x}}$ and covariance P_x, and $\boldsymbol{g} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ an arbitrary nonlinear function. The goal is to approximate the mean value $\bar{\boldsymbol{y}}$ and covariance P_y of the variable $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$. A set of $2n_x + 1$ weighted samples or *sigma points* $\mathcal{S}_i = \{\mathcal{W}_i, \boldsymbol{\mathcal{X}}_i\}_{i=0}^{2n_x}$ are chosen to completely represent the true mean and covariance of the variable $\boldsymbol{x}$, i.e.

$$\begin{aligned}
\boldsymbol{\mathcal{X}}_0 &= \bar{\boldsymbol{x}} \\
\boldsymbol{\mathcal{X}}_i &= \bar{\boldsymbol{x}} + (\sqrt{(n_x + k)P_x})_i \qquad i = 1, \dots, n_x \\
\boldsymbol{\mathcal{X}}_i &= \bar{\boldsymbol{x}} - (\sqrt{(n_x + k)P_x})_i \qquad i = n_x + 1, \dots, 2n_x \\
\boldsymbol{\mathcal{W}}_0^{(m)} &= \lambda/(n_x + \lambda) \\
\boldsymbol{\mathcal{W}}_0^{(c)} &= \lambda/(n_x + \lambda) + (1 - \alpha^2 + \beta) \\
\boldsymbol{\mathcal{W}}_i^{(m)} &= \boldsymbol{\mathcal{W}}_i^{(c)} = 1/[2(n_x + \lambda)] \qquad i = 1, \dots, 2n_x,
\end{aligned}$$

where: $\lambda = \alpha^2(n_x + k) - n_x$; $\alpha > 0$ provides one more degree of freedom to control the scaling of the sigma points and to avoid the possibility of getting a non-positive semi-definite covariance; $k \geq 0$ is another scaling parameter; β affects the weighting of the zero-*th* sigma point.

Each sigma point is then propagated through the function $\boldsymbol{g}(\cdot)$ ($\boldsymbol{\mathcal{Y}}_i = \boldsymbol{g}(\boldsymbol{\mathcal{X}}_i)$, for $i = 0, \dots, 2n_x$) and the estimated mean and covariance of $\boldsymbol{y}$, as well as the cross-covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$, are computed as follows:

$$\begin{aligned}
\bar{\boldsymbol{y}} &= \sum_{i=0}^{2n_x} \mathcal{W}_i \boldsymbol{\mathcal{Y}}_i \qquad P_y = \sum_{i=0}^{2n_x} \mathcal{W}_i (\boldsymbol{\mathcal{Y}}_i - \bar{\boldsymbol{y}})(\boldsymbol{\mathcal{Y}}_i - \bar{\boldsymbol{y}})^T \\
P_{xy} &= \sum_{i=0}^{2n_x} \mathcal{W}_i (\boldsymbol{\mathcal{X}}_i - \bar{\boldsymbol{x}})(\boldsymbol{\mathcal{Y}}_i - \bar{\boldsymbol{y}})^T.
\end{aligned} \tag{4.9}$$

The Unscented Kalman Filter is obtained by applying the SUT to the nonlinear functions f_t and h_t in (4.7)-(4.8).

In practice, in the UPF algorithm, given the mean $\boldsymbol{x}_{t-1|t-1}^{(i)}$ and covariance $P_{t-1|t-1}^{(i)}$ at time $t-1$, as well as the mean and covariance of the process disturbance $\boldsymbol{\omega}_{t-1}$, application of the SUT to the state transition equation (4.7) allows to compute an approximation of the predicted mean $\boldsymbol{x}_{t|t-1}^{(i)}$ and covariance $P_{t|t-1}^{(i)}$ at time t. In turn, given $\boldsymbol{x}_{t|t-1}^{(i)}$ and $P_{t|t-1}^{(i)}$ as well as the mean and covariance of the measurement noise $\boldsymbol{\nu}_t$, application of the SUT to the measurement equation (4.8) allows to provide an approximation of the predicted measurement mean $\boldsymbol{y}_{t|t-1}^{(i)}$ and covariance $S_t^{(i)}$ as well as of the state-measurement cross-covariance matrix $\Gamma_t^{(i)}$. Then, the updated mean $\bar{\boldsymbol{x}}_t^{(i)}$ and covariance $P_{t|t}^{(i)}$ are obtained by applying the standard Kalman filter correction step.

Since in practice it can happen that, after a few iterations, one of the normalized weights tends to 1, while the remaining weights tend to zero (*weight degeneration*), a selection or *resampling* stage is usually included in the particle filtering algorithm, in order to eliminate samples with low importance weights and replicate samples with high importance weights. Summing up, the resulting algorithm is reported in Algorithm 1.

4.2 Problem formulation

The object to be localized is assumed to be static during the measurement collection. This assumption is common to other works [31, 34, 35, 190] and is realistic, for instance, if the object is very heavy or is stuck on a support preventing any possible movement. Hence, the goal of the 6-DOF object tactile localization problem is to estimate in real-time the pose $\boldsymbol{x} \in \mathbb{R}^6$ of an object $\mathcal{O}$ of known shape, on the basis of the tactile measurements $\boldsymbol{y}^t = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$ collected up to the current time instant t. The minimal pose representation of the object is given by the 6-dimensional state vector $\boldsymbol{x}$, consisting of the coordinates of the center of the reference system attached to the object and the three Euler angles representing the orientation, i.e.

$$\boldsymbol{x} = \begin{bmatrix} x, & y, & z, & \phi, & \theta, & \psi \end{bmatrix}^T. \tag{4.10}$$

The measurements are collected by touching the object with the end effector of the robot. Each measurement $\boldsymbol{y}_t$ consists of the acquired Cartesian position of the contact point, i.e.:

$$\boldsymbol{y}_t = \begin{bmatrix} x_{t,p}, & y_{t,p}, & z_{t,p} \end{bmatrix}^T. \tag{4.11}$$

Algorithm 1 The Unscented Particle Filter

```
1: **for** $i = 1, \ldots, N$ **do**
2:     draw the state particles $\boldsymbol{x}_{0|0}^{(i)}$ from the prior $p_{0|0}(\boldsymbol{x})$ and set $P_{0|0}^{(i)} = P_0$,
3:     and $\tilde{w}_0^{(i)} = 1/N$;
4: **end for**
5: **for** $t = 1, 2, \ldots,$ **do**
6:     **1) UKF prediction and correction**
7:     **for** $i = 1, \ldots, N$ **do**
8:         - **Time update**:
9:         given $\{\boldsymbol{x}_{t-1|t-1}^{(i)}, P_{t-1|t-1}^{(i)}\}$, compute $\{\boldsymbol{x}_{t|t-1}^{(i)}, P_{t|t-1}^{(i)}\}$ by applying
10:        the SUT to the state transition equation (4.7);
11:        - **Measurement prediction**:
12:        given $\{\boldsymbol{x}_{t|t-1}^{(i)}, P_{t|t-1}^{(i)}\}$, compute $\{\boldsymbol{y}_{t|t-1}^{(i)}, S_t^{(i)}, \Gamma_t^{(i)}\}$
13:        by applying the SUT to the measurement equation (4.8);
14:        - **Measurement update**: set
               $K_t^{(i)} = \Gamma_t^{(i)} \left(S_t^{(i)}\right)^{-1}$
               $\bar{\boldsymbol{x}}_t^{(i)} = \boldsymbol{x}_{t|t-1}^{(i)} + K_t^{(i)} (\boldsymbol{y}_t - \boldsymbol{y}_{t|t-1}^{(i)})$
               $P_{t|t}^{(i)} = P_{t|t-1}^{(i)} - K_t^{(i)} S_t^{(i)} \left(K_t^{(i)}\right)^T;$
15:    **end for**
16:    **2) Weight update**
17:    **for** $i = 1, \ldots, N$ **do**
18:        sample from the proposal distribution:
               $\hat{\boldsymbol{x}}_t^{(i)} \sim q^{(i)}(\cdot|\boldsymbol{y}^t) = \mathcal{N}(\cdot; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)});$
19:        evaluate and normalize the importance weights:
               $w_t^{(i)} = \tilde{w}_{t-1}^{(i)} \frac{\ell_t(\boldsymbol{y}_t|\hat{\boldsymbol{x}}_t^{(i)})\ \varphi_{t|t-1}(\hat{\boldsymbol{x}}_t^{(i)}|\boldsymbol{x}_{t|t-1}^{(i)})}{\mathcal{N}(\hat{\boldsymbol{x}}_t^{(i)}; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)})}$
               $\tilde{w}_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)};$
20:    **end for**
```

```
21:     3) Resampling
22:     for $i = 1, \dots, N$ do
23:         draw $j \in \{1, \dots, N\}$ with probability $\tilde{w}_t^{(j)}$ and set:
                $x_{t|t}^{(i)} = \hat{x}_t^{(j)} \quad P_{t|t}^{(i)} = P_{t|t}^{(j)} \quad \tilde{w}_t^{(i)} = \frac{1}{N}.$
24:     end for
25: end for
```

It is worth noticing that the exploited measurements consist only of tridimensional contact point vectors. Notice also that, while for ease of presentation it is assumed that a measurement consists of a single contact point, the proposed approach is well-suited to being extended to measurements consisting of multiple contact points (corresponding to different fingertips touching the object). This would simply amount to processing, at each time t, a measurement vector of size $3n_t$, n_t being the number of fingertips touching the object at that time. Finally, notice that, in the sequel, all measurements and the object pose will be assumed to be expressed in the same, fixed, reference system.

4.2.1 Considerations on the motion model

Since the object is assumed to be static, the 6-DOF object tactile localization problem is basically a static parameter estimation problem. In this respect, it is well known that the use of particle filtering techniques for estimating static parameters requires special care, because a direct application of these techniques to the constant state equation $x_{t+1} = x_t$, corresponding to the Markov transition density $\varphi_{t+1|t}(x|\xi) = \delta(x - \xi)$, would incur in the so-called weight-degeneracy phenomenon. Many solutions have been proposed in the literature to circumvent such a problem, see for instance [195] and the references therein. A simple but effective approach consists of adding an artificial dynamic noise on the static parameter by considering a state-transition equation of the form

$$x_{t+1} = x_t + \omega_t, \tag{4.12}$$

where ω_t is the artificial dynamic noise which is modeled as a Gaussian random variable with zero mean and suitable covariance Q_t. The idea is that the artificial evolution provides a mechanism for generating at each time instant new particles with a sufficiently diffuse distribution. In this work, a time-invariant covariance matrix is used, i.e. $Q_t = Q$, since it proves effective in the considered case studies. However more elaborated solutions can be easily incorporated within the proposed algorithm [195].

Some considerations on the possibility of extending the approach to the case of moving object localization are provided in the subsequent Remark 2.

4.2.2 Measurement model

In order to apply the UPF to the tactile localization problem under investigation, it is necessary to define the measurement model both in terms of a likelihood function $\ell_t(\boldsymbol{y}_t|\boldsymbol{x}_t)$ and of a measurement function $\boldsymbol{h}_t(\cdot,\cdot)$. The proposed *likelihood function* is based on the so-called *proximity model*, in which the measurements are considered independent of each other and corrupted by Gaussian noise. For each measurement, the likelihood function depends on the distance between the measurement and the object, hence the name "proximity". This model is the adaptation of the likelihood proposed in [34] to the case of contact point measurements only.

Let the 3D object model be represented by a polygonal mesh consisting of faces $\{f_i\}$. For each face f_i, let $\ell_{t,i}(\boldsymbol{y}_t|\boldsymbol{x}_t)$ be the likelihood of the measurement $\boldsymbol{y}_t$ relative to that face when the object is in the pose $\boldsymbol{x}_t$. Then, the likelihood of the measurement is defined as the maximum likelihood over all faces, i.e.

$$\ell_t(\boldsymbol{y}_t|\boldsymbol{x}_t) \propto \max_i \ell_{t,i}(\boldsymbol{y}_t|\boldsymbol{x}_t), \tag{4.13}$$

apart from a normalizing factor which, however, is independent of the state $\boldsymbol{x}_t$ and needs not necessarily to be computed.

Each likelihood is assumed to be Gaussian, with variance σ_p^2, and can be computed as follows:

$$\ell_{t,i}(\boldsymbol{y}_t|\boldsymbol{x}_t) = \frac{1}{\sqrt{2\pi}\sigma_p} \exp\left(-\frac{1}{2}\,\frac{d_i(\boldsymbol{y}_t,\boldsymbol{x}_t)^2}{\sigma_p^2}\right), \tag{4.14}$$

where the quantity $d_i(\boldsymbol{y}_t,\boldsymbol{x}_t)$ is the shortest Euclidean distance of $\boldsymbol{y}_t$ from the face f_i when the object is in the pose $\boldsymbol{x}_t$. For instance, supposing that f_i is the representation of the i-th face in the object reference system, the distance $d_i(\boldsymbol{y}_t, \boldsymbol{x}_t)$ can be computed as

$$d_i(\boldsymbol{y}_t, \boldsymbol{x}_t) = \min_{\boldsymbol{p} \in f_i} \|\boldsymbol{y}_t^{\boldsymbol{x}_t} - \boldsymbol{p}\|,$$

where $\|\cdot\|$ is the Euclidean norm and $\boldsymbol{y}_t^{\boldsymbol{x}_t}$ denotes the transformation of the measurement $\boldsymbol{y}_t$ using the roto-translation matrix corresponding to the state $\boldsymbol{x}_t$.

