MULTI-CONTACT TACTILE EXPLORATION AND INTERACTION WITH UNKNOWN OBJECTS

DISSERTATION (2017)

SUBMITTED TO THE SCHOOL OF ENGINEERING, DOCTORAL PROGRAM ON ROBOTICS, CONTROL, AND INTELLIGENT SYSTEMS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE (EPFL)

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

by

NICOLAS SOMMER

THESIS COMMITTEE:
Prof. Silvestro Micera, president of the jury
Prof. Aude Billard, thesis advisor
Prof. Hannes Bleuler, examiner
Prof. Lorenzo Natale, examiner
Prof. Kenji Tahara, examiner

Lausanne, Switzerland
March, 2017

ABSTRACT

HUMANS rely on the sense of touch in almost every aspect of daily life, whether to tie shoelaces, place fingertips on a computer keyboard or find keys inside a bag. With robots moving into human-centered environment, tactile exploration becomes more and more important as vision may be occluded easily by obstacles or fail because of different illumination conditions. Traditional approaches mostly rely on position control for manipulating objects and are adapted to single grippers and known objects. New sensors make it possible to extend the control to tackle problems unsolved before: handling unknown objects and discovering local features on their surface. This thesis tackles the problem of controlling a robot which makes multiple contacts with an unknown environment. Generating and keeping multiple contacts points on different parts of the robot fingers during exploration is an essential feature that distinguishes our work from other haptic exploration work in the literature, where contacts are usually limited to one or more fingertips.

In the first part of this thesis, we address the problem of exploring partially known surfaces and objects for modeling and identification. In multiple scenarios, control and exploration strategies are developed to compliantly follow the surface or contour of a surface with robotic fingers.

Whereas the methods developed in the first part of this thesis perform well on objects with limited size and variation in shape, the second part of the thesis is devoted to the development of a controller that maximizes contact with unknown surfaces of any shape and size. Maximizing contact allows to gather information more rapidly and also to create stable grasps. To this end, we develop an algorithm based on the task-space formulation to quickly handle the control in torque of an actively compliant robot while keeping constraints, particularly on contact forces. We also develop a strategy to maximize the surface in contact, given only the current state of contact, i.e. without prior information on the object or surface.

In the third part of the thesis, we develop a new way to teach robots how to react to sensing information while performing a task, by modulating Dynamical Systems (DS) using external signals. We extend existing approaches to locally modulate DS to enable sensing-based modulation, so as to change the dynamics of motion depending on an external signal. The problem of autonomous grasping using only tactile data is tackled using this algorithm. We apply our approach by using the data collected from the tactile sensors with a particle filter for object state estimation, which is used to modulate the dynamics of the robot motion accordingly, changing from exploratory to grasping motions depending on task progress. This allows to generate fast and autonomous object localization and grasping in one flexible framework. We also apply this algorithm to teach a robot how to react to collisions in order to navigate between obstacles while reaching.

Keywords: Tactile Exploration, Active Compliance, Multi-contact Control, Dynamical Systems.

RÉSUMÉ

LES humains sont dépendants de leur sens du toucher dans tous les aspects de la vie quotidienne, que ce soit pour faire ses lacets, ajuster la position de ses doigts sur un clavier d'ordinateur ou pour trouver des clés dans un sac. Avec le déploiement de robots dans des environnements faits pour l'homme, l'exploration tactile devient de plus en plus importante puisque que l'utilisation de la vue peut facilement être bloquée par des obstacles ou ne pas fonctionner à cause de différentes conditions d'éclairage. Les approches traditionnelles reposent principalement sur le contrôle en position pour manipuler des objets, et elles sont adaptées pour de simples pinces robotiques et des objets connus. De nouveaux capteurs rendent possible l'amélioration du contrôle des robots pour s'attaquer à des problèmes non résolus jusqu'à maintenant: manier des objets inconnus et découvrir de manière autonome des caractéristiques géométriques sur leur surface. Cette thèse aborde le problème qui consiste à contrôler un robot qui entre en contact en plusieurs points avec un environnement inconnu. Générer et garder de multiples contacts sur plusieurs parties des doigts des robot pendant l'exploration est une particularité essentielle qui différencie notre travail d'autre travaux d'exploration haptique dans la littérature, pour lesquels les contacts sont la plupart du temps limités à un ou deux points sur le bout des doigts.

Dans la première partie de cette thèse, nous nous préoccupons du problème de l'exploration de surfaces et d'objets partiellement connus, pour les modéliser et les identifier. Dans divers scenarios, des stratégies de contrôle et d'exploration sont développées pour suivre la surface ou le contour d'une surface de manière compliante avec des doigts robotiques.

Alors que les méthodes développées dans la première partie de cette thèse fonctionnent bien pour des objets de taille et de variation de forme limitées, la seconde partie est dédiée au développement d'un algorithme qui maximise le contact avec des surfaces inconnues, de toute taille ou forme. Maximiser les contacts permet de récolter l'information plus rapidement et aussi d'attraper des objets de manière plus stable. Dans ce but, nous développons un algorithme basé sur la formulation de l'espace des taches pour rapidement permettre le contrôle en couple de robots activement compliants, tout en conservant des contraintes, en particulier sur les forces de contact. Nous développons aussi une stratégie pour maximiser la surface en contact, à partir seulement de l'état actuel des contacts, c'est-à-dire sans information additionnelle concernant l'objet ou la surface.

Dans la troisième et dernière partie de cette thèse, nous développons une nouvelle méthode pour apprendre aux robots comment réagir à des informations sensorielles pendant l'exécution d'une tâche, en modulant des systèmes dynamiques par l'utilisation des signaux externes. Nous étendons une approche existante pour moduler localement des systèmes dynamiques, en basant la modulation sur des signaux externes. Nous abordons le problème qui consiste à attraper des objets en utilisant uniquement l'information tactile et de manière autonome, avec cet algorithme. Notre approche utilise l'information collectée avec des capteurs tactiles et un filtre à particules pour estimer la position de l'objet. Ceci est utilisé pour moduler la dynamique du mouvement du robot, en variant d'un mouvement de recherche à un mouvement pour attraper un objet, en fonction du progrès de la localisation. Cela permet d'accomplir rapidement et de manière autonome une tâche de localisation et de préhension d'objet, le tout dans une seule structure algorithmique flexible.

Mots Clés: Exploration tactile, Compliance active, contrôle avec multi-contacts, Systèmes dynamiques.

Table of Contents

1 Introduction . . . 1
- 1.1 Motivation . . . 1
- 1.2 Main Contributions . . . 5
- 1.3 Thesis outline . . . 8
- 1.4 Publications . . . 9

2 Background . . . 11
- 2.1 Related work . . . 11
  - 2.1.1 Tactile sensing hardware . . . 11
  - 2.1.2 Tactile sensing for object and surface classification . . . 12
  - 2.1.3 Tactile sensing and control . . . 15
  - 2.1.4 Grasping under uncertainty with multiple DOFs hands and tactile sensors . . . 16
  - 2.1.5 Operational space framework and null-space . . . 18
  - 2.1.6 Dynamical systems for manipulation . . . 20
- 2.2 Our approach . . . 22

3 Tactile compliance and surface recognition . . . 23
- 3.1 Introduction . . . 23
- 3.2 Face classification using touch with a humanoid robot . . . 24
  - 3.2.1 The probing mechanism . . . 24
  - 3.2.2 Face identification . . . 27
  - 3.2.3 Results and discussion . . . 30
  - 3.2.4 Conclusion . . . 33
- 3.3 Experiments with stretchable tactile sensors . . . 35
  - 3.3.1 Robot integration . . . 36
  - 3.3.2 Results . . . 38
  - 3.3.3 Conclusion . . . 38
- 3.4 Bimanual compliant tactile exploration . . . 42
  - 3.4.1 Exploration strategy . . . 42
  - 3.4.2 Object Identification . . . 48
  - 3.4.3 Experiment . . . 49
  - 3.4.4 Results . . . 52
  - 3.4.5 Conclusion . . . 52
- 3.5 Conclusion . . . 53

4 Multiple tactile contacts control for exploration and grasping 59
- 4.1 Introduction . . . 59
- 4.2 Controller structure . . . 60
- 4.3 Increasing contact area . . . 65
- 4.4 Experiments . . . 66
  - 4.4.1 Exploration . . . 67

4.4.2 Compliance experiments 75
4.5 Discussion and conclusion 85

5 Learning Externally Modulated Dynamical Systems 89
5.1 Introduction 89
5.2 Approach 91
5.2.1 Locally Modulated Dynamical Systems 91
5.2.2 Externally Modulated Dynamical Systems 92
5.2.3 Design of the modulation function 92
5.2.4 Illustrative examples 94
5.2.5 Learning EMDS 97
5.3 Autonomous Localization and Grasping 98
5.3.1 Belief model 99
5.3.2 EMDS 99
5.3.3 DS coupling and hand DS 102
5.3.4 Active compliance algorithm 103
5.4 Experiment 1: Autonomous Localization and Grasping in simulation 103
5.4.1 Experimental setup 104
5.4.2 Results 105
5.5 Experiment 2: Reaching while avoiding obstacles 107
5.5.1 Experiment 2a: Avoiding obstacles 107
5.5.2 Experiment 2b: Navigating between obstacles 115
5.6 Discussion 121

6 Conclusions 123
6.1 Main Contributions 123
6.2 Limitations and Future Work 124
6.3 Final Words 126

Appendices 129

A Appendices for Chapter 4 131
A.1 Exp 1: Details of the control 131

B Appendices for Chapter 5 135
B.1 Likelihood computation for the particle filter's measurement step 135
B.2 Proofs of stability for EMDS 136

References 139

Curriculum Vitae 149

Chapter 1

INTRODUCTION

"For robots, the final frontier is not space, it is your living room."
Cynthia Breazeal
(Personal Robots Group at the Media Lab, MIT)

1.1 Motivation

Humans rely on the sense of touch in almost every aspect of life. Without even being conscious of tactile sensations, we carry out complex tasks that could not be achieved without it, as illustrated in Figure 1.1. We sometimes get to experience how much we rely on touch when our fingers are numb from cold. Ordinarily trivial tasks such as manipulating objects or balancing a phone between the fingers consequently become arduous. Thanks to our tactile sensations, we are able to tie shoelaces without looking at them. This requires to finely localize the shoelace's position on the finger, slightly twist it in different ways, while holding another shoelace with other fingers or another part of the same finger.

We can also find keys inside a bag full of different objects. Skin pressure guides the arm to control forces exerted on possibly fragile items, and we are able to detect the change of temperature due to touching the metallic surface of the keys compared to a similar object in shape but not in material. This is all possible because humans are equipped with highly sensitive and multimodal tactile afferents, which each specialize to respond to different interactions (see Figure 1.2), such as high-frequency skin deformation, static pressure, or skin stretch.

With robots moving into human-inhabited environments, touch becomes of

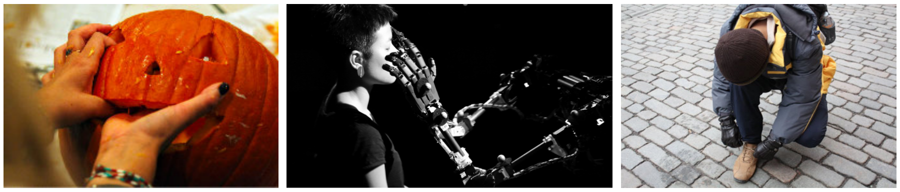
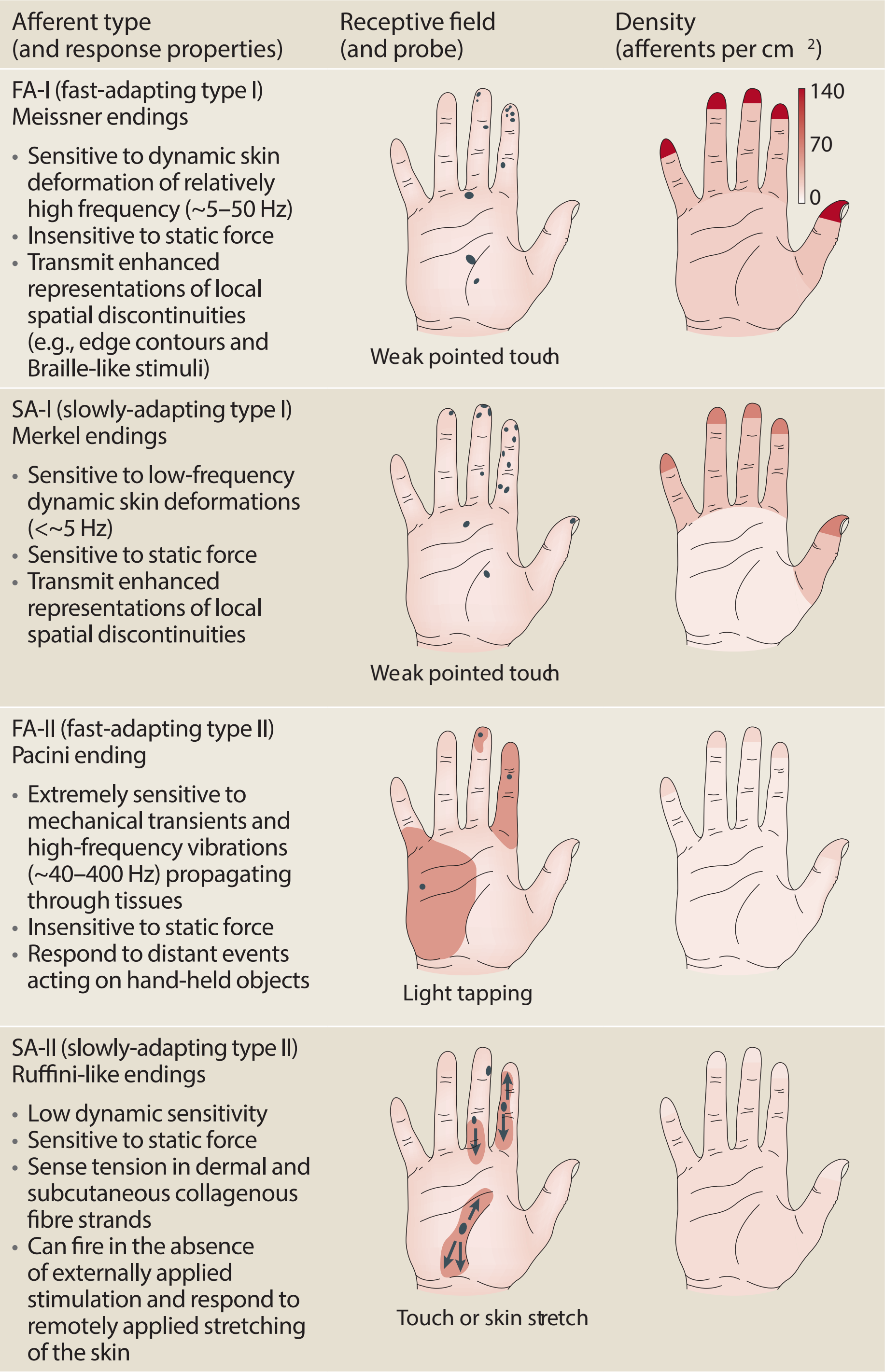


Figure 1.1: Examples of daily use of touch. From left to right, carving a pumpkin for Halloween (©Basilio Norris), a robot touching the face of a person to identify her (©Louis Philippe Demers), someone tying shoelaces while wearing gloves (©http://farofflands.wordpress.com).



Afferent type (and response properties)	Receptive field (and probe)	Density (afferents per cm ^{2})
FA-I (fast-adapting type I) Meissner endings • Sensitive to dynamic skin deformation of relatively high frequency (~5–50 Hz) • Insensitive to static force • Transmit enhanced representations of local spatial discontinuities (e.g., edge contours and Braille-like stimuli)	Weak pointed touch	140 70 0
SA-I (slowly-adapting type I) Merkel endings • Sensitive to low-frequency dynamic skin deformations (<~5 Hz) • Sensitive to static force • Transmit enhanced representations of local spatial discontinuities	Weak pointed touch	
FA-II (fast-adapting type II) Pacini ending • Extremely sensitive to mechanical transients and high-frequency vibrations (~40–400 Hz) propagating through tissues • Insensitive to static force • Respond to distant events acting on hand-held objects	Light tapping	
SA-II (slowly-adapting type II) Ruffini-like endings • Low dynamic sensitivity • Sensitive to static force • Sense tension in dermal and subcutaneous collagenous fibre strands • Can fire in the absence of externally applied stimulation and respond to remotely applied stretching of the skin	Touch or skin stretch	

Figure 1.2: Tactile sensory innervation of the hand (Johansson and Flanagan, 2009).

primary importance to be able to interact with everyday objects. Traditionally, robots have been confined to industrial settings where the environment can be precisely controlled. In uncontrolled environments, information is incomplete and should be gathered, e.g. by touch or vision. In an effort to handle unknown and dynamic environments, much effort has been put in providing obstacle avoidance skills to robots (Khatib, 1986), especially for safety purposes, for instance to navigate between humans (Trautman et al., 2015). However, in some cases, contact should not be avoided. On the contrary, contact is sometimes either necessary because we need to manipulate objects by touching them, or simply because it is the only way to obtain information. Other means of gathering information such as computer vision are limited by occlusion, illumination conditions and only provide partial information about texture and other surface properties. It also requires heavy and complicated software processing to handle the image data.

Fortunately, recent advances in tactile sensing offer a range of research directions in robotics for allowing robots to be in contact at multiple points on their body. Moreover, thanks to advances in the design of dexterous humanoid hands designed to be able to manipulate complex shapes, we can now consider manipulation that exploits the entire shape of the fingers. Such manipulation requires precise control of multiple contact points along the fingers.

However, up to now, the use of tactile sensing has been mostly limited to a few contact points on the end effectors of robots, often without maintaining contact during relative motion between the robot and the contacted area. The problem of exploring completely unknown surfaces by touch has also not been addressed until now.

Grasping, another crucial and widely studied area in robotics, has not yet benefited fully from advances in tactile sensing. While tactile sensing is essential in order to finely guide fingers on objects, especially under uncertainty, its use in robotic grasping is mostly limited to fingertip sensors. One reason for this may be the scarcity of reliable multi degree of freedom (DOF) robotic hands fully equipped with tactile sensors, besides the fingertips.

For most of this thesis, we consider robotic systems composed of an arm and hand with multiple dexterous fingers and equipped with artificial tactile skin to sense contact position and intensity. First, we investigate how to adapt the arm's position during the exploration. This is important in order to keep fingers in contact, as well as to avoid other collisions during the exploratory motion. For instance, how should the robot's arm move while exploring a fixed unknown surface, or both arms when exploring an object held by a bimanual humanoid robot. We also consider the problem of the motion of each link of the fingers in order to make contact and comply with unknown surfaces. In particular when robotic hands have many degrees of freedom, creating contacts on many parts of even a single finger can create improvements by speeding up the exploration process, as well as provide more tactile information simultaneously. This can be useful for the control of a robot or for the stability of a grasp. Indeed, in order to perform stable and strong grasps, tactile sensors should be available on the whole surface of the fingers to detect all the contacts and adapt them.

We also consider how to control the arm and fingers during contact in order to keep contact forces low and to avoid loosing contacts. This is crucial both for safety of the interaction and speeding up the exploration. The global exploratory motion of the arm should be compatible with the local finger motion on the surface of the explored shape.

Finally, we also study the use of tactile and force sensing to provide Dynamical Systems (DS) with the capacity to react to contacts. DS offer an efficient way to encode manipulation tasks such as reaching or grasping, with the advantage of a very fast computation time and the possibility to react instantly to perturbations. They can also benefit from the Learning from Demonstration (LfD) paradigm (Billard et al., 2016; Argall et al., 2009), which eliminates the need to code for the motion explicitly. Instead, the skills are acquired based on demonstrations of the task. In this thesis, tactile sensing is used to modulate DS while preserving important stability properties. This is important as it offers a way to include external sensing in the teaching process, so that the robot takes into account tactile information during task execution.

This thesis's approach is illustrated on Figure 1.3.

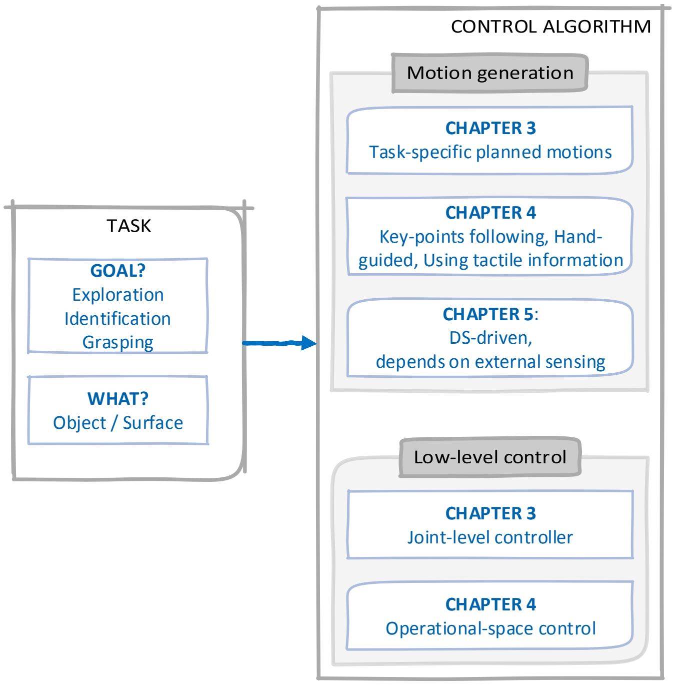


Figure 1.3: This thesis's approach. The tasks to achieve (left) require both motion generation and low-level control algorithms (right). This figure details in which chapters these problematics are addressed.

1.2 Main Contributions

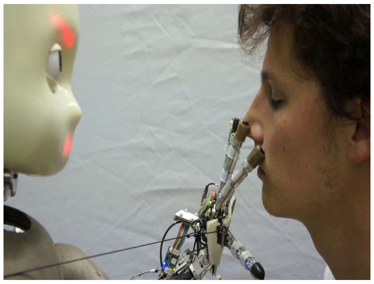
In this thesis we bring to light three main contributions:

Control and exploration strategies for compliant exploration

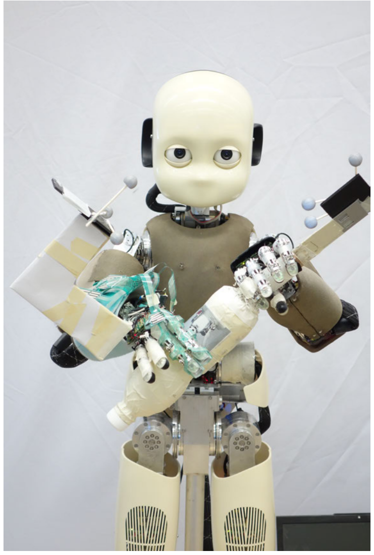
Haptic object exploration and modeling is traditionally performed through successive grasps (Schneider et al., 2009) or by following the surface with a probe-like robotic end-effector (Okamura and Cutkosky, 1999). We develop approaches to provide a more natural way to gather tactile data, namely with multiple contacts at the same time and in a continuous manner. This contrasts with alternative approaches using only one contact point (Jamali et al., 2016) (e.g. a fingertip), and "poking" strategies (Meier et al., 2011). In this thesis, two scenarios are studied in which we gather and process data about objects in two different ways.

In the first scenario, partial finger trajectories across human-like faces are used for classification. Because proprioceptive information is directly used to model the faces, this process does not require to probe iteratively different locations or to build a precise 3D model of the face, in contrast with most of the previous existing works.

In the second, geometrical contact information is gathered with robot kinematics during bimanual exploration of unknown objects, which has not been done before. An exploration strategy is developed to guide the motion of the two arms and fingers along the objects. The motion is generated on the go without planning for both arms to maximize the surface on the object that can be explored, while keeping compliant contact with multiple phalanxes of the fingers to gather tactile data. Because our method is not based on planning, it can run very fast at runtime and allows a fast exploration. In both these scenarios, we propose a low-level controller of the fingers which provides active compliance using tactile signals, This controller is also applied to the experimental trial of a stretchable tactile sensor prototype mounted for the first time on the dorsal side of a robotic hand.



(a) Face exploration



(b) Bimanual exploration

Figure 1.4: The iCub humanoid robot in some of our experiments. See Chapter 3 for more details.

A controller to maximize contact

We propose a framework to control a robot making multiple contacts with an unknown surface of arbitrary size and shape. It consists of an algorithm to compute torques given a task priority, using the original operational-space formulation, and hierarchical task priorities. We extend the closed-form null-space computation of torques with inequality constraints. This allows to control at a high frequency a robot in the interaction force null-space with desired contact points (equality constraint, keeping the normal interaction force constant), and undesired contact points (inequality constraint, setting a maximum allowed interaction force). This framework also consists in a strategy to bring the robot links in contact with the unknown surface. This algorithm can be applied to grasping, where the active adaptation of the fingers to the shape of the object ensures that the hand encloses objects with multiple contact points. We show that this improves the robustness of the grasp compared to simple enclosing strategies. We test these algorithms both in simulation and on a 16-DOF robotic hand customly equipped with tactile sensors on the whole inside surface of the fingers.

Externally Modulated Dynamical Systems

In order to modulate existing Dynamical Systems (DS) based on external sensory information, we present an extension of the existing Locally Modulated Dynamical System (LMDS), named Externally Modulated DS (EMDS). The EMDS accepts external input to automatically modulate between two DS while conserving important stability properties, according to an activation function. We also provide methods to learn this activation function from demonstrations.

In a first application, we reuse the previously developed hand controller to solve the problem of autonomous exploration and grasping from solely tactile data. We propose a framework integrating 1) the active exploration controller for the finger compliance, 2) a tactile-based particle filter for object localization and 3) the EMDS. The state of the probability distribution given by the state estimation of the particle filter provides an estimate of the progress of the localization task. This is used as the input of the EMDS to modulate between searching and grasping motions. A coupling between the arm and hand Dynamical Systems allows to conjugate the exploratory and grasping behaviors. This framework allows to generate natural and autonomous object localization and grasping in one flexible framework, hence without explicit segmentation. We also apply this algorithm to teach a robot how to react to collisions in order to navigate between obstacles while reaching.

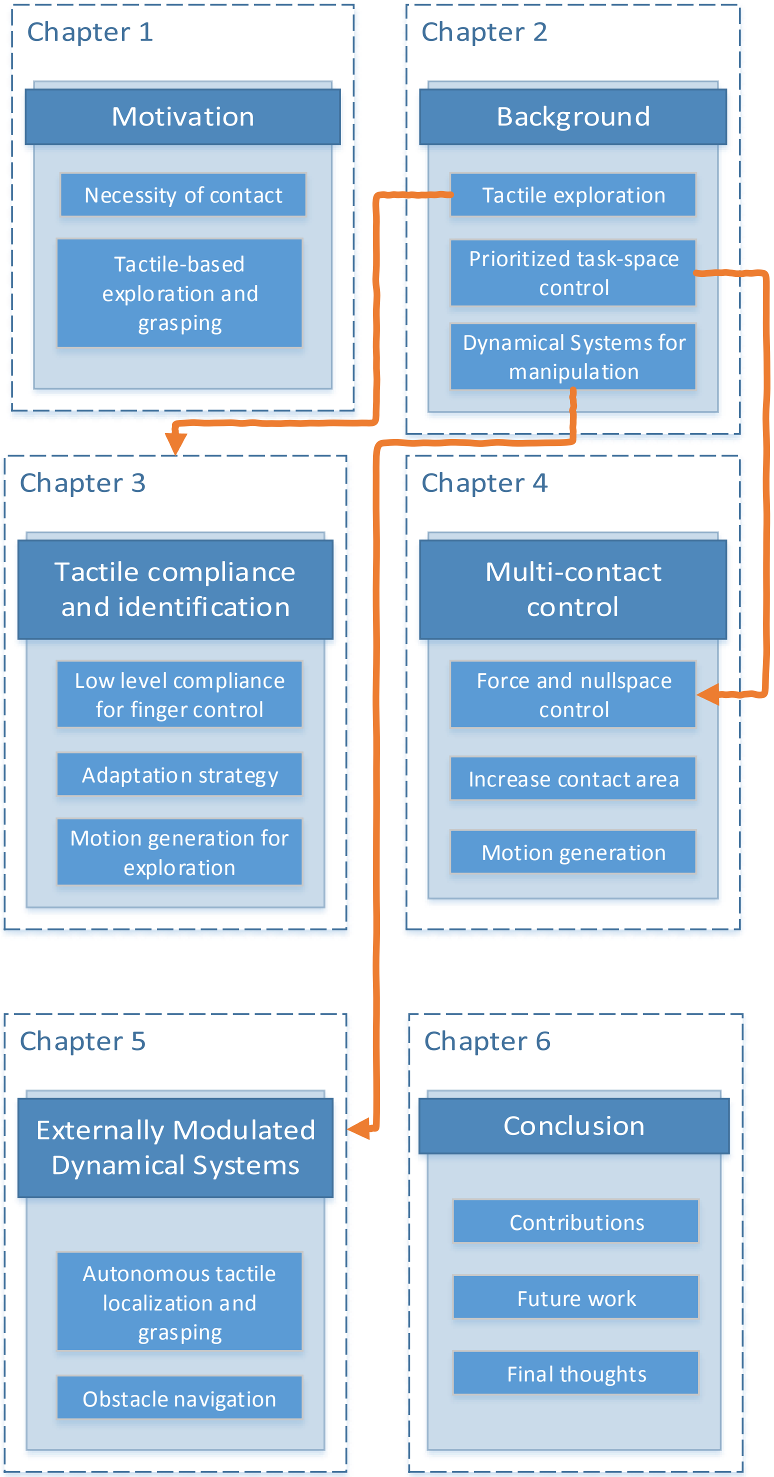

Chapter 1
Motivation
Necessity of contact
Tactile-based exploration and grasping
Chapter 2
Background
Tactile exploration
Prioritized task-space control
Dynamical Systems for manipulation
Chapter 3
Tactile compliance and identification
Low level compliance for finger control
Adaptation strategy
Motion generation for exploration
Chapter 4
Multi-contact control
Force and nullspace control
Increase contact area
Motion generation
Chapter 5
Externally Modulated Dynamical Systems
Autonomous tactile localization and grasping
Obstacle navigation
Chapter 6
Conclusion
Contributions
Future work
Final thoughts

Figure 1.5: Roadmap of the Thesis with key points.

1.3 Thesis outline

The thesis is structured according to three main contributions outlined in the previous section, each comprising a chapter. The following paragraphs give a detailed outline of the structure of this thesis, see Figure 1.5.

Chapter 2 - Background and related work

This chapter presents a review of related work. The novelty of our approach is discussed in detail in relation to the state of the art in the area.

Chapter 3 - Tactile compliance and surface recognition

In this chapter, we present an approach for exploring partially known objects or surfaces with tactile sensors, with the objective of modeling or identification. When only partial information about the surface to make contact with is available, close-form methods to compute robot trajectories in contact cannot be used because of the uncertainty. We present different scenarios for which control and exploration strategies are developed to compliantly gather tactile information from contact between the robot's fingers and an unknown object.

Human-like faces are explored with the hand of a humanoid robot and the finger trajectories are modeled with statistical series analysis for classification. Bi-manual exploration is also studied as a mean to increase the relative workspace of a robot holding an object for reconstructing its shape with tactile sensors mounted on the robot's fingers. The same compliance mechanism is also successfully applied to test an integrated stretchable tactile sensor skin mounted for the first time on the dorsal side of a robotic hand, in collision detection and surface recognition experiments.

Chapter 4 - Multiple tactile contacts control for exploration and grasping

In this chapter, we tackle the problem of exploring completely unknown objects, for which a robot must be controlled to a) make contact with the surface of the object and b) handle contact forces during the interaction. The framework developed in this chapter is particularly useful for robotic hands with many degrees of freedom (DOFs). It allows for rapid exploration of surfaces and is applied to grasping by generating highly stable enveloping grasps. This is achieved by keeping lower priority tasks in the null-space of the contact tasks and allows to keep interaction forces constant. The robot thus keeps multiple simultaneous contacts while moving to create additional contacts.

Chapter 5 - Learning Externally Modulated Dynamical Systems

This chapter presents the Externally Modulated Dynamical Systems (EMDS) algorithm and multiple applications in grasping and obstacle navigation. In the first application, we tackle the problem of autonomous grasping using only tactile data. First, our framework consists of a contact particle filter for object state estimation. Then, the EMDS algorithm, used to generate the arm motion according to external input, is detailed. The EMDS is coupled with a second DS that provides complementary information to the active compliant exploration algorithm described in Chapter 4. In a second series of experiments, we use more complex external signals to navigate between obstacles, depending on collision information from a force-torque sensor.

Chapter 6 - Conclusion

In the final chapter, we conclude by providing a summary of the work achieved, outlining the key contributions and limitations. We also discuss avenues for future work.

1.4 Publications

Large portions of this thesis have been published in peer-reviewed conferences and journals. The human-like face exploration experiments presented in Chapter 3 were published in Sommer and Billard (2012). Bimanual object exploration and identification has been published in Sommer et al. (2014). The results of Chapter 3 have also lead to a publication in collaboration with postdoctoral fellow Aaron Gerratt and Prof. Stephanie Lacour (Gerratt et al., 2014), part of the Laboratory for Soft Bioelectronic Interfaces at EPFL. The contents of Chapter 4 have been published in Sommer and Billard (2016). The contents of Chapter 5 are under submission.

Chapter 2

BACKGROUND

The use of tactile information is crucial to the development of robotics and is an increasingly active area of research. It involves a wide spectrum of fields: *tactile sensors development*, *object and surface recognition*, *robot control under contact*, *motion generation* and *grasping*. In this chapter, we will present the most relevant work to this thesis in each of these domains.

This chapter unfolds as follows: in Section 2.1.1, we begin by briefly presenting the current state of tactile sensor technology. In Section 2.1.2, we present the use of tactile sensing to classify objects or surfaces. In Section 2.1.3, we describe the existing work tackling robot control using tactile information. In Section 2.1.4, we present the grasping applications of tactile sensing. Finally, we present the operational space framework useful to control robots in contact in Section 2.1.5 and introduce Dynamical Systems in Section 2.1.6.

2.1 Related work

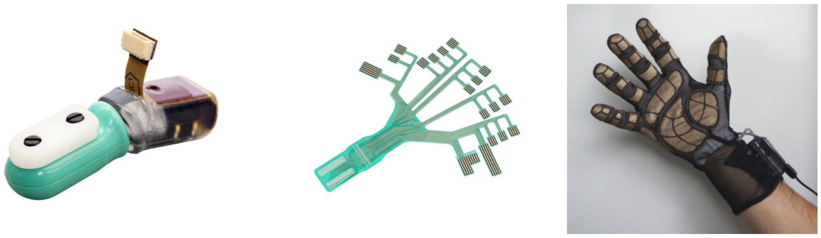
2.1.1 TACTILE SENSING HARDWARE

Tactile sensors encompass artificial devices that provide measurements of different modalities by contact[1]. Artificial skin has sparked interest in robotics for several decades (Harmon, 1982) and this section very briefly presents a few of the recent progresses. For an in-depth review of tactile sensing, please refer to Kappassov et al. (2015).

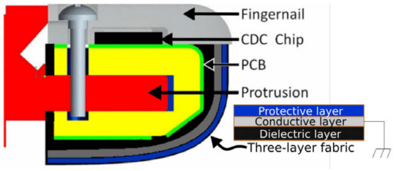
The sensing modalities provided by tactile sensors can consist of contact force – normal and tangential –, torque, temperature, vibrations or surface properties: texture, friction coefficient. Comparably to humans (refer back to Figure 1.2 from Johansson and Flanagan (2009)), different sensors are designed to measure different signals, and are based on different sensing types. Typically, the limits of current technology restricts the design of sensors by choosing a trade-off between spatial resolution, sensitivity, frequency response, multimodality and complexity of construction.

---

[1] The word *tactile*, derivated from *touch*, *toccare* (latin), comes from the Onomatopoeia “toc” (“knock” in English), evoking the sound of two objects colliding.



(a) SynTouch BioTac (b) Tekscan Grip system (c) Fabric-based glove sensor



(d) iCub tactile fingertip design

Figure 2.1: A few recent tactile sensors used in robotics. a) The Syntouch Biotac, b) the Tekscan Grip system, c) (Büscher et al., 2015), d) Jamali et al. (2015)

Tactile sensors are based on the measurement of a physical change of some material, deformed or influenced by contact. The sensing systems can measure a change of capacitance, resistance, optical distribution, pressure and electrical charge.

A few examples of tactile sensors can be found in Figure 2.1: the BioTac multimodal sensor provides temperature, high sensitivity and high frequency coarsely localized contact data, and low sensitivity and low frequency localized contact data. The Tekscan sensor is designed to fit an anthropomorphic hand and provides a wide range of pressure sensitivity with 2mm spatial resolution. The fabric-based sensor from Uni Bielfeld (Büscher et al., 2015) is based on the piezo-resistive effect. It is both flexible and stretchable to comply with the finger motion. The iCub tactile fingertip (Jamali et al., 2015) integrates sensing technology in the design of the fingertip. The sensor is composed of 12 sensing elements distributed on the fingertip, protected by a thin layer of fabric.

In this thesis, we mostly use the Tekscan tactile sensors which we fit to several robotic hands. We also use the integrated tactile sensors in iCub's hand, and a prototype of stretchable tactile sensor.

2.1.2 TACTILE SENSING FOR OBJECT AND SURFACE CLASSIFICATION

Touch brings important information when entering in contact with an object or surface, primarily the contact area and intensity. This is already enough to discriminate between different sorts of objects. Existing work in tactile exploration can be classified into two main categories: local and global exploration.

Local feature exploration

Local exploration strategies focus on gathering local information about an object's surface and extrapolating for identification. For instance, a robotic finger equipped with a tactile sensor is moved on the surface of an object to detect fine surface features (Okamura and Cutkosky, 1999). Another approach is to model the curvature of a surface at one point by using data from three differently oriented curves (Ibrayev and Jia, 2006). These curves are gathered from three trajectories concurrent at one interest point. The curvature profile is then matched to a database of objects for identification: the normal direction and the two main curvatures. Another type of local exploration consists in identifying surface properties, such as roughness, fineness and traction, and using these properties to classify materials (Fishel and Loeb, 2012). In Rosales et al. (2014), both shape and friction coefficients are modeled under a probabilistic framework during the exploration of an object's surface with both a tactile probe and an RGB-D camera. Using tactile array sensors, creating a tactile image of the contact between a grasped object and the gripper is popular: it is then for instance possible to differentiate between rough, flat, edge, cylindrical and spherical contact shapes using a neural network (Jiménez et al., 1997). Tactile images (2d-arrays) are also often processed with Self-Organizing Maps (SOM) in order to classify grasped objects: for instance, data from a few grasps with a 12-degrees of freedom robotic hand are used to classify 10 different objects in Johnsson and Balkenius (2007).

Global features

However, comparing a local feature is often not enough to distinguish between two complex objects. In Schneider et al. (2009), a bag-of-features approach is applied to generate object histograms describing their local features along their length, with tactile information retrieved from several grasps at different positions along the object's axis. This allows to keep a sense of continuity and to represent the object globally from local features without the need for precise localization during the exploration. A systematic approach is also used for reconstructing 3-D point cloud models of objects with a 3-fingered hand and tactile sensors (Meier et al., 2011). Tactile sensing is combined with proprioceptive information to obtain absolute contact locations. The precise 3D point clouds of the scanned objects are then compared with stored 3D models for recognition. However, this is a slow method because of the systematic probing (the fingers are opened and closed a hundred times for an object smaller than 9cm), and is restricted to small objects that can fit between the robot's fingertips. Self-organizing maps (SOM) can also take into account kinesthetic data in addition to tactile data: in Navarro et al. (2012), finger joint angles and touch information form a multi-sensory input to SOMs and are used to cluster grasped objects. Joint angles give information about the object's size whereas tactile data gives more precise shape information.

Continuous exploration

Finally, we are particularly interested in continuous exploration with several fingers, as it is better suited to reconstruct the shape of an object thanks to the flexibility of the multiple degrees of freedom available. Humans do not release and grasp several times an object in order to recognize it by touch, they rather follow the surface with their fingers. Indeed, iterative touches take more time and the object's position may be lost when the contact is broken. The first works focused on the reconstruction of parametric models of objects: already in 1990, Allen, inspired by exploratory procedures from Lederman and Klatzky (1987), explored objects modeled by superquadrics with a contour following method that used the model's parameters to compute a trajectory (Allen and Michelman, 1990; Stansfield, 1991). In Heidemann and Schopfer (2004), a tactile sensor array is moved around the surface of a convex-shaped object while passively rotating to follow the slope. The time-series of 2D pressure profiles are fed to several neural networks for classification after local PCA for feature extraction and dimensionality reduction. Vision can also be coupled to tactile information in order to reduce the data to lower dimensionality using a multi-modal dimensionality reduction technique (Kroemer et al., 2011) and help the classification of textures. Time series of tactile data are very high-dimensional and decreasing the dimensionality is done by using synchronized visual features with a multi-modal mapping method. This is achieved by finding lower dimensional representations where the classification performance is improved. In another application, continuous probing is used to identify surfaces by mobile robots (Giguere and Dudek, 2011). A probe uses an accelerometer attached near its tip in contact with the ground to collect data describing the surface on which the robot is moving. Classification is done by analyzing selected features of the data from fixed time windows. In Maekawa et al. (1995) and especially Okamura et al. (1997), tactile sensor arrays on fingertips and palm are used to gather data while rolling and sliding fingers on an object during haptic exploration. During the exploration, some fingers are responsible for grasping while the others explore the surface by rolling or sliding on it. Since the object's position and orientation are tracked by assuming pure rolling during the phase when the object is being moved, features detected by the tactile sensors can be added to a model of the object.