Notice that the considered measurement model does not take *negative information* into account. In other words, the points of the search space exploited to compute the likelihood function are only the ones on the object surface touched during the collection of measurements, while the information provided by the lack of contact in some sub-regions of the search space is not taken into account in the likelihood function. Even if the negative information can also support object localization, it is not exploited in this method in order to keep the computational complexity moderate.

As previously pointed out, the use of the UPF requires also the definition of a *measurement function*, namely a mathematical mapping giving the measurement $\boldsymbol{y}_t$ as a function of the current state $\boldsymbol{x}_t$ and a measurement noise $\boldsymbol{\nu}_t$, see (4.8). For the sake of simplicity, a measurement equation with additive noise is taken into account, i.e.,

$$\boldsymbol{y}_t = \boldsymbol{h}_t(\boldsymbol{x}_t) + \boldsymbol{\nu}_t \,. \tag{4.15}$$

In particular, the measurement function is required to compute the Scaled Unscented Tranform (SUT) in the measurement prediction step of the Unscented Kalman Filter.

It is important to highlight how the definition of a measurement equation is different from the one of a likelihood function: given the state and the measurement noise, the measurement equation provides a measurement value - a contact point in the present case - whereas the likelihood function is proportional to the probability of having a certain measurement for a given state.

Tactile sensors are atypical sensors from this standpoint. In fact, typical sensors, e.g. radars, are characterized by a mathematical relationship between the current state of the object and the provided measurement: given the state of the object, the measurement of the object position and orientation supplied by the sensor remains unchanged (neglecting the measurement noise).

On the other hand, the employment of tactile sensors makes the scenario quite different. The measurement is given by the tactile sensor pose itself, i.e., the forward kinematics of the end effector of the robot, only if the robot actually touches the object. Thus, if the object is in a generic state and the sensor in a specific pose, it cannot be taken for granted that such a configuration provides a contact measurement. Moreover, the sensor moves during the measurement collection, while the object is motionless. It is not possibile to predict unambiguously the measurement value without a model of the sensor motion: given the pose of the sensor and the object distance from it, the predicted measurement is not unique, since the sensor could touch the object in different points.

Nevertheless, in order to compute a predicted measurement for each possible configuration $\boldsymbol{x}$, it is necessary to define a measurement equation capable of handling also the case in which there is no actual contact between the sensor and the object in the considered pose $\boldsymbol{x}$ (in particular, the sigma point of the i-th predicted particle). Further, the predicted measurement should be consistent with the proximity-based likelihood (4.13).

To this end, it is useful to provide an alternative interpretation of the likelihood (4.13). Notice first that, due to the measurement noise, the measurement $\boldsymbol{y}_t$ does not represent the actual contact point between the sensor and the object, which however will be in the neighborhood of $\boldsymbol{y}_t$. The proximity model assumes that the actual contact point is the point on the object surface which is closest to the measurement $\boldsymbol{y}_t$. In fact, equation (4.13) can be rewritten as

$$\ell_t(\boldsymbol{y}_t|\boldsymbol{x}_t) \propto \exp\left(-\frac{1}{2\sigma_p^2}\ \|\boldsymbol{y}_t - \boldsymbol{h}_t(\boldsymbol{x}_t)\|^2\right) \tag{4.16}$$

where

$$\boldsymbol{h}_t\left(\boldsymbol{x}_t\right) = \arg\min_{\boldsymbol{p}\in\partial\mathcal{O}^{\boldsymbol{x}_t}} \|\boldsymbol{y}_t - \boldsymbol{p}\| \tag{4.17}$$

and $\partial\mathcal{O}^{\boldsymbol{x}_t}$ represents the object boundary in the pose $\boldsymbol{x}_t$ with respect to the robot reference system. Then, the likelihood of the measurement $\boldsymbol{y}_t$ depends on its distance from such a hypothetical contact point according to a Gaussian distribution. Accordingly, given a configuration $\boldsymbol{x}_t$, the corresponding predicted measurement is selected as the point of the object surface which is closest to the measurement $\boldsymbol{y}_t$. Such a choice turns out to be consistent with the proximity likelihood model. In fact, by taking the additive measurement noise $\boldsymbol{\nu}_t$ in (4.15) as a Gaussian random variable with zero-mean and covariance $\sigma_p^2 I$, with I the identity matrix, it is an easy matter to see that (4.15) and (4.17) give rise precisely to a likelihood of the form (4.16).

4.3 The Memory Unscented Particle Filter

The main challenges of the 6-DOF tactile localization problem are its dimension (6-DOFs), its multimodal nature, and the fact that individual measurements are relatively uninformative, since they are tridimensional vectors in a 6-DOF space. In particular, the latter fact implies that the standard UPF algorithm is not well suited to this problem. In fact, Algorithm 1 uses, at each time instant t, only the current measurement $\boldsymbol{y}_t$ in order to compute the importance weights $w_t^{(i)}$. However, since a single contact point measurement is unable to completely characterize the object's pose (lack of observability), the standard weights do not provide enough information to understand which particles must be replicated and which ones must be eliminated in the subsequent resampling step. Thus, performing the standard resampling step - and then discarding some particles - on the basis of such weights is problematic: some potential representative particles could be cut off and the algorithm could limit the search to wrong sub-regions.

In order to overcome such a drawback, we propose a novel variant of the UPF, referred to as Memory UPF (MUPF). The idea is to use also past measurements to update particle weights so as to preserve their ability to characterize particle goodness. Since the object is static, all the measurements refer to the same pose and, in principle, at each time t all the measurements $\boldsymbol{y}^t$ collected up to the current time could be used to compute the importance weights. However, this solution would entail a computational effort growing in time. To avoid such a growth of complexity, the proposed approach follows a *moving window* strategy, i.e., by using, at each time instant, a sliding window consisting of the most recent m measurements. In this way, at each time instant, the weight computation requires

$O(Nm)$ likelihood evaluations, and the size m of the sliding window can be chosen according to the available computational capabilities.

In practice, the particles $\{\hat{\boldsymbol{x}}_t^{(i)}\}_{i=1}^N$ and the set of independent measurements $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$, collected up to the current instant t, are used to compute the weights by:

$$w_t^{(i)} = \frac{\tilde{w}_{t-1}^{(i)} \cdot \prod_{k=\bar{k}(t)}^{t} \ \ell(\boldsymbol{y}_k | \hat{\boldsymbol{x}}_t^{(i)})}{\mathcal{N}(\hat{\boldsymbol{x}}_t^{(i)}; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)})}, \tag{4.18}$$

$$\tilde{w}_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)} \tag{4.19}$$

for $i = 1, \ldots, N$, where

$$\bar{k}(t) = \begin{cases} t - m + 1, & \text{if } t - m + 1 \geq 1 \\ 1, & \text{otherwise.} \end{cases} \tag{4.20}$$

Of course, the reuse of measurements in the update of the particles' weights modifies the nature of the approximation, and hence special care needs to be taken in order to retrieve the pose estimate in a theoretically sound way. To see this, observe preliminarily that the addressed problem is inherently of a multimodal nature, since in the presence of symmetries in the object, there might exist multiple values of $\boldsymbol{x}$ compatible with the measurements. Then, taking the expected value as estimate is not meaningful. Instead, a maximum *a posteriori* probability (MAP) criterion can be followed by taking as pose estimate at time t the corrected particle $\hat{\boldsymbol{x}}_t^{(i)}$ corresponding to the highest value of the estimated posterior distribution [196].

Recalling that each corrected particle after the weight update can be considered corresponding to a Gaussian distribution with mean $\hat{\boldsymbol{x}}_t^{(i)}$ and covariance $P_{t|t}^{(i)}$, one might be tempted to take as estimated posterior $\hat{p}_{t|t}(\cdot)$ the function

$$\hat{p}_{t|t}(\boldsymbol{x}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \mathcal{N}(\boldsymbol{x}; \hat{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)}). \tag{4.21}$$

Unfortunately, such a choice would not be theoretically sound due to the multiple use of measurements in the weight computation. In this respect, notice first that, since the object is static, the 6-DOF localization problem is a parameter estimation problem and, hence, the true posterior $p_{t|t}(\cdot\,)$ at time t takes the form

$$p_{t|t}(\boldsymbol{x}) \propto \prod_{k=1}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})\, p_0(\boldsymbol{x}), \tag{4.22}$$

where $p_0(\cdot)$ is a PDF reflecting the prior knowledge on the object configuration. Since at each time instant multiple measurements are used in the weight computation, the estimated posterior $\hat{p}_{t|t}(\cdot\,)$ does not approximate the true one $p_{t|t}(\cdot\,)$ but instead it approximates the PDF

$$\tilde{p}_{t|t}(\boldsymbol{x}) \propto \prod_{k=1}^{\bar{k}(t)-1} \ell(\boldsymbol{y}_k|\boldsymbol{x})^m \prod_{k=\bar{k}(t)}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})^{t+1-k}\, p_{0|0}(\boldsymbol{x}), \tag{4.23}$$

where $p_{0|0}(\cdot)$ is the prior density used in the generation of the initial particles, thus introducing an undesired warp in the form of the estimated posterior PDF.

This drawback can be circumvented by computing special weights $\bar{w}_t^{(i)}$, used only for the purpose of pose estimation extraction but not propagated in the recursion. In fact, by setting

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)} \cdot \prod_{k=\bar{k}(t)}^{t} \ell(\boldsymbol{y}_k|\hat{\boldsymbol{x}}_t^{(i)})^{m-t+k-1}}{\mathcal{N}(\hat{\boldsymbol{x}}_t^{(i)}; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)})}, \tag{4.24}$$

$$\bar{w}_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)} \tag{4.25}$$

for $i = 1, \dots, N$, and using the estimated posterior

$$\hat{p}_{t|t}(\boldsymbol{x}) = \sum_{i=1}^{N} \bar{w}_t^{(i)}\, \mathcal{N}(\boldsymbol{x}; \hat{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)}) \tag{4.26}$$

in place of (4.21), it turns out that such a $\hat{p}_{t|t}(\cdot\,)$ approximates the PDF

$$\bar{p}_{t|t}(\boldsymbol{x}) \propto \prod_{k=1}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})^m\, p_{0|0}(\boldsymbol{x}) \tag{4.27}$$

so that all measurements provide the same contribution to the estimation problem. Then, by choosing $p_{0|0}(\boldsymbol{x}) \propto p(\boldsymbol{x}_0)^m$, we obtain $\bar{p}_{t|t}(\boldsymbol{x}) \propto p_{t|t}^m(\boldsymbol{x})$

which implies that $\bar{p}_{t|t}(\boldsymbol{x})$ and $p_{t|t}(\boldsymbol{x})$ share the same maximum points. In turn, this implies that application of the MAP estimation criterion to $\bar{p}_{t|t}(\boldsymbol{x})$ is equivalent to computing the MAP estimate according to $p_{t|t}(\boldsymbol{x})$. These considerations allow concluding that, with the choice $p_{0|0}(\boldsymbol{x}) \propto p_0(\boldsymbol{x})^m$, the MAP estimate $\hat{\boldsymbol{x}}_t$ corresponding to the particle with the maximum a posteriori probability according to (4.26)

$$\hat{\boldsymbol{x}}_t = \arg\max_j \hat{p}_{t|t}(\hat{\boldsymbol{x}}_t^{(j)}) = \tag{4.28}$$

$$= \arg\max_j \sum_{i=1}^{N} \bar{w}_t^{(i)} \mathcal{N}(\hat{\boldsymbol{x}}_t^{(j)}; \hat{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)}). \tag{4.29}$$

is coherent with the true posterior PDF.

Remark 1 *The fact that (4.26) approximates (4.27) can be shown by noting that $\bar{p}_{t|t}(\boldsymbol{x})$ can be decomposed as follows*

$$\begin{aligned}\bar{p}_{t|t}(\boldsymbol{x}) &\propto \prod_{k=\bar{k}(t)}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})^{m-t+k-1} \\ &\times \prod_{k=\bar{k}(t)}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})^{t+1-k} \prod_{k=1}^{\bar{k}(t)-1} \ell(\boldsymbol{y}_k|\boldsymbol{x})^m \, p_{0|0}(\boldsymbol{x}) \\ &= \prod_{k=\bar{k}(t)}^{t} \ell(\boldsymbol{y}_k|\boldsymbol{x})^{m-t+k-1} \, \tilde{p}_{t|t}(\boldsymbol{x})\end{aligned}$$

which precisely corresponds to the weight update in (4.24).

A further modification, as compared to the standard UPF, pertains to the resampling step. Since in the first iterations only few measurements are available (thus providing insufficient information), all the particles are retained so as to account for more possibile solutions, in accordance with the multimodal nature of the problem. This amounts to skipping the standard resampling step for a certain number t_0 of initial time instants (in the experimental results reported in the following sections, for the first two time instants). The degeneration of the weights in the first iterations is avoided by setting the weights of all particles equal to $1/N$.