This paragraph tackled the use of tactile sensing for gathering information about surfaces and objects while touching them. Some methods involve poking, getting the image of the contact surface on a tactile array, some involve pinching or grasping with the purpose of getting an approximation of the size of the object and more tactile information. Some methods involve sliding on the surface, allowing at the same time to increase the area on which information is obtained and to collect dynamic data relative to the texture. In these approaches however, the sense of touch is only used passively: tactile information is gathered during predefined motions of the robot.

2.1.3 TACTILE SENSING AND CONTROL

In the previous section, we presented applications where tactile sensors are used to collect data, especially for classification, without actually being included in the control loop of the robot. However, humans use touch to drive the control of their fingers, not only to gather data, but also for instance to control slip when lifting object (Johansson and Westling, 1984). In robotics, many control-related tasks rely on tactile information. We present here a few of them, including slippage detection, adaptation of grasp posture, control of manipulator stiffness, or update of the end-effector orientation. By detecting small vibrations associated with slip while grasping an object, the grasping force can be adjusted to avoid slippage of the held object (Tremblay and Cutkosky, 1993; Schürmann et al., 2012; Su et al., 2015; Narendiran and George, 2015). This is achieved thanks to a tactile sensor equipped with an accelerometer. In addition, the normal and tangential force measurements at the time slip is detected can provide an accurate measurement of the coefficient of friction of the material. Touch information is in this case used both for control and for gathering information on the contact surface properties. Tactile signals can guide the motion of exploration, for instance to follow edges (Berger and Khosla, 1991; Chen et al., 1995). Stiffness control of an object aims at holding an object so that it acts as if suspended by a set of springs and dampers; tactile sensing can enhance this control by providing precise object-finger location information, which is required to achieve the desired impedance of a grasped object through torque commands (Son and Nowe, 1996). In that case, tactile sensors help determine the object's initial pose after grasping and track it as it rolls and slides against the fingers during manipulation, thus improving the control's accuracy. In Yamakawa et al. (2007), reaction forces on the fingers are measured through tactile feedback during a task of knot tying. This information is used to adapt the finger pinching force in order to follow a force profile optimized for a phase-based model of the task. In Jamisola et al. (2014), the authors tackle the task of exploring a discontinuous surface with a rolling end-effector and force-torque information with a compliant controller. This involves adjusting the controller to the orientation of the surface normal to maintain a desired normal force. While exploration with a single end-effector simplifies the control, it has limitations: when the probe is small, the exploration process is very slow, especially if the surface to be covered is large. However, if the probe is large, it cannot comply with arbitrary shapes (especially for convex objects) or cannot reach some areas. Another area of research focuses on using robotic hands and fingers or grippers, and tactile or force sensors to model the object's shape. Bierbaum et al. (2008) introduces the use of potential fields to drive the exploration of a five-fingered hand in simulation. While this allows autonomous reconstruction of several simple objects, the hand is controlled in velocity and thus the interaction forces are not taken into account. Besides, the exploration only uses the fingertips. There is no contact with the other finger links, as only fingertips are subject to the potential field. One of the most advanced works tackling tactile interaction is probably the one from Jain et al. (2013) in which multiple contacts occur on the arm of a robot, not for exploration explicitly but to help the robot reach trough cluttered space. They use model predictive control with a model of the contacts that assumes linear stiffness and optimizes for reaching a desired position with the end-effector, with constraints on contact forces. However, the objective of the controller is to reach a point with the end effector. It also ignores the posture of the rest of the arm: there is no focus on the tactile exploration itself. This exploratory approach is also limited as it requires to command the robot in position and thus is not ideally suited to the control of contact forces. Despite all these progresses, there is still no available control framework to create, manage and remove contacts with a robot and an unknown environment. In the following section, we present the grasping applications of tactile sensing, which share many similarities and problematics with the exploratory applications, especially under uncertainty.

2.1.4 Grasping under uncertainty with multiple DOFs hands and tactile sensors

Complying with the shape of an unknown object during grasping shares similarities with the exploration of unknown objects and can also benefit from tactile information. Indeed, both for exploration and grasping, external sensory information is necessary to actively comply if there are uncertainties in the position or the shape of the object, or in the robotic system itself.

Most of the work in grasping consist in planning grasps for known or partially known objects (Bicchi and Kumar, 2000; Goldfeder and Allen, 2011; Roa and Suárez, 2014). However, reliably controlling robotic fingers to realize generated grasps on a real platform with position and shape uncertainties remains a problem. Indeed, it is difficult to realize the planned grasps with a real robot hand, and this makes the quality evaluation less relevant in practice, as the realized grasps are less optimal than the planned ones (Kim et al., 2013). While *soft* systems approach this problem using passive mechanical compliance to adapt to position or shape uncertainties, active compliance is the only way to control rigid robotic hands with multiple degrees of freedom. Using additional sensory information, one can improve grasping success rates by detecting position errors. For instance in Hsiao et al. (2010) and Chen et al. (2015), torque or tactile sensors in the fingers are used to detect the first contact and compliantly pause the finger in contact before it tips over the object to be grasped. In Li et al. (2016), the authors exploit tactile sensors on the fingertips to control the finger contact force under shape uncertainty. From the perspective of using tactile data to drive compliant motions, Sauser et al. (2012) uses information from fingertip tactile sensors on a robotic hand to compliantly adapt the grasp of a selection of objects, by learning the non-linear correlation between finger position and tactile signature. After teaching correct grasping postures and fingertip forces by a human demonstrator guiding the robot, a model is learned to predict the expected finger joint configuration and tactile pressure given the contact normal at each fingertip. This model is then used to adjust the control of the fingers while holding an object. A feedback controller tries to satisfy both position and force (tactile pressure) constraints. The controller gives priority to the position control, so that force is taken into account as the position controller brings the finger into contact. Platt's null-space grasping control (Platt et al., 2010) uses local object geometry measurements to guide grasps and converge to unit frictional equilibrium. This involves following the negative gradient of two functions: force and moment residuals which are zero at this equilibrium. Because the force residual controller displacements are tangential to the surface, and the moment residual controller displacements are projected on the null space of the gradient of the unit frictionless force residual, the resulting motion corresponds to the fingertips sliding on the surface of the object. However, the ensured improvement of the chosen grasp metric is based on several assumptions: convex objects, 2nd order continuity of the surface and only two contact points. Finally, in these works, only the fingertips are taken into account and no attempt is made at controlling grasps with contacts on all links of the hand. When the whole hand, not only the fingertips, is used to grasp an object, realizing planned grasps becomes even more difficult as multiple contacts should be made between the fingers and the grasped object. Grasping synergies is an efficient concept to simplify control of high-dofs hands inspired by human grasping. Whether the synergy is integrated in the mechanical design of sub-actuated hands (Catalano et al., 2012; Grioli et al., 2012) or simulated in software (Ciocarlie et al., 2007; Bicchi et al., 2011), it decreases the dimensionality of the control problem. However, synergy-based grasping strategies can also lead to unsuccessful grasps and they do not seek to maximize the contact surface. Besides, the underactuation of synergy-based hands can be problematic when active control of all fingers and phalanxes is required, especially when the grasp controller can benefit from tactile feedback, for instance to place fingers in specific postures (eg. aligning the index along the handle of a knife, while the rest of the hand englobes the knife handle). For this reason, using tactile information can be useful and provide active compliance at all the desired contacts points on the fingers.

While grasping and local control of contact points require tactile information and can be seen as controlling contacts points separately, it is necessary to approach haptic exploration with constraints on the whole robot. In the next section, we present the operational space framework which allows to do so, and especially to take into account existing contacts in the problem.

2.1.5 OPERATIONAL SPACE FRAMEWORK AND NULL-SPACE

Haptic exploration fits well within the prioritized controller scheme, as some tasks – managing contact forces, avoiding joint limits – can be interpreted as constraints and be given a very high priority, while other tasks such as arm posture are less important. This framework is commonly used for humanoid control, including constraints on contact forces, but not for haptic exploration. Khatib's operational space framework (Khatib, 1987) allows to express the dynamics of the robot in task coordinates, and the prioritized simultaneous control of several tasks through cascaded null space projections (Khatib et al., 2004). More recently, this framework was used to control several contacts on different links of a robot arm (Park and Khatib, 2008), but there have not been results showing cases where the robot makes additional unpredicted contacts or looses some of its contacts. In Flacco et al. (2012), commands are automatically scaled down if they violate hard bounds at the joint level (position, velocity or acceleration constraints). This allows to have explicit hard constraints, which was usually not possible in that framework. Another approach towards prioritizing tasks is to formulate the inversion of the Jacobian as a quadratic problem. For instance, the Stack of Tasks approach (SoT) (Mansard et al., 2009) provides an interface to add and remove tasks automatically with a pre-specified hierarchy. Recently, hierarchical control schemes based on a sequence of quadratic programs (QP) can also handle inequality constraints for kinematic control (Kanoun et al., 2011) and dynamic control (Saab et al., 2013). Efforts have also been made to solve these problems fast enough for real-time control of humanoid robots with many degrees of freedom (Escande et al., 2014).

In this thesis, we follow the null space approach to prioritizing tasks and introduce a controller based on a modified null space projection matrix that allows to take into account inequality constraints. While this is not as efficient as the latest QP-based methods, it is an alternative approach simpler to implement and closer to the original idea since it only relies on matrix inversion and does not require an otherwise complex solver.

OPERATIONAL SPACE CONTROL

The dynamics of a manipulator in contact describe how the robot moves in response to torques applied at the robot joints and the contacts forces on the links. In this section, these dynamics are detailed in order to describe operational space control, task space and null space control of a robot.

Operational space control of a robot in contact

The equations describing the dynamics of a robot in contact are of the form:

$$M_q(q)\ddot{q} + b(q,\dot{q}) + g(q) + J_c^T(q)f = \tau \tag{2.1.1}$$

where $q, M_q(q), b(q,\dot{q}), g(q), f$ and τ are respectively the vector of joint angles, the joint-space inertia matrix, the Coriolis and centrifugal torques, the torques due to gravity, the contact forces and the vector of joint torques. $J_c(q)$ is the contact Jacobian, e.g. considering an operational point x on the robot where a contact occurs, the relationship between the virtual joint velocities $\dot{q}$ and the virtual velocity of the operational point, $\dot{x}$, is given by the Jacobian matrix at the contact point in this configuration:

$$\dot{x} = J_c(q)\dot{q} \tag{2.1.2}$$

The manipulator dynamics in the operational space are given by pre-multiplying Equation (2.1.1) with $J(q)M_q(q)^{-1}$. For better readability, we do not specify the dependency on the joint angles vector q and its derivatives from now on:

$$\ddot{x} - \dot{J}\dot{q} + JM_q^{-1}(b+g) = JM_q^{-1}(\tau - J_c^T f) \tag{2.1.3}$$

Task space and null space for redundant manipulators

The operational space framework for task-level control of redundant manipulators decomposes the overall motion behaviour into two components. The first is defined by the task behaviour, specified in terms of forces and moments in the operational space, F_{task}. This force is translated into a joint torque based on Equation (2.1.2): $\tau = J^T F_{task}$. This vector is however not completely specified in the case of redundant manipulators. The operational space framework allows to select from a set of task-consistent torque vectors to perform a secondary task. This secondary task is specified by an arbitrary torque vector τ_{sec}.

In order to ensure that this secondary torque vector does not affect the task behavior F_{task}, the additional torque is projected into the null space N of the task Jacobian J. The torque $N^T\tau_{task}$ resulting from the projection on the null-space does not affect the behaviour of the operational point. However, since the rank of N_{robot} is $N-k$ (k being the rank of J and N_{robot} the number of degrees of freedom of the manipulator), the behaviour of the secondary task is not guaranteed.

The operational space and secondary task are combined to obtain the general expression for the torque-level controller:

$$\tau = J^T F_{task} + N^T\tau_{sec} \tag{2.1.4}$$

Practically, the null-space projection matrix N can by obtained with:

$$N = I - \bar{J}J \tag{2.1.5}$$

with $\bar{J}$ the dynamically consistent generalized inverse of J, given by:

$$\bar{J} = M_q^{-1}J^T(JM_q^{-1}J^T)^{-1} \tag{2.1.6}$$

The use of this specific inverse ensures that the acceleration $\ddot{x}$ at the end effector is not affected by the projected torques.

Multiple task behavior

The concept of decomposing the control torques in separate tasks with different priorities can be extended to more than 2 tasks and priorities.

Assume a set of n tasks T_i, where T_i has higher priority than T_{i+1}. Every task is associated with a torque vector τ_i, Jacobian J_i and corresponding null space projection matrix N_i. These tasks can be performed simultaneously while ensuring strict hierarchy depending on the task's priority. The control torques are then given by:

$$\tau = \tau_1 + N_1^T(\tau_2 + N_2^T(\tau_3 + \cdots)) \tag{2.1.7}$$

or

$$\tau = \tau_1 + N_{prec(2)}^T\tau_2 + N_{prec(3)}^T\tau_3 + \cdots + N_{prec(n)}^T\tau_n \tag{2.1.8}$$

with $N_{prec(i)} = N_{(i-1)}N_{(i-2)}\cdots N_1$

This ensures that each task is executed as well as possible in the null space of all the tasks of higher priority. Note that τ_1 is not projected on the null space of any other task, therefore it is not altered by this projection process. The lower the task priority i, the lower the rank of the corresponding null space $N_{prec(i)}$, hence the task has less chances to be executed properly. Important tasks such as avoiding joint limits or keeping equilibrium in the case of a humanoid robot should thus have priority 1.

2.1.6 DYNAMICAL SYSTEMS FOR MANIPULATION

In robotics, Dynamical Systems (DS) have proven to be an interesting approach to motion generation, as an alternative to classical methods relying on separate planning and execution. They offer a simple way to integrate both steps into one formulation (Billard and Hayes, 1999; Selverston, 1980).

Dynamical Movement Primitives (DMPs) have recently gained popularity (Schaal et al., 2003; Ijspeert et al., 2013). They are a set of differential equations that can compactly represent a large variety of robotics tasks. Their mechanism also make it easy to incorporate in Reinforcement Learning, and learning without risking unstable behavior. They however rely on a phase variable acting as an implicit clock, forcing the system to converge to a linear system with ensured stability properties.

Time-invariant DS formulations (Gribovskaya et al., 2011a) allow to represent motions in a time-independent manner, in contrast with time-varying representations. Because stability is a major concern when dealing with DS, this has been addressed in Khansari-Zadeh and Billard (2011) for a specific parametric form of DS, Gaussian Mixture Regression (GMR). In Kronander et al. (2015), a formulation and an incremental learning method were introduced to represent motion from demonstrations, while ensuring bounded trajectories and no introduction of spurious attractors. This formulation does not base the stability analysis on a known Lyapunov function, therefore incremental demonstrations do not need to comply with an energy function. While asymptotic stability cannot be guaranteed, this has little influence on the resulting behavior and can be used to our advantage to create cyclic motion. In this thesis, we build upon this formulation to create DS with equivalent stability properties while depending on external signals.

Recently, Pastor et al. (2011) introduced a framework to react to sensory input while performing reaching-type motions with DMPs. In this work, the DMP's trajectory is adapted in order to match previously learned sensory signal, i.e. force information. This is done through a pre-defined mapping between sensing and end-effector accelerations, using the task Jacobian of the sensor. This approach is directly applicable to situations in which such mappings between sensory signals and control signal can be defined. This includes sensors with low-dimensional inputs such as force-torque sensors, but cannot be extended well to a tactile skin on multiple fingers for instance. For tactile data, it is generally not possible to define generic mappings from sensor signature to control response. In this thesis, we suggest to learn the mapping from external signal to modulation of the dynamics, which are provided by a time-invariant DS. More recently, the authors generalized their work in the Associative Skill Memory framework (Pastor et al., 2013), which switches between learned motor primitives based on sensory signature (using hard switches). This is based on the assumption that task representations should be stereotypical with as little variation as possible in order for the associated sensory recordings to have little variance. A wide range of tasks do not follow this description, including grasping objects with multi-fingered hands, for which contacts can occurs on many different parts of the fingers and in different order.

2.2 Our approach

In this chapter, we presented multiple uses of tactile sensing in robotics. In this thesis, we focus on exploration strategies to compliantly explore surfaces of objects, in order to identify them. In order to gather tactile information, the arm motion is generated depending on the scenario to avoid collisions and to provide configurations in which fingers can comply to the explored shape. For instance, we use tactile information to determine the optimal wrist orientation. We also provide a framework to control a robot with multiple contact points, while moving to explore and to create additional contact points. For this purpose, we rely on the operational space formulation and combine multiple task behaviors, including contact tasks and creating contacts. We propose a modified null space computation algorithm that allows to keep contact forces low during the exploration, differentiating between desired contact points and undesired contact points. Finally, we also use and extend an existing Dynamical System formulation in order to generate reaching and grasping motions while performing localization, in a single framework.

Chapter 3

TACTILE COMPLIANCE AND SURFACE RECOGNITION

3.1 Introduction

In this chapter, we tackle the control of robot's arms and fingers when using tactile sensors to explore partially known objects or surfaces. We propose different strategies to explore while compliantly gathering tactile information from contact between the fingers and objects. In different scenarios, we generate a motion for the exploration. First, simple linear trajectories to "scan" human-like faces for face identification by touch, or simple features on a flat surface. For more complex objects that should be touched from different angles, we develop a more complex bimanual exploration algorithm for reconstructing the object's shape using two arm and hands.

Concurrently, algorithms for object identification are provided to recognize the explored objects or surfaces: Hidden Markov Models to identify noisy data from face exploration, and point-cloud matching from 3D points generated with the robot's forward kinematics and a model of the tactile sensors.

All these experiments are carried out on the same humanoid robot platform, iCub (Metta et al., 2008), using several tactile sensors. In the different experiments, we use the capacitive tactile sensors integrated in the robot's fingertips, another set of sensors, Tekscan, customly fitted to the fingers of the robots, and a set of prototype stretchable tactile sensors mounted on the back of the robot's fingers.

This work lead to the following publications:

- N. Sommer and A. Billard. Face classification using touch with a humanoid robot hand. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 120–125, 2012. doi: 10.1109/HUMANOIDS.2012.6651508

- Nicolas Sommer, Miao Li, and Aude Billard. Bimanual compliant tactile exploration for grasping unknown objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6400–6407. IEEE, 2014

- A. P. Gerratt, N. Sommer, S. P. Lacour, and A. Billard. Stretchable capacitive tactile skin on humanoid robot fingers; First experiments and results.

In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 238–245, November 2014. doi: 10.1109/HUMANOIDS.2014.7041366

3.2 Face classification using touch with a humanoid robot

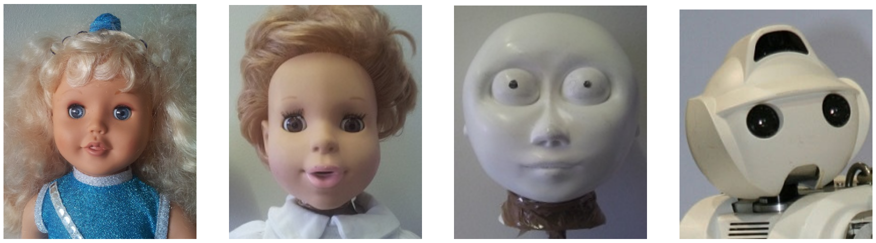
This section presents an experiment in which the iCub humanoid robot learns to recognize faces through proprioceptive information. We take inspiration in the way blind people recognize people's faces, i.e. through tactile exploration of the person's face. The iCub robot's tactile sensors are used to provide compliance in the hand motion so as to smoothly scan the facial features. The displacement of the fingers, as the robot explores the face, is used to build a model of the face using Hidden Markov Models. We show that the robot can successfully distinguish across the faces of a standard doll and the faces of three humanoid robots, the HOAP-3 robot, a Robota doll robot and MyDreamBaby, a commercial robotic doll.

This work combines the notion of continuous exploration of a surface and the idea of compliant control. Precisely, we use a) the tactile fingertips to introduce a compliant mechanism for the displacement of the fingers along the face and b) proprioceptive information, i.e. the position of the fingers during a motion, to classify the faces. In contrast with most of the previous works, this process does not require to probe iteratively different locations, or to build a precise 3D model of the face. Our approach rather relies on the essential characteristics of one continuous human-like motion across the face.

3.2.1 The probing mechanism

The goal of the experiment is to identify a face by touch. In order to do so, our humanoid robot - the iCub robot - moves its hand in a vertical plane, while its fingers actively follow the curve of the face to track its shape. There are four faces to classify in this experiment (see Figure 3.1).

The faces have been chosen because they all share similar basic features (eyes, mouth, roundness of the head), hence making the task to distinguish across their features more challenging: the traditional doll (*Doll1*) and the robotic doll (*Doll2*) have faces that are extremely similar from a tactile viewpoint, as the overall surface of the face and the distance across facial features are almost identical. Major differences lie in the shape of the mouth and nose of the two dolls. The face of the robot *Robota* is a scaled version of the *Doll1* robot and again differs from the previous faces mainly through its overall size, as well as the relative proportion of the face covered by the eyes and nose. The face of HOAP-3 robot is the most distinctive of all four faces, because of its protruding forehead and its lack of a nose.



(a) Doll1 (10x12cm)

(b) Doll2 (MyDreamBaby, 9x12cm)

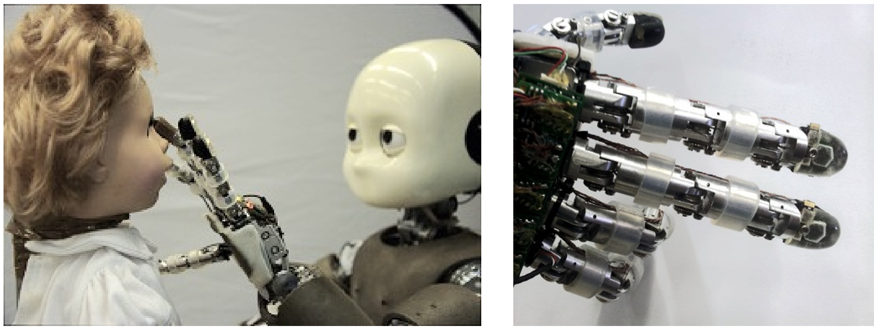
(c) Robota (plaster cast, 15x13cm)

(d) HOAP (12x13cm)

Figure 3.1: Faces to be sorted in the experiments(*width* x *heigth*)

iCub

The iCub robot is a 53-DOFs humanoid robot whose arms are composed of 7 joints, plus 9-DOFs hands (see Figure 3.2). The 7 arm joints are used to achieve the vertical motion of the hand while one joint per finger is used to follow the face (sole index and middle finger are used in this experiment).



(a)

(b)

Figure 3.2: Experimental set-up: a) The iCub robot moves slowly a finger over the face of *Doll1* and captures an image of the face of the robot through proprioceptive measurement. b) The iCub's hand is endowed with tactile sensors at the finger tips.

Hand trajectory

The hand is controlled so as to follow a predefined vertical line from the top to the bottom of the face, keeping a fixed orientation, palm facing the scanned head, pointing upwards, see Figure 3.4. The motion starts with the fingers at the level of the forehead and is stopped manually when the fingers reach the bottom of the face. For each face, this motion is repeated ten times: at each run, the hand is shifted horizontally so as to span homogeneously the whole width of the face (see Figure 3.3). These ten trajectories are used during the learning phase to create a model of the face (see Section 3.2.2).

During the motion, the index and middle fingers stay in contact with the

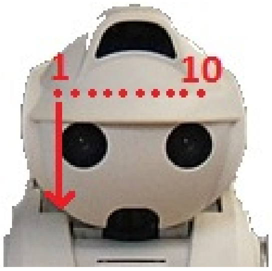


Figure 3.3: Front view scheme of hand trajectories and starting points – red dots – on the *HOAP* robot. Each dot represents the center of the middle and index fingertips on the head, at the beginning of the motion.

face by means of a pressure loop (detailed in the next section). Since the ring and little fingers are coupled and cannot be controlled independently, they are not used in the experiment: they cannot follow two different profiles simultaneously. The spacing between the fingers (adduction/abduction) is fixed during all the experiments. The angular values of the finger joints are recorded during the experiment. Each motion lasts approximately between 7 and 10 seconds, depending on the size of the face. The ten recordings of these angular values form the dataset used in the learning phase. Data are gathered at a rate of 50Hz, resulting in 400 datapoints on average.

PRESSURE CONTROL WITH TACTILE SENSING

The goal of the experiment is to record the motion of the fingers while they stay in contact with the face. This is achieved through tactile pressure control. Our iCub robot is endowed with capacitive tactile sensors on its fingertips (Jamali et al., 2015). Each of these sensors is composed of 12 *taxels* t_f^p (i.e. tactile pixels), $t_f^p \in [0, 255]$, with finger $f = 1, 2$ and taxel $p = 1..12$. The average pressure per finger s_f is used here as the controlled variable for the pressure loop:

$$s_f = \frac{1}{12} \sum_{p=1}^{12} t_f^p \tag{3.2.1}$$

Note that the faces used in the experiments have been covered with aluminum foil because this enhances the response of the capacitive sensors and hence ensures better tactile pressure control (refer to Figure 3.4).

A PD controller is implemented to follow a constant target pressure $\hat{s}_f$. This target pressure is manually adjusted so as to keep a contact with the face without damaging the fingers. Each finger f is thus controlled in current u_f following:

$$u_f(s_f, \hat{s}_f) = \kappa_p(\hat{s}_f - s_f) - \kappa_d \dot{s}_f \tag{3.2.2}$$

where $\dot{s_f}$ is the derivative of the total pressure at each finger, and $\kappa_p \in \mathbb{R}$ and $\kappa_d \in \mathbb{R}$ are the proportional and derivative coefficients[1].

---

[1] In our implementation, the gains κ_p and κ_d are hand-tuned.

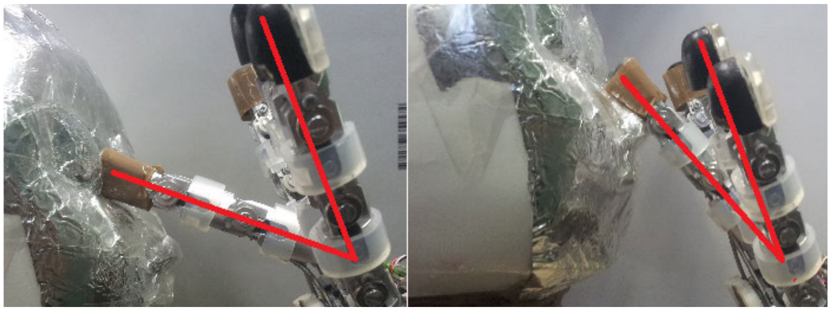


Figure 3.4: The fingers follow the curve of the face. On the left, a wide angle describes the depression of the eyes and on the right, the nose bump yields a smaller angle.

3.2.2 FACE IDENTIFICATION

DATA PRE-PROCESSING

The raw data from the experiments are the angles $\theta_f^{t,n}$, with $f = 1..F$ fingers ($F = 2$, index and middle fingers), $n = 1..N$ demonstrations and $t = 1..T$ timesteps. These values depend heavily on the distance between the hand and the face: the same face profile yields different results if the face to identify is slightly moved away from iCub's hand. A few pre-processing steps enable to get rid of this issue. First, we take the sinus of the angles in order to have a value linearly correlated with the distance between the hand and the face:

$$x_f^{t,n} = sin(\theta_f^{t,n}) \tag{3.2.3}$$

This gives us the data set $\left\{x_f^{t,n}\right\}_{t=0}^{T}$ (see Figure 3.5). The remaining constant shift following from the hand being further away during another motion can be removed by simply taking the derivative of x with respect to z, the vertical coordinate.

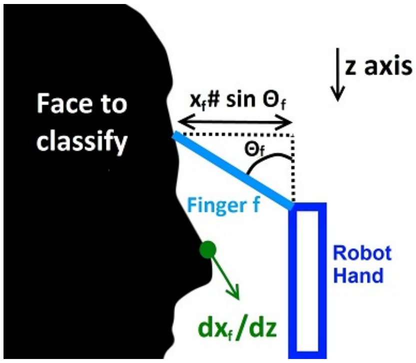


Figure 3.5: Scheme of the finger probing system for one finger. The hand moves along the z axis.

This linearized value is time dependent and the vertical velocity profile of the hand motion is not flat (the velocity is not exactly constant during the motion), we therefore re-sample the values according to the Cartesian vertical coordinate z. The new dataset $\left\{\tilde{x}_f^{g,n}\right\}_{g=0}^{G}$, indexed by g, spans regularly the vertical axis z. The data points $\tilde{x}$ are interpolated from x, with G the chosen number of sampled datapoints[2]. The linearized profile is then differentiated with respect to z to obtain a set of data independent from the velocity of the hand during the motion:

$$\mathcal{D}_r = \left\{ \frac{d\tilde{x}_f^{g,n}}{dz} = \frac{\tilde{x}_f^{g,n} - \tilde{x}_f^{g-1,n}}{\Delta z} \right\}_{f=1,n=1,g=1}^{F,N,G} \tag{3.2.4}$$

with r $\in \{Doll1, Doll2, Robota, HOAP\}$.

The data is then de-noised using a *lowess* filter (Cleveland, 1981) – local regression using weighted linear least squares, here with a 1st degree polynomial model. These pre-processing steps yield data containing velocity profiles which describe the slope of the faces along two vertical lines described by the fingers. This information is sufficient to recreate the original face profiles – sectional views as in Figure 3.5 – by integrating the slope.

The advantage of pre-processing the data is visible on Figure 3.6: while the raw trajectories are not aligned and vary in amplitude, the final data is much easier to compare. Note that the pre-processed curves are not perfectly aligned. This is expected, since the profiles differ depending on which part of the face is spanned by the finger (to recall, each of the trajectory is initialized at a different location along the width of the face).

LEARNING ALGORITHM

Due to the absence of reliable position measurement on our robot's end-effector, recognizing the essential characteristics of the motion of the finger when moving across the face (as opposed to recognizing the exact 3D trajectory) is preferable. To account for this inherent variability in the way we acquire data, we choose to encode the distribution of our datapoints through a density-based representation. Such probabilistic encoding offers a flexibility that conventional data-driven techniques do not have. For instance, computing the norm of the distance between two trajectories would be offset by a temporal shift if they are not properly aligned. A Hidden Markov Model (Rabiner, 1989, HMM) offers a probabilistic encoding of a sequence of values, and is hence well suited to encode the dynamics of motion of the fingers. To distinguish across faces, we compare the likelihood of each face's model in a winner-take-all approach. One advantage of HMM is the fact that it allows to recognize motions even when solely part of

---

[2] G was set to 140 points in the current implementation.

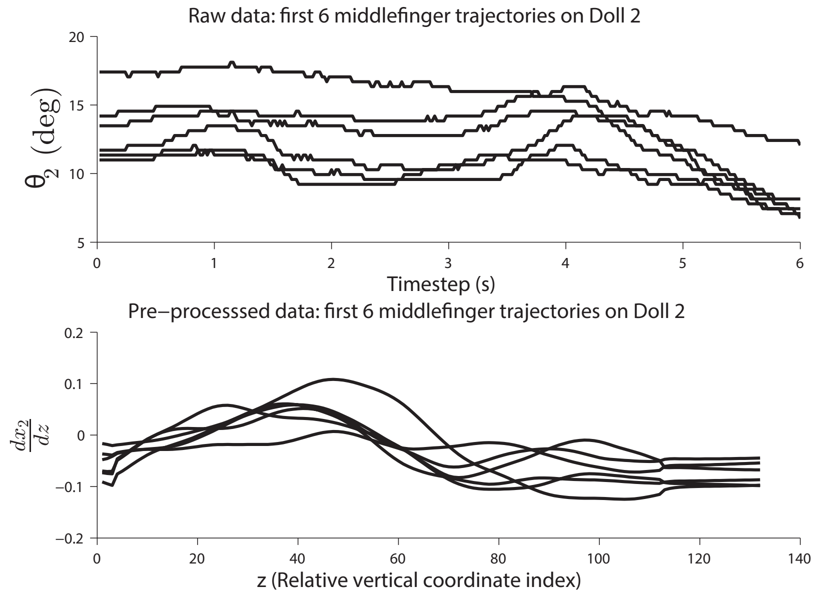


Figure 3.6: Comparison between raw and pre-processed data on the first 4 middle finger motions recorded on *Doll2*.

the motion is presented. This may prove very useful for face recognition, as it would allow to recognize faces even when the motion of the finger is initialized in a different location (e.g. in the middle of the face, as opposed to the top of the face) or when the fingers loose temporarily contact with the face as they swipe through the face.

Model description and learning

For each face r, a set of pre-processed data $\mathcal{D}_r$ is used to train a fully connected continuous Hidden Markov Model with 2-dimensional observations $\frac{d\tilde{x}_1}{dz}$ and $\frac{d\tilde{x}_2}{dz}$. The model takes as parameters the set $M = \{\pi, A, \mu, \Sigma\}$, representing, respectively, the initial states distribution, the states transition probabilities, the means of the output variables and the output covariance matrices. For each state, the output variables are described by K multivariate Gaussians:

$$p(x) \sim \sum_{k=1}^{K} \mathcal{N}(\mu^k, \Sigma^k) \tag{3.2.5}$$

The transition probabilities $p(q(t) = j|q(t-1) = i)$ and the observation distributions $p(x(t)|q(t) = i)$ are estimated by the *Baum-Welch* algorithm, an *Expectation-Maximization* algorithm that maximizes the likelihood that the training dataset can be generated by the corresponding model.

The HMM hyperparameters – number of states and number of Gaussians per state – are optimized through grid search with respect to the average classification performance on leave-one-out cross-validation (detailed in Section 3.2.3).

The HMM states are initialized through K-means and full covariance matrices are considered for the Gaussian distributions. The optimization resulted in 7-state models with 2 Gaussians per output (to ensure that the comparison of likelihood across the four face models is balanced, we fixed that all four HMMs had the same number of states). One HMM is thus defined by $n_S n_G \frac{dim_G(dim_G+1)}{2} = 7 \cdot 2 \cdot 3 = 42$ parameters, with n_S number of states, n_G number of Gaussians and dim_G the dimension of the Gaussians. Classification performance during testing is computed through a leave-one-out process: namely, each of the 10 trajectories for a given face model is tested against its corresponding HMM model (the latter being trained with the remainder 9 trajectories) and all the other 3 face models. This is repeated for each of the four face models. The cross-validation algorithm is detailed in Algorithm 1.

3.2.3 Results and discussion

We built 1 HMM for each of the four faces. Each model was trained using 10 examples of trajectories. We run the Forward-backward algorithm to determine the likelihood that any of the four models has generated the testing trajectory. A trajectory is said to be well classified if the likelihood of its associated model is larger than the likelihood of all other models. The testing is performed by leave-one-out cross-validation on the initial set of trajectories (10 for each of the 4 faces): each trajectory is a) compared to the fully trained models of the other faces and b) compared to a model of the same face built with the remaining 9 trajectories (the actual tested trajectory excluded from the model).

Since the construction of each HMM is not deterministic, training and classification are carried out ten times (also called here ten runs). In total, we built for each run 4 fully trained HMMs plus $4 \cdot 10$ partially trained HMMs for the testing phase detailed previously.

All trajectories describe a different section of the face since they are spread along the width of the face. We thus assume that the variation of the face's profile along its width is smooth enough so that new trajectories generated on other points of the face will follow a profile similar to those of the training trajectories and hence will be correctly classified by the HMM.

Performance in testing revealed very accurate results with an overall 91% recognition rate. 100% recognition rate is achieved for the *HOAP* face and 99% for the *Robota* face, while 77% and 88% recognition rate are obtained for the *Doll1* and *Doll2* faces. Figure 3.7 shows the median and quartiles of classification performance for each model across the ten runs. On average, the number of misclassified trajectories is 3.6 ± 2.7 out of 40 trajectories ($9\% \pm 7\%$ error rate). The best performance across the 10 runs is 2 misclassified faces (5% error rate).

These results are somewhat expected. The HOAP's face is not very humanlike and hence differs more dramatically from the three other heads. *Doll1* and

Algorithm 1: Leave-one-out cross-validation

```
for $run = 0$ to 10 do
  for $face \in \{Doll1, Doll2, Robota, HOAP\}$ do
    Build $HMM(face)$ using all $face$ trajectories.
    for $traj = 0$ to $N$ do
      for $faceToTest \in \{Doll1, Doll2, Robota, HOAP\}$ do
        if $faceToTest \neq face$ then
          Compute likelihood of $HMM(face)$ for $traj$.
        else
          Build model $HMM(face)_{\{\setminus traj\}}$ with trajectories $n \in \{1..N \setminus traj\}$
          and compute likelihood of this model for $traj$.
        end if
      end for
      Trajectory $traj$ is correctly classified if the likelihood of the true face is
      the highest.
    end for
  end for
end for
```

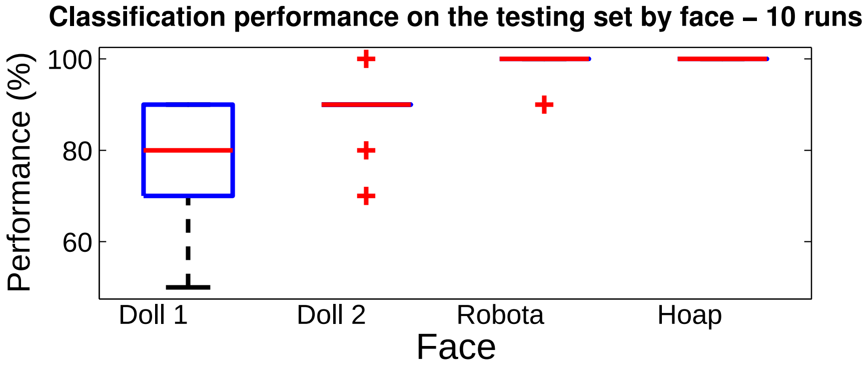


Figure 3.7: Boxplot representation of classification performance by face. (Median: red line, quartiles: blue lines, outliers: red crosses).

Doll2, while differing in some of their facial features are very similar in size, making it more difficult to discriminate across the two, especially when the fingers span the outer edges of the faces. As mentioned previously, the face of the Robota robot differs from the other dolls' faces mostly by its being wider and longer. Therefore, $\dot{\tilde{x}}$ (the profile slope) varies at a different rate when the fingers slide over *Doll1*'s face than when it does so over Robota's face. Here we see how our data encoding manages to encapsulate this relative difference in the temporal sequencing of finger motion, while remaining robust to absolute variation in the time it takes to span the face.

Looking more closely at the results, we find that one of the 40 trajectories is always misclassified (1st *Doll2* trajectory, classified as *Robota*) and another one is misclassified in 8 out of 10 runs (6th *Doll1* trajectory, also classified as *Robota*). The first one is a trajectory describing the side of *Doll2*'s face, therefore it is more likely to display few identifiable features, whereas the second one describes

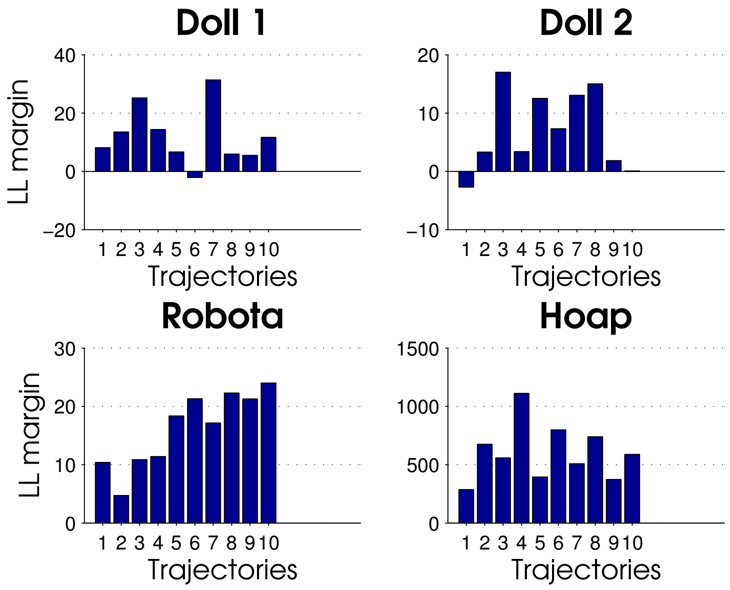


Figure 3.8: Margin of log-likelihood[3] for the first run: positive values correspond to correct classification.

the nose of *Doll1*'s face, which is narrow and might not have been described in the training set. This can be seen on Figure 3.8. In general, there may be several explanations to mis-classifications: a) the data of the corresponding trajectory is not reliable. This may happen, for instance if there is a failure in the tactile pressure feedback that leads to a finger leaving the face during the motion; b) a section of the robot's face is similar to a section from another face: each trajectory covers only a fraction of the face even if two fingers are used simultaneously to increase the specificity of one face's *signature*; c) these trajectories correspond to sections of the face that are very different from the rest of the face yet the model is not trained with this part of the face.

Aside from the binary classification result, it is important to estimate the confidence of the classification. Figure 3.8 and Table 3.1 give an indication on the margin of log-likelihood[3] between the true face and the face with the other highest log-likelihood for each trajectory: while *Doll1*, *Doll2* and *Robota* trajectories have a margin around 10, *HOAP*'s trajectories have a log-likelihood margin average of 564. As discussed previously, the *HOAP*'s face is very different from the other three and such can be identified with high confidence. This information could be used for instance to command the robot to perform a new measure of a face if the margin of log-likelihood, a measure of confidence in the model's prediction, is below a threshold.

Figure 3.9 shows the slopes measured by the index on the 4 faces; only

---

[3] The margin of log-likelihood of a trajectory is here defined as the difference between the log-likelihood of its associated model and the other best log-likelihood (i.e. the best if the classification is failed or the second best otherwise). The margin is positive if the classification is correct.

Face	Margin of log-likelihood
Doll1	8.77 ± 1.59
Doll2	6.23 ± 0.82
Robota	14.25 ± 1.19
HOAP	564.16 ± 40.16

Table 3.1: Average margin of log-likelihood per face over ten runs and ten trajectories.

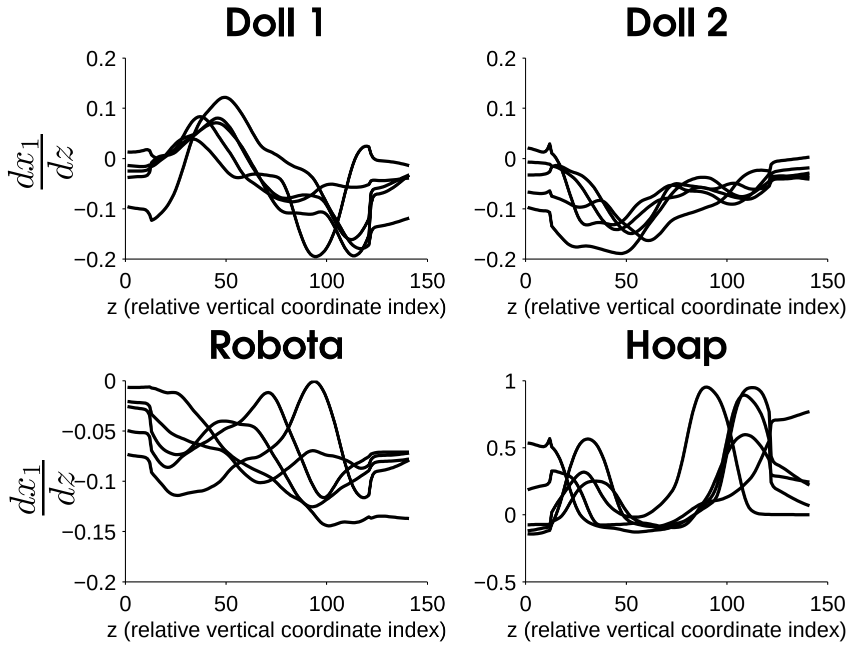


Figure 3.9: Comparison of the slopes from the first 5 trajectories on each face, index finger only.

the measures from the first 5 motions are displayed for clarity. As expected, the curves are not perfectly aligned. This results from both the noise in the experiments and the changes of profile along the width of one face.

Additionally, we tested our face exploration algorithm on real human faces, see Figure 3.10.

3.2.4 Conclusion

In this section, we presented an experiment in which faces are classified through proprioceptive information. Although the classification is not perfect, the algorithm gives good performance at discriminating across 4 very similar faces. The algorithm was shown to work flawlessly for the two faces that were most distinguishable. However, we can think of several ways to improve the classification performance.

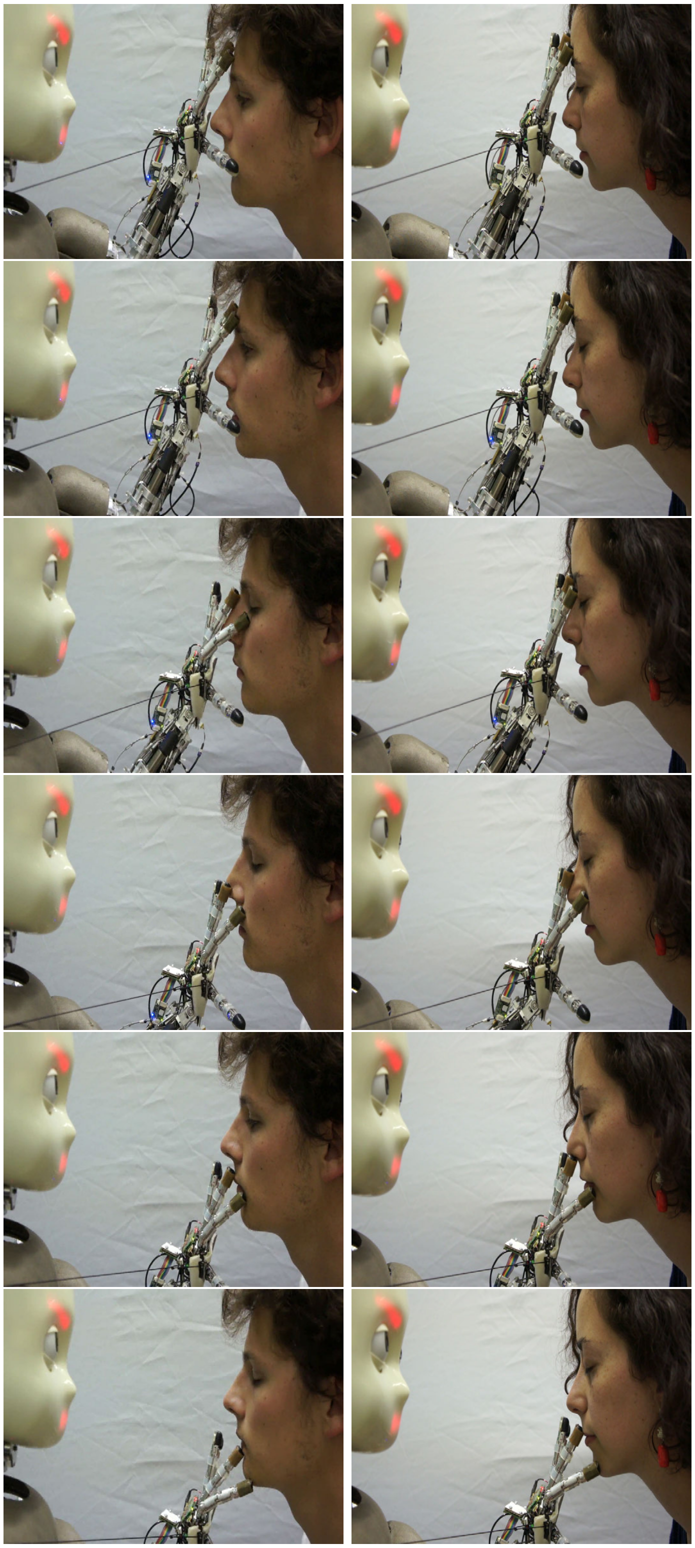


Figure 3.10: Face exploration of two real human faces. The string visible on the pictures pulls the hand backwards to discard the effects of joint slack in the wrist, which is causing wrong kinematic readings and perturbates the identification process.

As discussed in the result section, training 10 times a HMM may result in 10 different solutions (e.g. across ten runs performance varied from 95% recognition rate to 85%). This is due to the fact that the initialization of the HMM parameters is stochastic and the optimization leads only to local optimal solutions. To be less sensitive to the choice of initial conditions, one could perform crossvalidation on the choice of HMM during training (by training 10 HMM for each class and picking the one that yields best results). We did not do this in these experiments as the results overall were very satisfactory, but this may be required as one increases the number of faces to classify (as would be necessary if pursuing these experiments). Besides, HMM is not the only algorithm available to classify time-series, echo-state networks (Jaeger, 2001) usually give very good results in a large range of applications and could be used here to compare their performance with HMMs'.

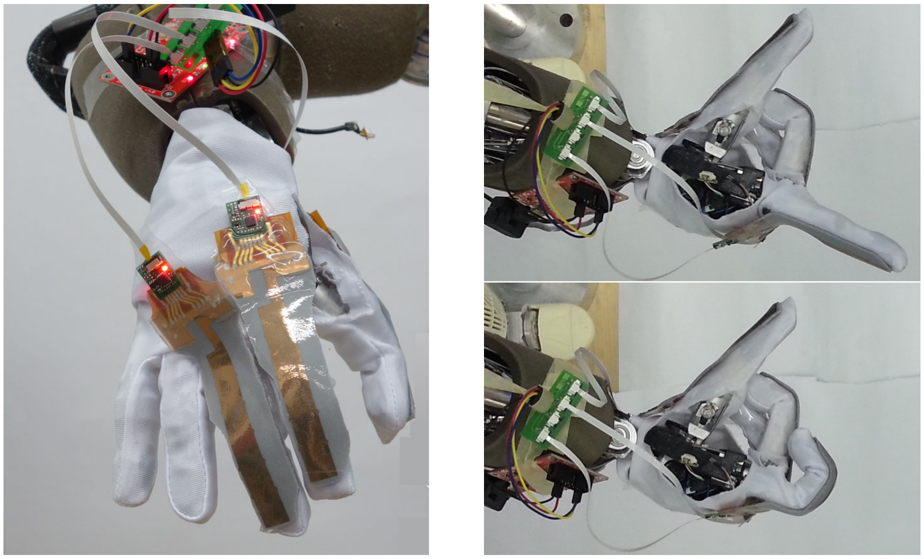
In the approach presented in this section, we cannot recognize which part of the face is touched. One could train one HMM per section of the faces and compare new data to each model, thus classifying the face and the part of the face being touched. A further drawback is the necessity to scan the face vertically from top to bottom, however, we can imagine that our method is robust to minor changes in the head orientation. In order to obtain true robustness to the changes in orientation of the motion or the face, one would require a different approach based on modeling the face and fitting new data with this model. This approach would also enable a more complex exploration strategy, i.e. choosing the direction of exploration or detecting the face's edges.

In Section 3.4, we tackle the problem of extending this approach to classify across objects. We take inspiration in the work by Meier et al. (2011): the idea is to fully model the object to identify with a 3-D point cloud. Because one of the drawbacks of exploration with one hand is the limited workspace relatively to the explored shape (in this case, it would be hard to even reach the side of the explored faces with iCub's hand), we use both hands of the robot to both *hold* the object and to *explore* it.

In the next section, we describe additional experiments using a prototype of stretchable tactile sensors, which can be placed between links of the robot, such as at the finger's knuckles.

3.3 Experiments with stretchable tactile sensors

The increasing demand for tactile sensing in robotics has led to robots almost entirely covered with artificial skin (Anghinolfi et al., 2013; Maiolino et al., 2013). However, tactile sensing technology is usually based on rigid materials, which do not allow to place them in areas of intricate motion, such as joints. In collaboration with another laboratory developing stretchable tactile sensors,



(a) (b)

Figure 3.11: a) The hand of the iCub humanoid robot with tactile sensors mounted on the back of its fingers. b) The sensors stretch and bend with the fingers.

we tested a prototype of these sensors mounted on the dorsal side of a robotic finger, in tactile exploration scenarios. We used the same tactile-based control as presented in the previous section.

The details of the manufacture and characterization of the sensors are not presented here. Two experiments, obstacle detection and contour following, are described in the next section.

3.3.1 Robot integration

In order to demonstrate the sensor's efficacy in a more generalized environment, compared to the laboratory characterization's setting, sensors are mounted on a stretchable textile glove and fitted onto the hand of the iCub humanoid robot. The sensors are manufactured in sets of six 9 mm x 5 mm nodes distributed along the length of the finger. The sensor acquisition rate during these tests was approximately 20 Hz, though this can be increased in future work by improving the serial communication. The iCub is used for two different applications of the sensors described above. In both experiments, we are using the tactile skin to detect contact on the back of the fingers and to provide compliance in the finger motion. In the first experiment, the sensors are used to detect contact with an obstacle during the arm's motion. In the second experiment, the fingers make use of the sensor pressure information to compliantly explore haptic features.

SETUP

The 7 arm joints of the iCub Humanoid robot are used to achieve the motion of the hand while one joint per finger is used to follow the surface in the second experiment (index, middle and thumb fingers can be used for this experiment). The tactile sensors are mounted on the back of the fingers: each finger is equipped with 6 tactile patches uniformly distributed from the first phalanx until the fingertip (see Figure 3.11).

Each finger has 3 degrees of freedom, controlled by two actuators: the second and third phalanx are controlled by one actuator and coupled together. However, only the actuator controlling the first joint can apply a force in the direction of the opening of the finger, the other actuator can only bend the finger, not bring it back. Similarly, springs bring back the joints to a straight position when the tendon for bending is released. This constrains us to use only the first actuator of each finger to apply a pressure on the outside of the finger. For this experiment, we also tie the second phalanx to the first one in order to rigidify the finger.

PROCEDURE FOR EXPERIMENT A

The goal of this experiment is to demonstrate the use of tactile sensors on the dorsal part of a robot for obstacle detection. The procedure is simple: the robot hand moves towards a flat surface (the obstacle) in a constant velocity Cartesian motion with the back of the fingers facing the obstacle. When contact is detected (the sensor value is above a threshold) with either finger (index or middle finger in that case), the motion stops to prevent collision and the hand is pulled back. We performed the obstacle detection experiment 20 times (see Figure 3.12), with the contact occurring either on the proximal or the distal knuckle: between the 1^{st} and 2^{nd}, or 2^{nd} and 3^{rd} joints. The knuckles can be seen on the index in the bottom of Figure 3.11(b). The capacitance (proportional to pressure) of the sensor during the experiment is displayed in Figure 3.14.

PROCEDURE FOR EXPERIMENT B

The experiment proceeds as follows: the robot positions its hand with its back towards a flat surface and while the hand is moving parallel to the plane, the fingers follow the contour of the surface, controlled in a pressure loop with the tactile sensors. The hand motion is a fixed linear Cartesian motion with constant velocity, while the fingers are controlled in current in order to maintain a desired tactile response.

A PD controller is implemented to follow a constant target pressure $\hat{s}_f$. This pressure is manually tuned so as to keep the fingers in contact without applying too much force on the object and finger tendons. Each finger f is controlled in current using the same controller as in Equation (3.2.2).

The two features can be seen on Figures 3.13 and 3.15(a): the arm and hand move parallel to the plane and the index follows the contours of the surface, including the features.

3.3.2 RESULTS

A video of the experiments can be found here: `https://www.youtube.com/watch?v=z512r3fDgX8`.

EXPERIMENT A

The experiment is successful if the robot detects the contact and stops; it fails if the hand tries to force into the obstacle and must be stopped manually.

The experiment succeeded 20 out of 20 times for the second knuckle, only 17 out of 20 for the proximal knuckle (see Table 3.2). The reason for the 3 failures is the lack of precision on the orientation of the hand: contact occurs on a part of the hand that is not covered with tactile skin and thus cannot be detected. This stresses the need for a tactile skin that covers all of the robot's surface. A noticeable delay of the robot reaction in the included video is a result of the robot control, as opposed to an insensitivity of the sensor.

Location of contact	**# trials**	**# success**
Proximal knuckle	20	17
Distal knuckle	20	20

Table 3.2: Exp a: Results of the obstacle detection.

EXPERIMENT B

Snapshots of the experiment can be seen on Figure 3.16. The two features are clearly extracted by the movement of the fingers thanks to the tactile sensors, as can be seen on Figure 3.15. The lack of a perfect straight line between the two features can be attributed to the imprecision in the proprioceptive measurements of the robot that are used in the forward kinematics to compute the position of contact. Also, the precision of the reconstructed feature is limited by the length of the sensor ($1cm$), which is the reason for the larger reconstructed feature around $200mm$ on Figure 3.15(b) compared to the true feature. The evolution of the sensed pressure can be seen on Figure 3.15(c).

3.3.3 CONCLUSION

In this section, we presented additional experiments using a prototype of an artificial skin mounted on the dorsal side of iCub's fingers. We showed another example where tactile sensors can be used to control robot fingers for surface following and shape reconstruction, even when the sensors are placed on areas of

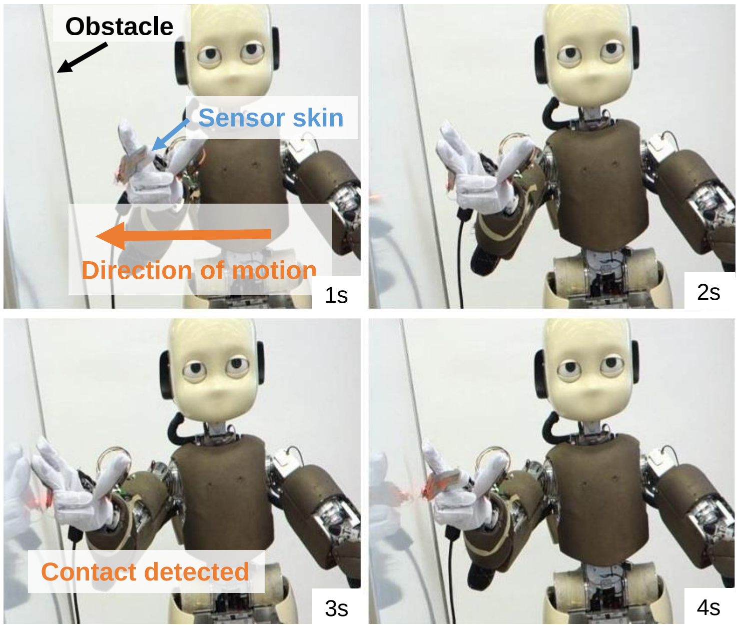


Figure 3.12: Exp a: Snapshots from the obstacle detection task (distal knuckle). The hand moves towards the obstacle until contact is detected by the tactile sensors on the back of the fingers (here, second knuckle), then it withdraws.

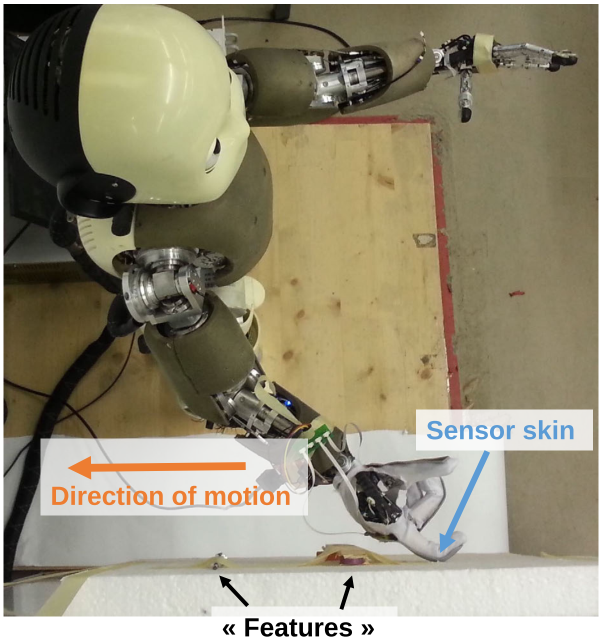


Figure 3.13: Exp b: the index follows the contour of the features on the surface. Two "features" are present: a small and a bigger bump.

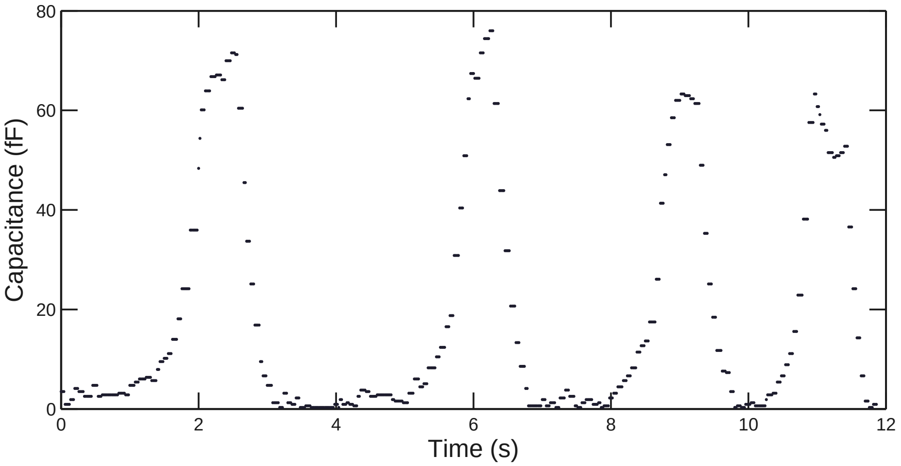
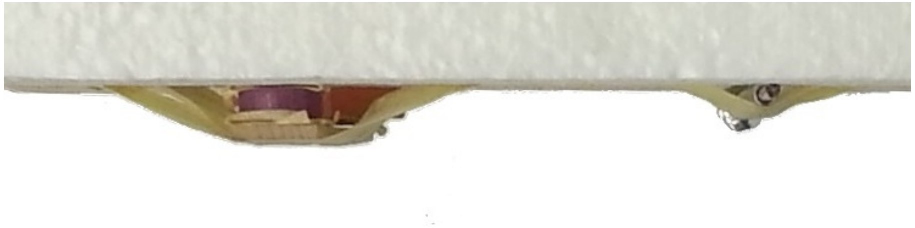
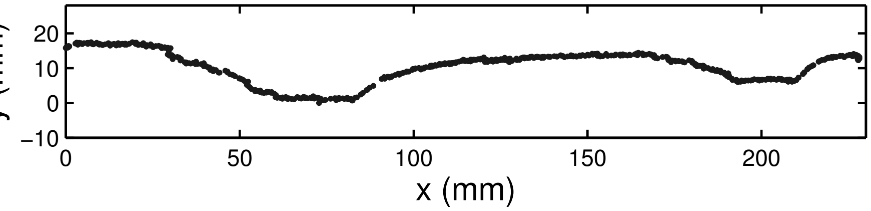


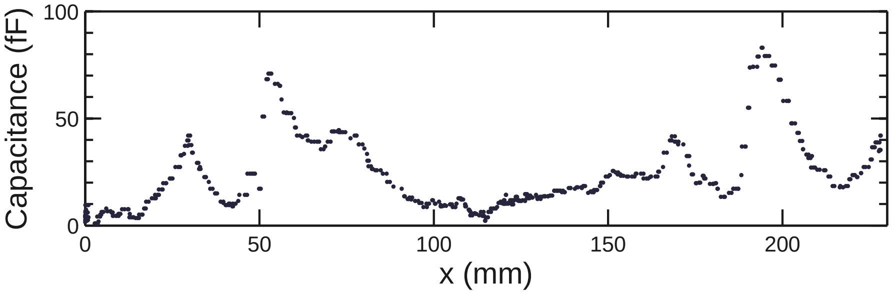
Figure 3.14: Exp a: The evolution of the capacitance value from the sensor that enters in contact during the obstacle detection experiment. The experiment is run 4 times in a row.



(a) Picture of the features

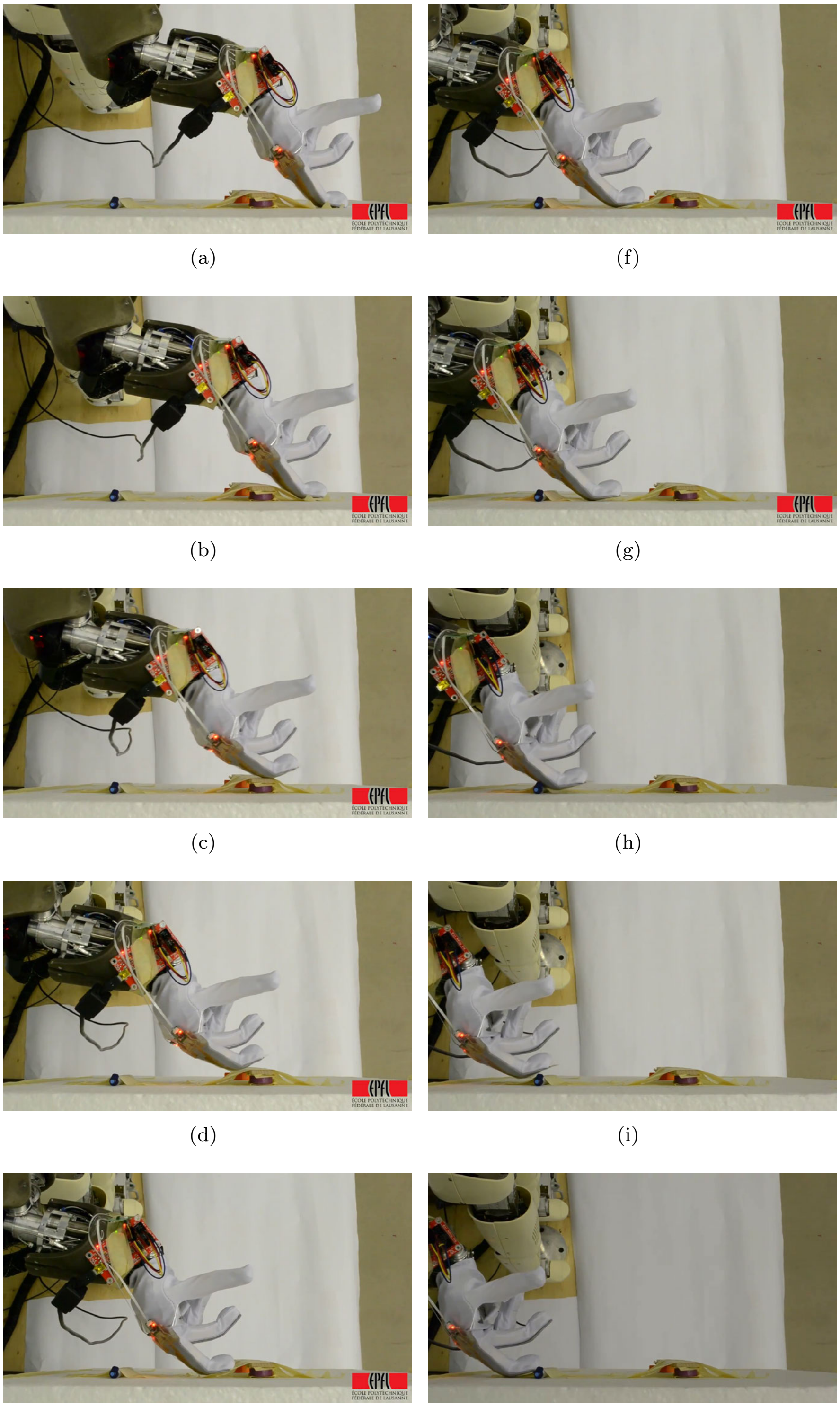


(b) Reconstructed features



(c) Response of the sensor in contact during the feature following task

Figure 3.15: Exp b: a) A top-down picture of the two features. b) The reconstruction of the surface and features from 578 tactile data points collected with the artificial skin. c) The sensor's response during the scanning: the response increases when the finger enters in contact with the feature, and decreases when the finger releases the applied force.



(a) (f)

(b) (g)

(c) (h)

(d) (i)

(e) (j)

Figure 3.16: **Exp b:** scan of the features with the back of the hand. Pictures from top to bottom, left to right.

the robot that are usually not suited to hosting sensors, such as on the knuckles at the back of the fingers.

3.4 Bimanual compliant tactile exploration

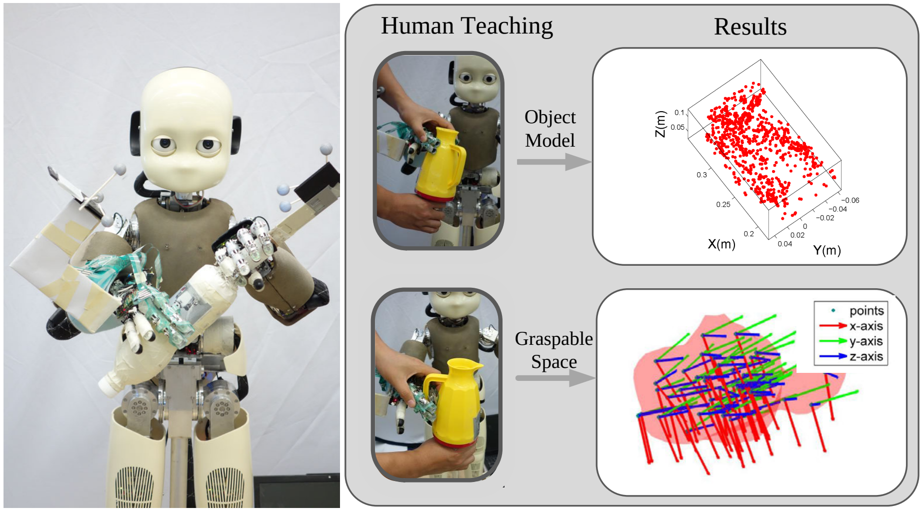
Humans have an incredible capacity to learn properties of objects by pure tactile exploration with their hands. Tactile exploration is crucial during manipulation, especially when handling objects with two hands. In this case, the objects are often obstructed from view by either or both arms and fingers and one can rely only on tactile information to direct the manipulation. In this section, we present a method whereby the two arms move in coordination so as to maximize the surface on the object that can be explored. This is used in conjunction with an algorithm for object recognition. An exploration strategy is proposed to guide the motion of the two arms and fingers along the object. Experiments on an iCub humanoid robot validate our approach.

In this section, we present a strategy for bimanual compliant tactile exploration of unknown objects. The object is held by one hand while the other hand is exploring it, see Figure 3.17. The bimanual coordination strategy consists in moving the hand holding the object so as to bring the interesting region on the object into the reachable space of the other hand so as to make it easier for the other hand to either explore or grab the object.

The rest of the section is organized as follows: in the next subsection, we introduce our bimanual control framework and the local finger exploration strategy. In Section 3.4.2, the method for object identification is presented. In Section 3.4.3, we present the experimental setup with our humanoid robot iCub. Further, we present our experimental results and discussion in Section 3.4.4.

3.4.1 Exploration strategy

Our objective involves the identification of objects through tactile exploration. However, the workspace of humanoid robots is usually limited. Most daily objects are too large and cannot be explored by a single arm and hand. In order for the robot to gather enough information on the object's shape to allow unambiguous identification, the hand needs to explore a large portion of the object. To this end, we must extend the reachability of the exploring hand relatively to the object. In order to achieve this, we use one hand of the robot to hold an object, while the other hand explores it. This allows to approach and touch the object from different angles and with higher dexterity relatively to the workspace of both arms.



(a) (b)

Figure 3.17: **Left:** The iCub humanoid robot is exploring an object with its two hands. **Right:** The robot is taught the shape of the object by a human teacher passively guiding the robot's hand along the object, emphasizing the object's part that can be grasped, e.g. the handle on the jar. The object model is stored as a point cloud and the graspable part is modeled using GMM (Gaussian Mixture Model).

Bimanual coordination

Let $\mathcal{T}_R^0$ and $\mathcal{T}_L^0$ be the homogeneous transformations from the robot root frame to the frames R and L attached respectively to the right and left arm's "interest points". In the rest of the section, we will refer to the "interest points" to denote a) the center of the palm on the exploring hand and b) the point to be reached on the object held by the other hand.

The goal is to have both frames coincide:

$$\mathcal{T}_R^0 = \mathcal{T}_L^0 \Leftrightarrow \mathcal{T}_R^L = \mathbb{I}$$

in which $\mathbb{I}$ is the identity matrix (see Figure 3.18).

Motion generation

We generate a kinematic constraint from the above static constraint in order for the system to converge to this state. We give the following translational and rotational velocities in Cartesian space until both frames coincide:

$$\mathbf{v_R} = \frac{\mathbf{t_L^R}}{\parallel \mathbf{t_L^R} \parallel} \cdot f(\parallel \mathbf{t_L^R} \parallel) \tag{3.4.1}$$

and $\mathbf{v_L} = -\mathbf{v_R}$. With $\mathbf{v_x}$ being the translation velocity vector of the frame x, expressed in the robot root frame, $\mathbf{t_L^R}$ the translation vector from R to L, and f a function from $\mathbb{R}^+$ to $\mathbb{R}^+$ designed to give a smooth and converging motion.

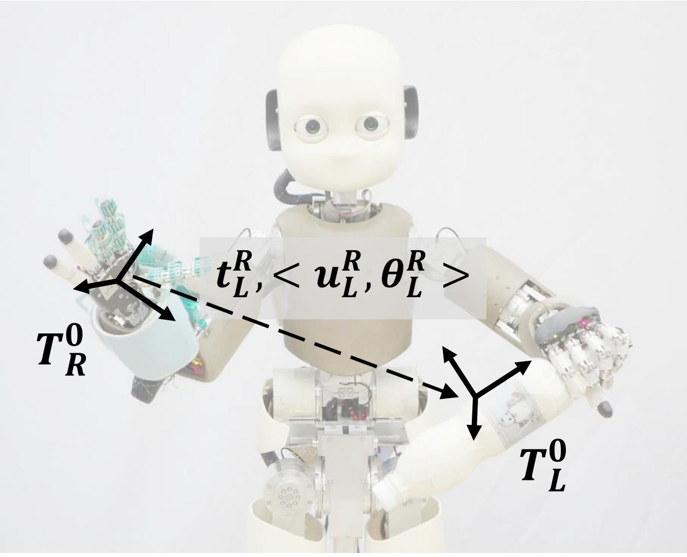


Figure 3.18: Schematic of the bimanual constraint. The frames $\mathcal{T}_0^L$ and $\mathcal{T}_0^R$ of the interest points should coincide to satisfy the bimanual constraint. The frame $\mathcal{T}_0^L$ changes depending on where the object should be scanned and the estimation of the object's diameter.

For the rotation, a similar constraint is expressed in the axis-angle notation which defines a rotation with an axis $\mathbf{u}$ and a rotation angle θ around this axis. Given the axis-angle rotation $< \mathbf{u_L^R}, \theta_L^R >$ equivalent to the usual rotation matrix notation R_L^R, the rotational velocity is defined in axis-angle notation: $\mathbf{w_R} =< \mathbf{u_L^R}, \omega >$ and $\mathbf{w_L} =< \mathbf{u_L^R}, -\omega >$ with $\omega = f(\| \ \theta_L^R \ \|)$. We chose to use the function $f(x) = a \cdot \exp(-\frac{w}{x})$, with parameters a and w determining respectively the velocity of the motion far from the target point and a measure of the closeness to the target.

Working at the limit of the workspace

Because the workspace of the robot's arms are often limited during bimanual manipulation (e.g. the hands of the iCub robot can barely reach each other), it is important to take into account the non-feasibility of a given inverse kinematics problem. Our IK solver uses a pseudo-inverse of the Jacobian with optimization, this way we can weight some constraints so as to satisfy them in priority. In our application, the position constraints are more important than the orientation constraints, and the orientation of the normal of the palm (the right hand's interest point) is more important than the orientation of the other axes of the interest point's frame. Therefore, we express the orientation Jacobian in the interest point's frame and the desired axes are weighted as indicated previously. These weights are only taken into account when the IK problem has no solution and a compromise between the constraints has to be found, therefore their choice is not very sensitive and they are set empirically.

Collision avoidance

During the scanning of objects, the goal is to keep one hand and fingers in contact with the scanned object at all times. However, when both arms are changing configuration to start a new “scan” from a different angle, there is a need for collision avoidance in order for the scanning hand not to hit the object while moving around it. Because we do not know the exact shape of the object held by one of the hands, we assume a cylinder with a sufficiently large diameter, and require the end-effector to move outside this cylinder during the motion, until the hand is aligned in front of the target reaching point, where it is allowed to enter the “collision area” (see Figure 3.19). The corrected velocities for collision avoidance are given in Table 3.3.

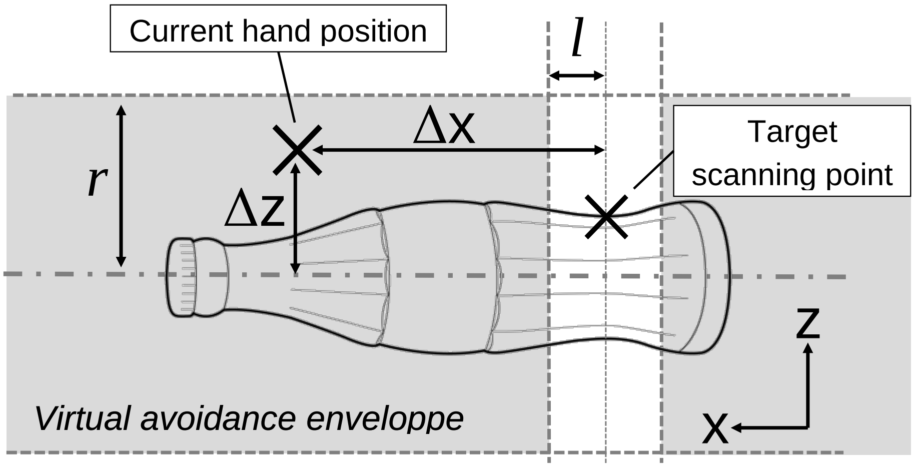


Figure 3.19: Scheme of an object and its collision avoidance virtual envelope. A lateral safety zone is delimited by the parameter l, in which the exploring hand can enter. The diameter of the virtual envelope is defined by r.

	$\lvert\Delta x\rvert \leq l$	$\lvert\Delta x\rvert > l$
$\lvert\Delta z\rvert > r$	$\mathbf{v}_\mathbf{L}^{\mathbf{R}\prime} = \mathbf{v}_\mathbf{L}^\mathbf{R}$	$v^{R\prime}_{L\,z} = v^{R}_{L\,z} \cdot \exp(\frac{-w_{out}}{\lvert\Delta z - r\rvert})$
$\lvert\Delta z\rvert \leq r$		$v^{R\prime}_{L\,z} = -v_{av} \cdot \exp(\frac{-w_{in}}{\lvert\Delta z - r\rvert})$

Table 3.3: Velocity correction for collision avoidance, with v_{av} a predefined avoidance speed, w_{out} and w_{in} parameters that regulate the transition from collision avoidance to the normal behaviour of reaching the starting scanning point, $\mathbf{v}_\mathbf{L}^{\mathbf{R}\prime}$ the modified relative velocity between the right and left frames to avoid collision, and $v^{R}_{L\,z}$ the z component of the velocity, expressed in a coordinate system rotating around the object. Z is constrained to be normal to the principal axis of the object (i.e. the scanning direction x, see Figure 3.19) and oriented around the x axis to point from the center of the object towards the other hand.

Compliant tactile control

During the exploration of the object, the tactile sensors provide contact information. In order to obtain this information, the fingers must apply enough pressure on the object. The tactile response is thus used in a pressure loop designed to apply sufficient force and obtain contact data, while not pressing too hard so as not to damage the object being touched (see Figure 3.20(a)). For n_s tactile sensor patches and n_a actuators, the motors are commanded in current with $\mathbf{u} \in \mathbb{R}^{n_a}$ following:

$$\mathbf{u} = \kappa \cdot \Phi(S, S^*) \tag{3.4.2}$$

with $S, S^* \in \mathbb{R}^{n_s}$ respectively the current and desired tactile response, $\kappa \in \mathbb{R}^{n_a}$ a vector of proportional gains for each actuator, $\Phi : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{n_a}$ a mapping between the tactile sensor patches and corresponding motors. The mapping Φ depends on the architecture of the robot hand – the number of actuated joints, the number and disposition of the sensors – and the desired behaviour

In our implementation, each finger is controlled the same way, $n_s = 3$ for each finger as we take directly the average value for each tactile patch as inputs (one per phalanx, each composed of 12 or 16 taxels), and $n_a = 2$: the first actuator of the finger controls the first phalanx and the second actuator controls the second and third phalanx coupled together. For each finger, Φ is defined as follows:

$$\Phi(s) = \{min(e_0, e_1), e_2\} \tag{3.4.3}$$

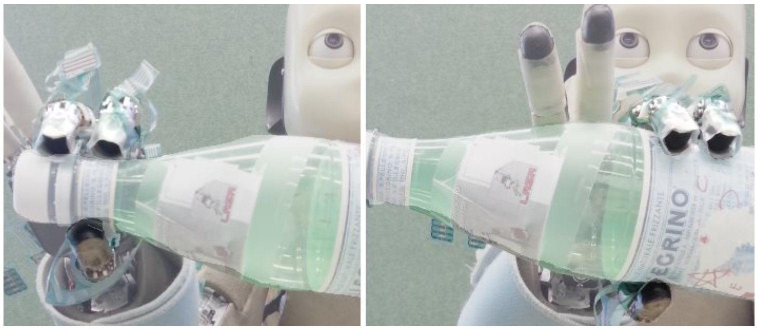
With s the average tactile response for each of the three phalanx (s_0, s_1 and s_2 are the average pressures on respectively the first, second and last phalanx and $e_i = s_i^* - s_i$, with s_i^* the corresponding desired pressures). This mapping allows to make contact on the three phalanxes, using only the two actuators, by using the passive compliance emerging from the coupling between the last two phalanxes.