Summing up, the proposed MUPF algorithm is shown in Algorithm 2. The term *Memory*, in the name of the proposed algorithm, is due to the computation of the weights: at each iteration a non-decreasing number of

Algorithm 2 The Memory Unscented Particle Filter

```
1: for $i = 1, \dots, N$ do
2:     draw the state particles $\boldsymbol{x}_{0|0}^{(i)}$ from the prior $p_{0|0}(\boldsymbol{x})$
3:     and set $P_{0|0}^{(i)} = P_0$ and $\tilde{w}_{0|0}^{(i)} = 1/N$;
4: end for
5: for $t = 1, 2, \dots$ do
6:     1) UKF prediction and correction
7:     for $i = 1, \dots, N$ do
8:         - Time update: set $\boldsymbol{x}_{t|t-1}^{(i)} = \boldsymbol{x}_{t-1|t-1}^{(i)}$ and $P_{t|t-1}^{(i)} = P_{t-1|t-1}^{(i)} + Q$;
9:         - Measurement prediction: like in Algorithm 1;
10:        - Measurement update: like in Algorithm 1;
11:    end for
12:    2) Weight update
13:    for $i = 1, \dots, N$ do
14:        sample from the proposal distribution:

               $\hat{\boldsymbol{x}}_t^{(i)} \sim q^{(i)}(\cdot|\boldsymbol{y}^t) = \mathcal{N}(\cdot; \bar{\boldsymbol{x}}_t^{(i)}, P_{t|t}^{(i)}),$

15:        evaluate and normalize the modified importance weights via
16:        (4.18) and (4.19);
17:    end for
18:
19:    3) Estimated pose extraction (optional)
20:    for $i = 1, \dots, N$ do
21:        evaluate and normalize the importance weights via (4.24) and
22:        (4.25);
23:    end for
24:    compute the estimated pose $\hat{\boldsymbol{x}}_t$ via (4.28);
```

```
25:     **4) Resampling**
26:     for $i = 1, \dots, N$ do
27:         if $t > t_0$ then
28:
29:             draw $j \in \{1, \dots, N\}$ with probability $\tilde{w}_t^{(j)}$,
30:             then set:
                    $x_{t|t}^{(i)} = \hat{x}_t^{(j)} \qquad P_{t|t}^{(i)} = P_{t|t}^{(j)} \qquad \tilde{w}_t^{(i)} = \frac{1}{N}.$
31:         else
32:             set :
                    $x_{t|t}^{(i)} = \hat{x}_t^{(i)} \qquad \tilde{w}_t^{(i)} = \frac{1}{N}.$
33:         end if
34:     end for
35: end for
```

measurements is exploited to evaluate the likelihood function. Notice also that the computation of the weights $\bar{w}_t^{(i)}$ is optional (since they are not used in the time propagation from t to $t+1$) and can be limited only to those time instants in which one wants to extract an estimate $\hat{x}_t$ of the object's pose from the approximated posterior.

Remark 2 *While the considered framework deals with static objects, the proposed algorithm is well-suited to being extended to the case of moving objects since it is based on Bayesian filtering and is inherently recursive in nature. When the object is not static, however, the use of a sliding window of the most recent measurements in the weight computation requires some caution because the past measurements do not refer to the current pose. In principle, this problem can be circumvented by considering particle states consisting of the whole object trajectory in the sliding window (similarly to what happens in particle-filtering-based solutions to the SLAM problem) so that the likelihood, with respect to the measurements in the sliding window, can be correctly computed. Further, when the object is static, a simple motion model like (4.12) makes sense only to model small random movements caused by probing. For truly moving objects (for example a rolling ball), more complex motion models are required including also the object velocity. Of course, the main challenge in this case is the increased complexity due to such modifications. Such generalizations are left for future research.*

4.4 Algorithm validation

In order to evaluate the performance of the developed Memory Unscented Particle Filter (MUPF), a C++ implementation of MUPF has been tested via simulations on different objects and collections of measurements. The tests have been run on a Linux platform, with a quadcore 3.40 *GHz* processor. The developed code, the exploited measurements and the reconstructed object models can be downloaded from *github*[1].

4.4.1 Simulation setup

The simulation setup consists of five objects: a rectangular box, a tetrahedron, a cleaner spray, a robot toy and a safety helmet (Fig. 4.1).

The mesh models of the first two objects, having a simple geometrical shape, are built from ruler measurements whereas the other three more complicated objects are approximated by triangular mesh models, reconstructed via image processing algorithms. In particular, the mesh models of the cleaner spray and safety helmet are obtained from 360 degree point clouds reconstructed with the RTM toolbox[2] [197]. The RTM toolbox merges together several partial 3D models - i.e. different views of the object - captured by rotating the object in front of a RGB-D camera and, in a few seconds, provides a 360 degree point cloud of the object. Conversely, the more complex point cloud of the robot toy is retrieved by making use of the AutoDesk 123d catch application[3] that, in several tens of minutes, processes different object photos taken from different views with a smartphone. Thus, the triangular mesh models of the three objects are extracted by applying the Poisson Surface Reconstruction algorithm [198] to the merged point clouds. The complete pipeline for model reconstruction is outlined in Fig. 4.2.

The contact point measurements exploited in the simulation tests are drawn by non-uniformly sampling random points on a subset of 3D model faces.

The MUPF algorithm requires setting the following parameters: the

[1] DOI:10.5281/zenodo.163860.

[2]Recognition Tracking and Modelling of Objects, by ACIN of Technische Universität Wien, http://www.acin.tuwien.ac.at/forschung/v4r/software-tools/rtm/.

[3]http://www.123dapp.com/catch.

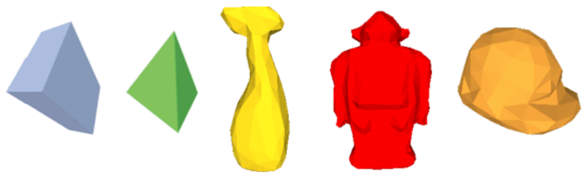


FIGURE 4.1: Simulation setup objects. From left to right: a rectangular box (0.1 × 0.3 × 0.2 [m]), a tetrahedron (equilateral triangular basis with the side of 0.33 [m] × height of 0.2 [m]), a cleaner spray (approximately 0.23 × 0.08 × 0.05 [m]), a robot toy (0.23 × 0.09 × 0.06 [m]), and a safety helmet (nearly a half-sphere with radius 0.1 [m]).

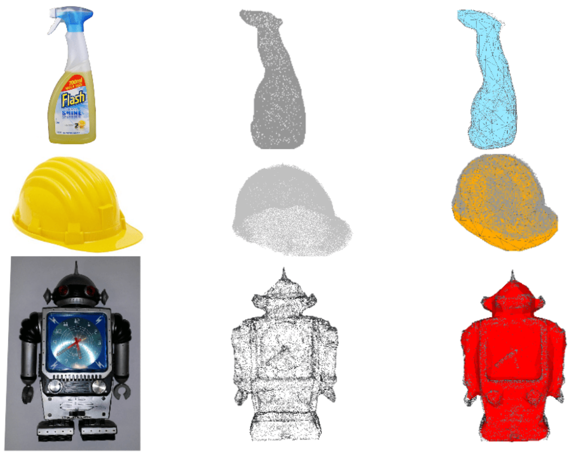


FIGURE 4.2: Pipeline for real object modelling. From left to right: real objects, 360 degree point clouds (obtained with a RGB-D camera and the RTM toolbox for the cleaner spray and the safety helmet, and with 40 photos from different views and the Autodesk 123d catch app for the robot toy), triangular mesh models matching the point clouds, computed by using the Poisson surface reconstruction. On the top: the cleaner spray, whose mesh model consists of 250 faces. In the middle: the safety helmet, featured by a mesh model of 250 faces. On the bottom: the robot toy, whose mesh model is made up of 750 faces.

artificial process noise covariance matrix Q; the measurement noise covariance σ_p^2 characterizing sensor accuracy; the initial covariance matrix P_0 to quantify the initial uncertainty and, hence, the extent of the search region; the parameters of the unscented transformation α, β, k; the number of particles N; the length m of the measurement window for the importance weight update.

As preliminary tests, the parameters are kept constant, as shown in Table 4.1. In particular, the chosen matrix Q is such that the artificial process disturbance spreads the particles with standard deviations of $1\,cm$ in position and about 5 degrees in rotation. Conversely, the covariance σ_p^2 assumes that the measurements of the end-effector position are affected by an error with standard deviation of $1\,cm$ in all Cartesian coordinates. Finally, the initial matrix P_0 indicates an initial uncertainty with standard deviation of 0.2 [m] for the position along the three coordinates and respectively $\pi, \pi/2, \pi$ for the three orientation angles ϕ, θ, ψ. The initial particles $\boldsymbol{x}_{0|0}^{(i)}$ for $i = 1, \ldots, N$ are drawn from the prior distribution $\mathcal{N}(\cdot; \boldsymbol{x}_0, P_0)$, where $\boldsymbol{x}_0$ is arbitrarily chosen (a 6D null vector in our tests). The choices of Table 4.1 have proven effective in all the considered simulations, thus indicating that the proposed algorithm works over a broad range of problems without a case-by-case parameter tuning.

It is worth pointing out how the exploitation of the UKF step in the UPF allows to considerably reduce the number of particles to $N = 700$ (with a standard particle filter it would be in the order of $N = 10^6$ for a 6-DOF problem). Section 4.4.6 provides a detailed analysis about the parameters influence on MUPF performance.

TABLE 4.1: Parameter set for the MUPF.

Q	diag($[10^{-5}, 10^{-5}, 10^{-5}, 10^{-4}, 10^{-4}, 10^{-4}]$) $[m^2]$, $[rad^2]$
P_0	diag($[0.04, 0.04, 0.04, \pi^2, (\pi/2)^2, \pi^2]$) $[m^2]$, $[rad^2]$
σ_p^2	$10^{-4}[m^2]$
α	1
k	2
β	30
N	700

4.4.2 Performance evaluation

The performance of the proposed algorithm is assessed in terms of both effectiveness and execution time, since the ultimate aim of this work is a real-time application of the algorithm.

In this respect, algorithm *reliability* is measured in terms of number of successes among trials, where a trial is considered failed whenever the estimated pose is substantially different from the real one.

In simulation tests, successes and failures can be discriminated by computing the distance between the estimated and the true object poses, since the knowledge of the latter is available. The situation is different in real experiments, wherein the true pose is often difficult (if not impossible) to be measured. In this case, the distinction between a successful or a failed trial is necessarily accomplished by the user by visually inspecting that the solution found by the algorithm is consistent with the real pose of the object. In the successful cases, a numerical evaluation of the localization can be done by relying merely on measurements without the need of the ground truth. This choice is by far preferable (sometimes the only viable solution) for an experimental assessment.

These considerations suggest the definition of the following *performance index*:

$$\mathcal{I}_L = \frac{1}{L}\sum_{i}^{L} d_i, \tag{4.30}$$

where L is the total number of collected measurements and d_i the distance between the i-th measurement and the object in the estimated pose. In other words, given the set of measurements and the estimated pose, the proposed performance index is the average of the distances between each measurement and the object in the estimated pose. It is worth highlighting that the performance index (4.30) is not in the standard least-square fit form in that it is a sum of errors (not of squared errors) and each error term d_i in (4.30) is a complicated nonlinear distance function $G(y, x)$ of two arguments (a contact point measurement y and the object pose x), not expressible in the classical residual form $y - g(x)$ of best-fit problems.

The performance index $\mathcal{I}_L$ has been adopted to evaluate the localization quality in simulation (together with the standard localization error measured as distance of the final estimated pose from the ground truth) and experimental tests, for the reasons listed below. First, the index $\mathcal{I}_L$ is the

FIGURE 4.3: On the left: a robot toy in the real pose. On the right: two different estimated poses, both featured by a performance index of $0.008[m]$ with respect to the set of measurements, coloured in black. The green one corresponds to the correct pose, whereas the red one is a local minimum, representing a completely wrong pose, but anyway consistent with the measurements.

only viable solution for the experimental tests, wherein the real pose cannot typically be known or measured with sufficient accuracy. Secondly, the use of a common error index for both simulation and experimental tests, makes easier the comparison between the two cases. Third, if simulation tests are carried out with noiseless measurements and a sufficient number of informative measurements is collected, then the performance index $\mathcal{I}_L$ can be related to the distance between the estimated and the true object poses, in the sense that $\mathcal{I}_L$ vanishes for large L if and only if the two poses coincide. Finally, the index $\mathcal{I}_t$ is easily computable on-line at each time t and could therefore be monitored in order to understand when to stop localization of the current object. As a further benefit, (4.30) provides a synthetic (scalar) indicator of the pose error, in terms of linear displacement (measured in units of length). Thus, the index computation is not affected by the problems related to the computation of angular displacements.

Nevertheless, it is worth pointing out that when the measurements are too inaccurate, the index (4.30) can be non-informative and the evaluation of the algorithm performance would necessarily require the ground truth object pose. In fact, if measurements are very noisy, the computed performance index might be low even if it is associated to local minima and corresponds to a completely wrong localization (Fig 4.3).

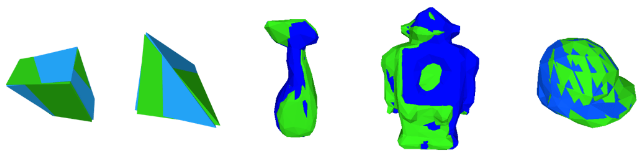


FIGURE 4.4: MUPF simulation results: the real poses are coloured in blue, whereas the estimated ones, featured by an error index of 0.002 [m], are coloured in green.

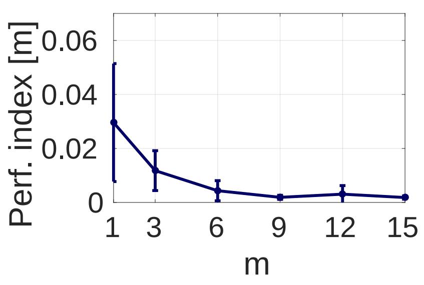
4.4.3 Simulation results

Table 4.2 provides, for each considered object, the following metrics averaged over 50 independent trials of the MUPF: standard localization error in both position and orientation, performance index $\mathcal{I}_L$ defined in (4.30), execution time and reliability. Table 4.3 reports the total number of measurements L and the MUPF window size m used for each object. The true object poses are fixed over trials and differ from the 6D null vector in translation (from 0.05 up to 0.1 [m] along one axis) and orientation (from 45 up to 90 degrees with respect to one axis).