Thumb motion

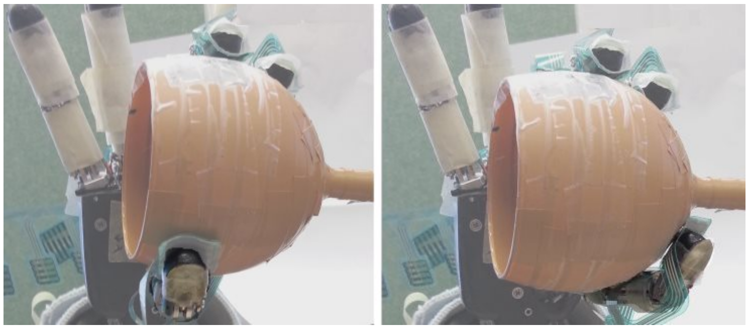
On anthropomorphic robotic hands, the thumb is usually equipped with an additional degree of freedom which enables it to control its opposition to the other fingers. During the scanning of objects, we use this DoF to increase the amount of the object's surface explored by the thumb, especially for reaching areas otherwise difficult to access (see Figure 3.20(b)). A periodic swiping motion is implemented and efficient enough to gather data more efficiently.

Detect loss of contact with the object

While scanning, the fingers might slide off the object (for instance when reaching an extremity). In that case, they might touch each other and record the contact



(a) Finger's compliance during scanning



(b) Thumb opposition

Figure 3.20: **Top:** The fingers adapt to the size of the object in order to follow compliantly the surface. **Bottom:** Illustration of the advantage of changing the thumb's opposition while scanning a glass: the thumb follows the high curvature of the surface.

as if they were touching the object. When this happens, the distance between the contacts points on the two fingers is close to 0 and this allows us to detect these events and to discard these contact points. This is also used to detect that the exploration has reached the end of the object and decide that the object can be scanned from another orientation.

Approaching the object

When the exploring hand comes in contact with the object, we need to detect precisely when the hand touches the object. Tactile sensors seem a good way to detect this contact. However, they should be extremely sensitive and detect very light pressure. Otherwise, when the exploring hand comes into contact with the object for the first time, it may apply too much force on the object – a small force on the object creates a high torque on the hand holding the object. While exploring, we overcome this problem by "pinching" the object so as to apply forces on both sides of the object. Since our tactile sensors are not sensitive enough, we use force-torque sensors embedded in the robot's arm to detect when the hand touches the object. We use a first order band-pass filter to remove both the low-frequency component of the signal due to the errors of estimation of the robot's limb's own weight and smooth the high-frequency component since the signal is very noisy.

3.4.2 OBJECT IDENTIFICATION

In order to identify an object, the data collected from tactile exploration is first filtered and smoothed using a GP-based filter. The data can then be aligned with previously known object models and the average distance after alignment is used as criterion for identification. After the identification, a grasp is computed from previously learned grasps. As the data acquired from tactile exploration is noisy and un-uniformly distributed, which is not ideally suitable for object identification, a GP-based filter was implemented by my colleague Miao Li to smooth the data. The data used for further identification is thus the filtered data.

For each of the objects to explore, we assume that there is already a point cloud model for it, which can be obtained either from a vision scanner or from human demonstrations. In Section 3.4.3, we give details on how we collect this point cloud model from human demonstrations. Herein, for the i-th object, the point cloud model is denoted as $\mathcal{O}^i = \{\mathbf{p}^{i,j}\}^{j=1..n_p}$. The object's identification algorithm tries to align the datapoints collected so far with the available object point clouds $\mathcal{O}^i$ and the one with the smallest alignment error is identified as the corresponding object. To this end, after each scanning, the points gathered so far $\mathcal{X} = \mathbf{x}^j, j = 1...n_x$ are transformed into the most similar pose using the *iterative closest point* (ICP) algorithm.

As described in Besl and McKay (1992), ICP can compute the optimal transformation $(R, \mathbf{q}_t)$ between two corresponding datasets that minimizes the following distance error:

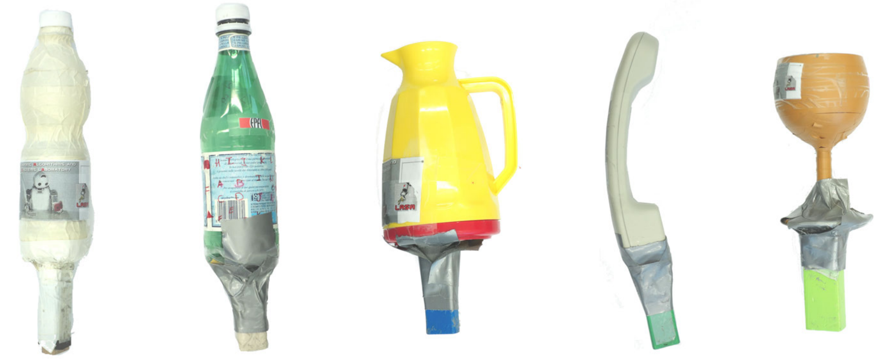
$$Dist(\mathcal{X}, \mathcal{O}^i) = \frac{1}{n_x} \sum_{j=1}^{n_x} \|\mathbf{p}^{i,j} - (R\mathbf{x}^j + \mathbf{q}_t)\|^2 \tag{3.4.4}$$

In our work, the correspondence between the measured points and the object point cloud models are chosen with the nearest neighbor match without replacement: the same point in the object point cloud model cannot be the correspondence point for two different measured points. In general, this method suffers from local minima. To counter this effect, we run 10 different comparisons with 10 different initializations of the initial points. These initial points are uniformly obtained from different rotations R around the object's principal axis and its normalized translation components are randomly sampled in $[-0.5, 0.5]$.

For each available object model, we compute the minimal distance after alignment, i.e. $Dist(\mathcal{X}, \mathcal{O}^i)$ and the object is identified as the object with the minimal distance.

3.4.3 EXPERIMENT

As in the previous experiments (see Section 3.2), the iCub humanoid robot (Figure 3.22(a)) is used to explore different everyday objects using both arms. We chose five objects: 2 bottles, 1 jar, 1 phone receiver and 1 glass, shown in Figure 3.21. The two bottles are very similar and can test the accuracy of the identification method. The phone's profile encompass sharp changes in curvature, a challenge for the compliant control of the fingers. Scanning the glass is even more challenging as it requires to control precisely for the thumb's motion in order to follow the edges. The jar has a much larger diameter and involves two particular features: the handle and the spout. During the exploration, one arm holds the object, while the other arm explores it with its fingers. The collected data are compared with data previously collected manually in order to identify the object. During the exploration, the robot attempts to identify the objects as well as their positions and orientations. Then, from the previously learned grasps, one grasp is selected and adopted by the free hand, on the object (see Algorithm 2).



(a) bottle 1 (b) bottle 2 (c) jar (d) phone (e) glass

Figure 3.21: Five different everyday objects are used in our experiment. Handles are mounted on the bottom of the objects in order to adapt to the size of iCub hand.

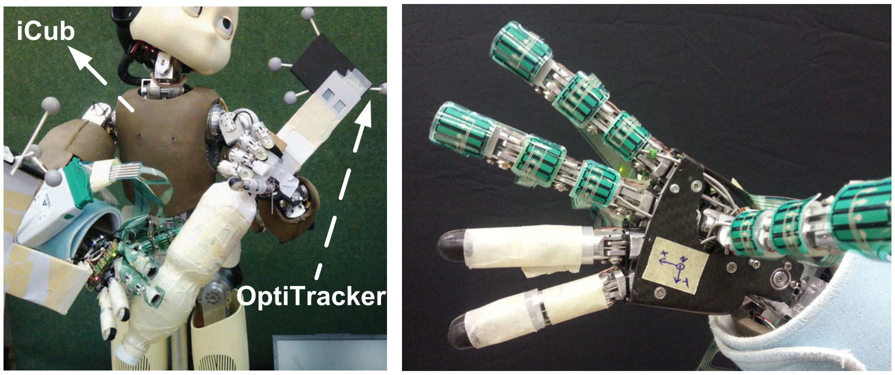
SETUP

We use both arms of iCub, each of which have 7 degrees of freedom (DoFs). Each hand has 9 DoFs, 3 for the thumb, 2 for the index finger, 2 for the middle finger, 1 for the coupled ring and little finger and 1 for the adduction/abduction. Only the thumb, index and middle finger are equipped with *Tekscan*[4] tactile sensors (see Figure 3.22(b)). The Tekscan sensors have a spatial resolution of $4mm$ ($6.2\ sensors/cm^2$), the fingers are equipped with $3*4$ taxels – tactile pixels – per phalanx, and $4*4$ taxels on their fingertip, which makes a total of 120 taxels on the hand. A motion capture system – *OptiTrack*[5] – is used to track

---

[4] http://www.tekscan.com/

[5] http://www.naturalpoint.com/optitrack/



(a) iCub and optical tracking markers (b) iCub's hand equipped with Tekscan tactile sensors

Figure 3.22: An iCub humanoid robot (**a**) is used in our experiment. The thumb, index and middle finger of the right hand are equipped with Tekscan sensors (**b**). During the experiment, the objects are firmly held by the left hand (no relative motion), while the right hand explores the object from different orientations.

the position and orientation of both hands to compensate for the inaccuracy of iCub's kinematics and obtain precise measurements. The contact positions are obtained through forward kinematics starting from the motion tracker, and given the geometry of the tactile sensors.

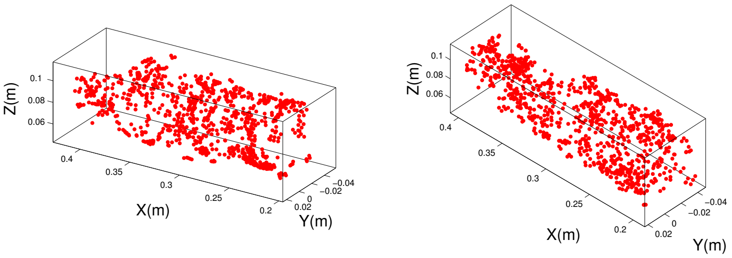
Manual data collection

Prior to the exploration, we manually collect data from the objects using the same setup with the difference that the object is held by a human demonstrator in place of the robot itself. An optical tracker is attached to the object while the fingers of iCub are pressed against the object to collect point cloud data all over the surface (see the top left image on Figure 3.17(b)). The acquired object point clouds are shown in Figure 3.23.

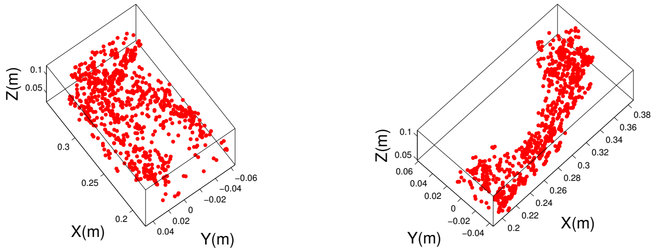
Exploratory procedure

For the exploration process, we only assume that the principal axis of the object is available, for instance through basic image processing. However, we know the precise position of the hand holding the object through our motion capture system – instead of using forward kinematics, imprecise because of slack in the joints. This system is also used to track the position of the right hand.

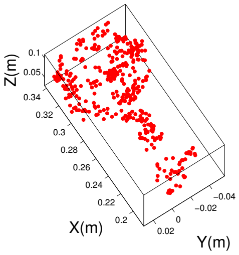
The right hand scans the object from one end to another along this principal axis, and changes the angle of approach iteratively around this principal axis at every scan. The procedure is described in Algorithm 2. During the whole exploration, both arms move simultaneously to achieve the desired relative position and orientation between the interest points (the palm and a point on the object's surface), therefore the indications in Algorithm 2 are given in relative



(a) bottle 1 (b) bottle 2



(c) jar (d) phone



(e) glass

Figure 3.23: The object point clouds obtained from human teaching.

terms between these two interest points.

Algorithm 2: Exploratory procedure

```
$\theta \leftarrow \theta_{min}$; \\ Angle of approach around the principal axis
while $\theta < \theta_{max}$ do
    Go above initial scanning point;
    while !contact do
        Move hand and object towards each other;
    Close fingers and activate finger compliant control;
    while fingers in contact with object do
        Slide the hand along the object's principal axis;
    Open fingers;
    Try to identify object;
    if object identified then
        Compute and reach a grasping posture;
        Grasp the object;
        return 1;
    $\theta \leftarrow \theta + increment$;
return 0;
```

3.4.4 RESULTS

Each object is scanned using Algorithm 2. The acquired point clouds are quite noisy and non-uniformly distributed. As mentioned above, the raw point clouds are not ideal for object identification, due to the difficulty in finding the correct corresponding points for the ICP algorithm. With the GP filter, the filtered point clouds become smoother, sparser and less noisy, as shown in Figure 3.27.

For each explored object, we chose 10 different initial configurations for the ICP algorithm, where the rotation R in Equation (3.4.4) is uniformly sampled around the principle axis of the object. The object is identified as the object with the smallest distance among the 10 different trials. The distance for each trial is shown in Figure 3.26 and the points after alignment are shown in Figure 3.27. We repeated the identification algorithm 10 times for each object and the success rate of identification is always above 90%. The failure happens when bottle 2 is misidentified as bottle 1 and the jar is misidentified as bottle 1. The statistics for the distances are shown in Figure 3.24.

After the object is identified, a grasp is chosen (this is work done by Miao Li) and the right hand moves to the selected grasp, as shown in Figure 3.28.

3.4.5 CONCLUSION

In this section, we presented a general approach for bimanual compliant tactile exploration, with applications to object identification, manipulation and grasping. The kinematic limitations of the system, i.e., workspace limitation and collisions, are considered in this exploration strategy, which is critical in tac-

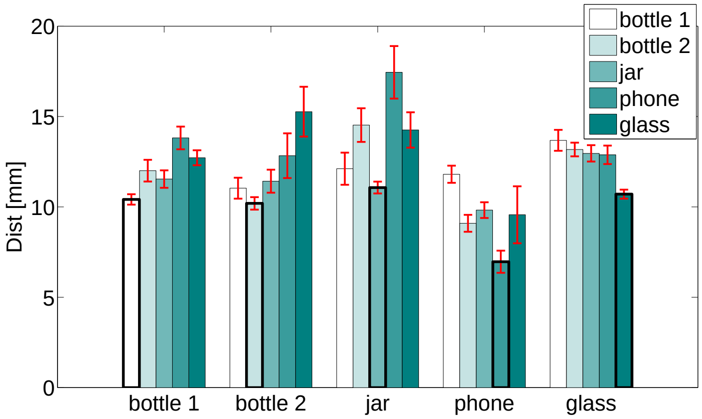


Figure 3.24: Comparison of the aligned distance from Equation (3.4.4) for all the objects. The object with the smallest distance is chosen as the identification result. For each object, we ran the identification algorithm 10 times.

tile exploration as suggested in Bierbaum et al. (2008). Because our method is close-loop, i.e. not based on planning, it can run very fast at runtime, and allow a fast exploration.

However, this approach is limited to objects of relatively small size that can be held by a robot, and the objects should have one principal axis along which to perform the exploration. In the next chapter, we tackle the exploration of completely unknown objects, trying to remove any possible constraint on the shape or size.

3.5 Conclusion

In this chapter, we presented multiple exploration strategies for different scenarios in which objects are scanned using touch, including for the first time bimanual tactile exploration on a robot. Given that those objects are partially or completely unknown, tactile-servoing methods are introduced to keep contact with the object while maintaining a constant pressure. We showed that we were able to differentiate between similar human-like faces from tactile-based trajectories, encoded as HMMs, as well as between similar objects using tactile point-clouds, both collected using tactile sensing as a primary control variable.

We also tested these methods using different tactile sensing technologies, including a prototype of stretchable tactile sensor that can be placed on an actuated part of the robot, in our case the back of the knuckles on the fingers.

In the next chapter, we tackle several additional problems, including the exploration of completely unknown objects, e.g. without any prior about them, and improving the compliance between hand-fingers and the explored objects, using tactile sensing and redundant kinematic chains.

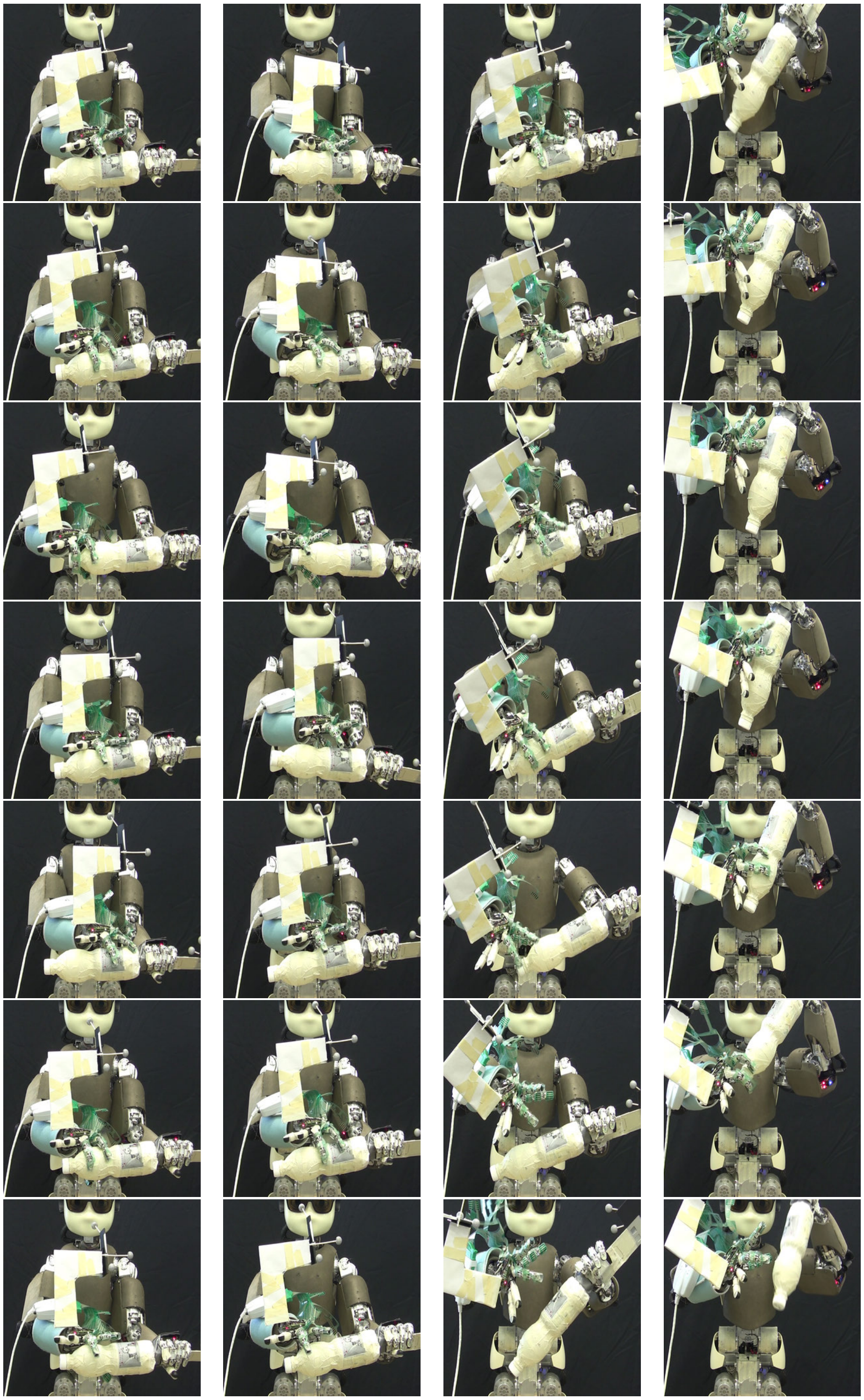
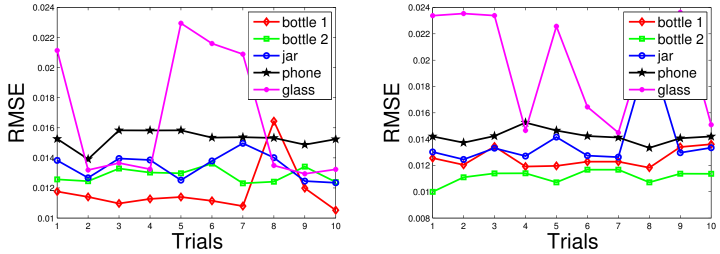
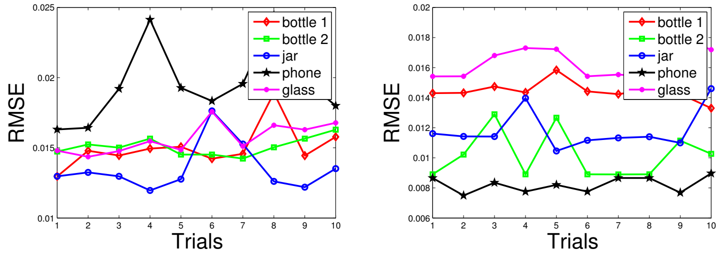


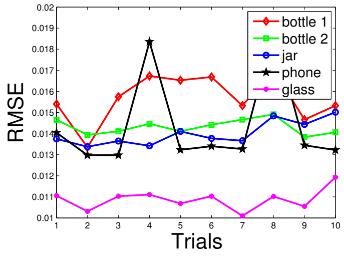
Figure 3.25: Exploration of bottle 1. Pictures from top to bottom, left to right. Total duration is 3 minutes. The robot explores the object from one end to the other, alternatively changing the relative orientation between the object and the hand that explores it.



(a) bottle 1 (b) bottle 2

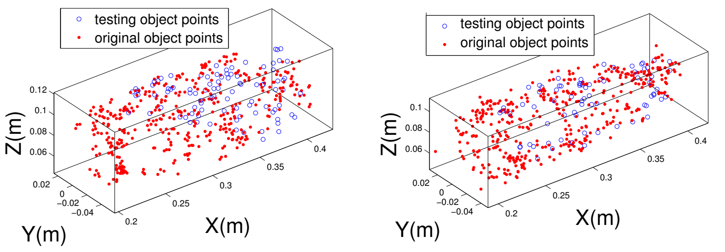


(c) jar (d) phone

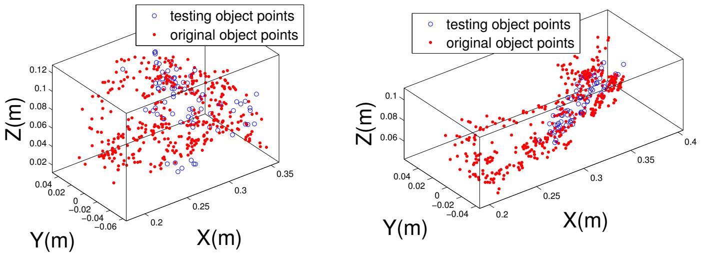


(e) glass

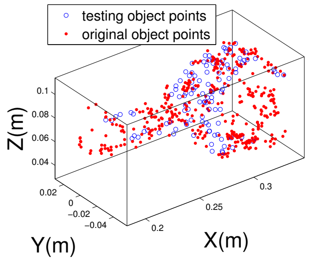
Figure 3.26: Object identification with sparse point cloud starting from 10 different initial configurations.



(a) bottle 1 (b) bottle 2

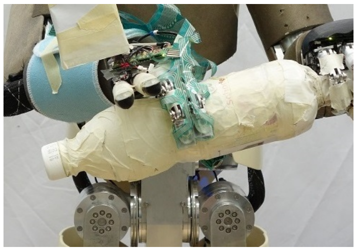


(c) jar (d) phone

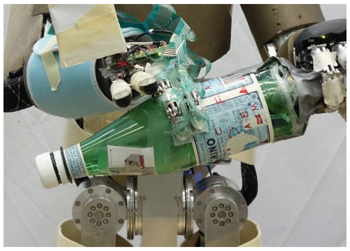


(e) glass

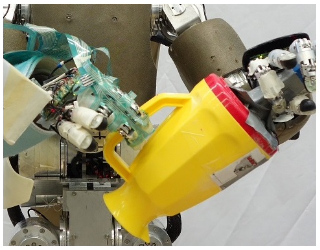
Figure 3.27: The filtered object point clouds aligned with the trained object points cloud. Only 400 datapoints from the trained object point clouds are displayed.



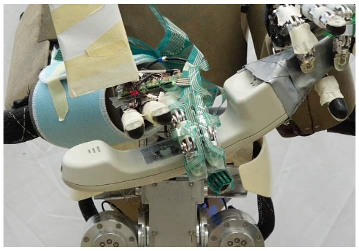
(a) bottle 1



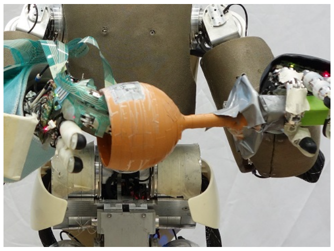
(b) bottle 2



(c) jar



(d) phone



(e) glass

Figure 3.28: The selected grasp (hand position and orientation) for each explored object after identification.

Chapter 4

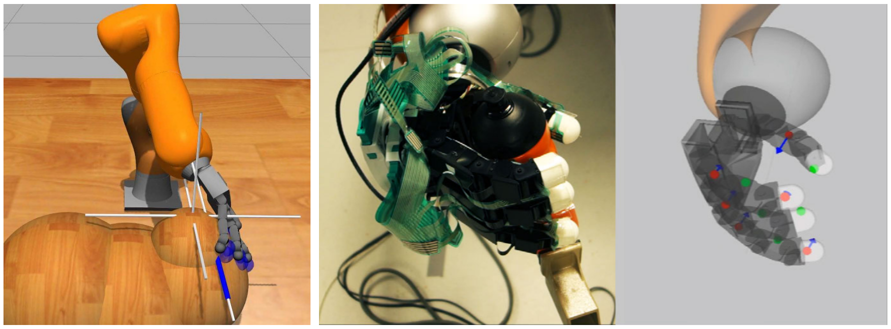
MULTIPLE TACTILE CONTACTS CONTROL FOR EXPLORATION AND GRASPING

4.1 Introduction

In robotics, collisions are ordinarily avoided and in the cases when contact is allowed, it is usually limited to a single contact point at the end-effector. However, recent progresses in tactile sensing offer a range of research directions in robotics for allowing robots to be in contact at multiple points on the body.

Most research on haptic exploration has focused on a single contact (Okamura and Cutkosky, 1999; Heidemann and Schopfer, 2004) on the end-effector or sequences of multi contact grasps (Meier et al., 2011). Much less work has been done on continuous multi contact exploration. In order to map a surface or search for an object on it, it is more efficient to keep all fingers in contact while moving than to touch sequentially several points. Increasing the number of contact points also improves the overall time for the search or the reconstruction. Keeping contact during exploration becomes particularly crucial when the mapping must be precise and when the object being scanned is moving. This allows to keep a precise estimation of the relative position between the robot and the object.

In this chapter, we develop an algorithm to maximize the number of points in contact when the hand is scanning or grasping an object (see Figure 4.1). To this end, we project the forces/torques required for the exploration in the null-space of the contact forces. Additionally, we control the forces at each contact point to prevent an uneven distribution of contact force. Given existing contact points, other desired contact areas of the robot are moved towards the estimated surface of the unknown object. This enables the robot to perform a rapid exploration of complex, non convex shapes while maintaining low contact forces. This controller requires only to know the parts of the robot on which it is desirable to make contact and does not need a model of the environment besides the robot itself. We show that this improves the robot's ability to make contact with unknown surfaces by using tactile sensors. This is crucial for tactile exploration and is very useful for grasping under uncertainty as tactile signals can guide the fingers to actively comply with the sensed shape.



(a) Surface exploration in simulation.

(b) Compliant grasping and attractors.

Figure 4.1: Two of our experiments: a robotic arm and hand system explores a shape in simulation and compliantly grasps an object on a real platform.

This work lead to the following publication:

- Nicolas Sommer and Aude Billard. Multi-contact haptic exploration and grasping with tactile sensors. *Robotics and Autonomous Systems*, 2016. ISSN 0921-8890. doi: 10.1016/j.robot.2016.08.007. URL `http://www.sciencedirect.com/science/article/pii/S0921889016301610`

4.2 Controller structure

In order to explore its environment or grasp an object, the robot needs to create contacts. However, some contact points can be desired while others might just occur during the exploration and not be desirable. Here, we explain how we differentiate between these two types of contact and how both of them are taken into account by our controller. First, we introduce the operational space coordinates using the contact points.

Operational space coordinates using contact normals

At each timestep, for each contact point $i \in \{1, .., N_c\}$ detected by tactile sensors, we define its position $p_c^i \in \mathbb{R}^3$, normal direction $n_c^i \in \mathbb{R}^3$, parent joint $l_c^i \in \{1, \ldots, N\}$, and Jacobian $J_c^i \in \mathbb{R}^{1 \times N}$. The set C contains the joints attached directly above a link that currently hosts a contact point and the set n_c the normals of contact:

$$C = \{l_c^i\}_{i=1}^{N_c}, \qquad n_c = \{n_c^i\}_{i=1}^{N_c} \tag{4.2.1}$$

with N_c the number of contact points and N the total number of DOFs of the robot. The operational space contact Jacobian J_c^i is computed as $J_c^i = {n^i}^T J^i$, where J^i is the contact Jacobian expressed in the robot base frame.

The Jacobian for the operational space coordinates is given by the concatenation of all these contact Jacobians:

$$J_c = \begin{pmatrix} J_c^1 \\ J_c^2 \\ \vdots \\ J_c^{N_c} \end{pmatrix} \tag{4.2.2}$$

Set of areas that can be in contact

During the task execution, the more contact points between the robot and its environment, the more information is retrieved at the same time. For instance, it would be inefficient to try to localize an object on a table using only one fingertip. However, increasing the number of contacts also decreases the manipulability of the robot as each contact introduces a dynamic constraint, as detailed in Section 4.2. A mechanism for deciding whether a contact is desired or not is thus required.

In addition to the contacts points defined in the previous section, we define the desired contacts points $i \in \{1, .., N_d\}$ with position $p_d^i \in \mathbb{R}^3$, parent joint $l_d^i \in \{1, \ldots, N\}$, and Jacobian $J_d^i \in \mathbb{R}^{1 \times N}$. The set D contains the joints which directly control a link that hosts a desired contact point.

$$D = \{l_d^i\}_{i=1}^{N_d} \tag{4.2.3}$$

with N_d the number of desired contact points.

The mechanism of choosing the desired contact points depends on many criteria, including the robot platform, the task and possible prior on the shape to explore or grasp[1]. The combinations of the sets C and D and what they represent are detailed at the beginning of Table 4.1.

Control of the robot in contact

The dynamics of the robot are of the form:

$$M_q(q)\ddot{q} + b(q, \dot{q}) + g(q) + J_c^T(q)f = \tau \tag{4.2.4}$$

where $q, M_q(q), b(q, \dot{q}), g(q), f$ and τ are respectively the vector of joint angles, the joint-space inertia matrix, the Coriolis and centrifugal torques, the gravity torques, the contact forces and the vector of joint torques[2]. The torques τ

---

[1] In this work, we assigned a desired contact point on each link of the robot's hand, hence 3 per finger.

[2] We do not model joint friction, this is an approximation, especially for the joints in the hand. See the discussions about neglecting friction in the results section of the experiments.

Table 4.1: Notation table

Variable	**Description**
C	Current contact points
D	Desired contact points (not time dependent)
$C \setminus D$	Current undesired contacts
$D \cap C$	Current desired contacts
$D \setminus C$	Desired contacts not yet in contact
$p_c^i \quad (p_d^i)$	Position of (desired) contact point i
$n_c^i \quad (n_d^i)$	Normal direction of (desired)contact point i
$l_c^i \quad (l_d^i)$	Parent link of (desired) contact point i
$J_c^i \quad (J_d^i)$	Jacobian of (desired) contact point i
$N_c \quad (N_d)$	Number of (desired) contact points
J_c	Operational space Jacobian
N	Total robot DOFs
q	Vector of joint angles
M_q	Joint-space inertia matrix
b	Coriolis and centrifugal torques
g	Gravity torques
f	Contact forces
τ	Commanded robot torques
$\ddot{x}$	Operational space acceleration of contact points
τ_0	Lower priority torques
N_{τ_0}	Null-space projection matrix, dependent on τ_0
$J_{C \cap D}$	Jacobian of desired contact points
$J_{C \setminus D}$	Jacobian of undesired contact points
J_{τ_0}	Modified operational space Jacobian
K_d	Stiffness matrix of impedance controller
D_d	Damping matrix of impedance controller
p_r^j	Reference position of desired contact point j
p^j	Current position of desired contact point j
v^j	Desired velocity of desired contact point j

applied to the robot are chosen in the form of a prioritized controller:

$$\tau = \tau_1 + N_{\tau_0}(q)\tau_0 \tag{4.2.5}$$

where τ_1 are the torques for the highest-priority task, e.g. the contact force control, N_{τ_0} a modified null space projection matrix that depends on τ_0, the vector of lower priority torques.

The null space projection matrix $N(q)$ is usually chosen so that any torques projected on it do not affect the operational space acceleration $\ddot{x}$. The manipulator dynamics in the operational space are given by pre-multiplying Equation (4.2.4) with $J(q)M_q(q)^{-1}$. For better readability, we do not specify the dependency on the joint angles vector q and its derivatives from now on:

$$\ddot{x} - \dot{J}\dot{q} + JM_q^{-1}(b+g) = JM_q^{-1}(\tau - J_c^T f) \tag{4.2.6}$$

The terms $\dot{J}\dot{q}$ and $JM_q^{-1}(b+g)$ can be compensated for, therefore $\ddot{x}$ is not affected by projected lower-priority torques whenever:

$$JM_q^{-1}N = 0 \tag{4.2.7}$$

We want to avoid creating accelerations or forces for existing desired contacts. This is in order to avoid disturbing the control of the contact force. However, the robot might also host contacts on links which do not accept contacts, i.e. $i \in C \setminus D$. In that case, strictly positive operational space accelerations – towards the surface – should be avoided, but negative accelerations can be accepted as they will break the undesired contact. The strategy adopted here is to take into account the null-space torques τ_0 in the computation of a modified null-space projection matrix. We separate the Jacobian space operational matrix J_c in two sub-matrices $J_{C\cap D} \in \mathbb{R}^{N\times N_{C\cap D}}$ and $J_{C\setminus D} \in \mathbb{R}^{N\times(N_c - N_{C\cap D})}$ containing the concatenated desired contact Jacobians and undesired contact Jacobians, with $N_{C\cap D}$ the number of existing desired contacts. The new conditions are expressed with a modified null space projection matrix N_{τ_0} that respects the following constraints:

$$J_{C\cap D}M_q^{-1}N_{\tau_0} = 0 \tag{4.2.8}$$

$$\text{and } \forall \tau_0 \in \mathbb{R}^N, \quad J_{C\setminus D}M_q^{-1}N_{\tau_0}\tau_0 \leq 0 \tag{4.2.9}$$

This can be ensured by constructing a modified operational space Jacobian matrix J_{τ_0} that contains only the Jacobians of the desired contacts, plus the Jacobians of the undesired contacts which would create undesired contact forces because of the torques τ_0.

Finally, we compute the modified null space projection matrix N_{τ_0} from the

Algorithm 3: Modified Jacobian for null space computation

```
Data: The Jacobians $J_{C\cap D}$ and $J_{C\setminus D}$
Result: The modified Jacobian $J_{\tau_0}$
$J_{\tau_0} = \begin{bmatrix} J_{C\cap D} \end{bmatrix}$;
for $i \leftarrow 1$ to $(N_c - N_{C\cap D})$ do
    if $J^i_{C\setminus D} M_q^{-1} \tau_0 > 0$ then
        $J_{\tau_0} = \begin{bmatrix} J_{\tau_0} \\ J^i_{C\setminus D} \end{bmatrix}$ // Concatenate Jacobians
return $J_{\tau_0}$;
```

modified Jacobian J_{τ_0}, obtained with Algorithm 3:

$$N_{\tau_0} = I - J_{\tau_0}^T \bar{J}_{\tau_0}^T \tag{4.2.10}$$

$$\text{with} \quad \bar{J}_{\tau_0} = M_q^{-1} J_{\tau_0}^T (J_{\tau_0} M_q^{-1} J_{\tau_0}^T)^{-1} \tag{4.2.11}$$

The Equation (4.2.5) for controlling the robot is finally detailed as:

$$\tau = \tau_c + N_{\tau_0}(\tau_d + \tau_r + \tau_e) \tag{4.2.12}$$

The torques τ_c, τ_d, τ_r, and τ_e are described in the following sections, they respectively represent torques for the contact forces, for increasing the number of contacts, for tracking a rest position (for some of the joints), and for driving the exploration of the robot in the corresponding experiment.

This is easily extended to multiple levels of task priorities. The procedure is described in Algorithm 4. This is important for instance to add joint limits and joint centering tasks.

Algorithm 4: Multi-priority algorithm

```
Data: For each task $task$, its priority level $i$, torques $\tau_{task}$ and
      Jacobian $J_{task}$. For the contact tasks, this Jacobian is
      computed according to Algorithm 3
Result: Torques $\tau$
$\tau_{prev} = 0$
for priority $i \leftarrow 1$ to $p_{max}$ do
    $J_i = []$,        $\tau_i = 0$
    for $task \in tasks_i$ do
        $J_i = \begin{bmatrix} J_i \\ J_{task} \end{bmatrix}$
        $\tau_i \mathrel{+}= \tau_{task}$
    $N_i = I - J_i^T \bar{J}_i^T$
    $\tau = \tau_i + N_i \tau_{prev}$
return $\tau$;
```

4.3 Increasing contact area

In order to gather information about its environment by means of touch, or to grasp an object, the robot must at first make contact. Then, once in contact, increase as much as possible the area in contact. To this end, we proceed to switching across two modes of control: one mode controls links not yet in contact, and the other mode controls the contact force at the joints already in contact. We start by explaining how we determine which mode of control to use.

All joints in the fingers that affect control of the force at the contact points and the desired contact points are controlled in torque according to Equations (4.3.2) and (4.3.3). The other joints in the fingers are controlled by a PD controller.

At each time step, the control mode for each joint depends on its position relative to the contact points and desired contact points. The set of joints with existing desired contact points is given by $C \cap D$ – both part of the current existing contacts and desired contacts –, the set of joints with desired contact points is given by $D \setminus C$ – desired contact but not a contact yet –. The set K contains all joints that are not on the same kinematic chain as either a contact point or a desired contact point, and all joints when the robot is not in contact:

$$K = \{i \mid \forall j \in S_i, j \notin C \cup D \parallel C = \varnothing\}_{i=1}^{N} \tag{4.3.1}$$

with $N \in \mathbb{R}$ the total number of DOFs, and S_i the set of joints belonging to the kinematic chain starting from the base of the robot, passing through the joint j until one of the fingertips.
The set K hence contains all joints not used for control of force at contact or desired contact point. These joints track a predefined rest position with a PD controller. In the case that the robot is not in contact, a higher-level controller is in charge of bringing the robot into contact, as explained in the experiments section.

Control of existing contact points

For each existing contact point that is located on a link i which accepts contact points, e.g. for $i \in C \cap D$, we apply a small normal force to maintain the contact:

$$\tau_c = \sum_{i \in C \cap D} J_c^{i^T} f_n \tag{4.3.2}$$

where $f_n \in \mathbb{R}$ is the desired normal force applied at the contact point[3].

---

[3] In our experiments, we chose $f_n = 0.5N$. This value must be above the sensitivity of the tactile sensors to be able to sense contacts. If there is friction in the joints, a higher value is useful because friction can lead to a non-zero acceleration at a contact point despite the null-space projection, and potentially loose contact.

Control of desired contact points

The desired contact points are used to increase the number of contact areas. For each desired contact point that is not on a link where there is already a contact point, a corresponding Cartesian reference position is computed and tracked with a Cartesian impedance controller. The position of each desired contact point is predefined for each link of the robot[4].

The desired contact points are controlled by impedance control, thus the total torque for the desired contacts:

$$\tau_d = \sum_{j \in D \backslash C} {J_d^j}^T (K_d x_e^j + D_d \dot{x}_e^j), \tag{4.3.3}$$

$$\text{with } x_e^j = p_r^j - p^j \tag{4.3.4}$$

with isotropic stiffness and damping matrices $K_d = k_d \cdot I_{3\times3}$ and $D_d = d_d \cdot I_{3\times3}$[5], p_r^j, $p^j \in \mathbb{R}^3$ the reference and current Cartesian positions of the desired contact point j. The reference Cartesian position is computed by integrating the desired velocity v^j:

$$p_r^j = p_{init}^j + \int_{t_{init}^j}^{t} v^j \, \mathrm{d}t \tag{4.3.5}$$

The desired velocity v^j should be chosen according to the task and can take into account prior about the explored surface. The details are given for each of our applications in the experiments section. The initial Cartesian position p_{init}^j and initial time t_{init}^j are reset to the current values p^j and t when the desired contact j is created. This happens either when an existing contact is lost and turns into a desired contact or when the set of desired contact changes.

4.4 Experiments

The algorithm we propose here is meant to be used with robots that have the ability to sense contacts at multiple joints. Unfortunately, to date, there is no commercially available technology to cover a robot entirely with a sensitive skin. In our experiments, we hence first conduct simulations, emulating a perfect sense of touch on all sides of the fingers. We then perform smaller scale - proof of concept - implementations using a real robotic hand covered with patches of tactile sensors. The first setup is a simulation of a robotic arm with a robotic hand attached at the end: there are 7 DOFs for the arm and 4*4 DOFs for 4 fingers. Second, a similar configuration with a real robot equipped with tactile sensors. The simulation allows to control the whole robot and hand in contact with an unknown environment, without the risk of contacts not being detected

---

[4] In this work, the position is defined as the geometrical center of the links.

[5] We choose $k_d = 5N.m^{-1}$, $d_d = 1N.s.m^{-1}$

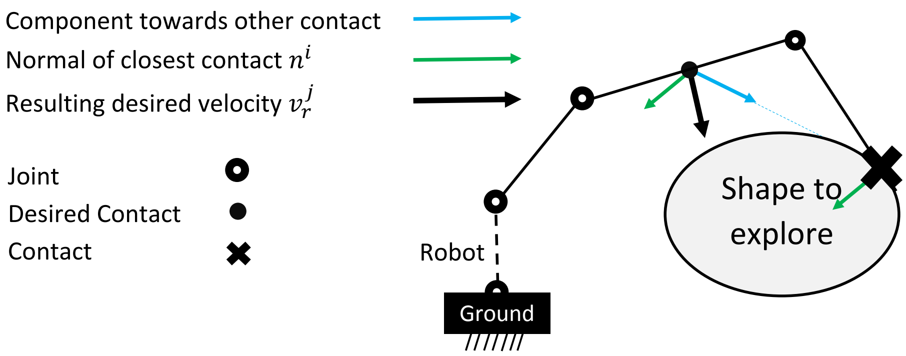

Component towards other contact
Normal of closest contact n^i
Resulting desired velocity v_r^j
Joint
Desired Contact
Contact
Robot
Ground
Shape to explore

Figure 4.2: Exploration: illustration of the computation of the velocity vector v^j for the desired contact points.

(e.g. if they occur on an unsensorized part of the robot) and cause damage[6]. The experiments were conducted in order to prove the effectiveness of the control strategy and the ability to interact with unknown objects and surfaces.

The experiments consisted of an exploration part, conducted in simulation, and an active compliance for grasping part, both in simulation and on a robot.

4.4.1 EXPLORATION

The first applications consist of the exploration of unknown surfaces, using our algorithm to actively comply with the unknown shape of the surfaces. The first experiment consists in the full autonomous exploration of a random shape, and the second experiment in the exploration of the inside surface of cups, using all edges of the fingers. A third experiment tests our algorithm with two different hand models on two new objects. All three of these experiments are carried out in simulation. We aim to demonstrate that maximizing the number of contacts during exploration helps at reconstructing the surfaces and thus gaining information about it, while keeping all contact forces low.

For the exploration, the desired velocity v^j introduced in Equation (4.3.5) of the desired contact point p_r^j is defined by the average of the closest point's normal and the direction towards the closest point (see Figure 4.2).

$$v^j = \lambda \cdot \frac{n^{i^*} + (p^{i^*_c} - p^j)}{\|n^{i^*} + (p^{i^*_c} - p^j)\|}, \quad i^* = \operatorname*{argmin}_{i \in \{1,\ldots,N_c\}} \{p^j - p^i_c\} \tag{4.4.1}$$

where n^{i^*} is the normal of the closest contact point $p^{i^*_c}$, and λ a predefined scalar velocity.

We use a simulated 7 degrees of freedom (DOF) Kuka Light Weight Robot

---

[6] We do not currently have tactile sensors covering the robot's arm. However, we are designing a method to reconstruct the point of contact from the torque sensing at joint level.

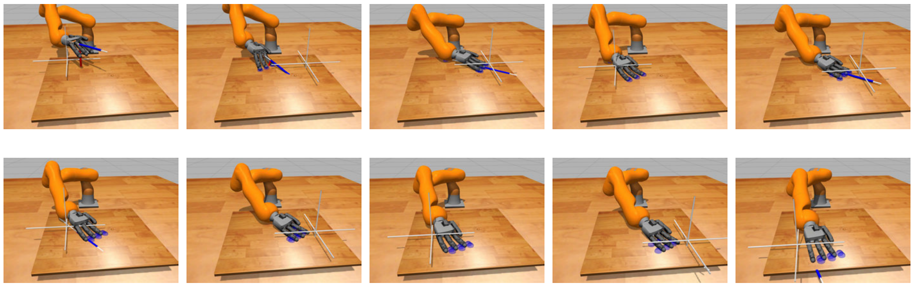


Figure 4.3: Exp 1 Exploration of the table (35 seconds). On each image, the thin white frame indicates the currently tracked reference frame.

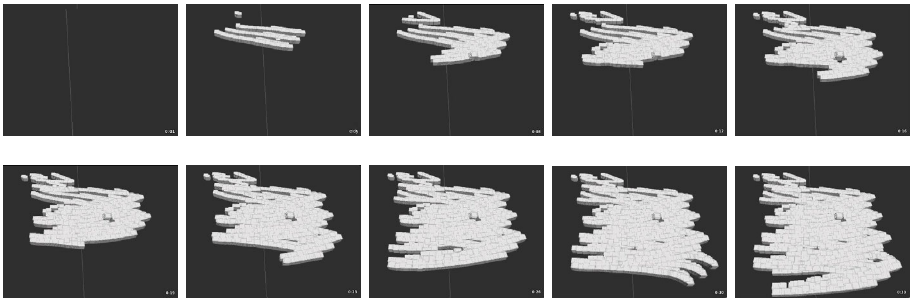


Figure 4.4: Exp 1 3-D reconstruction of the flat surface during the exploration.

arm with a 16 DOF AllegroHand robotic hand. We aim at showing that the robot can autonomously reconstruct a random shape. Because these experiments are carried out in simulation, there is no error in the reconstruction as each datapoint lies perfectly on the explored surface. The coverage of the surface reconstruction however depends on the chosen exploration strategy, which is in our case very simple. It also directly depends on the number of contacts during the exploration, hence the goal to maximize it. The simulation is run at 1000 Hz in Gazebo with ODE and the simulated robot is directly controlled in torque. The computation of the modified null-space projection matrix also runs at about 500-1000Hz (depending on the number of contacts) in a different thread on a PC with a Core i7 processor at 3.6Ghz.

Since one of the desired constraints is to avoid high contact forces, we record the average and the maximum interaction forces during the experiment (at each timestep, among all contact points). We also record the number of contact points during the exploration.

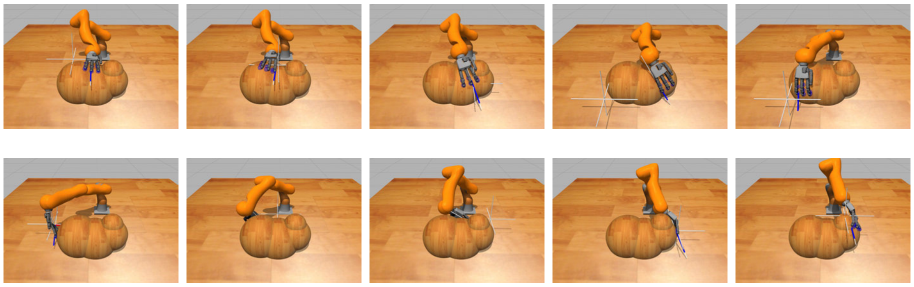


Figure 4.5: Exp 1 Exploration of the shape (50 seconds). On each image, the thin white frame indicates the currently tracked reference frame.

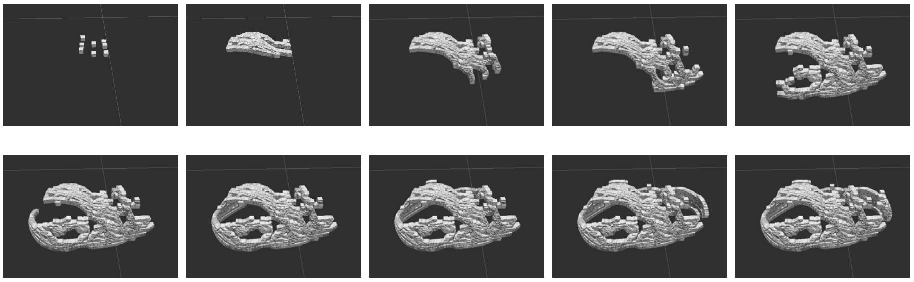


Figure 4.6: Exp 1: 3-D reconstruction of the shape during the exploration.

EXP 1: EXPLORATION OF A SURFACE

In this experiment, the first surface to explore is a flat square and a small bump located on top of it (see Figure 4.7). The goal of the experiment is to find the bump by mapping the surface, in order to show the usefulness of this algorithm for a searching task. The reference frames are located on the borders of the surface to explore, they are indicated as white 3D-crosses on Figure 4.3.

Another complex shape composed of several spheres of different diameters is also explored. The composition of spheres with variable radiuses creates a non-convex shape, hence the orientation of the hand is critical since the shape needs to be approached from different angles. The reference positions are distributed sequentially around the shape (see Figure 4.5).

The goal of the experiment is to autonomously explore and reconstruct the surface of an unknown arbitrary shape, with only a few given key reference positions around the shape to drive the direction of exploration. In order to follow the surface to be explored, the reference position and orientation of the controller need to be defined. They are determined using information from the tactile contacts. A controller is implemented to direct the hand towards the desired exploration locations. This controller is detailed in the appendix Appendix A.1. The exploration is performed 10 times.

Results

The shapes are properly reconstructed from the tactile sensing information,

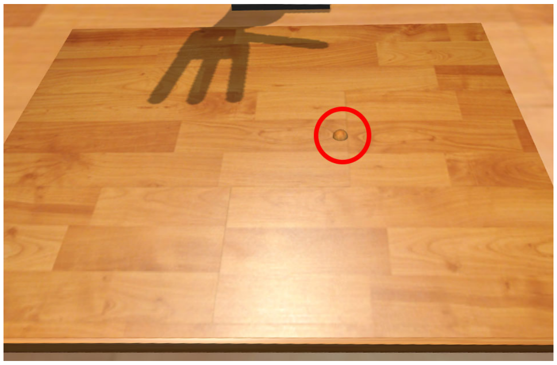


Figure 4.7: **Exp 1** A flat surface to explore. The bump to be localized is circled in red.

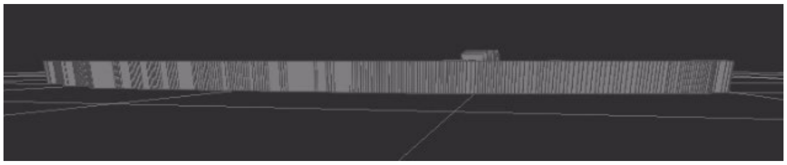


Figure 4.8: **Exp 1** Reconstructed flat surface in Rviz (side view). The bump is clearly visible on top of the table.

progressively as can be seen in Figures 4.3 to 4.6 after on average 35 seconds of exploration for the table and 45 seconds for the spherical shape[7]. The bump on the table can clearly be seen on the reconstructed shape in Figure 4.8 and the reconstructed shape on Figure 4.9.

The hand successfully changes orientation autonomously to explore the different faces of the explored shape. In a few runs of the experiment with the shape, one of the fingers bends and the contact occurs on the back of the finger, see Figure 4.11. While this does not lead to an increase in the contact force, it creates a hand configuration that is less optimal for exploration as less contacts can be made. This issue is further discussed in the discussion section.

During the exploration of the flat surface, the maximum contact force is in average of $0.80 \pm 0.22N$ with a maximum of $1.32N$. During the exploration of the second shape, the maximum simultaneous contact force is on average of $1.00 \pm 0.57N$, with a maximum of $4.59N$, while the average contact force is of $0.56 \pm 0.12N$. The distribution of these forces and the number of contact points can be seen as a boxplot representation in Figure 4.10. These forces should be compared with the desired force at each contact point, controlled in open-loop through the Jacobian transpose method, and set to be 0.5N. Indeed, the null-space projection prevents the commanded torques from influencing the contact forces by construction. However, the forces due to the dynamics of the robot are not compensated and can therefore influence the contact torques. Because the robot moves slowly in this experiment, as a robot should while it is in contact,

---

[7] Video of the experiments:
http://lasa.epfl.ch/videos/downloads/sommer_he.mp4

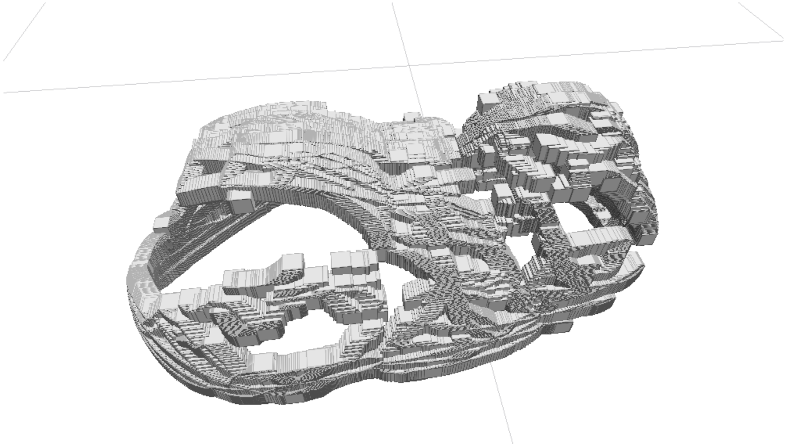


Figure 4.9: Exp 1 Reconstruction of the second shape in Rviz.

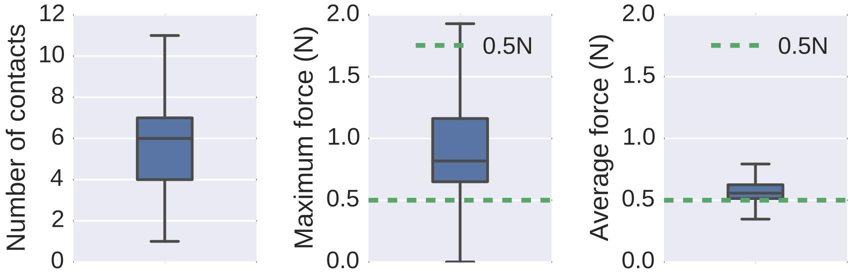


Figure 4.10: Exp 1: Number of contact points, maximum and average contact force during the exploration. The green dotted line represents the desired contact force of 0.5N.

the forces due to the dynamics of the robot are low and the contact forces do not vary far from the reference contact force of 0.5N.

The number of contacts points during the experiment oscillate between 1 and 11, including when the hand starts making contact with the object at the beginning. The thumb is not used in the experiment because its kinematic configuration does not allow it to comply properly with the shape. The average of 6 simultaneous contacts means that each finger has on average two links in contact.

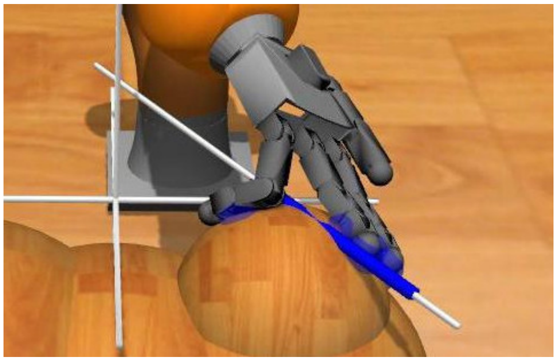


Figure 4.11: **Exp 1**: In some runs of the experiment, one of the fingers makes contact with its back side. While this is not a problem in terms of contact forces, it leads to a hand configuration that is less optimal for the exploration of certain surfaces.

Exp 2: Exploration of the inside of a cup

This experiment consists in exploring the inside of a cup with a robotic hand. This requires to establish contacts on all sides of the fingers and comply with very curvy and non-convex shapes.

The arm controls the position and orientation of the hand. The hand is introduced vertically inside the cup at its center until all fingers are in contact, then it is slowly rotated around the axis of symmetry of the cup for a given angle. The hand is then moved up out of the cup. Before any contact occurs, the fingers are kept in a resting position with all joints slightly bent. The objective is to be able to gather information about the explored object without creating too high contact forces. We hence record at each timestep the number of simultaneous contacts and the current average and highest contact force (of all the current contacts). While the arm is controlled in position, we compared our algorithm, which tries to maximize the contacts, on the hand, against a simple compliant controller with two different sets of gains for the finger joints (see Table 4.2).

The reference position of the joints can be seen in the first image of Figure 4.13, which also displays the progression of the experiment. It is also the rest position used by the active exploration algorithm. We explored 4 different models of cups presented in Figure 4.12.

Results

Table 4.3 sums up the results for each control method, with the average and standard deviation of the two previous values. Figure 4.14 gives the distribution of the number of contacts and the maximum contact force by cup and by control method.

Our method provides more contact points during the exploration of the cups than the other two methods (5.4±1.5 vs 2.6±1.3 and 2.4±1.1). Since the contact forces are set to be at $0.5N$ with our algorithm, the average measured force is of $0.53 \pm 0.09N$. It also keeps a lower maximum contact force, $0.77 \pm 0.26N$,

Table 4.2: Exp 2: Parameters of the tested controllers

	Active adaptation	**Low compliance**	**High compliance**
P gain	-	0.05	0.005
D gain	-	0.01	0.01

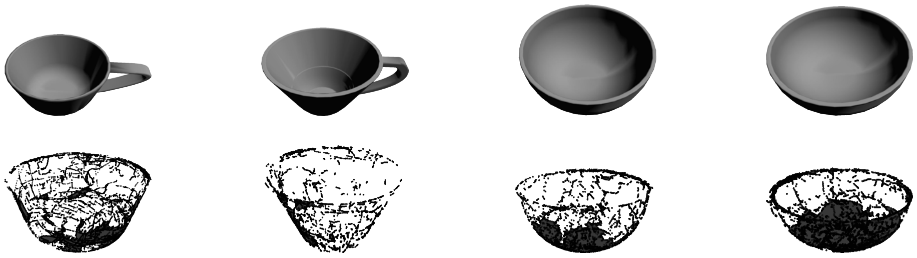

(e) Cup1 (f) Cup2 (g) Cup3 (h) Cup4

Figure 4.12: Exp 2: Models of the explored cups above, and reconstructed versions in Rviz after exploration below.

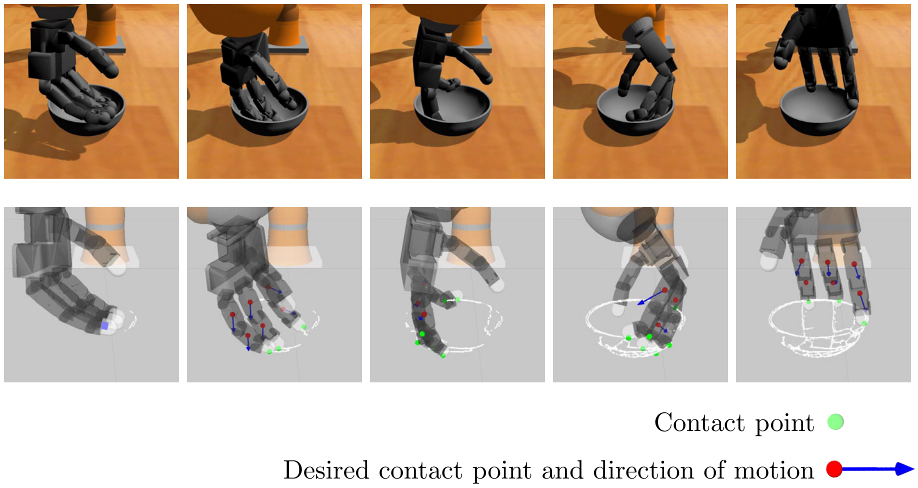
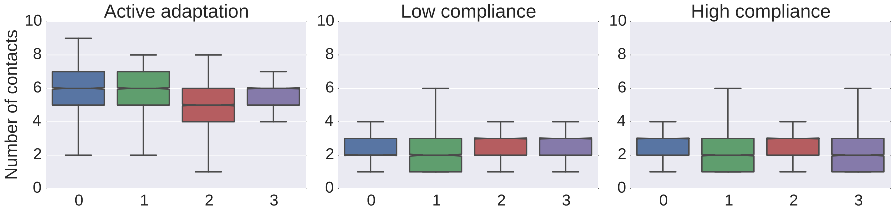
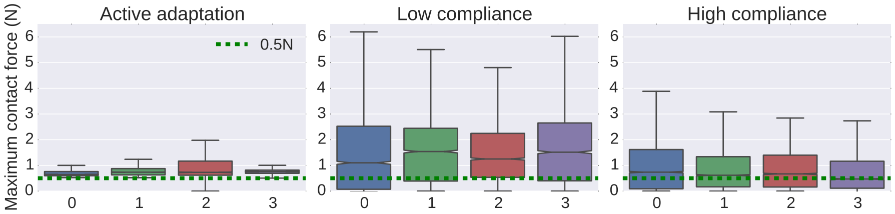


Figure 4.13: Exp 2: Progression of the exploration of *cup 4* with the active compliance algorithm in gazebo (top), with reconstruction of the shape with a tactile point cloud in Rviz (bottom). Green dots are the actual contact points, red dots are desired contact points and the blue arrows their desired direction of motion v^j.

slightly higher than the $0.5N$ reference (go back to Section 4.4.1 for a discussion about the maximum contact forces). As expected, the controller with a higher compliance has lower contact forces ($0.61 \pm 0.0.67N$) than the one with a lower compliance ($1.15 \pm 1.32N$), but both are higher than with our method. Besides, the number of contacts is similar for both of the compliance controllers, which makes the one with high compliance more interesting. However, this might not hold on a real robot as friction in the joints might prevent the use of low gains.



(a) **Number of simultaneous contacts** during the cup exploration



(b) **Maximum contact force (N)** during the cup exploration

Figure 4.14: Exp 2: Number of contacts and maximum contact force for each control method and cup.

Table 4.3: Exp 2: Results

	Active adaptation	**Low compliance**	**High compliance**
Nb. contacts	5.39 ± 1.46	2.62 ± 1.25	2.41 ± 1.14
Max force (N)	0.77 ± 0.26	1.67 ± 1.71	0.86 ± 0.84
Average force (N)	0.53 ± 0.09	1.15 ± 1.32	0.61 ± 0.67

Exp 3: Exploration with other hand configurations

In addition to the exploration of the shape based on spheres in Exp 1 and cups in Exp 2, we also tested our algorithm with two different hand models. The first configuration, called here *Hand 1*, is the same hand as used in the previous experiments, with released joint limits in order to allow the phalanxes to bend both forward and backward. This allows the fingers to take new configurations during the exploration. This can be especially interesting to explore non-convex objects. The second configuration, *Hand 2*, is also based on the same hand, with an additional finger. This gives a total of 4 fingers and thumb, similar to a human hand. For this experiment, we tested our algorithm and other hand models on two new objects, an *IRobot Create* robot and a mailbox, see Figure 4.15. The vacuuming robot contains a concave shape created by the empty dust holder, while the mailbox is made of many sharp edges, which makes it more difficult to keep all contacts.

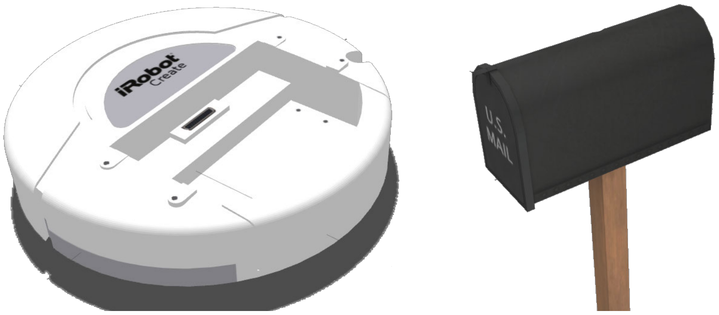


Figure 4.15: Exp 3: Objects explored: the IRobot Create and a mailbox.

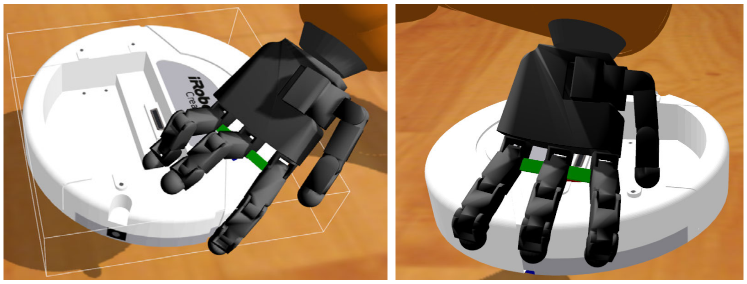


Figure 4.16: Exp 3: Snapshots of the exploration of the vacuuming robot with *Hand 1*. The released joint limits allow the fingers to comply to the concave shape in the middle of the robot.

Results

Predictably, there are in average a few more contacts made with *Hand 2* than with *Hand 1*, since it has one additional finger (Figure 4.18). The difference between *Hand 1* and *Hand 2* in terms of number of contacts is smaller on the first object as the released joint limits of *Hand 1* give it an advantage for complying to the complex shape of the vacuuming robot. The maximum and average contact forces are in the same range and match the results obtained with the original AllegroHand model in the previous experiments. Snapshots of the experiments can be seen on Figure 4.16. The point cloud representation of the objects after exploration can be found in Figure 4.17.

4.4.2 COMPLIANCE EXPERIMENTS

Another application of our active compliance algorithm is to grasp objects by enclosure, maximizing the contacts between the object and the hand and fingers. We perform two experiments in which we study in detail the performance of the compliance.

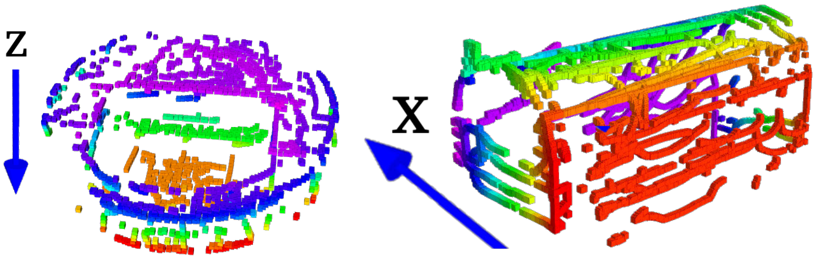


Figure 4.17: **Exp 3**: Reconstructed point cloud of the objects in Rviz. For readability of the 3D representation, the color of the points corresponds to their coordinate on the axis indicated by the blue arrow.

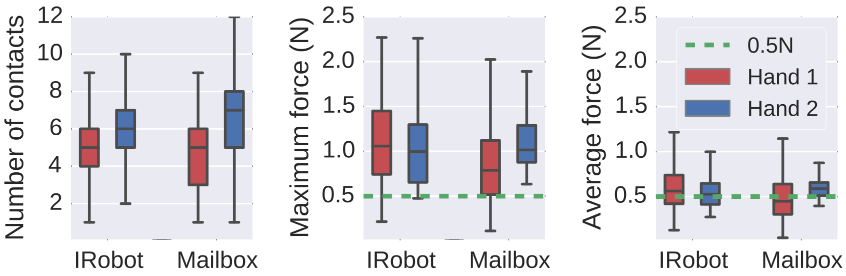


Figure 4.18: **Exp 3**: Number of contact points, maximum and average contact force during the exploration of the IRobot Create and Mailbox models. The green dotted line represents the desired contact force of 0.5N.

Exp 4: Shape compliance

In this first experiment, one finger of a robot hand complies with several objects, and we compare the resulting configuration with the best possible configuration.

Setup

We use the 16-DOFs *AllegroHand* controlled at 300 Hz using open-loop torques, and partially covered with Tekscan tactile sensors on the inside surface of the phalanxes, see Figure 4.19. Because the Tekscan sensors are designed for a human hand, which is smaller than the *AllegroHand*, we use two sets of sensors (i.e. normally for two human hands) and adapt them to the fingers of one *AllegroHand*.

The *AllegroHand* has 4 fingers with 4 DOFs each. The sensors come in patches of 4 by 3-15 taxels which are designed to fit the human hand, but we adapted their configuration for this particular hand. The density of taxels allows to determine the position of contact in addition to an estimate of the pressure. Knowing the geometry of the fingers, we are also able to determine the normal of the contact. For each link, when the summed response is above a

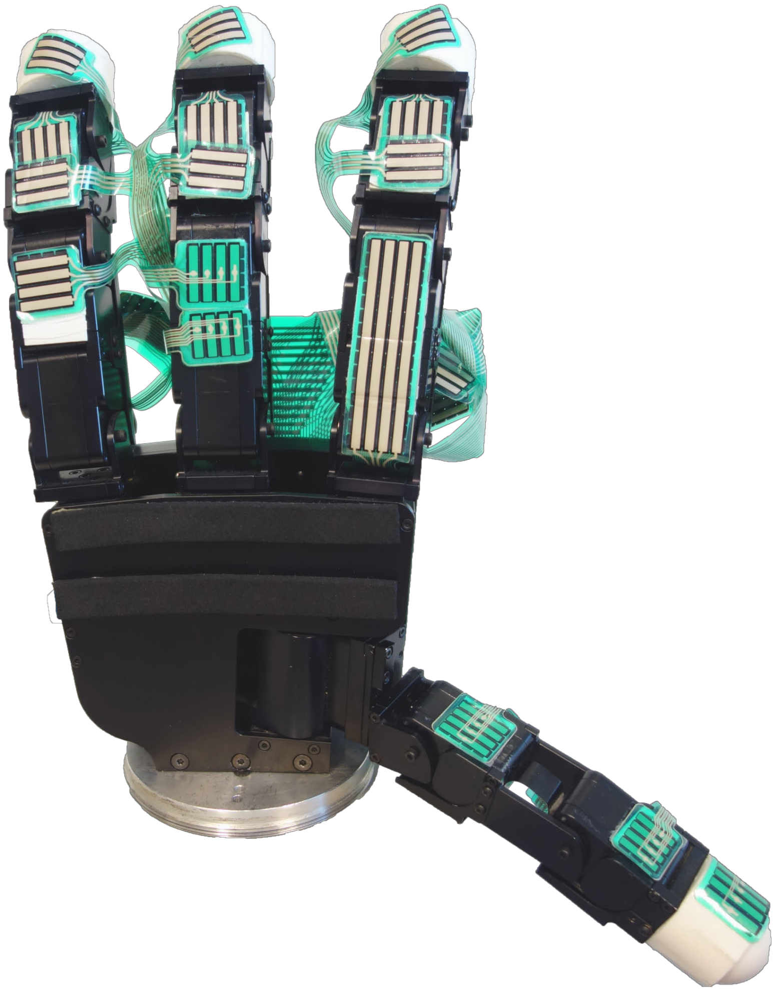
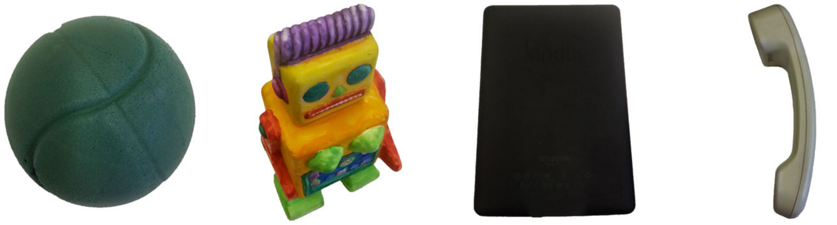


Figure 4.19: The *AllegroHand* with fingers covered with Tekscan tactile sensors. Each patch is a matrix of 4*3 or 4*4 taxels (4*15 for the base of the index finger). The Tekscan sensors are designed for human hands, so the mapping from tactile patch to fingers is customly adapted to the *AllegroHand*.

noise threshold, we define one contact point as the weighted average of all taxels readings on that link.

For each one of a set of 4 objects: a foam ball (soft and round), a robotic statuette (sharp edges), a tablet (flat, all contacts are aligned on a plane and the pressure is evenly distributed on the sensors, so they need to be sensitive) and a phone handle (non-convex curves), see Figure 4.20, a finger complies with their shape, given three different predefined positions/orientations (labeled a,b,c in Figure 4.21) of each object relative to the hand. We chose three different orientations for the phone, tablet and the statuette, and three different positions for the ball since it is symmetric. For each object-pose combination, we measure how many contacts are made with the object, out of the optimal possibility: maximum 3, one per phalanx, determined by manually back-driving the fingers



(a) Foam ball (b) Robot statuette (c) Tablet (d) Phone handle

Figure 4.20: Exp 4: Objects used in the experiment.

Object	Configuration	Number of contacts: reached/optimal
Foam ball	a	**2/2**±0.0
	b	**2/2**±0.0
	c	**3/3**±0.0
Robot statuette	a	**2/2**±0.0
	b	**3/3**±0.0
	c	**2/2**±0.0
Tablet	a	**3/3**±0.0
	b	**2/2**±0.0
	c	**3/3**±0.0
Phone handle	a	2.3/3±0.5
	b	**3/3**±0.0
	c	**3/3**±0.0

Table 4.4: Exp 4: Results of the compliance experiment (10 trials per object and configuration).

and looking for the optimal configuration. This is done 10 times per object for each configuration.

Results

The results are given in Table 4.4: the hand complies with the objects in an optimal way in most configurations. One failure to converge to an optimal solution occurs with the phone handle in configuration (a), because the sensor on the second phalanx can hardly make contact with the surface at that point on the object which forms a depression (see Figure 4.21, the first configuration of the phone). Another possible cause of failure is when contact is made with an unsensorized area, as in Figure 4.22: the top of the fingertip is not covered with tactile sensors and thus the finger gets stuck.

There is not yet a technology that allows to entirely cover a robot with

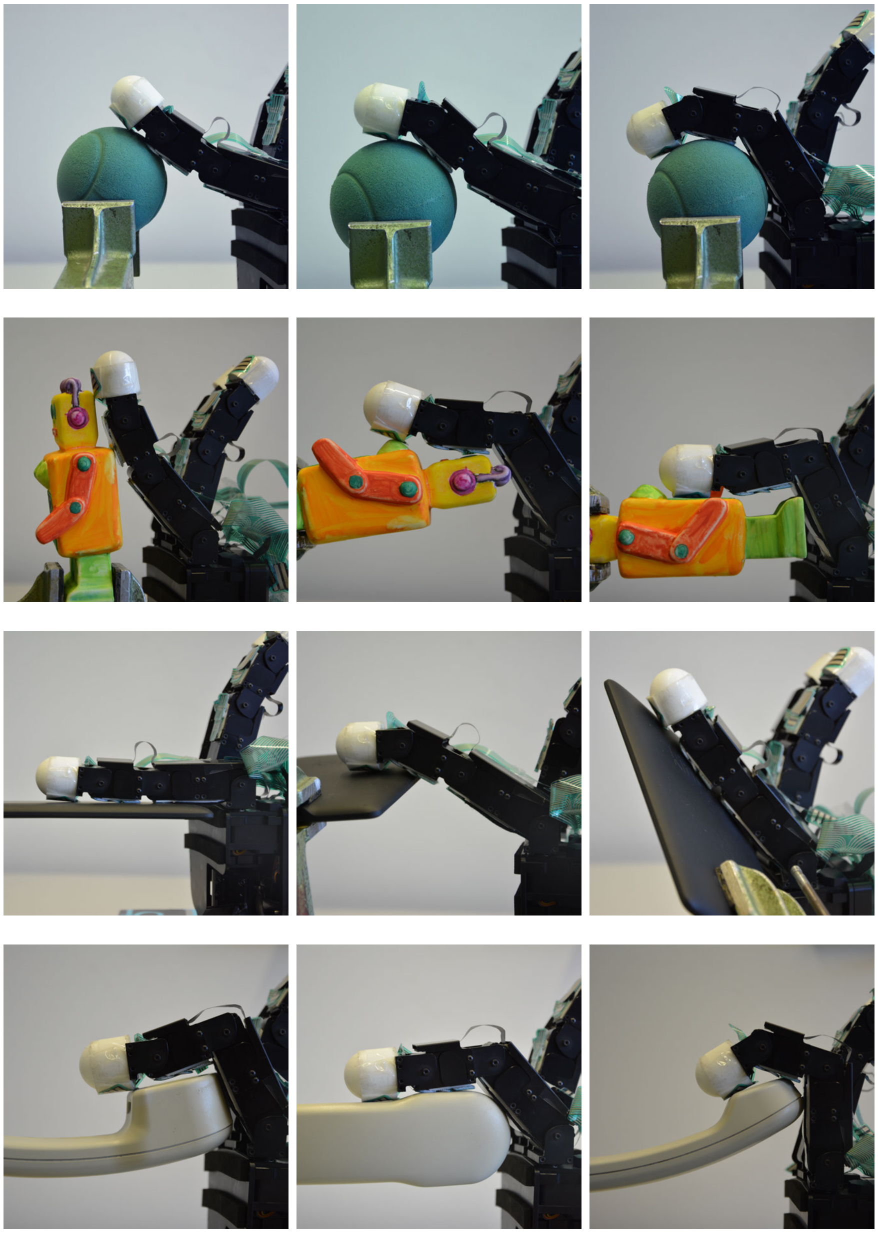


Figure 4.21: **Exp 4:** The 3 poses/orientations for each object (in order a,b,c), with the corresponding ”optimal” contact posture obtained by active compliance with the object. The initial finger configuration is palm flat open upwards, see Figure 4.19.

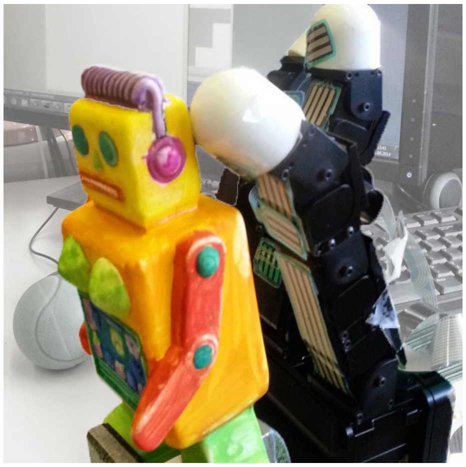


Figure 4.22: Exp 4: Failure with the statuette: the part in contact (the tip of the fingertip) is not sensorized.

artificial skin, especially the fingers and other parts that stretch and bend. Even when an area is sensorized, the sensors might not be sensitive enough to detect light touch. This is problematic as we need to know whether any part of the robot is in contact. Otherwise, there are still important safety issues, for the environment and for fragile parts of the robot such as the fingers. For this reason, the follow-up experimentations were carried out in simulation.

EXP 5: COMPLIANT GRASPING

We compared our method, called here *active adaptation*, with two simple grasping heuristics for enclosing, which can correspond to very simple synergy-based grasps.

The first method (*Enclose1*) consists in closing the joints of the fingers one by one from the base to the tip, until a contact is reached. The second method (*Enclose2*) is similar, but all joints close simultaneously until there is a contact above the joint on the same finger. It is thus faster, but there is a risk that less contacts are made if a link at the end of the finger touches first.

The chosen grasp preshape is inspired by the grasp opposition of the thumb vs. the other fingers from de Souza et al. (2015), which defines a grasp intention by a sum of patches (finger links) oppositions, with for each opposition set, a dominant patch per side. The chosen grasp intention is that of a power grasp for a cylinder of about 3cm of diameter, which is a description that roughly matches all of our tested objects. This provides us with a grasp preshape. It also provides our algorithm with a list of desired contact points (all the patches) and an opposition direction to help define our desired contact points velocities v^j. We only use one opposition direction, therefore we have two dominant patches (one per side of the opposition) and use the virtual line between them

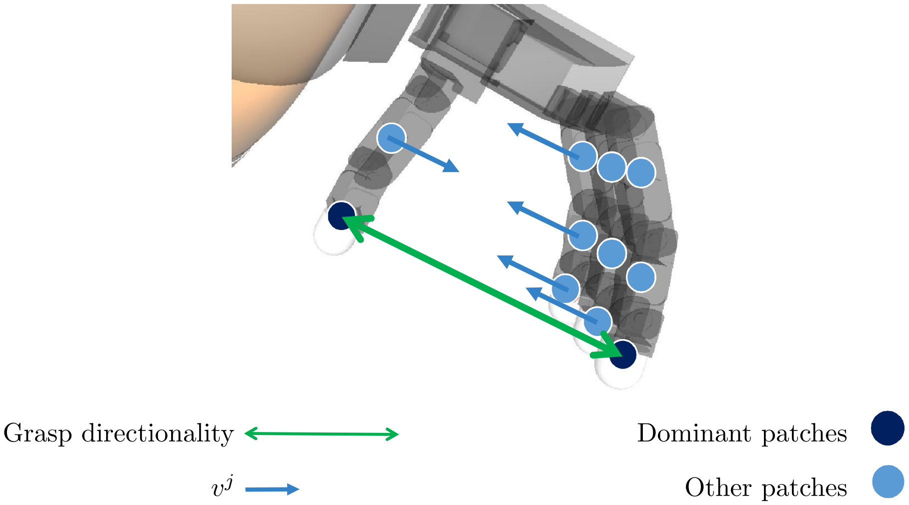


Figure 4.23: Exp 5, Grasping: illustration of the grasp preshape position and the computation of the velocity vector v^j (aligned with the grasp directionality) for the desired contact points using grasp opposition.

to define the direction of the desired velocity[8] for each desired contact point (see Figure 4.23).