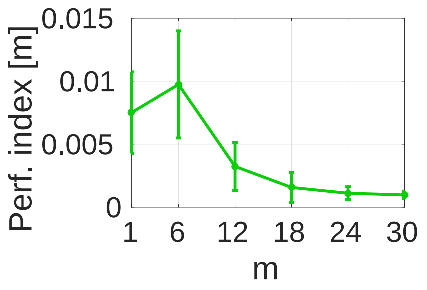
It is worth underlining how, when an adequate choice of m is adopted (see Fig. 4.5 and 4.6), the localization errors averaged over trials are small (e.g. the index $\mathcal{I}_L$ is less than $2\,[mm]$, see Fig. 4.4), the execution time is acceptable and the reliability is high. In Fig. 4.5, the behavior of the performance index $\mathcal{I}_L$ is shown as a function of the memory m ranging from 1 to L (the total number of available measurements). Such plots highlight how MUPF is capable of solving the problem even with small m ($1 < m \ll L$) whereas the standard UPF (i.e. MUPF with $m = 1$) doe not converge at all. In addition, Fig. 4.6 demonstrates that the algorithm is reliable even with small values of m (provided that $m > 1$).

TABLE 4.2: Simulation results for the MUPF algorithm.

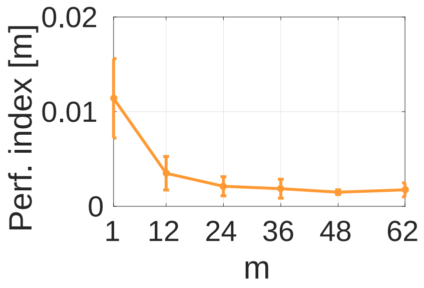
Object	Standard error [deg], [m]	$\mathcal{I}_L$ [m]	Time [s]	Succ./Trials
Box	0.30 - 0.0036	0.0025	1.61	50/50
Tetra.	17.1 - 0.0061	0.0021	3.63	50/50
Cleaner	0.78 - 0.0027	0.0025	7.32	50/50
Robot	19.6 - 0.0072	0.0021	3.95	50/50
Helmet	0.06 - 0.0023	0.0017	4.82	50/50



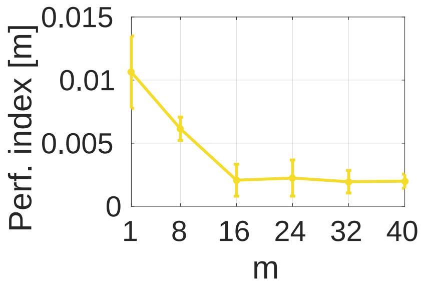
(a) Box



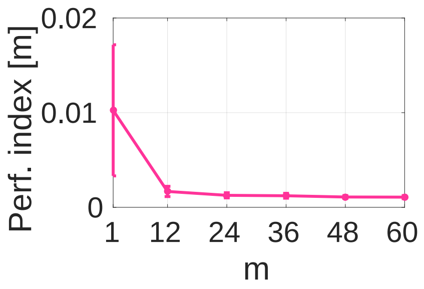
(b) Tetrahedron



(c) Cleaner

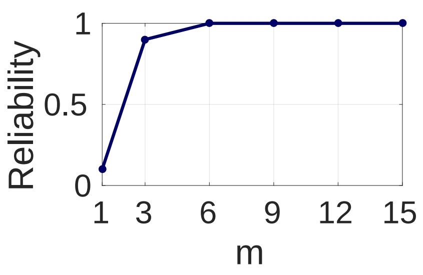


(d) Robot

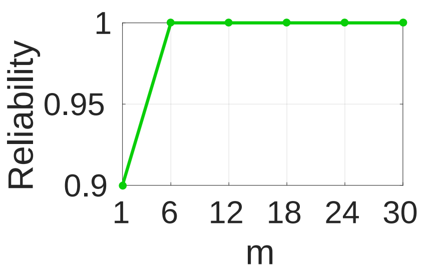


(e) Helmet

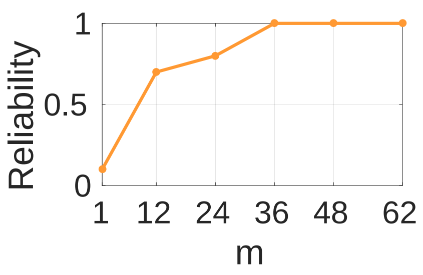
FIGURE 4.5: MUPF simulation results: average performance index on fifty trials by varying m, ranging from 1 up to the total number of measurements L.



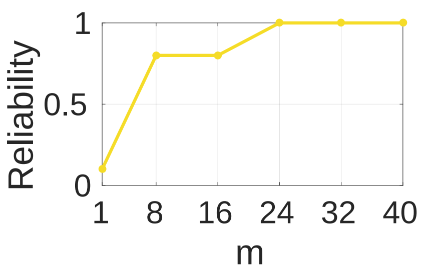
(a) Box



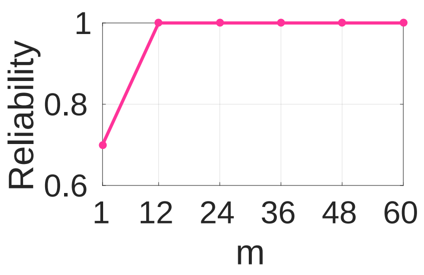
(b) Tetrahedron



(c) Cleaner



(d) Robot



(e) Helmet

FIGURE 4.6: MUPF simulation results: reliability on fifty trials by varying m, ranging from 1 up to the total number of measurements L.

TABLE 4.3: Simulation results: measurements and m values.

Object	L	m	Object	L	m
Box	15	9	Tetra.	30	12
Cleaner	62	36	Robot	40	24
Helmet	60	36			

For the sake of comparison, a simple batch baseline, the *Iterative Closest Point (ICP)* algorithm [199], and a state-of-art approach, the *Scaling Series* algorithm presented in [34] specifically for tactile localization, have been applied to the same simulation scenario.

In order to adapt ICP, which is originally designed for shape registration, to the tactile localization problem, two point clouds are considered: one consisting of the measurements, and the other representing the object model in the right pose. To this end, suitable models have been obtained by sampling 1000 points on the object mesh models of Fig. 4.1. However, it was found that a standard implementation of ICP does not converge in such a scenario. ICP fails because there is a large uncertainty in the object initial pose. Lacking a good initial guess close to the optimal solution, ICP gets trapped in local minima.

The results obtained with the Scaling Series are reported in Table 4.4 for the same sets of measurements used in the MUPF simulation tests (Table 4.3). For the sake of conciseness, only the values of the performance index $\mathcal{I}_L$ are shown.

TABLE 4.4: Simulation results for the Scaling Series algorithm.

Object	$\mathcal{I}_L$ [m]	Time - Max. Time [s]	Successes/Trials
Box	0.001	3.47 - 13.2	45/50
Tetra.	0.001	0.05 - 1.03	50/50
Cleaner	0.006	0.03 - 5.64	42/50
Robot	0.003	0.02 - 3.64	43/50
Helmet	0.005	0.04 - 4.20	32/50

Notice that the execution time of the Scaling Series algorithm significantly changes over the trials as the algorithm generates quite different numbers of particles from trial to trial. Hence, Table 4.4 reports both *average* and *maximum* (worst-case) execution times. Nevertheless, the Scaling Series algorithm proves to be relatively faster than MUPF. In terms of localization precision in the successful trials, the MUPF and Scaling Series algorithms exhibit comparable results. It is worth pointing out, however, that in a non negligible number of trials the Scaling Series algorithm diverged and failed to find a solution. The low reliability of the Scaling Series algorithm can be attributed to the automatic process of particle generation. Particularly, a rough parameter tuning can easily lead to the generation of an insufficient number of particles or, on the contrary, to their exponential growth, thus precluding the final convergence of the algorithm. This is somewhat surprising as MUPF has always been executed with the same parameters, whereas the parameters of the Scaling Series algorithm have been specifically tuned to each case in order to achieve better performance. In summary, MUPF turned out to be more reliable than the Scaling Series algorithm.

An extensive evaluation of the MUPF algorithm is performed by tackling the 6-DOF tactile localization problem for real objects via actual tactile measurements. For these experiments, the employed code implementation and hardware computing platform are the same ones exploited for the simulation tests.

4.4.4 Experimental setup

Four everyday objects are considered: two toys, the cleaner spray and the robot toy. The experimental tests on the safety helmet are not shown since many local minima, corresponding to different poses and featured by the same localization error, are wrongly given as possible solutions. The reasons of this behaviour will be explained in detail in Section 4.4.5. The mesh models of the first two objects are reconstructed from ruler measurements (Fig. 4.7), since they are well-represented by geometrical solid figures. The cleaner spray and robot toy mesh models are the same ones exploited for the simulation tests. Note that in order to avoid object's slip caused by the robot's movements, each object was strictly fixed to a support during measurement collection.

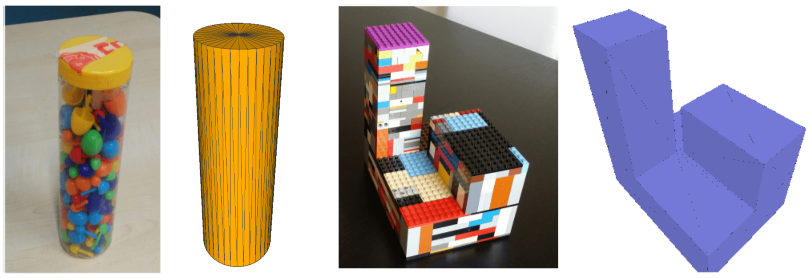


FIGURE 4.7: Mesh models of real geometric objects. On the left: cylindrical tube, with a diameter of 0.06 $[m]$ and height of 0.2 $[m]$, 144 triangular faces. On the right: a Lego object, made up of three parallelepipeds (total dimensions of $0.2 \times 0.1 \times 0.2\,[m^3]$), 36 triangular faces.

The platform used for the collection of tactile measurements is the iCub humanoid robot [2]. Tactile measurements are supplied by fingertips on the iCub hands (see Chapter 3), that are covered with capacitive tactile sensors capable of providing accurate contact point measurements, once contact with the object is detected. Due to the object complexity, tactile measurements are collected through a user-guided strategy, consisting of predefined points approximately located around the objects. This strategy was necessary since a completely blind exploration of the objects turned out to be unfeasible and often caused the robot to hit the object with part of the hand not covered with sensors. It is important to remark that, for this work, the final goal of the experimental tests is the extensive evaluation of the proposed MUPF algorithm through realistic measurements, without focusing on the design of an autonomous measurement collection strategy.

Before providing experimental results, it is worth discussing the main sources of measurement uncertainty, in order to better appreciate the performance of the proposed algorithm and to understand how to set the parameters. In this respect, one relevant source of uncertainty is given by the tactile sensors themselves. The contact point measurement, in fact, is given by the kinematics of one of the fingers and the supplied x, y, z coordinates are affected by calibration offsets. In addition to this, the kinematics provides the x, y, z coordinates of the center of the fingertip. Thus, the retrieved point is always the center of the fingertip even if the tactile taxel activation - and thus the contact detection - has taken place on the extremity or on the side of the fingertip. Taking into account all these considerations, tactile measurements were empirically estimated to be affected by a noise with standard deviation of 0.015 $[m]$. Such sources of error and uncertainty suggest to choose a slightly larger variance $\sigma_p^2 = 4\ 10^{-4}\ [m^2]$ in order to characterize iCub tactile sensor accuracy. The other MUPF parameters used for the experimental tests are shown in Table 4.1, except for the number of particles N, set equal to 1200 in the experimental tests unless differently specified.

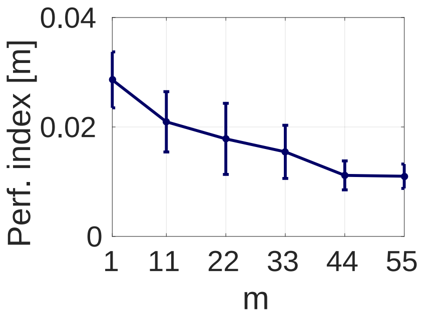
4.4.5 Experimental results

In Tables 4.5 and 4.6, the average performance index, along with the execution time and the algorithm reliability are provided for fifty trials of both the MUPF and Scaling Series algorithms on the four considered objects. The results obtained with the ICP algorithm are not shown due to the lack of convergence. In addition, only the performance index $\mathcal{I}_L$ is computed in the real experiments, where the true pose is difficult to be measured. Figs. 4.8 and 4.9 show the average performance index and the reliability on fifty trials by varying m, ranging from 1 (standard UPF) up to the total number of measurements $m = L$.

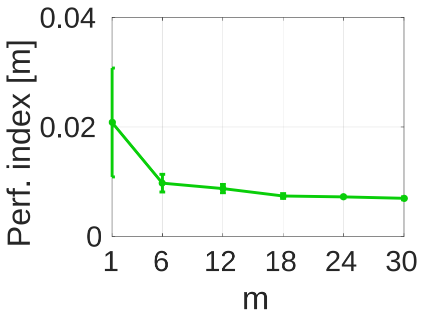
TABLE 4.5: Experimental results for the MUPF.