We tested the three methods systematically both on a simulated and a real robot. The first part of the experiment (**Exp 5a**) consists in enclosing objects at a predefined grasp position. The object is then sequentially released and grasped again in four other configurations shifted by 2cm in two different directions, and shifted by 17° in two different orientations. These shifts correspond to potential position and orientation uncertainties that the robot might have to deal with in a real application.

In the second part of the experiment (**Exp 5b**), the robot first grasps the objects at the initial position, and we apply sequentially the position and orientation perturbations while the object is grasped. This is important to test how the algorithm adapts to external perturbations. This creates five possible enclosure configurations for Exp 5a, and 4 possible perturbations for Exp 5b. Similarly to the previous experiments, we record the number of contacts made between the object and the hand (when the grasp is finished), and the contact force. However on the real robot, our tactile sensors do not provide values convertible into contact forces since their output depends on the type of material and the area in contact. They do not depend directly on the contact force. We however provide the average and maximum values for the signal given by the tactile sensors. This signal corresponds to the sum of all taxels readings for each patch.

In addition to the number of contacts made with the object, we also compute two grasping metrics based on the Grasp Wrench Space (GWS) (Pollard, 1996): the largest-minimum resisted wrench (or largest ball, or ϵ quality metric), and

---

[8] The norm of the desired velocity is set to $5cm/s$.

the volume of the GWS. These metrics describe what external wrenches can be applied to the object without loosing stability. Thus, the higher, the better. While the ϵ metric considers only the weakest direction, the volume of the GWS provides information about the global robustness of the grasp. Since we expect our method to make more contacts around the grasped object, we also expect a better performance on the grasp metrics.

SIMULATION

Setup

The perturbations on the objects are applied in the simulation by changing their position in the simulation environment. To avoid discontinuities, the position of the object is defined by attaching a virtual spring and damper to it (i.e. Cartesian impedance with high stiffness). Its position and orientation are changed by moving the reference pose.

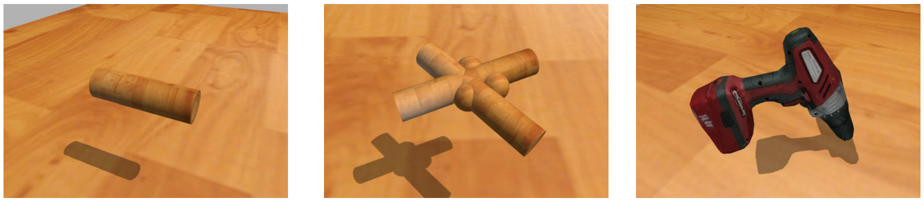
The selected objects are presented in Figure 4.24: we start with a simple cylinder, then an artificial more complex shape – very non-convex – composed by cylinders and spheres, and a drill. The grasping of *Object 1* can be seen in Figure 4.25.

Results

Figure 4.27 details the number of contact points for each control method and object. These results are summed up in Table 4.5 with additional data about the contact forces and grasp metrics. Our algorithm allows to create a high number of contacts with the object compared to the other two controllers. On average, about 9 contacts are made (i.e. a little more than 2 per finger out of 3 possible contacts for the 3 separate links), while only 5 to 6 for the other methods (a little more than 1 contact on each finger).

If we go more into details in the transition from Exp 5a to Exp 5b, our algorithm keeps about the same number of contacts (8.73 vs. 8.67), whereas *Enclose1* (6.00 vs. 5.75) and especially *Enclose2* (5.67 vs. 4.92) loose a lot of contact points. This is expected as Exp 5b is about adapting to perturbations after the grasp, which the other algorithms cannot do properly. The results are similar for the volume of GWS: our algorithm outperforms the other approaches, by providing a larger volume in both sets of experiments, and the difference increases in Exp 5b. The ϵ metric also follows the same trend.

The contact forces are pretty similar for each algorithm in Exp 5a ($< 1N$), whereas for Exp 5b, the other methods based on position control do not adapt to the perturbations and hence create high contact forces.



(a) Cylinder (b) Object 1 (c) Drill

Figure 4.24: Exp 5: Models of the grasped objects in simulation

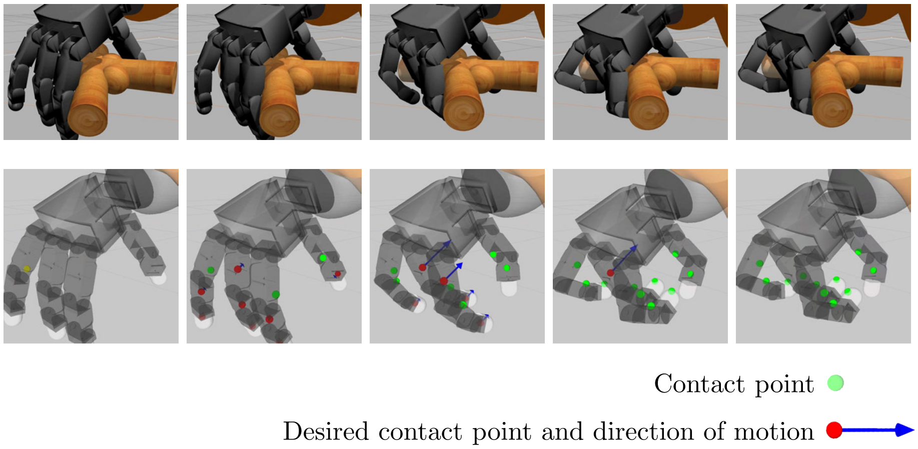


Figure 4.25: Exp 5a: Progression of the enclosure of *Object 1* with the active compliance algorithm in gazebo (top) and Rviz (bottom).

On the robot

Setup

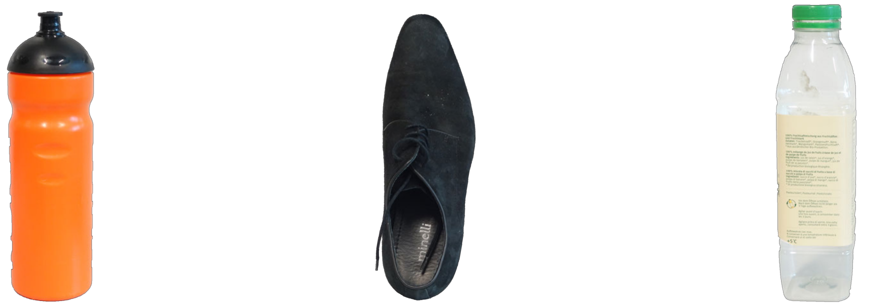
We equipped the 7 DOFs Kuka LWR Robot with the 16-DOFs AllegroHand presented in the previous experiment.

On the real robot, the perturbations are applied by giving the inverse perturbation command to the robot (the arm moves instead of the object). The chosen objects are presented in Figure 4.26(abc): we start with a cylindrical bottle, we also grasp a soft shoe, which is easily deformable, and a plastic bottle with a square section.

Results

The number of contact points is detailed in Figure 4.28 and summed up in Table 4.6 with the other measures and quality metrics. The number of contacts is again higher with our algorithms. On average, about 7 to 8 contacts are made, while only 4 to 6 for the other methods (a little more than 1 contact on each finger). The results for Enclose2 are still slightly worse than for Enclose1, as predicted.

On the real robot, our algorithm performs this time better during the pertur-



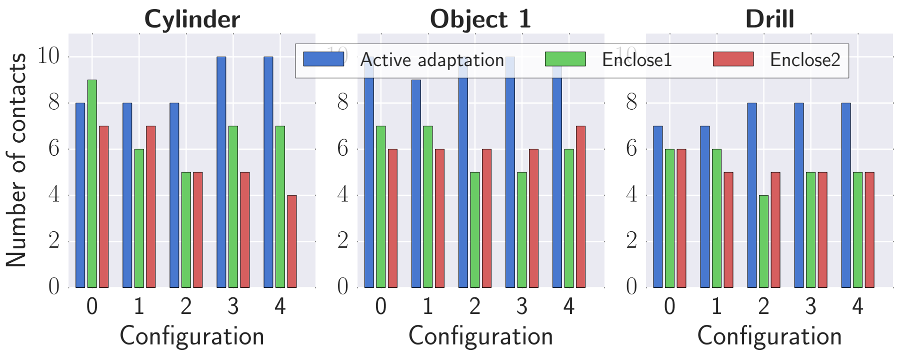
(a) Bottle (b) Shoe (c) Square bottle

Figure 4.26: Exp 5: Models of the grasped objects by the real robot.

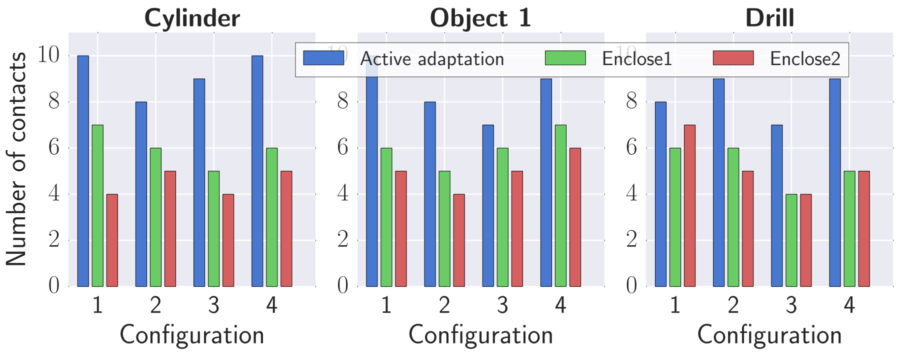
Table 4.5: Exp 5 simulation: Results

Exp 5a: Enclose	**Active adaptation**	**Enclose1**	**Enclose2**
Nb. contacts	8.73 ± 1.12	6.00 ± 1.21	5.67 ± 0.87
Max force (N)	0.92 ± 0.32	0.93 ± 0.58	0.72 ± 0.31
Average force (N)	0.55 ± 0.06	0.62 ± 0.56	0.52 ± 0.27
GWS volume	1.35 ± 0.45	0.95 ± 0.40	0.82 ± 0.52
GWS ϵ metric	0.13 ± 0.02	0.12 ± 0.03	0.13 ± 0.05
Exp 5b: Perturb.	**Active adaptation**	**Enclose1**	**Enclose2**
Nb. contacts	8.67 ± 1.03	5.75 ± 0.83	4.92 ± 0.86
Max force (N)	0.93 ± 0.31	2.13 ± 1.30	1.65 ± 0.86
Average force (N)	0.54 ± 0.07	1.16 ± 0.67	1.21 ± 0.58
GWS volume	1.31 ± 0.35	0.64 ± 0.40	0.51 ± 0.33
GWS ϵ metric	0.13 ± 0.03	0.10 ± 0.04	0.10 ± 0.02

bations (Exp 5b, 7.8 contacts) than the simple grasping (Exp 5a, 7.4 contacts). This can be explained by the effect of the perturbations helping the fingers slide on the surface of the object and thus creating more contacts. In simulation, this behavior relying on friction may not have been properly simulated. For *Enclose2*, the number of contacts actually increases with the perturbations (5.1 vs. 4.3 contacts). This is due to the fact that deformable objects can naturally comply with a non compliant controller and create more contacts, at the expense of high contact forces. Indeed, the set of objects is here more compliant than in simulation, especially the shoe. These results are also reflected in the grasp metrics, with the active adaption creating more robust grasps than the other methods. The tactile signal values are similar in range for all algorithms, with higher values during the perturbations – going from about 12 during en-



(a) **Exp 5a - Enclosing**, Simulation



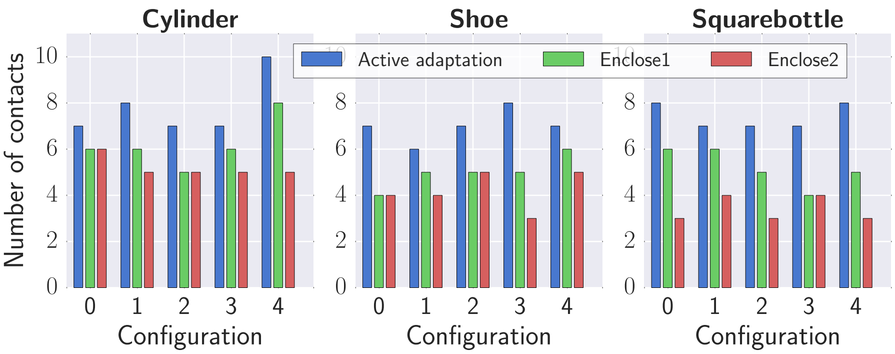
(b) **Exp 5b - Perturbation**, Simulation

Figure 4.27: Exp 5a-b, Simulation: number of contacts for each position/orientation configuration

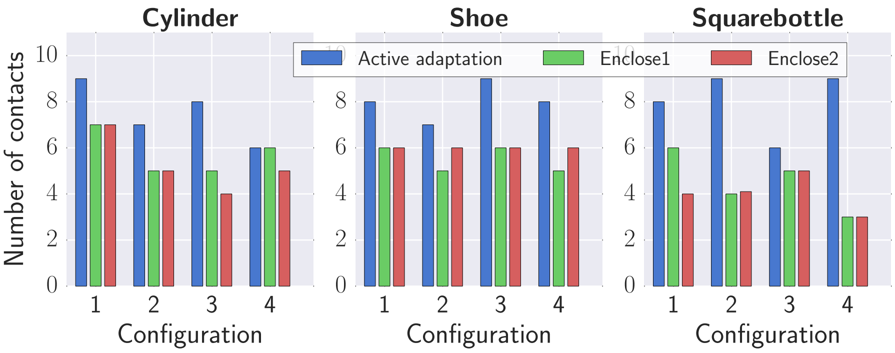
closing to 20 (no unit). Similarly as in the simulation, it is expected that the values increase during the perturbations, and it seems that the tactile readings increase for the active adaptation algorithm too, probably due to friction in the joints and with the object. However, these values from the tactile sensors cannot be precisely translated into contact forces: it is not possible to decouple the intensity of the signal, the area in contact and the material in contact.

4.5 Discussion and conclusion

In this chapter, we presented a method to actively comply with unknown surfaces with a multi-fingered robot equipped with tactile sensors. This method has applications both in haptic exploration and in grasping. To our knowledge, this is the first demonstration of active compliance between a complex system such as a robotic arm and hand, and unknown surfaces, by keeping and creating desired new contacts using tactile information. Our method allows to create and maintain contacts at desired positions on the robot while having unilateral constraints on undesired contacts, in the prioritized tasks framework. While the



(a) **Exp 5a**, real robot



(b) **Exp 5b**, real robot

Figure 4.28: Exp 5a-b, real robot: number of contacts for each position/orientation configuration (Exp 5a: enclosing and Exp 5b: perturbation)

high priority tasks take care of the interaction forces and contact constraints, the lower priority tasks allow to increase the contact area and to drive the exploration motion. Contacts occurring on parts of the robot that are not desired do not disturb the exploration nor create undesired forces thanks to the modified null-space control. We demonstrated the possibility to actively explore around arbitrary shapes with a simulated robot arm and hand. This is useful in the context of search, particularly for occluded areas, by only providing approximate positions for the robot to explore. The robot can then manage to move around the surface creating and loosing contacts while keeping low contact forces.

In the current implementation of the exploration strategy, there are situations when the robot can get stuck in local minimum. We did not tackle here the high-level planning as it is not purpose of this work. Simple approaches based on information gain, coupled with detection of local minimum would probably be enough to further automatize the exploration process.

It is then the task of a high-level planner to change the direction of exploration. We did not tackle here the high-level planning as it is not purpose of

Table 4.6: **Exp 5 real robot**: Results

Exp 5a: Enclose	**Active adaptation**	**Enclose1**	**Enclose2**
Nb. contacts	7.40 ± 0.88	5.47 ± 0.96	4.27 ± 0.93
Maximum *signal*	12.8 ± 6.6	13.0 ± 6.3	11.4 ± 7.4
Average *signal*	6.2 ± 3.6	7.2 ± 3.1	7.0 ± 3.3
GWS volume	1.4 ± 0.6	0.6 ± 0.5	0.4 ± 0.5
GWS ϵ metric	0.13 ± 0.05	0.10 ± 0.05	0.05 ± 0.05
Exp 5b: Perturb.	**Active adaptation**	**Enclose1**	**Enclose2**
Nb. contacts	7.83 ± 1.07	5.25 ± 1.01	5.09 ± 1.11
Maximum *signal*	19.9 ± 7.7	20.8 ± 14.7	18.3 ± 12.0
Average *signal*	9.3 ± 3.8	9.6 ± 5.9	9.0 ± 5.6
GWS volume	1.2 ± 0.5	0.7 ± 0.4	0.7 ± 0.6
GWS ϵ metric	0.14 ± 0.05	0.09 ± 0.05	0.08 ± 0.06

this work, but simple approaches based on information gain, coupled with detection of local minima would probably be enough to automatize the exploration completely [9].

The algorithm does not currently handle several desired contact points on one link. This could be useful for large areas on one link (for instance the palm of the hand) that could host several contact points simultaneously. Currently, if there is already an existing desired contact point on a link, it is not possible to deliberately increase the number of contacts points on that link. This would involve classifying whether each existing contact corresponds to a particular desired contact point.

One particularity of the high-DOFs platforms such as robotic hands is that they can take many different configurations during the exploration, some of which are not optimal to maximize the area in contact. For instance, simultaneous contact on the back of one finger and the front of another finger while exploring a flat area. However, this is an advantage for the exploration of certain shapes, for instance the inside of a cup in which some fingers make contact with one side while other stick to the other side. It also allows to hold two objects at the same time between the fingers[10], see Figure 4.29.

---

[9] We have experimented with planning exploration trajectories using tactile data gathered on the go, encoding tactile data as Octomaps (Hornung et al., 2013) for fast collision checking, and using MoveIt (Ioan A. Sucan and Sachin Chitta) to generate trajectories. The desired arm joint positions coming from the planned trajectories were then fed to our null-space controller as desired joint configurations. The result was not satisfactory as planning takes a lot of time (easily above 1 second, whereas the robot is constantly updating our 3D tactile map with new points). Besides, each new plan may be contradictory with the previous one, hence the robot would start moving in one direction, then switch to another direction when receiving a new plan.

[10] For holding two objects, the closest point of contact used to compute the velocity of a

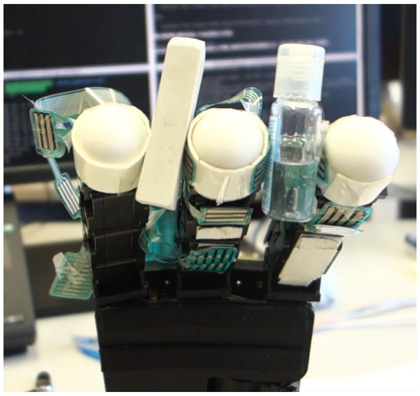


Figure 4.29: Additional illustration of use of the algorithm. The fingers hold two objects between them.

We also demonstrated the ability of this algorithm to comply to arbitrary shapes with an application to grasping. While a lot of the grasp planning research does not consider in detail the actual control strategy, uncertainties make precise grasp planning less relevant on the execution side. Our controller resulted in more contact points and provided more stable grasps than other uninformed enclosing algorithms. It could be a possible solution to implement planned grasps on actual robotic platforms.

---

desired point is valid only if its normal is opposite to the direction from the desired point to this point: $n^i \cdot (p_c^i - p^j) < 0$ as a condition to Equation (4.4.1).
Video of the experiments: http://lasa.epfl.ch/videos/downloads/sommer_he.mp4

Chapter 5

LEARNING EXTERNALLY MODULATED DYNAMICAL SYSTEMS

5.1 Introduction

In order to generate robot motion, the traditional methods based on planning and execution are not well suited to uncertain and changing environments. For instance, grasping traditionally relies on several separate steps: computing a grasp configuration, planning a collision-free robot trajectory and executing that grasp (Bicchi and Kumar, 2000; Roa and Suárez, 2014; Ciocarlie et al., 2014). If the object moves, or the pose is uncertain, the whole process may have to be started over. Also, because planning methods can yield very different results with small configuration differences, the new planned trajectory might be completely different.

Dynamical Systems (DS) offer an efficient way to encode reaching (Mohammad Khansari-Zadeh and Billard, 2014) and grasping motions (Shukla and Billard, 2012), which do not require to re-plan when the configuration changes. This allows to continuously and instantaneously update the trajectory. Furthermore, DS can be learned from demonstrations, instead of programming the robot explicitly. Instead of defining robot tasks as timed trajectories, or as dynamical systems that are indirectly driven forward by time, it is possible to define tasks as time-invariant dynamical systems. The latter have been shown to have numerous advantages for tasks that involve temporal and spatial perturbations (Gribovskaya et al., 2011a).

In order to successfully model the robot motion, the possibility to incrementally perform the demonstrations allows the teacher to refine her demonstrations depending on the robot's current performance. In previous work from colleagues in the lab (Kronander et al., 2015), a way to locally reshape an existing, stable nonlinear autonomous DS, while preserving important stability properties of the original system, was offered. This approach also included a method to enable incremental learning based on Gaussian Processes, for learning to reshape dynamical systems using this representation.

When executing a motion in a real environment, there is also a need to react to external sensory events, besides simply re-planning after a perturbation. For instance, when reaching for something and detecting contact with the robot arm, the trajectory of the robot may need to be adapted online, by modulating the arm dynamics depending on the sensed contact. One way to introduce a dependency from an external signal is through coupling across DS. However, we target here dependency on an external signal whose dynamics may not be known and hence cannot be done through coupling with another DS. Approaches to DS control with external sensing is used primarily for free-space motion and to update the state of the robot and the state of the attractor. Only a few attempts used an external sensing – force – as an input to the system (Gribovskaya et al., 2011b; Ureche et al., 2015). However, this was used to generate the desired trajectory and was then combined into a traditional impedance controller. Moreover, the sensing modulation was global. Here, we generalize this approach to enabling modulation from different types of sensing – not just force – and to allow the modulation to act locally, so as to provide modulation only in relevant parts of the task. Another approach consists in directly including external sensing to the inputs of the regression when learning a DS from demonstrations. By learning a mapping between end-effector position, tactile sensing, and velocity from demonstrations (Sommer, 2012), we were able to generate behaviors based on tactile sensing, including grasping. However, because the resulting system is not autonomous and there are no constraints on the DS formulation, it is hard to ensure stability. We address this by proposing a novel DS representation, called Externally Modulated Dynamical Systems (EMDS). We extend previous work (LMDS) to integrate external input in the DS and use it to reshape its dynamics. Using this external input, we can adapt the DS's behavior, for instance depending on sensory input. We also propose a method to learn how the dynamics are modulated depending on the external signal. Although introducing a dependency on an external signal, we can still guarantee preservation of the stability properties of the (original) dynamics.

With robots moving into human-centered environments, the use of sensory information becomes more and more important to interact with everyday objects. In particular, providing robots with the skill of autonomous grasping, especially under uncertainty, is one of the prerequisites for robots to be useful (Kemp et al., 2007). When the object to grasp must be localized, computer vision based methods do not always work, especially when vision is occluded or illumination conditions are bad (Galleguillos and Belongie, 2010). However, recent progresses in tactile sensing provide a range of possibilities to gather information by touch (Kappassov et al., 2015). In this chapter, we use touch to localize objects in a task of autonomous localization and grasping. We also illustrate our algorithm by using force-torque sensors to modulate the robot's trajectory when navigating between obstacles. An EMDS is used to drive the arm motion depending on the current variance of the estimate of the object's position. The hand configuration is given by a coupling between the EMDS and a DS for the hand. This configuration is used as a input to a controller based on work from chapter 4 to maximize contacts between the fingers and the object while actively complying to the surface. In summary, the main contributions of this chapter are:

1. Introduction of the EMDS framework, allowing the modulation of dynamical systems based on external signals while conserving important stability properties.
2. An interactive learning method for capturing *how* the dynamical system should be modulated by the external signal.
3. The application of EMDS to several challenging tasks, including blind reach-and-grasp, using only tactile input for object state estimation, and navigating through obstacles using contact information only.

The remainder of this chapter is organized as follows: In Section 5.2, we introduce the EMDS formalism and a possible design of the modulation function. We also illustrate the possibilities offered by this formulation in several 2D examples. In Section 5.3, we detail a complete framework used to autonomously localize and grasp objects, in which the EMDS plays a key role. The corresponding experiments and results are presented in Section 5.4. We also present experiments to avoid and navigate between obstacles in Section 5.5.

5.2 Approach

5.2.1 LOCALLY MODULATED DYNAMICAL SYSTEMS

In the previous work from our laboratory (Kronander et al., 2015), the Locally Modulated Dynamical Systems (LMDS) formulation was introduced. Since this chapter extends this work, we first provide a brief overview of the LMDS formulation in this section. LMDS allows to apply arbitrary local learning algorithms to reshape motion dynamics without loss of stability.

Let $x \in \mathbb{R}^N$ represent a N-dimensional kinematic variable, e.g. a Cartesian position or a joint angle vector. Let a continuous function $f\colon \mathbb{R}^N \to \mathbb{R}^N$ represent the original dynamics:

$$\dot{x} = f(x) \tag{5.2.1}$$

We define the *Locally Modulated Dynamical Systems* by multiplying (5.2.1) by a matrix-valued continuous function $M(x) \in \mathbb{R}^{N\times N}$, yielding:

$$\dot{x} = g(x) = M(x)f(x) \tag{5.2.2}$$

Using modulation functions that depend only on the state variable $x \in \mathbb{R}^N$ allows us to prove a number of interesting properties of the reshaped dynamics, including boundedness preservation of stability properties (Kronander et al., 2015).

Importantly, with an appropriate parameterization of the modulation function M, LMDS can be used with non-parametric learning algorithms without constraints.

5.2.2 EXTERNALLY MODULATED DYNAMICAL SYSTEMS

In the LMDS formulation, the input to the system is the state of the robot, x. In many tasks, it is necessary to be able to react to sensory input in a task-specific manner. The goal of the Externally Modulated Dynamical Systems (EMDS) is to provide a DS formulation that allows to learn reactions to sensory events such as contact detected with tactile sensing arrays or force-torque sensors.

Let $s \in \mathbb{R}^M$ be a M-dimensional external signal, independent of the state of the dynamical system. In EMDS, the dynamics are reshaped by a modulation field $M(x, s)$. The form of the dynamics follows the same reshaping structure as LMDS:

$$\dot{x} = g(x, s) = M(x, s)f(x) \tag{5.2.3}$$

where $M(x, s) \in \mathbb{R}^{N \times N}$ is a continuous matrix valued function that modulates the original dynamics $f(x)$. The difference to LMDS in formulation is hence that we allow the modulation to be a function not only on the DS state but also on our external signal.

As the resulting DS is not autonomous, we cannot expect the same stability properties as in the case of the autonomous LMDS formulation. However, by constructing the modulation matrix appropriately, we can achieve guaranteed boundedness and convergence of the dynamics by ensuring that M is full rank and locally active[1]. Stability properties and definitions are given in Table 5.1.

5.2.3 DESIGN OF THE MODULATION FUNCTION

The modulation field $M(x, s)$, as introduced in the previous section, has few design constraints. In this section, we introduce one possible way to design and parametrize this function. We also discuss whether this design conserves the stability properties of the original DS.

MODULATING ROTATING AND SPEED-SCALING DYNAMICS

In Kronander et al. (2015), the modulation function was proposed to be defined as a composition of a speed scaling and a rotation matrix. It is always possible to represent this modulation function compactly as a parameter vector $\theta \in \mathbb{R}^L$, where $L \geq D$ will depend on the chosen parameterization and the dimension of the state $x \in \mathbb{R}^D$. Complex reshaping of the original dynamics

---

[1] This will be discussed in the next section tackling the design of M.

Table 5.1: Definitions and stability propositions

These definitions differ from Kronander et al. (2015) since the DS in Equation (5.2.3) is non-autonomous.

Definition 1 (Locally active). A matrix-valued function $M(x)$: $\mathbb{R}^N \to \mathbb{R}^{N \times N}$ is said to be *acting locally* or to be locally active if there exists a compact subset $\chi \subset \mathbb{R}^N$ such that $M(x) = I_{N \times N}$ for all $x \in \mathbb{R}^N \setminus \chi$.

Definition 1b (Locally active, extended to multivariate matrix). A matrix-valued function $M(x, s)$: $\mathbb{R}^{N \times M} \to \mathbb{R}^{N \times N}$ is said to be *acting locally* or to be locally active on x if there exists a compact subset $\chi \subset \mathbb{R}^N$ such that $M(x, s) = I_{N \times N}$ for all $x \in \mathbb{R}^N \setminus \chi$ and for all $s \in \mathbb{R}^M$.

Let a DS be defined by $\dot{x} = f(x, s)$, where f: $\mathbb{R}^{N \times M} \to \mathbb{R}^N$ is a continuous real-valued function. The following are a set of standard definitions related to the properties of this DS.

Definition 2 (Boundedness). A DS is bounded if for each $\delta > 0$, there exists $\epsilon > 0$ such that:

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| < \epsilon, \quad \forall t > t_0, \quad \forall s_t, t > t_0$$

Definition 3 (Equilibrium point). An equilibrium point for a DS is a point $x \in \mathbb{R}^N$ such that $f(x, s) = 0, \quad \forall s \in \mathbb{R}^M$.

Definition 4 (Stability). An equilibrium point x^* is said to be *stable* if for each $\epsilon > 0$, there exists $\delta(\epsilon) > 0$ such that:

$$\|x(t_0) - x^*\| < \delta(\epsilon) \Rightarrow \|x(t) - x^*\| < \epsilon, \quad \forall t > t_0, \quad \forall s_t, t > t_0$$

Definition 5 (Local Asymptotic Stability) An equilibrium point x^* is called *locally asymptotically stable* if it is stable and, if in addition there exists $R > 0$ such that:

$$\|x(t_0) - x^*\| < R \Rightarrow \|x(t) - x^*\| \to 0, \quad t \to \infty, \quad \forall s_t, t > t_0$$

If R can be chosen arbitrarily large, the equilibrium point is *globally asymptotically stable.*

Proposition 1 (Equilibrium points). *If $M(x, s)$ has full rank for all x and s, the reshaped dynamics has the same equilibrium point(s) as the original dynamics.*

Proposition 2 (Boundedness). *Assume that the original dynamics is bounded (See. Def 2). Assume further that $M(x, s)$ is locally active on x in a compact subset $\chi \subset \mathbb{R}^N$ (See Def. 1b). Then, the reshaped dynamics is bounded.*

Proposition 3 (Lyapunov stability). *Consider a system $\dot{x} = f(x)$ that has a single equilibrium point. Without loss of generalization; let this equilibrium point be placed at the origin. Assume further that the equilibrium point is stable. Assume that the criteria for Propositions 1 and 2 are satisfied. If in addition, χ does not include the origin, the reshaped system is stable at the origin.*

Proposition 4 (Local asymptotic stability). *Consider a system $\dot{x} = f(x)$ that has a single equilibrium point. Assume that the conditions of Propositions 1, 2 and 3 are satisfied. then, the reshaped system is locally asymptotically stable at the origin.*

The proofs of the corresponding propositions without external modulations are trivially extended to EMDS with these modified definitions. They can be found in Appendix B.2.

can then be learned by using non-linear regression to learn a function mapping from the state to this parameter vector.

The rotation angle ϕ can always be recovered from θ, e.g. as the norm of a sub-vector of θ (angle axis representation), or as an independent element of θ. Hence, given a learned function from the state to the reshaping parameter vector, we can find the rotation angle as a function of the state, $\phi(x)$. In EMDS, we let the external signal modulate the rotation angle and the speed scaling before reconstructing M and applying the modulation to the original dynamics:

$$\theta(x, s) = h_s(s)[\phi(x)\mu_R, \kappa(x)] \tag{5.2.4}$$

with μ_R the normed rotation vector. The modulation function $M(x, s)$ is defined as:

$$M(x, s) = (1 + \kappa(x, s))R(x, s) \tag{5.2.5}$$

with $R(x, s)$ the rotation matrix associated with the rotation vector $\phi \cdot \mu_R$. The mappings $\phi(x) : \mathbb{R}^N \to [-\pi, \pi]$ and $\kappa(x) \to (-1, +\infty)$ are continuous functions from a robot position to respectively a rotation angle and a speed-scaling.

As in standard LMDS, the state dependent maps $\phi(x)$ and $\kappa(x)$ should be locally active. The parameters are also influenced by the continuous external activation function $h_s : \mathbb{R}^M \to [0, 1]$, that depends on the external signal s.

By construction, because $R(x, s)$ is a rotation matrix, it has full rank. So does $M(x, s)$ and therefore all the stability properties are guaranteed for any x and s.

5.2.4 ILLUSTRATIVE EXAMPLES

Here, we give a few 2d illustrative examples of how the external input in EMDS can influence the dynamics of the original DS, using the modulation matrix design presented in the previous section. Consider the following linear original dynamics:

$$\dot{x} = -Ax = -\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} x \tag{5.2.6}$$

Let the following function $h_x : \mathbb{R}^2 \to \mathbb{R}$ describe the influence of the modulation and impose the locally active property:

$$h_x(x) = \begin{cases} 0 & \text{if } \|x\| < 0.08 \\ 20 \cdot \|x\| - 1 & \text{if } 0.08 \leq \|x\| < 0.1 \\ 1 & \text{if } 0.1 < \|x\| < 0.7 \\ -20 \cdot \|x\| + 15 & \text{if } 0.7 \leq \|x\| < 0.85 \\ 0 & \text{otherwise.} \end{cases} \tag{5.2.7}$$

The value of $h_x(x)$ is visible as a Grey-scale for the following examples in Figures 5.2 and 5.3.

The external signal s influences the local modulation according to the following activation function[2]$h_s\colon \mathbb{R} \to [0,1]$:

$$h_s(s) = \begin{cases} 1 & \text{if } s < 0.0 \\ 1 - 10s^3 + 15s^4 - 6s^5 & \text{if } 0.0 \le s \le 1.0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.2.8}$$

This function is illustrated in Figure 5.1.

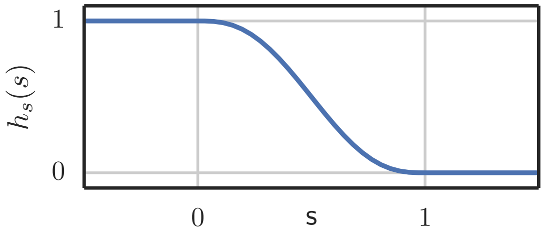


Figure 5.1: External activation function $h_s(s)$.

MODULATING ROTATING DYNAMICS

The modulation function in two dimensions is defined as the following rotation matrix without speed-scaling:

$$M(x,s) = \begin{bmatrix} cos(\Phi(x,s)) & sin(\Phi(x,s)) \\ -sin(\Phi(x,s)) & cos(\Phi(x,s)) \end{bmatrix} \tag{5.2.9}$$

Introducing the local activation function $\phi(x) = h_x(x)\phi_c$, with $\phi_c \in [-\pi, \pi]$ a constant angle, the rotation angle $\Phi(x,s)$ is given by:

$$\Phi(x,s) = h_s(s)h_x(x)\phi_c \tag{5.2.10}$$

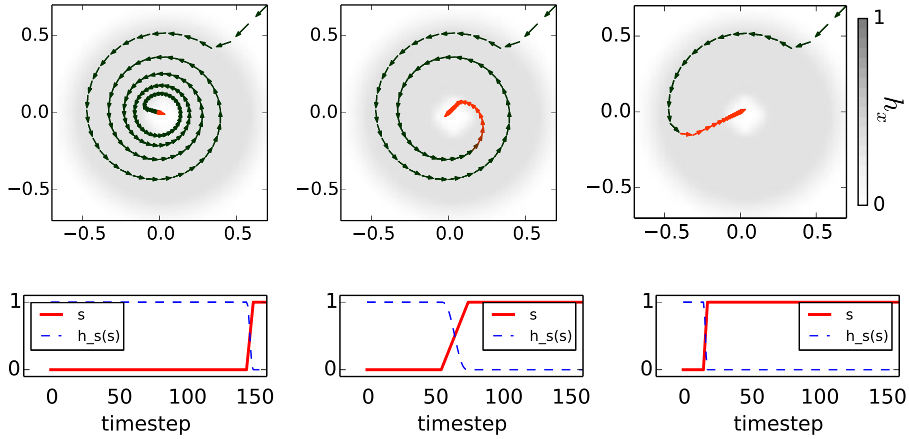
This results in a spiraling behavior where and when the DS is modulated.

It is modulated where the norm of x is above 0.08, below 0.85 (see Equation 5.2.7) and when there is no external signal inhibiting the modulation through $h_s(s)$ (see Equation 5.2.8). When the external signal is active, the rotation is inhibited and thus the system converges much faster with the original dynamics – a straight line.

The resulting dynamics are given in Figures 5.2(a) to 5.2(c) using $\phi_c = 81°$and different arbitrary profiles of external signal s. The evolution in time of the external signal s and consequently of the activation function $h_s(s)$ is

---

[2] The corresponding equation is a fifth order polynomial interpolating from 1 to 0 as s goes from 0 to 1.



(a) (b) (c)

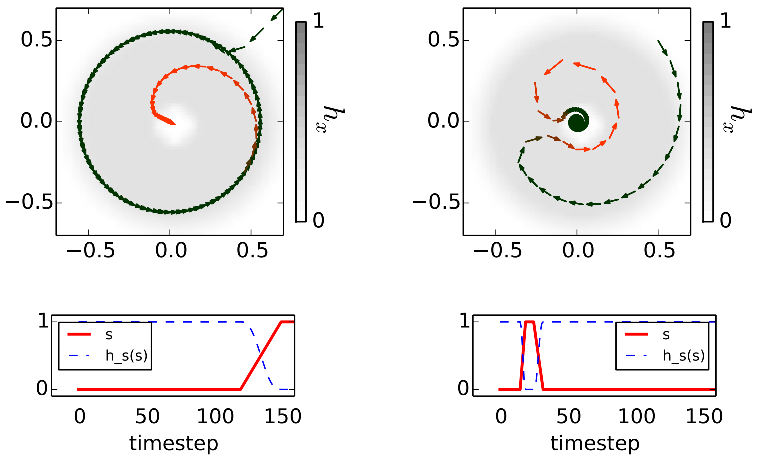
Figure 5.2: **Top:** Examples of the DS modulated by an external signal. The DS is the same in (a)-(c) but the external signal's profile is different in each plot. The color of the arrows indicates the value of the external modulation signal. The background color represents the value of $h_x(x)$. **Bottom:** Corresponding profiles of the external signal s and the function $h_s(s)$ which inhibits the rotational modulation.

drawn on the plots below the evolution of the DS in 2d, and the value of s is also represented on the top figures by the color of the arrows. When the signal s becomes high, the activation function $h_s(s)$ goes to 0 and the system switches from spiraling to the original linear dynamics, converging rapidly. The resulting behavior can be used for instance to switch between searching and reaching motions on a robot. In Figure 5.2(a), s is activated late, hence the system follows the modulated dynamics – spiraling – until it reaches the origin. In Figure 5.2(b), s increases earlier and at a slower rate, therefore the system switches gradually from spiraling to reaching directly with the original dynamics. Finally, in Figure 5.2(c), s changes abruptly and so does the dynamics of the system.

We also provide an example where the system is not globally asymptotically stable in Figure 5.3(a), by setting the maximum modulation angle ϕ_c to 90°. When the external signal is 0, the DS goes into a limit cycle. Boundedness is however enforced thanks to the locally active property.

In Figure 5.3(b), we illustrate another example behavior of our modulated system using different original dynamics (with $A = \begin{bmatrix} 0.05 & 0.2 \\ -0.2 & 0.05 \end{bmatrix}$ in Equation (5.2.6)), a maximum modulation angle $\phi_c = 160°$ and a signal s varying between 0 and 1.