Object	$\mathcal{I}_L$ [m]	Time [s]	Successes/Trials	L	m
Lego toy	0.0090	12.8	46/50	55	55
Cylinder	0.0063	6.71	50/50	30	18
Cleaner	0.0090	13.7	50/50	62	30
Robot	0.0054	12.3	43/50	60	36

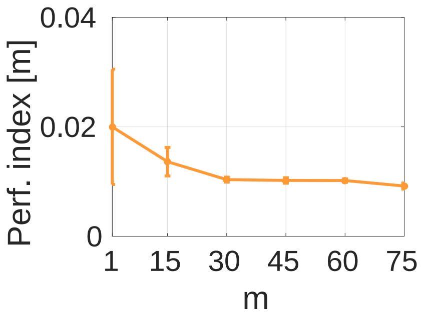
The experimental tests confirm the MUPF behavior exhibited in the simulation tests, even if the experimental solutions are unavoidably affected by a slightly worse performance index, due to the high measurement noise (Fig. 4.10). The measurement noise is also responsible for the deterioration of algorithm reliability for the Lego and robot toys. This effect can be ascribed to the fact that the measurement noise is comparable with the dimension of the distinctive details of these two objects. In fact, the distinction between a good or a wrong solution is strongly influenced



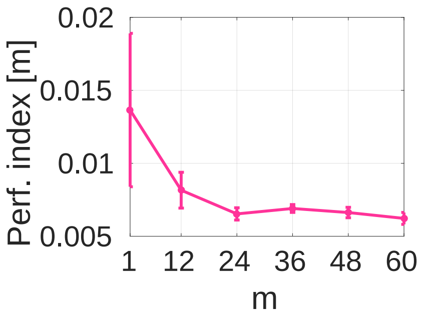
(a) Lego Box



(b) Cylinder



(c) Cleaner



(d) Robot

FIGURE 4.8: MUPF experimental results: average performance index on fifty trials by varying m, ranging from 1 up to the total number of measurements L.

TABLE 4.6: Experimental results for the Scaling Series algorithm.

Object	$\mathcal{I}_L$ [m]	Time/ Max. Time [s]	Successes/Trials
Lego toy	0.0073	5.03 - 29.71	40/50
Cylinder	0.0059	4.02 - 13.22	40/50
Cleaner	0.0139	4.02 - 13.22	23/50
Robot	0.0027	0.81 - 8.72	43/50

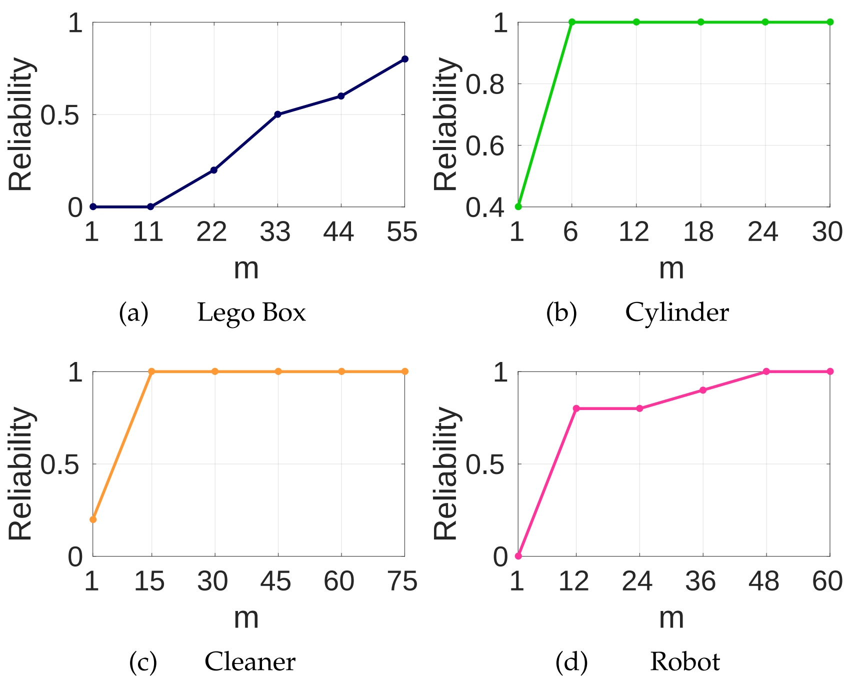
by the object details, since the only exploited information consists of tridimensional points, without taking advantage of surface normals. In such scenarios, a measurement noise of the same entity of the detail dimensions prevents the user from localizing the object even via visual inspection. As mentioned above, this is also the reason why experimental tests on the safety helmet are not shown: due to the strongly symmetric shape and the measurement noise, the measurements are not informative enough in the sense that there are many different poses compatible with the measurements (i.e. corresponding to local minima).

On the contrary, the Scaling Series performance turns out to be much worse compared to what reported in the simulation tests, particularly in terms of reliability. The failures of the Scaling Series algorithm are mainly caused by the generation of an insufficient number of particles. Often, it is not simple to set the Scaling Series parameters so that the number of generated particles is sufficient to reliably localize the objects. This shows how parameter tuning can actually be a weakness of the Scaling Series approach.

4.4.6 Further analysis

In this section, additional results are provided, with the aim of better analyzing MUPF performance.

First, the algorithm robustness has been tested by varying some algorithm parameters, such as the covariance Q of the artificial process noise and the number of particles N. The box-plots of Figs. 4.11(a) and 4.11(b) point out how the performance index and reliability are not significantly affected by varying the covariance matrix Q. Fifty trials of the MUPF have



(a) Lego Box (b) Cylinder

(c) Cleaner (d) Robot

FIGURE 4.9: MUPF experimental results: reliability on fifty trials by varying m, ranging from 1 up to the total number of measurements L.

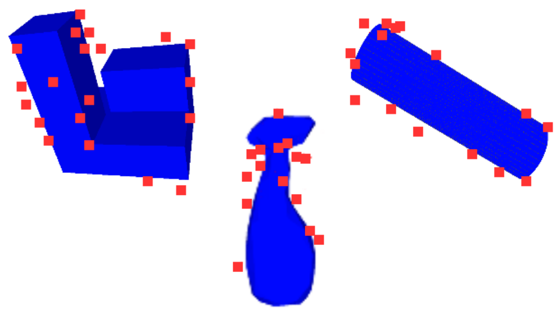


FIGURE 4.10: MUPF experimental results: tactile measurements are coloured in red, the estimated poses (performance index of 0.008 [m]) in blue.

TABLE 4.7: Q matrices used in the tests.

	Simulated tests		
Q_1	diag([10^{-6}, 10^{-6}, 10^{-6}, 10^{-5}, 10^{-5}, 10^{-5}])	[m^2],	[rad^2]
Q_2	diag(5×[10^{-6}, 10^{-6},10^{-6},10^{-5}, 10^{-5},10^{-5}])	[m^2],	[rad^2]
Q_3	diag([10^{-5}, 10^{-5}, 10^{-5}, 10^{-4}, 10^{-4}, 10^{-4}])	[m^2],	[rad^2]
Q_4	diag(5×[10^{-5},10^{-5}, 10^{-5},10^{-4}, 10^{-4},10^{-4}])	[m^2],	[rad^2]
Q_5	diag([10^{-4}, 10^{-4}, 10^{-4}, 10^{-3}, 10^{-3}, 10^{-3}])	[m^2],	[rad^2]
	Experimental tests		
Q_1	diag([10^{-6}, 10^{-6}, 10^{-6}, 10^{-4}, 10^{-4}, 10^{-4}])	[m^2],	[rad^2]
Q_2	diag(5×[10^{-6}, 10^{-6},10^{-6},10^{-4}, 10^{-4},10^{-4}])	[m^2],	[rad^2]
Q_3	diag([10^{-5}, 10^{-5}, 10^{-5}, 10^{-3}, 10^{-3}, 10^{-3}])	[m^2],	[rad^2]
Q_4	diag(5×[10^{-5},10^{-5}, 10^{-5},10^{-3}, 10^{-3},10^{-3}])	[m^2],	[rad^2]
Q_5	diag([10^{-4}, 10^{-4}, 10^{-4}, 10^{-2}, 10^{-2}, 10^{-2}])	[m^2],	[rad^2]

been carried out for five different Q matrices shown in Table 4.7 (a total of 5×50 trials). The performance index $\mathcal{I}_L$ and reliability averaged over the 50 trials - 5 values for each object - are used in building each box. The boxplots of Fig. 4.11 show the performance obtained with real measurements.

Fig. 4.12 shows the influence of the number of particles N on MUPF performance, in terms of localization error and reliability when real measurements are exploited. Performance deteriorates for $N \leq 400$, while slight changes have been found by varying N from 600 up to 1200 (with the exception of the legobox, see Fig. 4.12 (b)).

Secondly, MUPF execution time has been studied by varying the number of particles N and the MUPF window size m. Figs. 4.13(a) - 4.13(d) and 4.13(e) show the average execution time over fifty trials versus m (with N=1200) and, respectively, N (with m=L), in the case of real measurements.

Finally, given the recursive nature of the algorithm, it is worth to analyze the evolution of the performance index $\mathcal{I}_t$ during the MUPF iterations in order to check if it could be used as an appropriate stopping criterion for recursive, on-line localization. Fig. 4.14 shows how the index $\mathcal{I}_t$ evolves in time, i.e. while new measurements are being processed. It turns out that, after a burn-in period, $\mathcal{I}_t$ quickly converges to a small value. This suggests that the localization could be terminated whenever the addition of a new

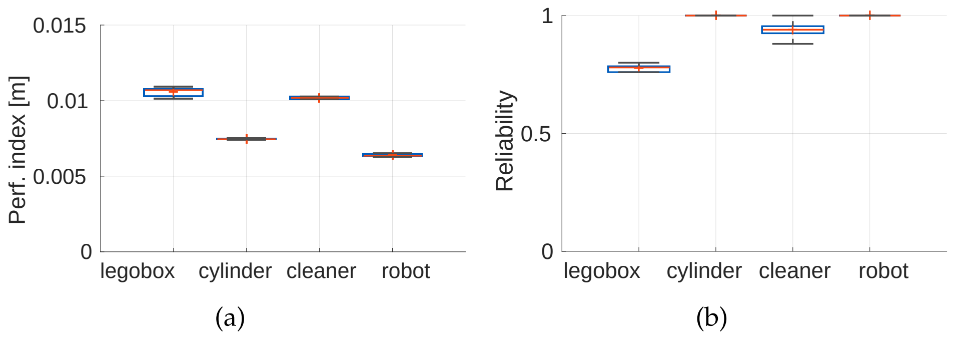


FIGURE 4.11: MUPF robustness analysis: (a) performance index and (b) reliability (number of successes among trials) on fifty trials for 5 different Q matrices, shown in Table 4.7.

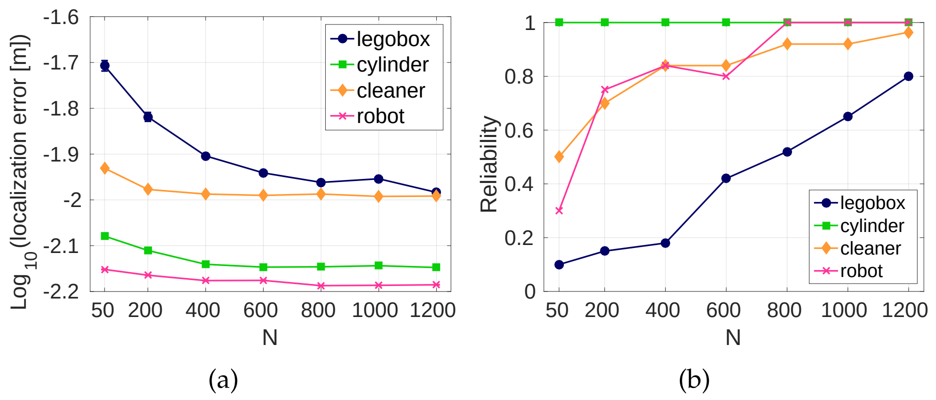
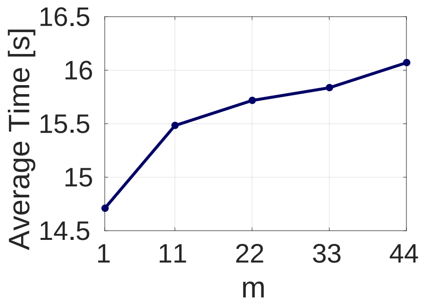


FIGURE 4.12: MUPF performance analysis on fifty trials by varying N from 50 up to 1200 with real measurements, in terms of localization error (a) and reliability (b).

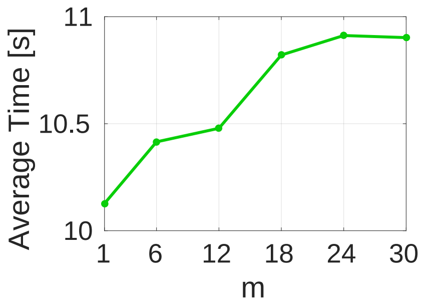
measurement (or a sequence of measurements) does not corresponds to a significant reduction of $\mathcal{I}_t$.