The modulation also applies a speed-scaling of factor 3, visible on the top figure from the length of the arrows changing with s. Depending on s, the direction of the rotation is changed. The resulting behavior is already more complex, while it is still based on linear original dynamics. By introducing non-



(a) Limit cycle example

(b) Different original dynamics

Figure 5.3: (a) Example of a modulation of the DS, using $\phi_c = 90°$. The system does not converge while s is 0 (i.e. the external activation function $h_s(s)$ is 1). If that is the case, the system stays in a limit cycle. (b) An example with different original dynamics and a modulation that also applies a speed-scaling.

linear dynamics, even more complex behavior can be obtained. By construction, the system is still bounded for any s and no spurious attractor can be introduced.

Comments on local modulation

In these examples, the modulation function $M(x, s)$ acts locally according to $h_x(x)$, as is represented in Figures 5.2 and 5.3 by the Gray background color. The local property ensures boundedness and local asymptotic stability for any chosen modulation matrix. It may thus be useful to keep locality even when not required by the desired dynamics, but only for the provided stability purposes. In a searching task such as illustrated in the above examples, it also makes sense to only activate the searching behavior in a subregion of the state-space corresponding to the searching region, hence the local activation in a donut-shape around the attractor at 0, according to Equation (5.2.7).

5.2.5 Learning EMDS

Using the design of the modulation function presented above, it is possible to retrieve a normal LMDS by removing the dependency on the external signal, i.e. by replacing $h_s(s)$ with 1 (i.e. never inhibiting the local modulation). Conversely, an EMDS can be created by associating an existing LMDS with the function $h_s(s)$.

Therefore, an EMDS can be based on an LMDS learned the same way as in the original LMDS formulation, using GP-MDS based on Gaussian Processes Regression (GPR), or any arbitrary local learning algorithm. The external signal activation function $h_s(s)$ can then be provided or learned separately to form the EMDS. To sum up, one way to to learn a complete EMDS from scratch with training data can be the following procedure:

1. Learn a DS –the original dynamics– from demonstration data, e.g. with SEDS (Khansari-Zadeh and Billard, 2011).
2. Learn new dynamics from other demonstration data to represent different dynamics, expressed as a modulation of the original dynamics, using a local learning algorithm or GP-MDS as presented in Kronander et al. (2015).
3. Learn the function $h_s(s)$.

See the following Sections 5.3.2 and 5.5.1 for examples of learning $h_s(s)$ in our searching and grasping task as a non-linear regression problem using Gaussian Process and human demonstrations to train the model on.

5.3 Autonomous Localization and Grasping

As an application of EMDS, we consider a highly challenging autonomous search-and-grasp task. We rely entirely on tactile sensing to localize the object to be grasped. In such a task, there are two subtasks involved: 1) to estimate the pose of the object and 2), when certain enough of the object's pose, to attempt to grasp the object. The searching and grasping behaviors are first modeled by LMDS (search as the original dynamics, and grasp as the reshaped dynamics, see Figure 5.7). Then, to incorporate our confidence in the object pose's estimate in the dynamics, we learn an activation function $h_s(s)$ from demonstrations and use the EMDS as described in Section 5.2.

During the exploration motion of the arm, generated by our EMDS (more about this in Section 5.3.2), tactile data from contacts with the robot is fed to a particle filter responsible for localizing the object. The state of the estimation is fed back to the EMDS: the variance of the object pose's distribution is used to modulate between searching and grasping behaviors, see Figure 5.4. Besides, the current estimate of the object's pose defines the position of the attractor of the EMDS. When the object's pose is known with certainty, the system's dynamics autonomously change to the grasping behavior. The hand and finger's behavior is controlled by a coupling mechanism (Shukla and Billard, 2012) between the EMDS generating the arm motion and a coupled DS generating the finger motion. The whole framework is illustrated on Figure 5.4.

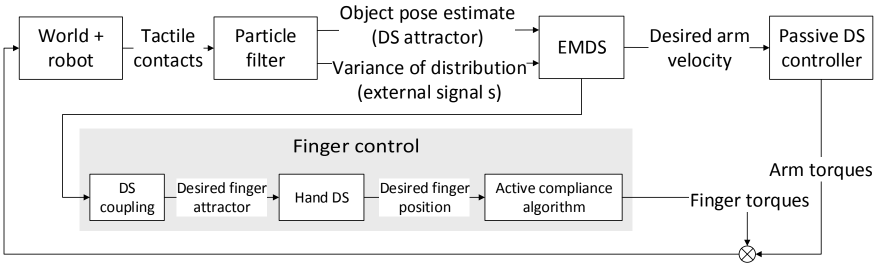


Figure 5.4: The framework for autonomous localization and grasping with EMDS.

5.3.1 Belief model

In order to model the belief over the object's position, we chose to represent the probability distribution with a particle filter. A particle filter is a Bayesian probabilistic method which recursively estimates the posterior from a prior probability distribution by integrating dynamics and sensing. A particle filter is composed of two main elements, the first integrates the dynamics of the system using a motion model $p(\xi_t|\xi_{t-1}, \dot{\xi}_t)$, $\xi_t \in \mathbb{R}^6$ representing the object's state at timestep t (position and orientation). The second integrates the sensing using a measurement model $p(y_t|\xi_t)$, y_t representing a measurement, to update the probability distribution. The two steps are depicted below:

$$p(\xi_t|y_{0:t-1}, \dot{\xi}_{0:t}) = \int p(\xi_t|\xi_{t-1}, \dot{\xi}_t)p(\xi_{t-1} y_{0:t-1}, \dot{\xi}_{0:t-1})d\xi_{t-1} \tag{5.3.1}$$

$$p(\xi_t|y_{0:t}, \dot{\xi}_{0:t}) = \frac{p(y_t|\xi_t)p(\xi_t|y_{0:t-1}, \dot{\xi}_{0:t})}{p(y_t|y_{0:t-1})} \tag{5.3.2}$$

The probability distribution over the state $p(\xi_t|y_{0:t}, \dot{\xi}_{0:t})$ is represented by a set of weighted particles which correspond to possible poses of the object.

The sensing model gives us the likelihood $p(y_t|\xi_t)$ of a particular set of tactile contacts $y_t \in \mathbb{R}^{7\times N_c}$(position, normal direction and intensity of contact for each of the N_c contacts) given an object's pose ξ_t. This likelihood function is based on generating a virtual sensation $\hat{y}_t = G(\xi_t)$ from a possible object pose ξ_t corresponding to one of the particles and comparing it to the sensed measurement. We detail the likelihood and comparison function $p(y_t|\xi_t) = C(y_t, \hat{y}_t)$ in the appendix, see Section B.1. In particular, y_t and $\hat{y}_t$ can have different dimensionality depending on the number of contacts and how this is tackled is explained there.

5.3.2 EMDS

An externally modulated DS as formulated in Section 5.2 generates the arm motion. It uses the design presented in Section 5.2.3, based on modulating rotation dynamics. Its attractor is translated to the latest expected value of

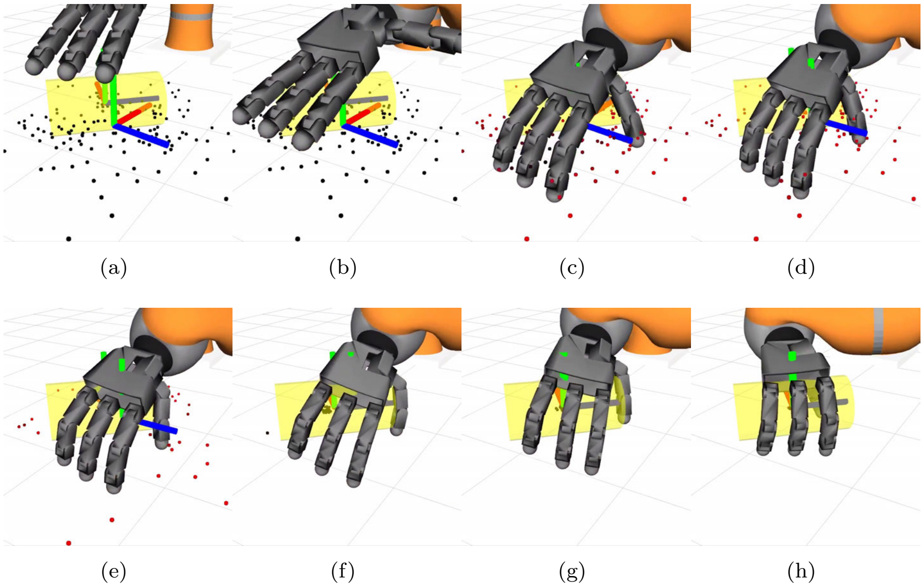

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.5: Exp 1: Illustration of the search behavior. On frame (**c**), the particles located around the hand's position start being discarded as no contact occurs: they have very low probability. The best estimate is then updated towards the real object's position, and the attractor of the DS is set to that position. With additional contact information, the estimate is updated more precisely on frame (**g**). The references frames made out of three RGB segments represent the real object's position and the estimated one.

the robot's pose, represented by $\hat{\xi}_t = \mathbb{E}[p(\xi_t|y_{0:t}, \dot{\xi}_{0:t}])$, i.e. the weighted average over each particle. The external signal is the norm of the covariance matrix of the object's position estimate: $s_t = \| \operatorname{Var}[p(\xi_t|y_{0:t}, \dot{\xi}_{0:t})] \|$.

Both the original dynamics and the modulation of the LMDS are provided as modeled reaching and grasping behaviors[3]. A searching behavior is implicitly provided by the evolution of the particle filter's best estimate. When the hand goes to a location and there is no sensed contact, the particles become depleted in that area. The robot then goes to the new best estimate at a different location and this process results in a searching behavior. An illustration of this behavior[4] can be seen in Figure 5.5.

In the modulated trajectory corresponding to the grasping behavior, the hand approaches the current attractor from above in order to implicitly avoid collisions with the fingers and to properly enclose the object. Typical trajectories of the original and the modulated dynamics can be seen on Figure 5.7. The output of the EMDS is the desired velocity of the end-effector, directly fed to the

---

[3] In our case, the original dynamics DS – reaching – is a simple linear DS. The grasping behavior is coded as a modulation of the original dynamics to generate a grasping behavior. Both can also be learned from demonstration, as described in Section 5.2.5.

[4] In this example, the *grasping* dynamics is not emphasized since the end-effector is already close to the attractor when it is updated. The grasping dynamics is illustrated more clearly on Figure 5.10.

passive DS controller (Kronander and Billard, 2016), designed to perform closed-loop control of DS while ensuring passivity, and ideally suited for uncertain manipulation tasks such as this.

Learning the external activation function

The function $h_s(s)$ mapping the external signal to the activation of the modulation can either be programmed or learned, with the constraint that its values lie between 0 and 1. We chose to learn this function since the values of the covariance matrix are not necessarily easy to link to the task. On the contrary, our graphical visualization (see Figures 5.5 and 5.10 for instance) of the object's pose estimated distribution is easy to interpret. It provides a way for the user to perceive the current particle's filter uncertainty value s through the visualization of the particles. The user also has access to the value s through a live plot showing the evolution of the value.

In order to learn the function $h_s(s)$, we go through a short learning phase during which a teacher manually selects the desired behavior while the task is being executed. During this phase, the teacher chooses how much to inhibit or not the local modulation of the original LMDS system using a graphical user interface (GUI) with a slider control. The teacher chooses continuous values between 0 (reaching approach) and 1 (grasping approach). We record pairs of values $(s_t, h_s(s_t))_{t=1..t_{max}}$ during these experiments in order to model the function h_s.

The recorded data are used to train a Gaussian Process Regression (GPR) model, using the squared exponential covariance function. The kernel's hyper-parameters are determined manually by using prior knowledge from the training data. Such a learned function $h_s(s)$ and the training data can be seen on Figure 5.6.

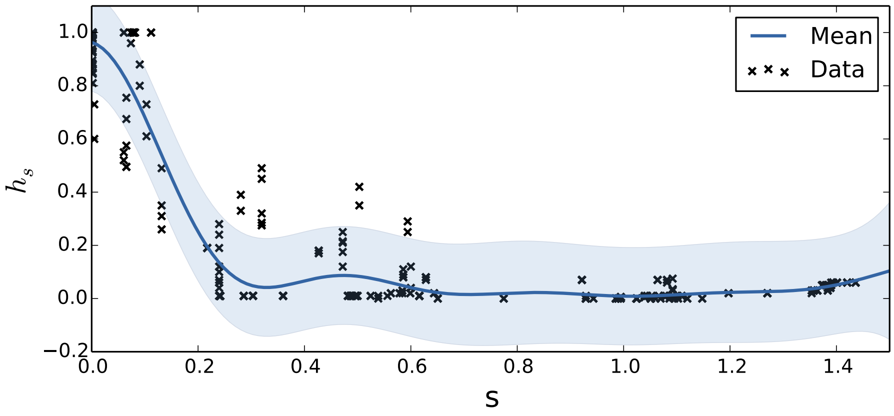


Figure 5.6: Learning $h_s(s)$ from demonstration data in **Exp 1**. The blue envelope around the mean represents the variance of the GP function.

The value of h_s starts to increase from 0 to 1 when s is below 0.2. Its shape is quite similar to the function manually defined in the 2d-examples, see Equation (5.2.8), however no parameters have to be hand-chosen. During runtime, we use the stored GP model to predict the value of h_s given an input s. We additionally enforce the output of the GP to be between 0 and 1 in order to follow the formulation from Section 5.2.3, since the output of the GP may lie outside demonstrated points.

5.3.3 DS coupling and hand DS

Coupled dynamical systems provide an efficient way to model a motion in which two dynamical systems are dependent on each other. The coupling of the two dynamical systems is given by $f_c : \mathbb{R} \rightarrow \mathbb{R}^{N_s}$:

$$\tilde{x}_s = f_c(\phi_{coupling}(x_m)) \tag{5.3.3}$$

And the equation describing the dynamics of the slave system with $f_s : \mathbb{R}^{N_s} \times \mathbb{R}^{N_s} \rightarrow \mathbb{R}^{N_s}$:

$$\dot{x}_s = f_s(x_s - \tilde{x}_s) \tag{5.3.4}$$

with $x_m \in \mathbb{R}^{N_m}$ the master's state, $x_s, \tilde{x}_s \in \mathbb{R}^{N_s}$ the slave's state and attractor, $\phi_{coupling} : \mathbb{R}^{N_m} \rightarrow \mathbb{R}$ the coupling function, and N_m, N_s the dimensions of the master and slave systems. Here the master system is the EMDS controlling the arm motion and the slave system is the hand DS controlling the fingers desired position. The function $\phi_{coupling}$ is designed to keep the hand open while far from the object, and close when approaching the object. The slave DS then allows to generate smooth finger trajectories. This hand configuration is fed to the compliant controller for the fingers described in the following section.

5.3.4 ACTIVE COMPLIANCE ALGORITHM

The active compliance algorithm's goal is to maximize contacts between the robot's fingers and the surface to explore or grasp. This is useful both to speed up the exploration by providing better measurements to the particle filter, and to achieve a stable grasp by enclosing the object. The algorithm takes the current desired hand configuration from the hand DS as an input. This allows the fingers to move in space or along the object's surface to reach the desired hand configuration. We use a variation of the method presented in Chapter 4 by solving the following optimization problem:

$$\min_{\tau} \; w_1 {\Delta\tau_1}^2 + w_2 {\Delta\tau_2}^2 \tag{5.3.5}$$

$$\textit{subject to} \tag{5.3.6}$$

$$\tau_{min} < \tau < \tau_{max}, \tag{5.3.7}$$

$$f_{c_j} < f_{c_{max}}, \qquad \forall j = 1..N_c, \tag{5.3.8}$$

$$J_{c_j} M^{-1}(J_{c_i}^T f_{c_i} - \tau) = 0, \qquad \forall i = 1..N_c, \forall j = 1..N_c \tag{5.3.9}$$

$$\tau = \sum_j J_{c_j}^T f_{c_j} + \tau_1 + \tau_2 \tag{5.3.10}$$

with $\Delta\tau_i = \tau_i - \tau_{iref} \qquad \forall i = 1, 2.$
We try to minimize the desired torques τ_{1ref} and τ_{2ref} for tasks 1 and 2 with associated weights w_1 and w_2 (Equation (5.3.5)).

And $\tau_{min}, \tau_{max} \in \mathbb{R}^{N_s}$ the torques limits, $f_{c_i} \in \mathbb{R}$ the i^{th} contact point's force with corresponding $J_{c_i} \in \mathbb{R}^{1 \times N_s}$ Jacobian matrix, N_c the number of contacts points on the robot. Equation (5.3.7) enforces joint torque limits, Equation (5.3.8) sets a maximum limit on the contact forces, Equation (5.3.9) represents the dynamics of the robot and Equation (5.3.10) links together the sub-variables of the optimization problem.

Task 1 torques are for keeping the desired joint configuration of the fingers x_s provided by the hand DS. They are computed using a PD controller. Task 2 torques are for creating new contacts, given by an impedance controller for each of the desired contacts points. Finally, the torques τ_f resulting from the optimization problem are used to control the robot's fingers.

5.4 Experiment 1: Autonomous Localization and Grasping in simulation

In this section, we present simulations to evaluate the proposed approach. We apply the framework presented in Section 5.3 to a task which consists in localizing and grasping an object in simulation. A reaching behavior is encoded with the original dynamics, consisting of a linear dynamical system. A grasping behavior is encoded by a local modulation of this dynamical system. Two example trajectories can be seen on Figure 5.7: the reaching dynamics lead to a direct trajectory to the object's estimated position, while the grasping behavior always approaches the object from the top. We compare this approach with using LMDS only, i.e. without taking into account the external signal. This leads to the dynamics always following the modulated system, i.e. the grasping approach.

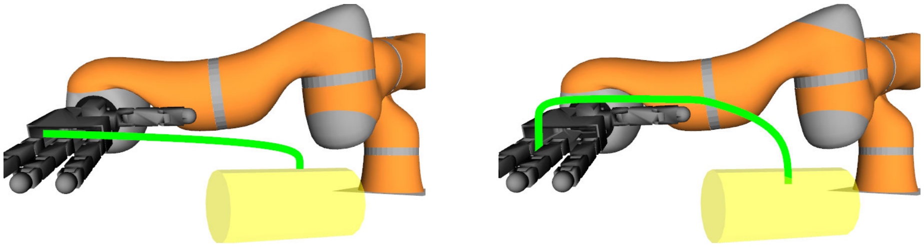

(a) Original dynamics
(b) Modulated dynamics

Figure 5.7: Exp 1: In green, reaching trajectories for the original and modulated dynamics. The modulated dynamics force the trajectory to approach the object from above in order to avoid collisions with the fingers.

5.4.1 EXPERIMENTAL SETUP

The simulation environment is Gazebo (Koenig and Howard, 2004), in which a Kuka LWR robotic arm with 7 degrees of freedom (DOF) and 16-DOFs AllegroHand are controlled in torque, see Figure 5.7.

The particle filter for estimating the object's pose requires the evaluation of the likelihood of a measurement for a potential object pose. This is achieved by generating a virtual measurement for the virtual object pose of each particle. This virtual measurement is generated by a second instance of gazebo, running a copy of the simulated world and keeping the robot's configuration synchronized. For each particle and corresponding object position, the object is moved in the second world and the state of the contacts is updated to compare it with the real simulated world.

We were able to run these measurements at a rate of about 1000 particles per second on one thread with a Core i7 cpu. This step being the bottleneck for the particle filter update rate, our 300-particles filter could run roughly at 3Hz.

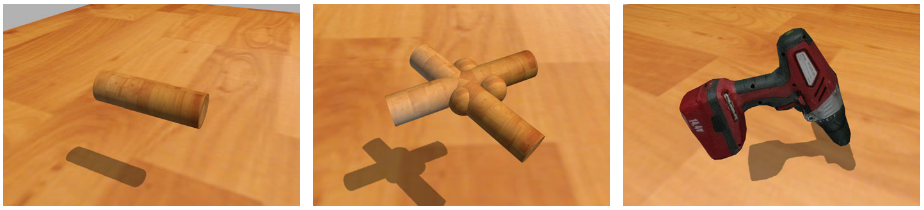
The objects used in this experiments are presented on Figure 5.8, we begin the experiment with a simple cylinder, then a more complex artificial object, non convex, in the shape of a cross composed of cylinders and spheres, and a drill.

For each trial, the object's pose ξ is randomly generated in the simulation environment from a uniform distribution in a plane:

$$\xi \in [-x_l, x_l] \times [-y_l, y_l] \times [-\theta_l, \theta_l]^5. \tag{5.4.1}$$

---

[5] $x_l = 0.1m, \quad y_l = 0.1m, \quad \theta_l = \frac{\pi}{2} rad$



(a) Cylinder (b) Cross (c) Drill

Figure 5.8: The three objects used in **Exp 1**.

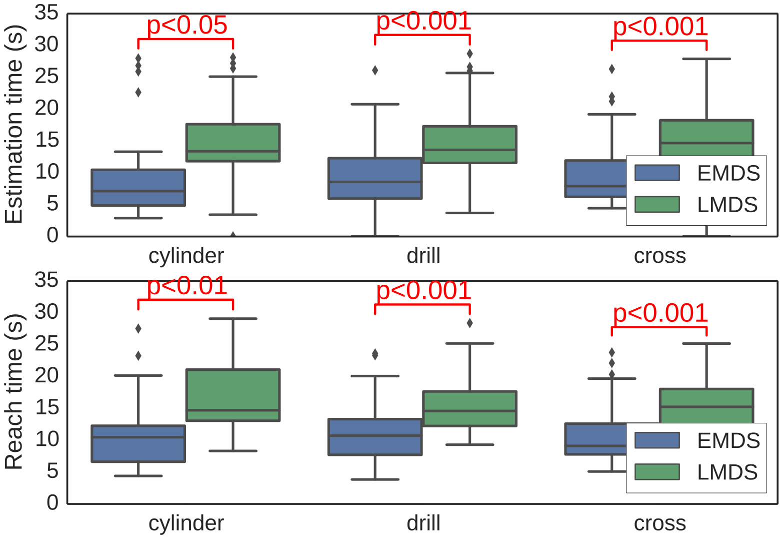


Figure 5.9: **Exp 1:** Results of the experiments: time to estimate and time to reach the object's position using LMDS only or EMDS, for the three objects.

5.4.2 RESULTS

For each tested condition – EMDS or LMDS – and object – cylinder, drill or cross – the trials are carried out 50 times in simulation. Each trial lasts 30 seconds. This represents 150 minutes of simulated experiments. The results are reported on Figure 5.9 with boxplots[6].

We measure both the time to estimate the object's position and to reach to that position, with a threshold of 1.5cm. For each of the explored objects, the EMDS strategy estimates and reaches the real object's position significantly faster than with LMDS. The estimation takes $14.7 \pm 6.2s$ with LMDS, while only $9.6 \pm 5.6s$ with EMDS. It takes a little more to reach the object, with $15.7s \pm 5.0$ for LMDS and $11.0s \pm 4.9$ for LMDS. Because with LMDS, the robot tends to perform a grasping approach even though the object's pose is not known with certainty, it is not surprising that this method takes more time.

---

[6] Boxplots show the median, the interquartile range (IQR: 25^{th} to 75^{th} percentile), as well as 2.5IQR range.

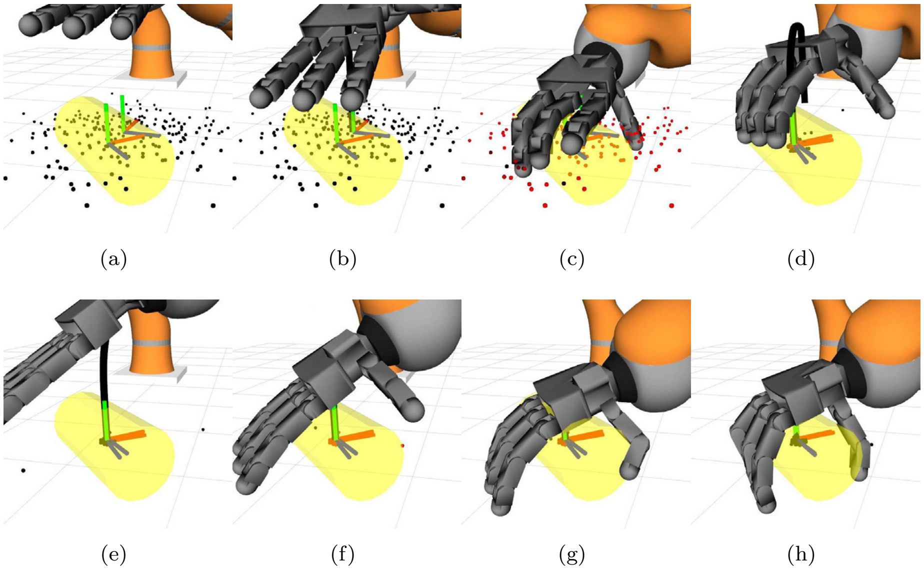


Figure 5.10: Exp 1: Images of the exploration and grasping procedure with the cylinder. On frame (**d**), the hand automatically switches to a grasping motion, and moves up to approach the object from above, until grasping on frame (**h**). The references frames made out of three RGB segments represent the real object's position and the estimated one.

One example of a trajectory for localizing and grasping the cylinder can be seen on Figure 5.10 and a video of multiple trials can be found here: `https://youtu.be/_DdUCsiTn0E`. The particles are represented as red or black dots[7], the color representing their current respective weight. The current object's real and estimated positions are represented by RGB frames. The estimated trajectory given the current external signal can also be seen as a thick black line. The corresponding evolution of the external signal s, activation function $h_s(s)$ and the altitude of the hand during the exploration are given in Figure 5.11. On frames (**a**)-(**c**), the hand approaches the current estimated pose of the object directly, until it touches the object, the number of contacts increases, and the object's current estimated position is updated. On frame (**d**), the particles have all gathered at the real object's position, hence the variance of the object's estimated position distribution decreases and h_s increases. Therefore, the dynamical system automatically adapts to a grasping motion, the altitude increases and the the hand approaches the object from above until grasping it on frames (**g**)-(**h**).

---

[7] The orientation of the particles cannot be seen here.

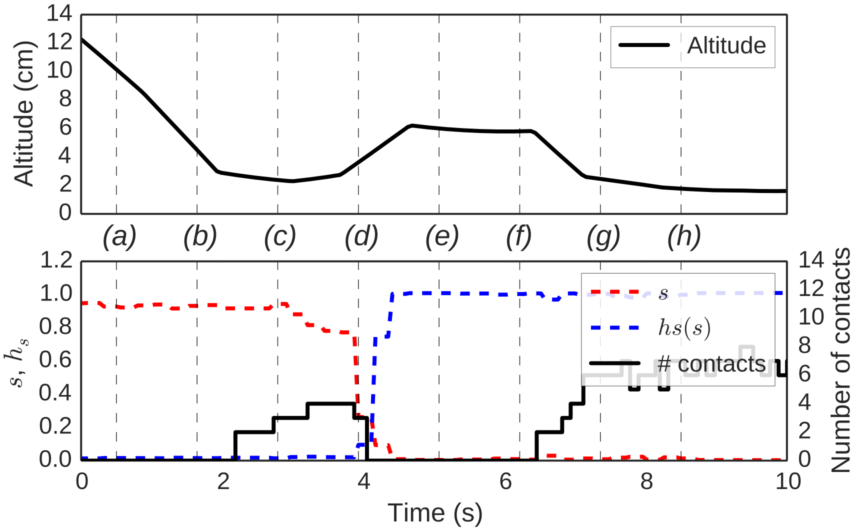


Figure 5.11: Exp 1: Evolution of the external signal s, activation function $h_s(s)$ and the altitude of the hand during one exploration trial. The labels (a)-(h) correspond to the steps of exploration in Figure 5.10.

5.5 Experiment 2: Reaching while avoiding obstacles

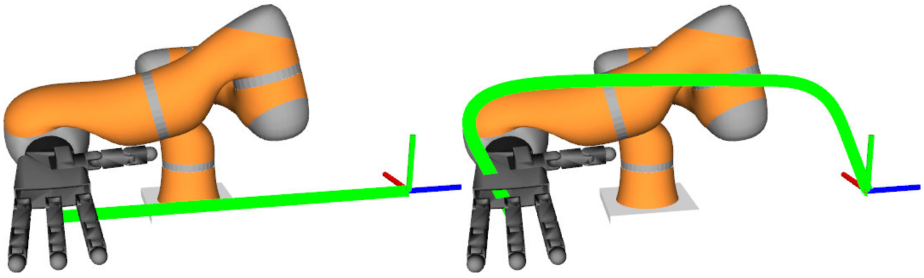
In this section, we present another application of our algorithm in which a task of reaching while avoiding obstacles is encoded. The task consists in going from point A to point B for the robot end-effector with the desired dynamics, while there are obstacles with unknown position on the path.

We present two experiments: in the first experiment, the avoidance behavior is achieved by going over the obstacles. This is demonstrated both in simulation and on the real robot. In the second experiment, the robot's avoiding behavior depends on characteristics of the collision: it takes a different trajectory depending on the detected angle of the collision with the object. The second experiment is done on the robot.

5.5.1 EXPERIMENT 2A: AVOIDING OBSTACLES

The original dynamics used in **Exp 2a** are simple linear dynamics – reaching in a straight line – while the modulated dynamics correspond to a maneuver to avoid obstacles. Instead of trying to reach through obstacles, the modified dynamics encode trajectories for which the end-effector moves back and goes over them, thus doing a detour while still reaching for the target at point B (see Figure 5.12).

This strategy allows to reach directly to the target while only taking a detour when necessary.



(a) Original dynamics (b) Modulated dynamics

Figure 5.12: Exp 2a: In green, reaching and avoiding trajectories corresponding to the original and modulated dynamics. The RGB frame on the right corresponds to the target. The modulated dynamics force the end-effector to go above obstacles and avoid collision with it.

The start and end points are $60cm$ away from each other on the y axis. Obstacles are placed on the path, see Figures 5.16, 5.18 and 5.19.

Similarly as in **Exp 1**, the function $h_s(s)$ mapping the external signal to the activation of the modulation is learned from demonstrations. Because the robot cannot stay in contact when there is a collision, and the system does not have a memory of the last contact, we choose to encode as the external signal s the time since the last collision[8]. This allows the system to learn how to avoid obstacles given the memory of the last time a collision occurred.

Learning the activation function from GUI demonstrations

During the execution of the task **in simulation**, a teacher specifies continuous values for the modulation signal $h_s(s)$ between 0 (reaching directly) and 1 (reaching indirectly with the avoiding maneuver). The learned function $h_s(s)$ and the training points can be seen on Figure 5.13.

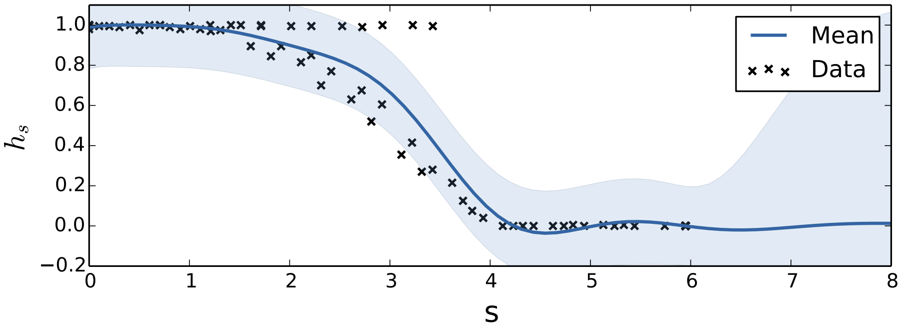


Figure 5.13: Exp 2a simulation: Learning $h_s(s)$ from demonstration data. The blue envelope around the mean represents the variance of the GP function. The external signal s corresponds to the time since the last collision.

The learned modulation function corresponds to choosing the avoiding dy-

---

[8] When no collision has yet occurred, this value is set arbitrarily high.

namics ($h_s(s) = 1$) when a contact occurred less than 2 seconds ago, and slowly switching to the original linear dynamics until 4 seconds after a contact. Then, the system converges again in a straight line towards the target ($h_s(s) = 0$).

Learning the activation function from real robot demonstrations

On the real robot, we make an attempt at learning the function $h_s(s)$ from demonstrations, by back-driving the robot arm. In simulation, there is no practical way to let a human guide or tele-operate a robot easily while providing feedback, for instance about collisions. Yet, it is crucial for the user to be aware of the external signals on which depends the activation of the modulation.

On the real robot, we can however let a human perform demonstrations by back-driving the robot. The objective is to define a mapping from the external signal s, chosen in this experiment as the time since last contact, to an activation value h_s. However, this value is only implicitly given by the user through the demonstrated trajectories. Hence, at each timestep, we need to find the value h_s which produces the demonstrated velocity $\dot{x}_s$ closest to the demonstrated velocity $\dot{x}$.

One way is to look for an inverse mapping of the EMDS. Let's go back to the original formulation:

$$\dot{x} = M(x, s)f(x) \tag{5.5.1}$$

Because s is always expressed through the activation value h_s, as in Equation (5.2.4) for instance, we can rewrite the previous equation as:

$$\dot{x} = M(x, h_s)f(x) \tag{5.5.2}$$

Hence, for a fixed x, the whole DS can also be expressed as a function G_x specific to that x:

$$\dot{x} = G_x(h_s) \tag{5.5.3}$$

If the function G_x is injective, there exists an inverse function G_x^{-1}:

$$h_s = G_x^{-1}(\dot{x}) \tag{5.5.4}$$

We know with certainty that the function G_x is not injective for all x. For instance at $x = 0$, $f(x)$ is 0 since the original dynamics is stable at the origin, hence G_x cannot be injective. This is also the case for regions where the original DS is not modulated, i.e. the function $M(x, s)$ is constant for all s.

Practically, h_s does not influence the dynamics in these regions, therefore our estimate is not important there. Instead of computing a closed-form inverse function, we estimate the value of h_s at each timestep by performing a line-search in the input space of $G_x(h_s)$, and search for the closest output:

$$h_s = \underset{h_{s\,i} \in \{0,\ldots,1\}}{\operatorname{argmin}} \|G_x(h_{s\,i}) - \dot{x}\| \tag{5.5.5}$$

We perform the line-search with values from 0 to 1 by increments of 0.1, and interpolate between the two closest values.

Finally, we learn the function $h_s(s)$ in the same way as in the method demonstrated previously, using Gaussian Process Regression. The result can be seen on Figure 5.14.

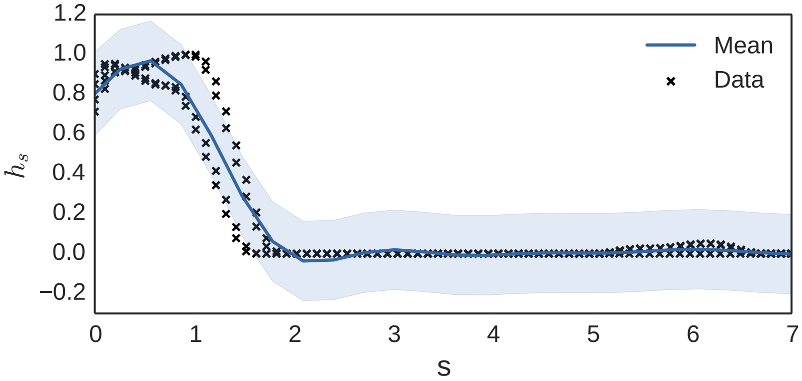


Figure 5.14: Exp 2 real robot: Learning $h_s(s)$ from demonstrated trajectories on the real robot. The blue envelope around the mean represents the variance of the GP function. The external signal s corresponds to the time since the last collision.

In comparison with the function learned in the simulation, the activation function goes back to zero only after 2 seconds, instead of 4 seconds. The slope of the decrease is also much more abrupt. This is probably due to the different method of teaching: while in the first case teaching is done through a graphical interface with a slider, in this case the user directly back-drives the robot arm.

RESULTS, EXPERIMENT 2A

In simulation

The progression of the task can be seen on Figure 5.16. A video of the experiment can be seen here: `https://youtu.be/gfL7aqogU0k`. The corresponding evolution of the external signal s, predicted modulation $h_s(s)$, altitude of the hand, progression towards the target, and collision status are given on Figure 5.15. The predicted modulation $h_s(s)$ directly depends on s, following the learning process described above. The altitude of the hand depends mostly on the modulation or not of the dynamics: if the activation function is high, the hand follows the avoidance dynamics and altitude increases. The progression towards the target is the y coordinate normalized between 0 (start point) and

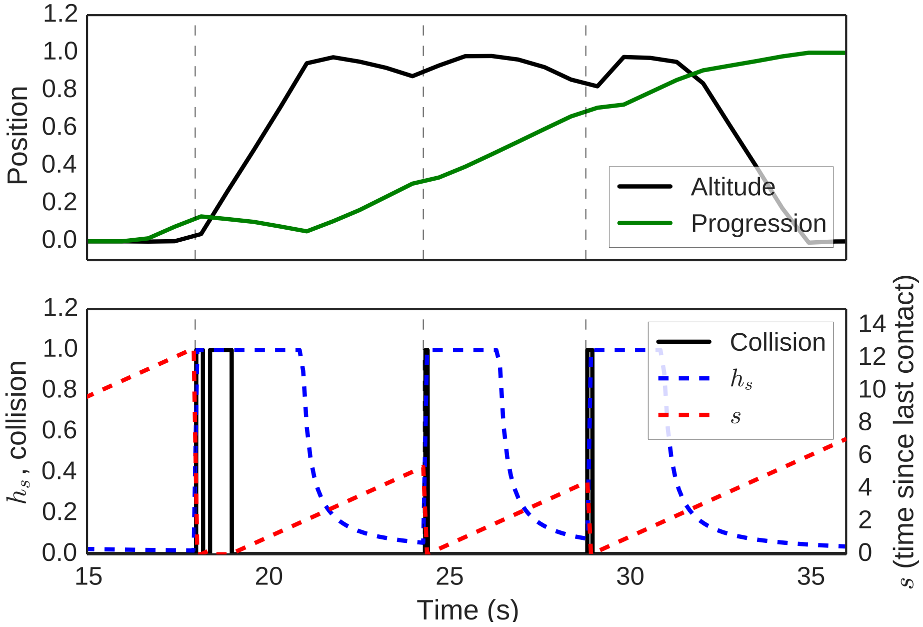


Figure 5.15: Exp 2a simulation. Top: Evolution of the altitude of the hand (normalized) and the progression of the task (horizontal position between start and end points, normalized between 0 and 1). **Bottom:** Evolution of the external signal s (time since last collision), activation function $h_s(s)$, and the collision status. The vertical dotted lines correspond to collisions with obstacles, see Figure 5.16.

1 (target point). The collision status is 1 if the number of contacts is higher or equal than 1, 0 otherwise. Objects from the first experiment (see Figure 5.8), namely the drill and the cylinder, are used as obstacles.

The hand starts moving right in a direct line towards the target (to the right of the cylinder), until it collides with the drill on frame (b). It then moves up and backwards according to the modulated dynamics (see Figure 5.12(b)) until it reaches the height encoded in the modified dynamics; it then moves right at the same z-coordinate. After a few seconds without contact, the dynamics slowly changes back to the original DS: the hand starts moving downwards towards the target frame. It collides again on frame (f), and the same process repeats itself twice until the end-effector reaches the target on frame (i).

The change between the two dynamics can be seen through the evolution of several variables on Figure 5.15. When a contact occurs, s is reset to 0, and $h_s(s)$ goes to 1. Simultaneously, the altitude starts increasing and the progression of the task starts regressing, as the avoidance behaviour is triggered. Then, as the time since the last contact increases and goes over 2 seconds, $h_s(s)$ goes back to 0 and the altitude decreases: the end-effector tries to reach for the object directly again.

For comparison, we ran the same experiment without the learned function $h_s(s)$, instead fixing the value of h_s. We either set the h_s value to 1 (always take

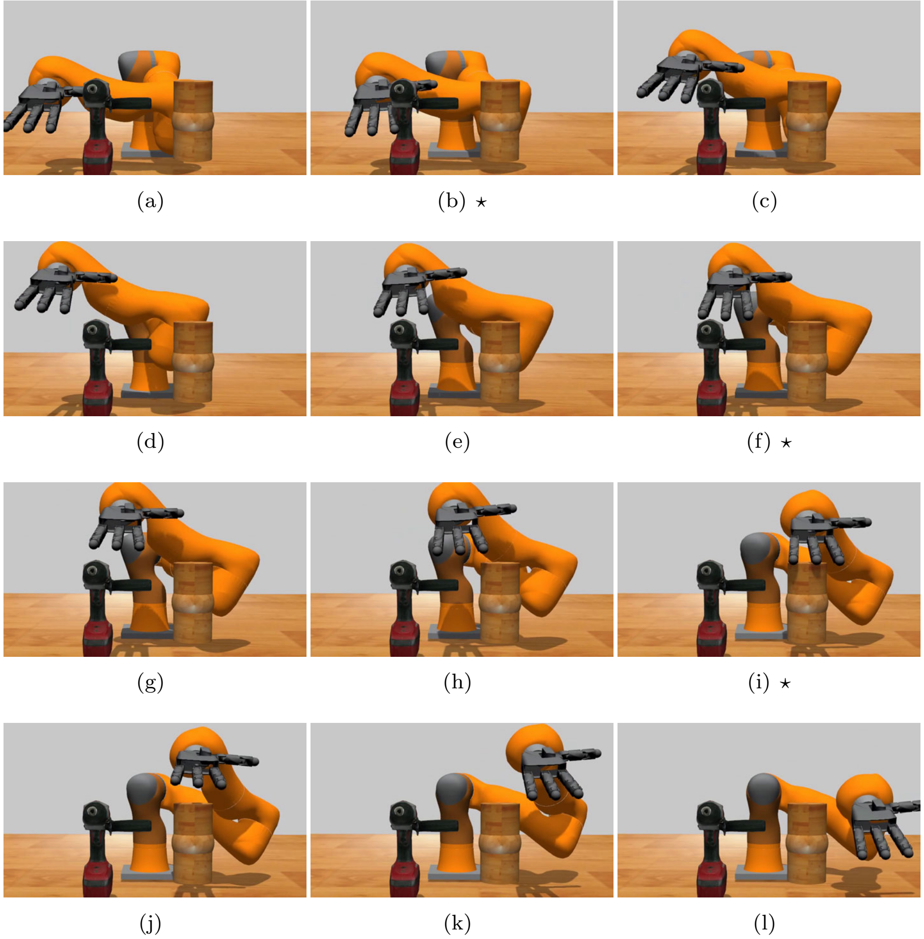
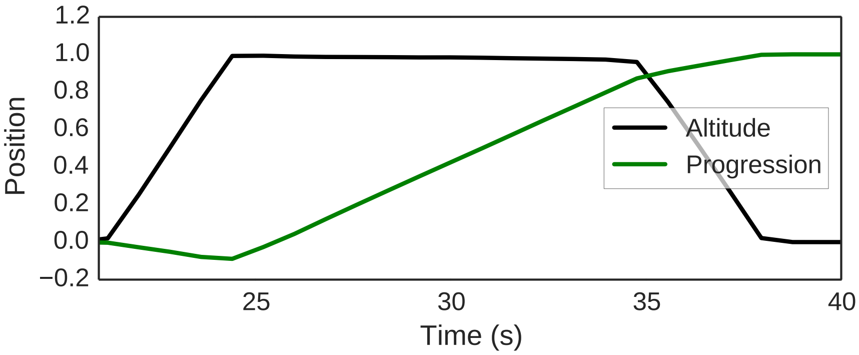
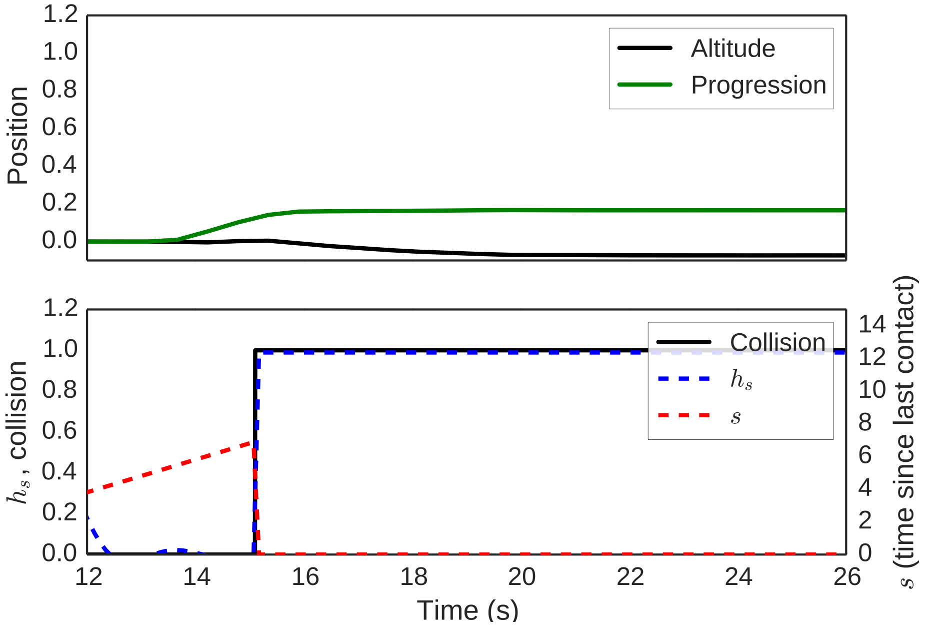

(a) (b) ⋆ (c) (d) (e) (f) ⋆ (g) (h) (i) ⋆ (j) (k) (l)

Figure 5.16: Exp 2a simulation: Progression of the experiment in Gazebo. Collisions occur on frames (b), (f) and (i). Corresponding evolution of the variables on Figure 5.15.

the avoiding trajectory) or to 0 (always reach directly). Unsurprisingly, the first one leads to the end-effector going above the obstacles without even touching them, but doing a unnecessarily large detour, see Figure 5.17(a). The trajectory starts by going away from the target until the hand reaches the desired height, as encoded in the avoiding DS. In the second condition ($h_s(s) = 0$), the end-effector gets stuck in contact with the object after the collision, see Figure 5.17(b), as no avoidance maneuver is triggered. This shows that the learned activation function improves the execution of the task.



(a) The activation function $h_s(s)$ set to 1 (always avoiding)



(b) The activation function $h_s(s)$ set to 0 (never avoiding)

Figure 5.17: Exp 2a simulation: Reaching while avoiding obstacles with h_s forced to 1 (a) or 0 (b). Collisions are not displayed for (a) since none occur. On (b), the progression gets stuck at second 16 when the hand enters in collision with the first obstacle and gets stuck. It cannot progress further and thus the progression does not increase anymore.

On the real robot

For the experiment on a real platform, we use a KUKA LWR robot with 7 DOFs with a force-torque sensor mounted at the end of the arm. We add a probe-like end-effector after the sensor to be the contact point during collisions. The force-torque sensor is used to detect collisions[9].

We ran the same experiment on the real robot. We used objects from different sizes as obstacles. The images of the experiments can be seen on Figures 5.18 and 5.19. A video of the experiment can be seen here: `https://youtu.be/scvDiqfETRc`.

In the first one, the end-effector collides with the first obstacle, then with the second obstacle. In the second one, the end-effector collides first with the large box, then again on top of the large box.

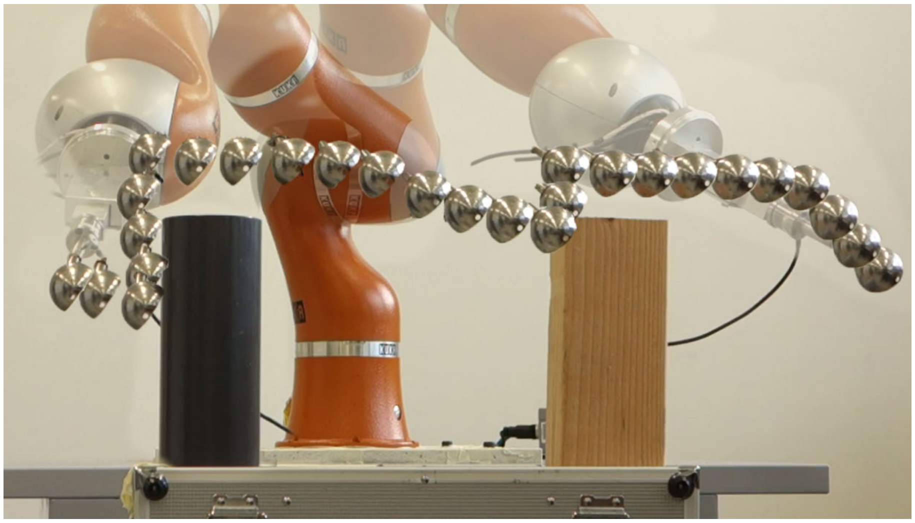


Figure 5.18: Exp 2a real robot: Trajectory of the end-effector during the task with two obstacles. The end-effector collides with the two objects.

---

[9] We simply threshold the norm of the force vector to detect a collision. The threshold is set to $2.0N$

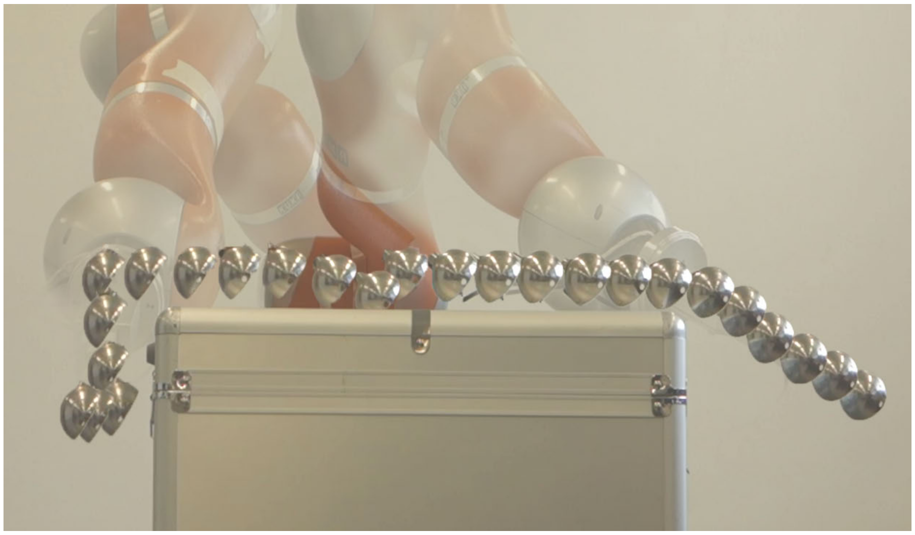


Figure 5.19: Exp 2a real robot: Trajectory of the end-effector during the task with one large obstacle. The end-effector collides on the side of the object and another time on the top.

5.5.2 Experiment 2b: Navigating between obstacles

In this experiment, we increase the complexity of the external modulation by taking into account two variables: the time since last contact, and the angle of the last contact[10]. We aim at learning how to avoid obstacles depending on information from the collision, here the force direction during contact.

For this purpose, we encode the original dynamics and the modulated dynamics as two opposite velocity fields in a central region, where the experiment is taking place. Both dynamics converge to the target when going far enough from that region. The first one is directed perpendicular to the direction between initial and target frames, in a horizontal plane. The second one is directed in the opposite direction, see Figure 5.20.

A whole range of dynamics is reachable by changing the activation of the modulation. For instance, by setting the activation to 0.5, the resulting trajectory is a straight line, see Figure 5.21. The angle of the deviation can be adjusted by modifying the activation value between 0 and 1.

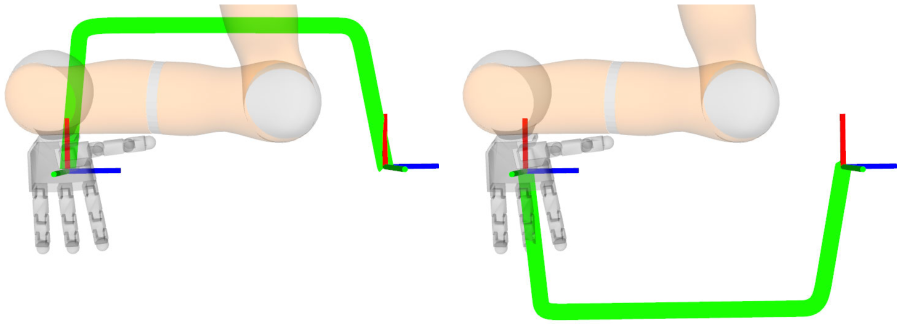
Learning the activation function

In order to learn the mapping $h_s(s)$, we perform demonstrations on the robot in a similar way as presented in the previous experiment. Because the input variable s is now two-dimensional (time since last contact and angle of last contact), more demonstrations must be given, spanning the whole input space. For this purpose, we perform 8 demonstrations with different collision angles.

The behaviour taught to the robot is the following. When no collision occur,

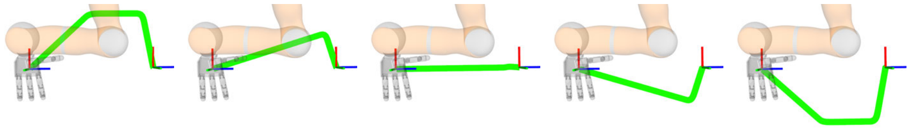
---

[10] Since all of our motion is taking place on a plane, the angle is taken between the vector of the force as measured by the force-torque sensor, and the direction from starting point and target point. In practice, we compute it using $atan2(f_x, f_y)$, f being the force vector.



(a) Original dynamics (b) Modulated dynamics

Figure 5.20: Exp 2b: Seen from above, in green, trajectories corresponding to the original and modulated dynamics. The RGB frames on the left and right correspond respectively to the starting point and the target.



(a) $h_s = 0.25$ (b) $h_s = 0.40$ (c) $h_s = 0.50$ (d) $h_s = 0.60$ (e) $h_s = 0.75$

Figure 5.21: Exp 2b: The resulting dynamics with different levels of activation. With $h_s = 0.50$, the dynamics reach in a straight line.

the robot moves in a straight line, i.e. $h_s(s) = 0.5$. After a collision, the end-effector adjusts its trajectory depending on the collision angle. If the angle is small (see Figure 5.22 left), the robot does a large detour, hence picks an extreme value of the activation function (0 or 1 depending on the direction of avoidance). If the angle is large (see Figure 5.22 right), the robot only slightly adjusts its trajectory.

Demonstration data are plotted on Figure 5.23. Each horizontal line of datapoints corresponds to a demonstration. We can see that the input space corresponding to the angle of collision is not perfectly spanned by the demonstrations, due to the demonstrated collision angles not being spread perfectly evenly. Looking at the horizontal axis, we can see that we recorded about 5 seconds of data after a collision per demonstration. The color of each datapoint indicates its corresponding h_s value, estimated using the method described in Section 5.5.1. We then learn a model with GPR, illustrated on Figure 5.24.

From these plots, we can extract a few observations:

1. The collision angles are not centered on 0. The median value, for which the output $h_s(s)$ switches from values below 0.5 to values above, is at about $0.4 rad$. It corresponds to a frontal collision. This is due to a mis-calibration of the force-torque sensor's orientation. Thanks to the learned mapping, this is not an issue.

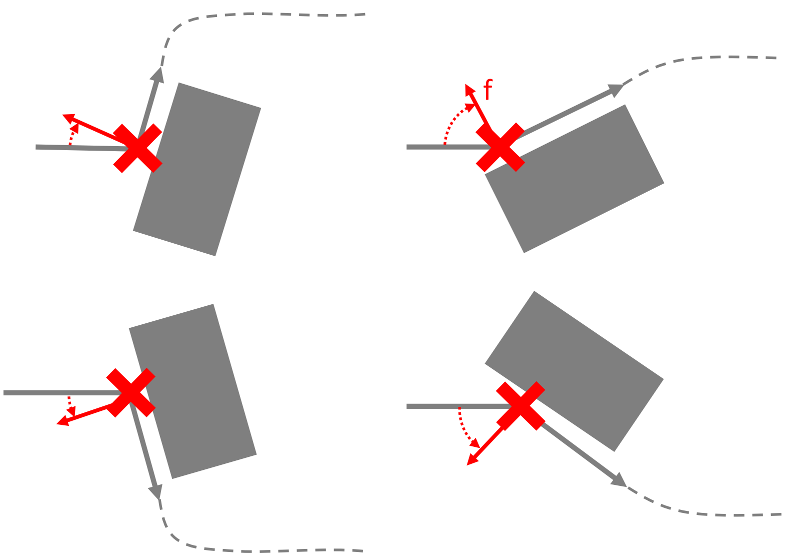


Figure 5.22: Exp 2b: Schematic of the demonstrated trajectories. In red, the collision point and the force sensed during the collision. The end-effector follows the shape of the objects after contact, then continues again in a straight line after a few seconds.

2. After a few seconds, all datapoints converge back to a value of $h_s(s) = 0.5$, i.e. a straight line. This is the desired behaviour.

3. A few "wrong" datapoints are demonstrated: on the top right of Figure 5.23, there are some "red" points ($h_s(s) > 0.5$), and on the bottom right, a few "blue" points($h_s(s) < 0.5$). However, the value of h_s for those points is close to 0.5. This is inherent to the demonstration method, back-driving the robot: it is sensitive to human demonstration errors.

4. The learned mapping, visible on Figure 5.24, corresponds to the desired behaviour: small angles (close to the median value of 0.4) yield extreme values of h_s, i.e. trajectories close to either one of the original or the modulated dynamics (see Figure 5.20). Bigger angles yield less extreme values, leading to less modified trajectories.

5. Because the learned function is highly non-linear, especially close to the median value, the kernel width of the Gaussian Process Regression is tricky to choose. A too high value does not allow the rapid function change close to the median angle value. A too low value overfits the data and removes the ability to generalize over collision angles not demonstrated (e.g. at $-0.5rad$). We chose a kernel width of 0.5[11].

---

[11] Because the GP's output falls back to 0 when far away from training data, we also first center our output data before learning the GP, so that its range lies between −0.5 and 0.5, instead of 0 and 1. This way, the DS moves in a straight line ($h_s(s) = 0.5$) when far away from demonstration data.

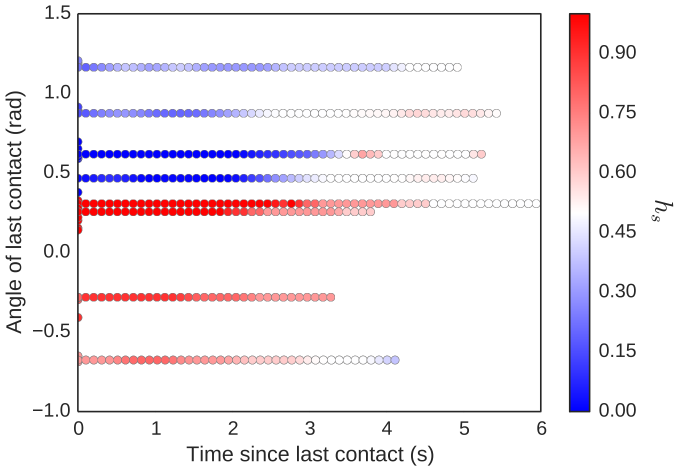


Figure 5.23: Exp 2b: Data from the demonstrations. Each horizontal line corresponds to one demonstration.

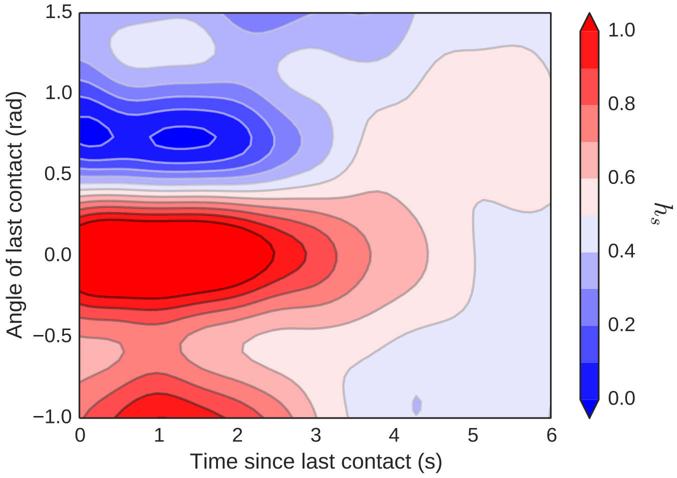
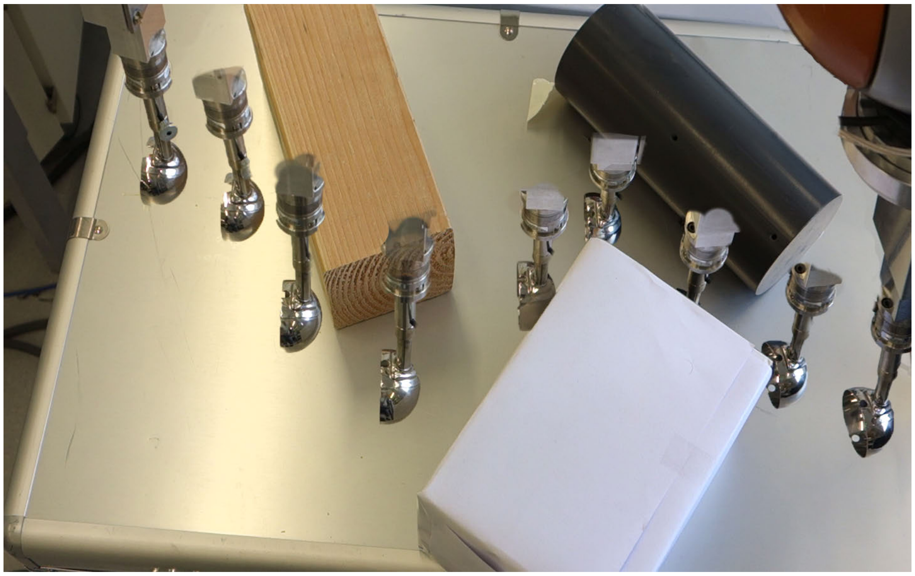


Figure 5.24: Exp 2b: Learned function $h_s(s)$, using Gaussian Process Regression (GPR).

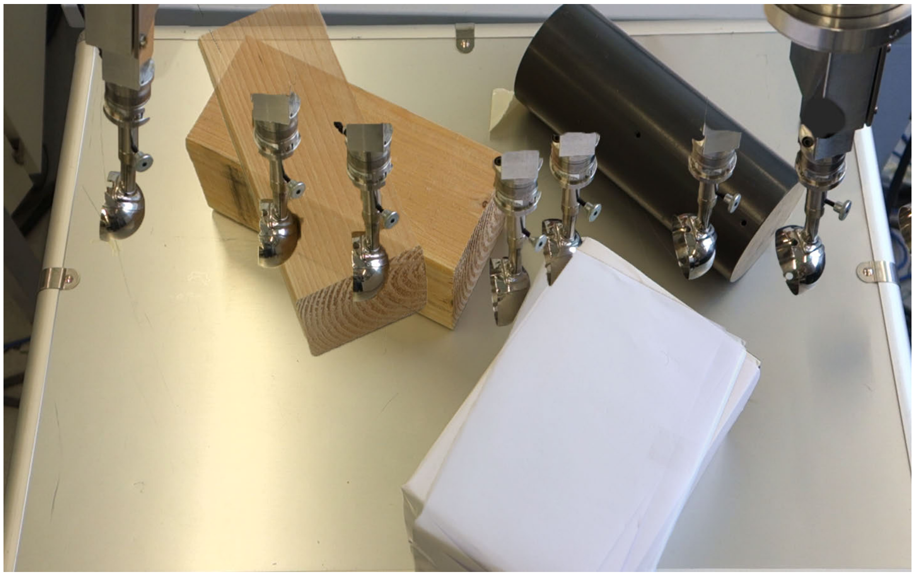
Results

Images from the experiment can be seen on Figure 5.25. The task is executed both activating the modulation depending on the input s, or fixing the value of the activation to 0.5, hence ignoring external signals for a comparison purpose. When the external signal is ignored, the end-effector moves in a straight line. Because of the passive DS controller, the robot is compliant. However, when ignoring the external signal, here the contact information, the robot does not avoid the obstacles. The friction in the joints prevents it to be really deviated from its trajectory and the robot displaces the obstacles while colliding with them.

When using information from the external signal, the robot adapts its trajectory after each collision and navigates between the obstacles. Depending on the collision angle, the robot adapts the avoidance trajectory. Therefore, it sometimes slides along the object (here with the second obstacle), or moves away from it (first and third obstacles). The video of these experiments can be found here: `https://youtu.be/Aiz3dUADcbw`.



(a) $h_s(s)$ learned from demonstrations



(b) $hs(s) = 0.5$ (fixed)

Figure 5.25: **Exp 2b:** Evolution of the obstacle avoidance task with the controller activated (a) or not (b). The end-effector reaches from left to right. In both cases, the desired velocity is tracked using a passive DS controller. In the second example, the objects are moved during the collisions.

5.6 Discussion

In this chapter we proposed a framework to modulate a certain type of dynamical systems depending on external signals. Desired behavior is achieved by modifying local modulations applied to an existing dynamical system. This allows to conserve important stability properties for any external signal, assuming that the modulation function is full-rank. We have proposed one way to design the modulation function based on regulating a rotational and speed-scaling modulation, inspired by the type of modulation presented in previous work from our team. This design ensures that the modulation matrix is always full-rank, and hence the system is stable for any external signal. As such, this work can be directly applied to an existing LMDS, provided a mapping between an external signal and the desired regulation of the modulation.

For this purpose, we also suggested a method to capture how the dynamical system should be modulated by the external signal, based on learning the corresponding mapping by teaching the desired behavior during task execution. We tested this teaching method by demonstrating both through a graphical interface and through physical demonstrations on the real robot. We applied this algorithm to a task of simulated blind reach-and-grasp, using only tactile input for estimating the object's pose. In this task, the modulation between the reaching and grasping behaviors was encoded as a learned function of the variance of the pose estimate. The regulation of the modulation allowed to find and grasp the object faster than when always modulating the DS and following the grasping motion.

We also applied this algorithm to tasks of reaching while avoiding obstacles. The system learns when to bypass obstacles depending on the last contact. We show that the task execution depends on learning a proper activation function, otherwise the behaviour is inadequate. The learned function depends on the time since the last contact, and thus depends implicitly on the size and shape of the obstacles seen during the task. The robot would perform less well with different obstacles as it would either collide again before bypassing (bigger obstacles), or make unnecessarily big detours (smaller obstacles). The teaching hence depends on the type of obstacles met during a specific type of task. We further studied this task by introducing the angle of collision into our activation mapping. With this two-dimensional external signal, our robot is able to navigate between obstacles and choose its trajectory by adjusting the level of activation of the DS's modulation, depending on the external signals.

In this work, the mapping from external signal to modulation is learned by teaching. We specify that the function can also be provided manually. An alternative to learning this function by a teacher would be to use reinforcement learning (Sutton and Barto, 1998) (RL) because the dimension of the problem is low and hence the problem fits particularly well the RL framework.

Additionally, the input in the first experiment (variance of the object's pose estimate) of this learned function could be multidimensional, i.e. use all the terms of the covariance matrix and the object type. This could allow to modulate the reaching for grasping trajectory object-wise and depending on the direction of the object's pose uncertainty. Indeed, it makes sense to approach objects differently whether the position's uncertainty is along their grasping axis or along a different direction.

Chapter 6

CONCLUSIONS

In this final chapter I highlight the main contributions, limitations and possible future directions of the work presented in this thesis, as well as personal insights.

6.1 Main Contributions

This thesis addressed the need for control algorithms and strategies when robots make multiple contacts with their environment, especially in tasks of exploration and grasping. This thesis leveraged the recent availability of commercial tactile sensors to cover robots with pressure-sensitive skin and develop new control algorithms.

First, we presented multiple scenarios in which tactile sensing is used to compliantly and continuously follow surface's contours while gathering tactile information, in order to identify the surface in contact. We showed that partial tactile data is sufficient to correctly classify human-like faces. We presented a bimanual exploration strategy that does not require planning and allows fast exploration of objects. We applied this method to what is to the best of our knowledge the first example of bimanual haptic exploration with a robot. We also believe that this is the first use of tactile sensors on parts of robot fingers other than the fingertips during continuous tactile exploration, which has the advantage of speeding up the exploration by collecting more tactile data simultaneously.

Second, we focused on how to create and keep multiple contacts on robotic fingers during haptic exploration. We presented a strategy to generate additional contacts, using information from other current tactile contact, namely position and normal of contacts. We developed a computationally efficient algorithm to compute control torques for moving in the null-space of existing contacts, putting constraints on maximum contact forces. We also demonstrated benefits of these methods for grasping tasks, showing that we obtain more contacts and thus better stability than by simple using enclosing strategies, given unknown objects to make contact with.

Finally, we presented the Externally Modulated Dynamical Systems (EMDS) algorithm, to take into account external signals beside time to modulate DS. This method is based on activating or not local modulations of the DS in order to change the dynamics while guaranteeing important stability properties. Along with EMDS, we presented an interactive learning method for capturing how the DS should be modulated by the external signal, using demonstrations of the task. We demonstrated the usefulness of this algorithm in a challenging blind reach-and-grasp task, using only tactile input for estimating the state of the object. We also tested this method with a more complex two-dimensional external signal describing collisions with the end-effector, which modulates the trajectory of the robot to navigate between obstacles on a real robotic platform.

6.2 Limitations and Future Work

HUMAN-LIKE FACE RECOGNITION AND BIMANUAL EXPLORATION

The exploration strategies presented in Chapter 3 depend on the type of object or surface to explore. In particular, the algorithm for the exploration of human-like faces is designed for flat surfaces such as faces, and it is limited in its workspace by the length of the robot's arm. Reaching for the side or back of a head to find more features to explore would for instance not be feasible. Because the identification method (HMM) is based on a sequence of exploration signals, the trajectories must be roughly comparable with each other in length. This restricts the exploration to predefined trajectories. Changing identification method would allow to relax constraints on the exploration trajectory, but should be robust to the noise inherent from the inaccurate kinematics of the robot.

In comparison, the bimanual exploration is based on 3D point-cloud reconstruction, hence the identification does not depend on the exploration trajectory. However, the fast trajectory generation for both arms requires objects to have one principal axis along which to perform the exploration, and of relatively small size in order to fit inside a robot's hands. We believe that the latter issue is inherent to bimanual manipulation settings, however the constraint on the shape of the object could only be released at the cost of a much more complex algorithm. Indeed, since the shape of the object is unknown at first, contacts should be allowed on all parts of the robot, especially the wrists and sides of the fingers. This is not yet possible on existing platforms. Furthermore, with more complex shapes, planning would probably be required to generate collision-free trajectories. Because planning takes time – notably with two arms – and re-planning should occur as long as new tactile information is gathered, it would requires a very slow exploration-planning iterative procedure.

One potential solution would be to plan only for the re-positioning of the object (motion of the arm holding the object), and let the other arm perform exploration using our multi-contact controller presented in Chapter 4.

One of the main obstacles encountered while working with our multi-contact exploration approach is the existence of local minima for the exploration motion. This is due to the fact that the shape of the object is only roughly known, and the exploration is sequentially directed towards *key frames* spread around the estimated shape. Although we did not tackle the high-level planning aspect of exploration, we briefly tried to merge our exploration strategy with a planner[1], to generate trajectories given the continuously updated point-cloud gathered from tactile information. This first problem we encountered is that because planning takes time, and the model is continuously updated, the plans are already irrelevant once available (the starting point has changed, as well as the world model). The second problem comes from the fact that the planner does not necessarily produces two similar consecutive plans, hence a new plan can contradict the previous one (e.g. go around a object one way or the other way.), and render the exploration process extremely inefficient. Both these issues could find a solution with continuous motion planning approaches (Steffens et al., 2016), although current methods are not fast enough for continuous exploration. Another issue comes from compromising collision avoidance and exploration: the parts of the robot equipped with tactile sensors (particularly the fingers) must be kept close to the estimated surface of the object to explore. This could be incorporated into the objective function of the planning algorithm (if there is one), although further increasing the planing time.

In the current version of the presented algorithm, we assume one contact per link, hence one constraint when computing null-space torques. In case of multiple contacts on one link of the robot, we would need to be able to discriminate between multiple contacts and one large contact. Indeed, if there are multiple contacts, each should generate a constraint, or be clustered together to generate for instance one planar constraint if the points are aligned (hence three constraints, one position and two rotations). One solution would be to perform clustering on the tactile sensors' signals for each link, and decide how many clusters exist, hence how many constraints.

In our grasping application, the list of *desired contacts* points on the robot's fingers is provided by another algorithm (de Souza, 2016), depending on the desired type of grasp. For our exploration experiments, we manually defined the *desired contact points* as points on each of the links of the robot's fingers. However, some of these may never be able to be in contact, hence creating useless secondary tasks, which may decrease the chances of other secondary tasks to succeed. One could try to teach which *desired points* should be activated depending on the type of surface to explore.

Finally, null-space torque control is sensitive to the dynamic model of the robot. Because all robots have friction in the joints, especially robotics hands,

---

[1] This is not part of the thesis. There is however a reference to our attempt in the conclusion of Chapter 4.

this model is never perfect. This may lead to constraints not being respected, depending on the amplitude of the errors in the model. In practice, the contact forces do not become high, even on the real robot, because friction is low compared to contact forces. However, one could formulate our null-space computation as a robust optimization problem to take this into account.

EMDS

One inherent limitation of EMDS is the requirement that the second DS must be expressed as a local modulation of the first one. When using a rotational modulation, this can be problematic if the two DS are strictly opposed. This can produce some computational instability since one can find several rotations that transform one DS into the other. In practice, the two DS are rarely exactly opposite, and in that case, it is possible to define a favorite axis of rotation in that case.

We suggested to generate the activation function $h_s(s)$ from demonstrations, however giving demonstrations in a high-dimensional space is cumbersome. It requires more demonstrations to cover the whole space (especially when learning the mapping with GPR). Because of that, the more demonstrations, the higher the chances that some of them contain mistakes from human error. An alternative would be to use reinforcement learning (RL) to obtain an optimal mapping. Since the modulation function is low-dimensional, RL is particularly suited.

6.3 Final Words

During my PhD, I have spent a long time working with tactile sensors and robots. I have been gathering insights into the current limitations and the promising uses of artificial tactile sensing for robotics, which I try to list below:

- Tactile data's **high dimensionality** is both a blessing for extracting a lot of information and a burden when using tactile sensing in a control system. Many dimensions make it difficult to use in learning frameworks, especially with learning from demonstrations since many demonstrations should be given to span the whole space. On the other hand, the high dimensionality of tactile signals provide a lot of information to be extracted, and could for instance be very useful to communicate with the robot, for instance during teaching.

- **Tactile sensors** are **difficult to work with** on real robots. Indeed, sensors as efficient as human skin are very difficult to produce. The existing designs do not yet offer full coverage, they often suffer from drift and are not able to detect very light contacts.

A potential improvement could come from designing the tactile sensors along with the robotic hands' design. The fingertip sensors of the iCub humanoid robot are a good example of such a strategy, but even though iCub arms and torso are now also covered with artificial skin, that is not the case of the side of the hands and most of the fingers, where contacts mostly occur. If a robot is expected to make contact with unknown objects, the whole surface should be able to detect contacts.

- Commonly used **Software for processing tactile sensing** do not exist yet. Although there are many different types of tactile sensors, there are many common operations required to process the data. For instance, tactile sensors are usually composed of multiple taxels, each outputting a pressure value. The combination of these values along with the geometrical information of these taxels provide a way to know the location, force and distribution of the contact points. A push to towards a unification of methods could be useful for the community using tactile sensors .

- The combination of **tactile sensors** and **compliant control** will enable robots to really **be in contact with the environment** without requiring perfect a world model. That is also the way to safely physically interact with humans. The example presented at the end of Chapter 5 using a passive DS control for the compliance and sensors to detect collisions illustrates well that a robot can interact with its environment, be perturbed and keep performing its task.

Appendices

Appendix A

Appendices for Chapter 4

A.1 Exp 1: Details of the control

This appendix describes the controller used in experiment 1 for surface exploration.

Median of normal of contact and attached frame

The median normal of contact $n_m \in \mathbb{R}^3$ corresponds to the average of the two most distant normals of contact between the robot and the surface. It is used both for determining the desired orientation of the hand during the exploration and the allowed plan of motion to reach the final Cartesian target (detailed in the next paragraphs).

$$n_m = \begin{cases} \frac{n_i + n_j}{2} & \text{if } n_i + n_j \neq 0 \\ n_i & \text{else.} \end{cases} \tag{A.1.1}$$

where $(i, j) = \operatorname{argmax}_{(i,j)\in n_c}\{acos(n_i, n_j)\}$ are the indices of the two contacts which have the most different normals.

This is useful, because taking only the average of all the contact normals would give little weight to outliers, which are very important as they represent crucial information about the surface's profile.

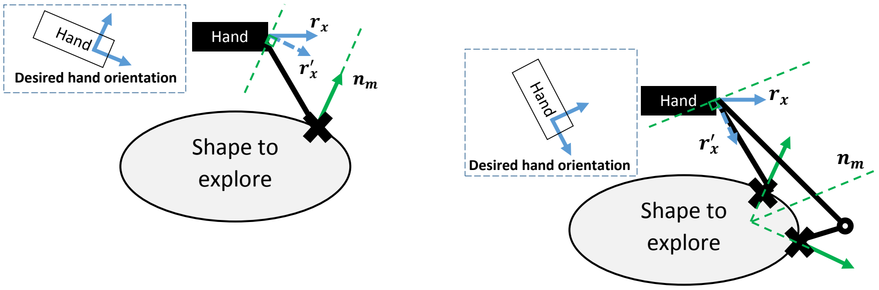
Orientation reference of the impedance controller

We create a rotational frame R_{n_m,r_x}, using the above normal direction of contact n_m and the orthogonal projection of the hand's proximo-distal direction (palm towards fingers) r_x on n_m:

$$r'_x = r_x - (r_x \cdot n_m)n_m \tag{A.1.2}$$

$$R_{n_m,r'_x} = \begin{bmatrix} n_m, & r'_x, & n_m \times r'_x \end{bmatrix} \tag{A.1.3}$$

This ensures that the palm of the hand stays perpendicular to the contact normal, see Figure A.1.



(a) One contact (b) Two contacts

Figure A.1: Schematic of the computation of the reference hand orientation for the impedance controller. In the case of one contact, the normal of the contact is chosen as a reference for the desired hand orientation. For two or more contacts, the average of the two most distant normals is chosen.

POSITION REFERENCE OF THE IMPEDANCE CONTROLLER

The reference position $p_r \in \mathbb{R}^3$ of the impedance controller is computed from the desired final position of the motion $p_f \in \mathbb{R}^3$, given by a higher-level controller, the current position $p \in \mathbb{R}^3$, and the computed median normal of contact n_m. The error p_e between the current and final position is projected on a plane normal to n_m in order to create a motion tangential to the surface:

$$p_e = p_f - p \tag{A.1.4}$$

$$p'_e = p'_e - (p'_e \cdot n_m) \cdot n_m \tag{A.1.5}$$

The reference position is then proportional with gain $G \in \mathbb{R}^+$ to the projected error, and saturated if that distance is bigger than a scalar threshold $d \in \mathbb{R}^+$.

$$p_r = \begin{cases} p + \frac{p'_e}{\|p'_e\|} \cdot G & \text{if } \parallel p'_e \parallel > d \\ p + p'_e \cdot \frac{G}{d} & \text{else.} \end{cases} \tag{A.1.6}$$

IMPEDANCE CONTROL

Because the robot operates in contact with its environment, a compliant controller provides a safe way to interact with the areas in contact.

Given the reference and actual positions $p_r, p \in \mathbb{R}^3$ and orientations $R_r, R \in \mathbb{R}^{3\times3}$ of the end effector (here defined at the base of the middle-finger), we define the Cartesian error term as:

$$x_e = \begin{bmatrix} p_e \\ \Psi \end{bmatrix}, \quad \Psi = \text{angleaxis}(R^T R_r) \tag{A.1.7}$$

where angleaxis(R^*) represents the angle-axis representation corresponding to a rotation matrix R.

The torques for the Cartesian impedance control task are computed by multiplication of the transposed Jacobian $J_e(q)$ with the Cartesian feedback control forces:

$$\tau_e = J_e^T(Kx_e + D\dot{x}_e) \tag{A.1.8}$$

The stiffness and damping matrices $K, D \in \mathbb{R}^{6\times 6}$ are symmetric positive definite:

$$K = \begin{bmatrix} K_p & 0 \\ 0 & K_r \end{bmatrix}, \; D = \begin{bmatrix} D_p & 0 \\ 0 & D_r \end{bmatrix} \tag{A.1.9}$$

where $K_p, K_r \in \mathbb{R}^{3\times 3}$ and $D_p, D_r \in \mathbb{R}^{3\times 3}$ are sub-matrices respectively relating forces to positional errors, torques to rotational errors, forces to positional velocity and torques to rotational velocities.

Definition of the stiffness and damping matrices

During the exploration, the purpose of the impedance control is to drive the motion of the robotic hand, not to ensure contact with the surface. For this reason, the stiffness matrices are defined in the rotational frame R attached to the end-effector, as:

$$K'_p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & k_p \end{bmatrix}, \qquad K'_r = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_r & 0 \\ 0 & 0 & k_r \end{bmatrix} \tag{A.1.10}$$

where the first axis corresponds to the dorso-palmar direction, and the second axis to the proximo-distal direction. In the robot's frame, we use the rotated stiffness matrices $K_p = RK'_pR^T$ and $K_r = RK'_rR^T$. The rotational stiffness value is zero in the dorso-palmar direction as the orientation of the hand along that axis is not crucial for the exploration and this releases a degree of freedom and allows more dexterous motions.

Both positional and rotational damping matrices are isotropic[1]:

$$D_r = d_r \cdot I_{3\times 3}, \qquad D_p = d_p \cdot I_{3\times 3} \qquad \text{with } d_r, d_p \in \mathbb{R}. \tag{A.1.11}$$

---

[1] We used $k_p = 300N.m^{-1}$, $d_p = 300N.s.m^{-