4.5 Discussion

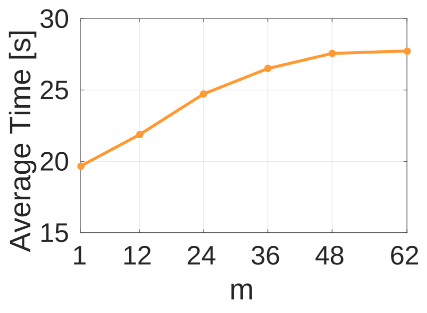
The proposed solution to the 6-DOF tactile localization is based on a novel recursive Bayesian estimation algorithm, the Memory Unscented Particle Filter (MUPF). In contrast to optimization techniques, Bayesian filtering turns out to be a successful approach to account for noisy sensors and inaccurate models. A further advantage of the Bayesian approach is that it can be naturally extended to consider the case in which the object moves, by introducing a suitable probabilistic model for the object motion. The multimodal nature of the problem makes particle filtering techniques more



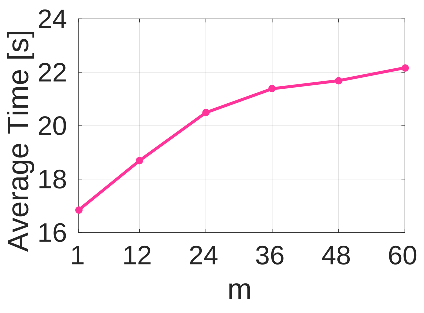
(a) Lego Box (N fixed).



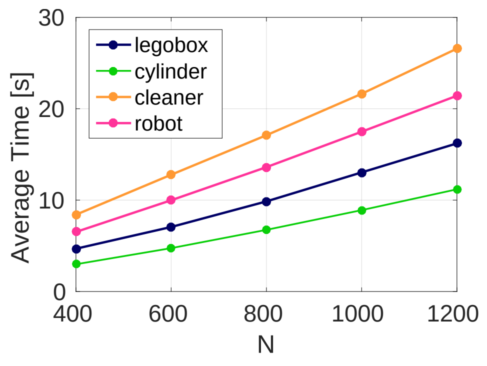
(b) Cylinder (N fixed).



(c) Cleaner (N fixed).



(d) Robot (N fixed).

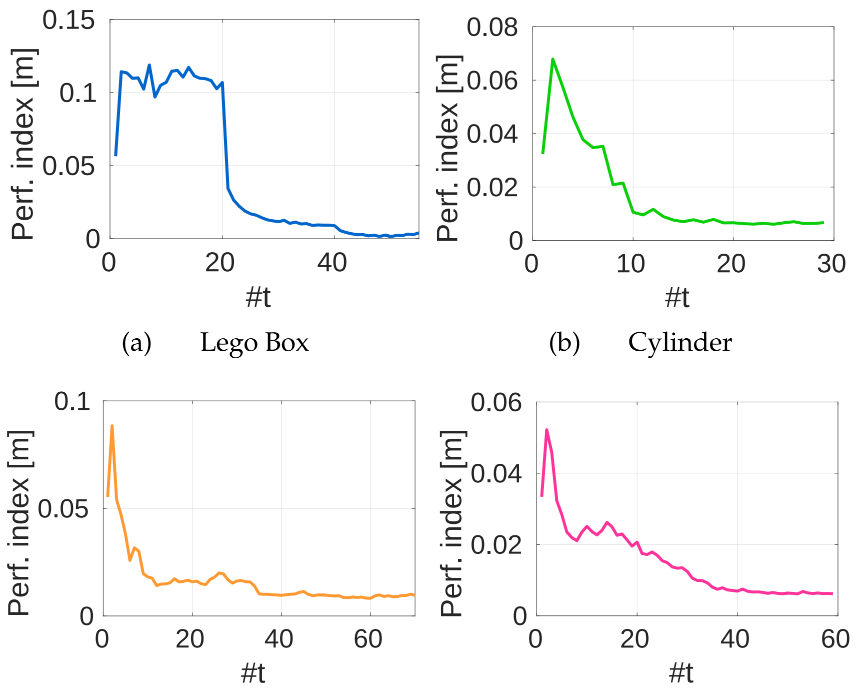


(e) Comparison between all objects (m fixed).

FIGURE 4.13: Average execution time on 50 trials (a) - (d) by varying m (with $N = 1200$) and (e) by varying N (with m equal to the maximum value, i.e. $m = L$).

suitable for tactile localization than nonlinear Kalman filtering approaches. However, the exploitation of standard particle filtering for 6-DOF tactile localization would require a number of particles in the order of 10^6 which, in turn, might entail an unaffordable computational load for real-time operation.

The proposed MUPF algorithm is capable of localizing tridimensional objects through tactile measurements with good overall performance and



(a) Lego Box (b) Cylinder

(c) Cleaner (d) Robot

FIGURE 4.14: Performance index trend at each algorithm time step (with real measurements). After a burn-in period, the performance index decreases and converges to an asymptotic value.

by exploiting a reduced number of particles (in the order of hundreds). The MUPF algorithm relies on the Unscented Particle Filter suitably adapted to the localization problem of interest. The Unscented Particle Filter jointly exploits the potentials of the particle filter for approximating multimodal distributions and of the unscented Kalman filter for efficiently generating the proposal distribution. It is worth to point out that, for measurement update purposes, the particle filter requires a probabilistic sensor description in terms of likelihood function while the unscented Kalman filter needs a measurement function allowing to predict the measurement given the estimated state. In the specific problem of interest, it is quite natural to characterize the tactile sensor in terms of likelihood (i.e. probability distribution of the sensed contact point given the object pose) while it is clearly not possible to uniquely predict the sensed contact point given the estimated object pose. To circumvent this difficulty and be able to apply UPF to tactile localization, the following idea has been pursued: for given object pose and measured contact point, define the likelihood in terms of distance between the object and the measured contact point and take the predicted contact point as the point on the boundary of the object at minimum distance from the measured contact point. As a further contribution, the standard UPF algorithm has been modified by the inclusion of a suitable sliding *memory* (hence the name MUPF) of past measurements in the update of the particle importance weights. In this respect, it was found that the *memory* feature is crucial for a careful exploitation of the available contact point measurements with consequent improvement of localization accuracy.

Furthermore, it is worth underlining how the proposed algorithm succeeds in solving the problem by using only tridimensional contact point measurements, without requiring the knowledge of surface normals.

Performance evaluation, carried out via simulation tests on two geometric objects and three everyday objects by using simulated measurements and tridimensional mesh models reconstructed by vision, demonstrates that the algorithm is reliable and has good performance with an average localization error less than $0.002\ [m]$ and a computing time of a few seconds. Moreover, the algorithm manages to localize real objects with actual tactile measurements collected with the humanoid robot iCub. The results of experimental tests on four real objects confirm the results of the simulation tests, providing localization errors less than $0.01[m]$ with a computing time less than $8\ [s]$.

The same simulation and experimental tests have been carried out also with a reference algorithm in the literature, called Scaling Series. The obtained results show how the MUPF is competitive with the state of art for 6-DOF tactile localization, and also exhibits several advantages with respect to the Scaling Series algorithm.

The MUPF described in this Chapter has been successfully applied also on a challenging tactile recognition task (see Chapter 5 for more details). This is in fact a natural extension of the localization problem. A robot able to localize an object using tactile sensors can also recognize it among a finite set of possible objects, using the same information. For example, given an effective localization algorithm, the robot can run it with different known object models and select the one that best matches the observations.

A useful feature of the proposed algorithm consists of the nature of measurements it processes. As we said in Section 4.2, the MUPF take advantage solely of the 3D contact positions of its fingertip with the object surface, instead of requiring also the surface normal measurements or other kinds of tactile informations, such as exerted force of texture information, as commonly happens in literature. The usage of this limited information can be considered as a limitation, being the responsible of lack of observability. However, this fact has the silver lining to making the algorithm agnostic to the source of measurements: it just needs to be fed with 3D points collected on the object surface, regardless of the exploited sensors. As a result, the MUPF can localize object by processing 3D point clouds. This turns out to be very useful in practical robotic manipulation since visual perception is often the first source of information for manipulating objects. Object point clouds can be used for estimating the initial object pose. After that, since vision occlusion is very likely to happen when interacting with the object, tactile information is exploited to assist or even replace visual feedback. Chapter 6 shows a practical usage of the MUPF to estimate the object in-hand pose using point clouds during the execution of bi-manual handover tasks.

The contributions detailed in this Chapter suggest other perspectives for future work on 6-DOF object tactile localization. First of all, dealing with the localization of objects in presence of slippage or even tracking moving objects is fundamental in real applications. When filtering techniques (e.g. variants of particle filtering) are employed in place of optimization methods, the extension to this case can be achieved by further considering a suitable model for the object motion. Moreover, the nearly recursive nature and the promising computing time of the proposed algorithm would allow reducing localization uncertainty on-line during measurement collection. This partial information could be in fact exploited to guide tactile exploration in order to maximize the information on the object pose.

Chapter 5

Applications of the Memory Unscented Particle Filter to object tactile recognition

This Chapter shows the application of the Memory Unscented Particle Filter, presented in Chapter 4 on tactile object recognition [200]. The robot explores an object using its tactile sensors, registering the 3D coordinates of the finger-object contact locations. The contact locations collected during the exploration are, then, compared with different object models. The solution of the recognition problem is given as the object whose model better fits the measurements, i.e., the object model with the lowest localization error.

As we already stressed out in Chapter 4, also in this application the measurements consisting only of a set of 3D contact point coordinates. Such data provide very basic, and noisy information, making the tactile recognition task more challenging.

The Chapter is organized as follows. Section 5.1 provides our formulation of tactile recognition as a multi-object localization problem. Section 5.2 presents the exploration strategy for acquiring measurements. Section 5.3 demonstrates the effectiveness of the proposed solution by means of simulation and experimental tests on the iCub humanoid robot. In Section 5.4 we suggest possible future directions.

5.1 Methodology

We introduce hereinafter the problem of tactile object recognition. Let k denote the number of objects of interest, each object being represented by a mesh model consisting of triangular faces $\{f_i\}$. A set of measurements $\{\boldsymbol{y}_t\}_{t=1}^{L}$ is collected using the tactile sensors by detecting contacts on the surface of object k^* (one of the k objects). It is assumed that object is attached to a surface and, thus, does not move during the exploration. Each measurement provides the 3D coordinates of the contact point, i.e. $\{\boldsymbol{y}_t = (x_t, y_t, z_t)\}_{t=1}^{L}$. The goal is to infer on which object the measurements have been collected. In the described scenario, the solution is given by the object model that best fits the available measurements.

5.1.1 Recognition as multi-object localization

We address the tactile object recognition problem as a localization problem applied to multiple objects, where the solution is provided by the object whose localization error is the lowest among all the considered objects.

The localization algorithm we use is the Memory Unscented Particle Filter, described in Chapter 4. Object recognition is achieved by simply running the MUPF for each of the given object models using the same set of measurements. For each possible object $j \in \{1, \dots, k\}$, the algorithm finds the pose $\hat{\boldsymbol{x}}_l$ that makes the object model representing the j^{th} object best fit the set of measurements. After k executions of the algorithm, k different solutions $\hat{\boldsymbol{x}}_j$, for $j = 1, \dots, k$, are computed. Once the pose $\hat{\boldsymbol{x}}_j$ is calculated for each object models $j \in \{1, \dots, k\}$, the corresponding *performance index* $\mathcal{I}_{L,j}$ introduced in 4.4.2 is used in order to measure the fitness of each object model in the estimated pose j. We recall the *performance index* to be defined as:

$$\mathcal{I}_{L,j} = \frac{1}{L} \sum_{t}^{L} d_{t,j}, \tag{5.1}$$

where L is the number of measurements and $d_{t,j}$ is the distance between the t^{th} measurement and the object model j in the estimated pose $\hat{\boldsymbol{x}}_j$. For the sake of clarity, from now on we refer to the performance index by simply using $\mathcal{I}_j$, omitting the number of measurements L in the subscript. In other words, given the set of measurements and the estimated pose, the proposed performance index is the average of the distances between each measurement and the object model in the estimated pose. Finally, after the k executions of the localization algorithm, the quantities $\mathcal{I}_j$ for $j = 1, \dots, k$, are available and the solution $\hat{k}$ for the tactile recognition problem is given by:

$$\hat{k} = \arg\min_j \mathcal{I}_j. \tag{5.2}$$

Clearly, the recognition is successful when $\hat{k} = k^*$. The steps of our algorithm for the tactile recognition stated as a localization problem are outlined in Algorithm 3.

Algorithm 3 Tactile recognition algorithm

```
1: **Data:** $k$ object models, a set of tactile measurements $\{\boldsymbol{y}_t\}_{t=1}^{L}$ on object $k^*$;
2: **for** $j = 1, \ldots, k$ **do**
3:     **Localization algorithm:**
4:     data: object model $j$, set of measurements on object $k^*$;
5:     output: $\hat{\boldsymbol{x}}_j$ ;
6: **end for**
7: Choose $\hat{k}$ as:
       $$\hat{k} = \arg\min_j \mathcal{I}_j,$$
   where $\mathcal{I}_j = \frac{1}{L}\sum_t^L d_{t,j}$ and $d_{t,j}$ is the distance between the $t^{th}$ measurement and the object $j$ in the estimated pose $\hat{\boldsymbol{x}}_j$.
8: Recognition is successful if $\hat{k} = k^*$.
```

5.2 Data Acquisition

The experimental setup consists of the iCub robot [201](Fig. 5.1) and six objects of interest (i.e. $k = 6$) for acquiring tactile data in our experiments. The objects, as shown in Fig. 5.2, are made of wooden geometric shapes. The objects are deliberately selected to have overlapping shapes with strong similarities in order to test our method in a challenging setting. For example, objects (a) and (b) have similar geometric configurations: one has a smooth arched surface and the other a saw-tooth surface, respectively. With the same principle we also selected objects (c) and (d), that have same general shape, the only difference being in the smoothness of the surfaces. Objects (e) and (f) can only be discriminated by the bottom edge: one has a straight edge, while the other has a curved edge.

The robot touches the object at various locations with the tip of its index finger. The fingertip is 14.5 [mm] long, 13 [mm] wide. Each finger is equipped with tactile sensors [22]. A contact location is registered when

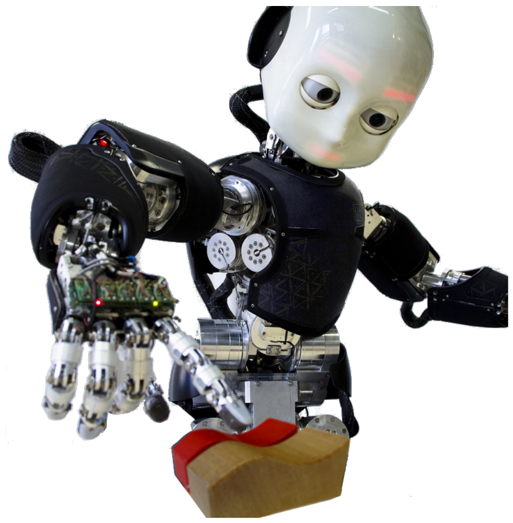
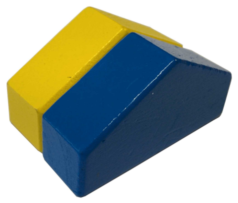


FIGURE 5.1: Experimental setup for data collection: the iCub robot is touching the object with its index fingertip.

the tactile sensors are activated. In our experiments the object is anchored to the surface of a table in front of the robot, hence, it does not move during the exploration. The choice of the exploratory area depends on the the size of the object. We sample an area of 40×50 [mm^2] (Fig. 5.4), using a grid search with a cell size of 2.5×2.5 [mm^2].

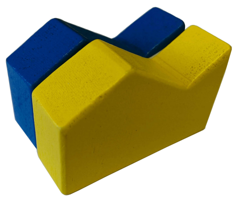
At the beginning of the exploration, the robot's index finger is placed at an arbitrary position close to the object. Then, the robot is commanded to sample a location of interest. We will refer to the location of interest as a waypoint. Since we do not have a priori knowledge of the shape of the object, the height of the waypoint is set to an arbitrary value larger than the height of the object. As reported in the flow chart of Fig. 5.3, the robot moves the finger toward the waypoint. After that, the robot extends its finger downward to detect a contact. If no contact is detected when the finger is fully extended, the robot sets the waypoint to the current location of the finger and retracts it. This process is repeated until the finger makes a contact with a surface – either the object or the table. When a contact is detected, the location of the contact is registered and the next waypoint is set to the next point in the grid. This process is repeated until the area is entirely covered. The tactile data collected for each object with this exploration strategy are shown in Fig. 5.4.

(a)

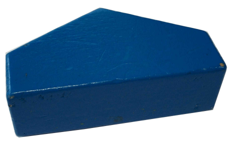


(b)

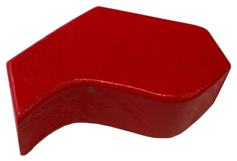
(c)



(d)



(e)



(f)

FIGURE 5.2: Objects used for experimental evaluation of the method.

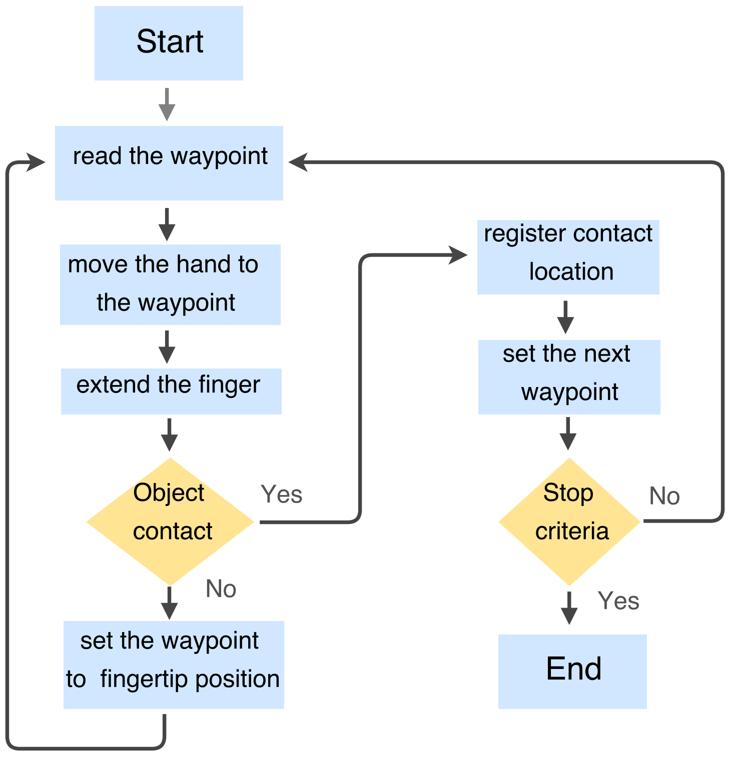
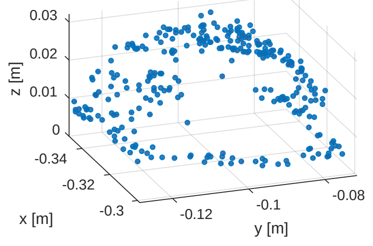
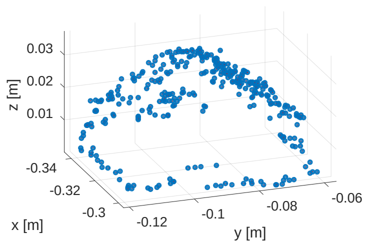


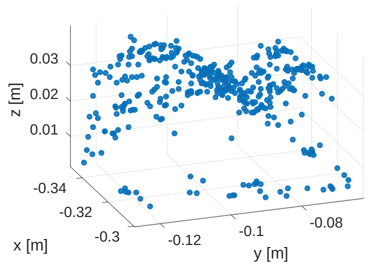
FIGURE 5.3: A flow chart showing the object-surface sampling.



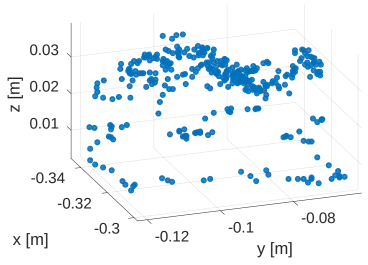
(a)



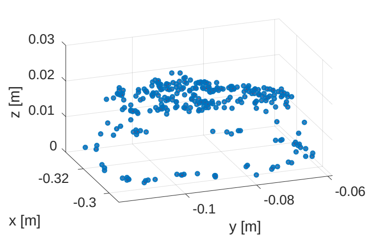
(b)



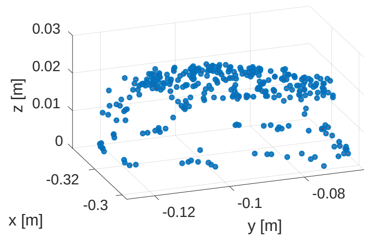
(c)



(d)



(e)



(f)

FIGURE 5.4: For each object, the tactile data collected by using the exploration strategy of Section 5.2 are shown. The letters identifying the different plots ((a) - (f)) correspond to the objects according to the notation of Fig. 5.2.

5.3 Results

The algorithm evaluation is performed first with synthetic measurements (Section 5.3.1) and then with real measurements (Section 5.3.2), collected through the exploration strategy described in Section 5.2. In both scenarios, the aim is to recognize the true object labeled as k^*, among the set of six objects shown in Fig. 5.2.

The C++ implementation of the MUPF algorithm used to carry out our experiments is publicly available on GitHub[1].

5.3.1 Simulation results

The synthetic measurements consist of six sets of 3D points (around 170 triplets for each set), each sampled on the surface of one specific model. We refer to the 3D points sampled on object (a) as *set of measurements (a)*. The same notation is used for the other objects. The synthetically-generated data are noiseless.

In Table 5.1, the MUPF parameter set used for running the simulated tests is shown. Matrix Q and σ_p are respectively the covariances of the process noise $\boldsymbol{\omega}_t$ and measurement noise $\boldsymbol{\nu}_t$; P_0 is the covariance matrix representing the initial uncertainty and N is the number of particles. The covariance Q is chosen such that it takes into account the stationarity of the object, similarly, the value of the covariance σ_p models the measurement noise. An arbitrarily large value is instead chosen for P_0 matrix. The selected number of particles N is a trade-off between algorithm execution time and reliability. In order to determine a good value for m, which is the number of most recent measurements used at each time instant, we run the MUPF algorithm for each object. Fig. 5.5 displays how the localization errors vary with different values of m in the range from 1 to L. The figure is for the data collected in the real experiments. The results of the simulated data, which were similar, have been omitted for clarity. Since the localization errors do not decrease significantly for $m > L/2$, $m = L/2$ has been chosen. Such a value leads to accurate performance regardless of the order of the measurements under consideration, thus not requiring the m measurements to be uniformly sampled on the object surface.

[1]DOI:10.5281/zenodo.45493.

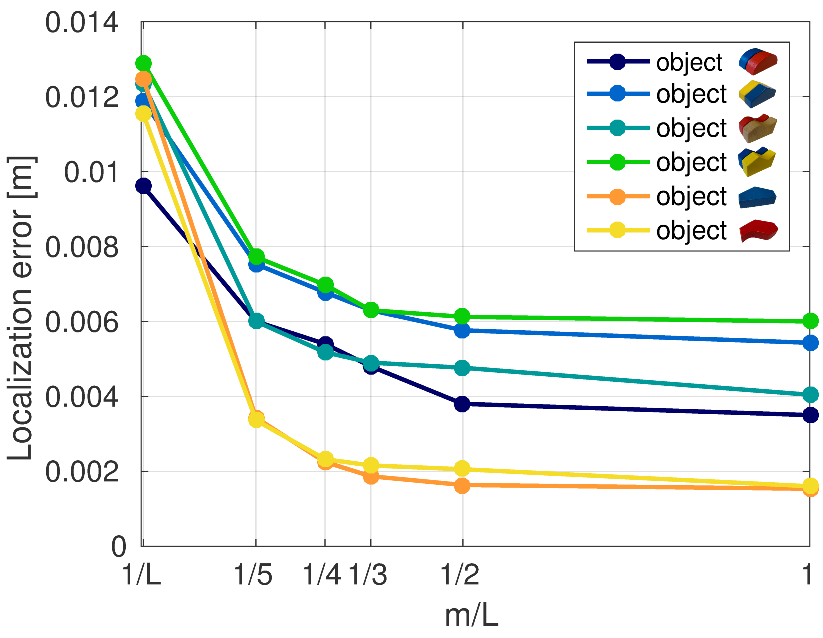


FIGURE 5.5: The localization errors obtained with real measurements with different values of m, from 1 to L. For the sake of clarity, the results from the simulated data have not been plotted as it exhibits a similar trend.

TABLE 5.1: Parameters set for the MUPF in simulation.

Q	diag($[10^{-4}, 10^{-4}, 10^{-4}, 10^{-2}\ 10^{-2}, 10^{-2}]$) [m], [rad]
σ_p	10^{-4} [m]
P_0	diag($[0.04, 0.04, 0.04, \pi^2, (\pi/2)^2, \pi^2]$) [m], [rad]
N	700
m	$L/2$

Fig. 5.6 shows the performance achieved with the simulated measurements in the shape of confusion matrix. Each row i of the matrix corresponds to a different set of measurements, respectively sampled on each object model surface. The column j instead stands for the j^{th} object model used by the localization algorithm. Each block of the matrix (i, j) contains the average localization errors on 10 trials obtained by running the MUPF with the set of measurements sampled on the object i and the model of object j, i.e. $\mathcal{I}^i_j$. Therefore, the correct behavior of the algorithm can be easily deduced by checking if the localization errors on the diagonal of matrix are the minimum for each row. More tightly, the recognition of object i is

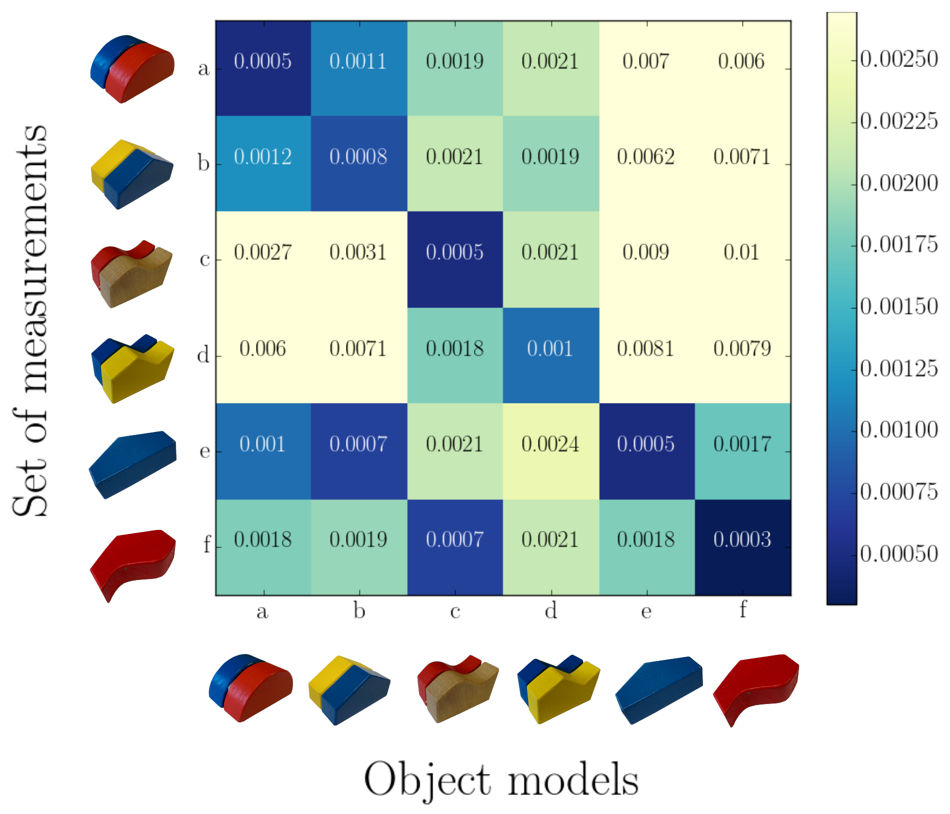


FIGURE 5.6: MUPF performance with simulated measurements. The performance achieved with the simulated measurements are shown in the shape of confusion matrix. Each row i of the matrix corresponds to a different set of measurements, respectively sampled on each object model surface. The column j instead stands for the j^{th} object model used by the localization algorithm. For each experiment, the average localization errors on 10 trials obtained for all the object models are shown.

successful if $\mathcal{I}_i^i = min_j \mathcal{I}_j^i$. Fig. 5.6 confirms that this condition is satisfied when simulation measurements are exploited.

5.3.2 Experimental results

Before showing the performance achieved using the real measurements, we provide a synthetic experiment to point out, from a quantitative viewpoint, that the task at hand is indeed challenging. The results of the experiment are shown in Fig. 5.7. The test consists of calculating the localization error of three different object models: (a), (b) and (c), using the set of *real* measurements (b). More precisely, the three profiles depicted in Fig. 5.7 represent how the localization error varies as the object models slide along the y axis of the frame attached to the object basis. Therefore, Fig. 5.7 reports the localization error versus the y displacement: a displacement equal to 0 represents the correct pose for the object (b), with respect to the set of measurements (b). By observing the trend of the localization errors, we can see how the localization error for object (b) is minimum for a displacement equal to 0, that is in fact the correct pose. However, object (a) and (c) provide an even lower localization error in correspondence of small displacements along y. This fact highlights how the similarity of objects and the noisy nature of the measurements could lead to wrong recognitions.

We discuss hereinafter the performance achieved with real data. The MUPF parameters used for the experimental tests are provided in Table 5.2. The parameters have been chosen by taking into account considerations similar to those explained in Section 5.3.1. In particular, covariances Q and σ_p are tuned differently in order to take into account the measurement noise of the real data. The value of m is determined as described in the previous section, see Fig. 5.5.

TABLE 5.2: Parameters set for the MUPF in real experiments.

Q	diag([$8\ 10^{-6}, 8\ 10^{-6}, 8\ 10^{-6}, 8\ 10^{-4}\ 8\ 10^{-4}, 8\ 10^{-4}$]) [m,rad]
σ_p	$4\ 10^{-4}$ [m]
P_0	diag([0.04, 0.04, 0.04, π^2, $(\pi/2)^2$, π^2]) [m], [rad]
N	1200
m	$L/2$

Fig. 5.8 shows the results of the real experiments, which can be interpreted similarly to the data of Fig. 5.6. Two main differences can be noticed

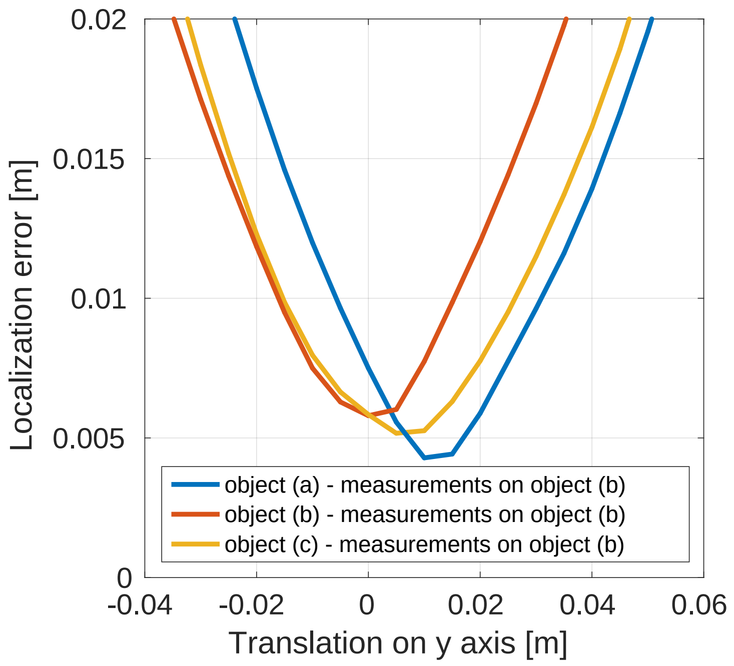


FIGURE 5.7: Synthetic test showing the challenging nature of tactile recognition problem. We compute the localization errors with respect to the set of real measurements (b) and three object models: (a), (b), and (c). Each model is sliding along the y axis of the ground frame. Object (b) results in the lowest error at zero displacement, whereas, notably, object (a) and (c) give lower values for small nonzero displacements.

by comparing Fig. 5.6 and Fig. 5.8, though. First, the measurement noise causes higher average localization errors. Therefore we manage to correctly recognize only 4 objects out of 6 in the real scenario, compared with the 100% overall classification score achieved in simulation. In particular, when the MUPF is executed using set of measurements (b), the solution $\hat{k}$ is given by object (a) and, analogously, when measurements belong to object (d), $\hat{k}$ comes out to be object (c). However, we could reasonably consider these two misclassifications acceptable, considering the high level of similarity between the pairs of objects and the noise in the measurements. In addition, the limited resolution of the tactile sensor and the size of the fingertip (approximately 6×6 [mm^2]) allow only a coarse discrimination of the shape of the object and hide finer details. It is expected that the performance of the recognition would increase using a smaller fingertip or sensors with higher resolution. Given these limitations, however, the carried out experiments demonstrate that the proposed algorithm achieves good performance.

5.4 Discussion

We addressed the problem of tactile recognition as tactile localization on multiple objects using the nonlinear filtering algorithm, Memory Unscented Particle Filter. The algorithm is capable of recognizing objects by exploiting only contact point measurements. The effectiveness of our approach is demonstrated both in simulation and with a real robot.

The promising results presented in this Chapter encourage possible future applications. For example, the model could be extended by including local features, such as surface classification (e.g. local curvature, edge, corners) or material properties (e.g. stiffness, texture). At this aim, the pressure values collected by the robot tactile sensors when contact with the object is detected could be exploited in an extended version of our algorithm. Pressure values provide useful information on stiffness properties, thus facilitating tactile recognition. A further extension of the work we presented consists of taking advantage of a more complex exploration strategy for data collection, by using multiple fingers at the same time. In fact, the exploitation of the knowledge of which finger has caused each

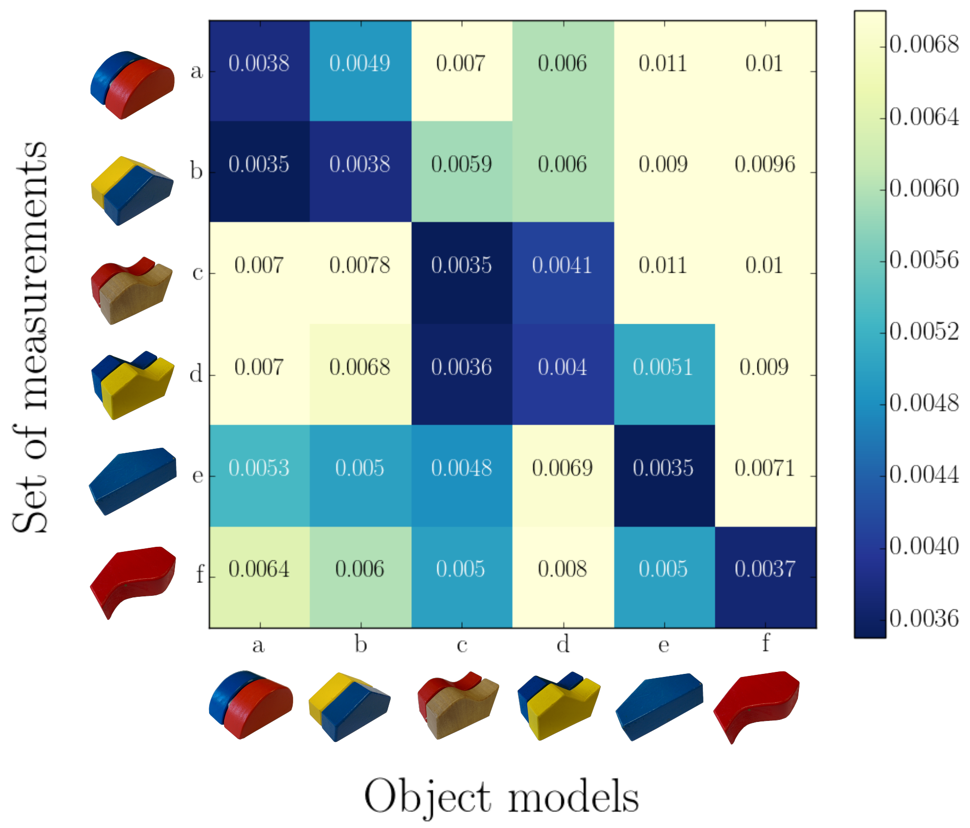


FIGURE 5.8: The performance achieved with the real measurements are shown in the shape of confusion matrix. Each row i of the matrix corresponds to a different set of measurements, respectively collected on each object surface. The column j instead stands for the j^{th} object model used by the localization algorithm. For each experiment, the average localization errors on 10 trials obtained for all the object models are shown.

tactile measurement could be very powerful and considerably improve the performance of our approach.

Part III

Bi-manual coordination: a new pipeline for the execution of handover tasks

Chapter 6

In-hand object localization using vision: bi-manual handover

In the previous Chapters, we presented an algorithm able to localize and recognize objects placed on a support (e.g. a table) by using 3D points sampled on their surface. Both the applications we showed in Chapter 4 and 5 make use of tactile measurements, in terms of the 3D positions of the robot fingertips in contact with the object surface. However, a blind collection of tactile measurements requires accurate planning or, alternatively, needs to be applied on controlled scenarios as shown in Section 5.2. Since our testing platform - the iCub humanoid robot - is provided with stereo vision, a most effective way to collect points belonging to the object surface is by extracting 3D visual point clouds.

In addition, there is no reason to focus only on localizing objects placed on a table. As long as we are able to collect visual and/or tactile measurements, an interesting scenario consists of in-hand object localization, i.e. the goal of estimating the object pose with respect to the hand holding it. In fact, the success, or not, in accomplishing a manipulation task is also determined by how the robot is holding the object. For example, we could image in the future (hopefully not too remote) to give our personal service robot a bottle of wine and ask it to place it on the table in the kitchen. If the robot holds the bottle in the wrong way and *not aware of that*, when the robot will put the bottle on the table, it'll likely fall down, staining the table and the floor and wasting some good wine. This is a simple example showing how in-hand pose localization and re-grasp are fundamental for the achievement even of basic manipulation tasks.

In-hand re-grasp is very challenging and still an open problem. If the robot is provided by more than one arm - as stands for the iCub -, a possible way to address the problem is by performing *bi-manual handover*. If the object pose in one hand of the robot is not suitable for the current task, the object could be grasped by the other hand in a proper configuration.

Another interesting application of bi-manual handover takes place in a pick-and-place scenario. If the robot is given an object in its left hand and is asked to put it on a target location in right hand workspace, the most reasonable movement in this case requires passing the object from the left to the right hand.

All these considerations encouraged us to study the bi-manual handover problem. The result we achieved is a novel pipeline that allows performing the handover task with the iCub humanoid robot and with different every-day objects. In this work, we did not explicitly take into account handover for re-grasp, but this could be surely a straightforward extension.

The proposed pipeline takes advantage of our previous work on tactile-driven object localization (Chapter 4) as well as other prior works on in-hand tactile manipulation [202] and self-touch [203], conveniently adapted and connected together for tackling the entire handover problem. The core part of the pipeline consists in the pose selection method for selecting the best pose for the handover task among a set of *a-priori* poses. The chosen pose maximizes the distance between the two hands and the manipulability index of a two-arms kinematic chain.

The Chapter is organized as follows. Section 6.1 introduces the pipeline we designed, together with a detailed description of all its steps. Section 6.2 validates our approach by analyzing the results of each pipeline steps and showing a set of successful handovers performed by the robot with different every-day objects. Finally, Section 6.3 ends the Chapter with some discussions about the main limitations of our approach and suggestions for improvements.

6.1 Pipeline

The pipeline for bi-manual object transfer we propose is outlined in Fig. 6.1. In practice, we ask the robot to pass a known object from one hand (that we refer to as *first hand*) to the other hand (named *second hand*). The entire pipeline can be divided in the following steps: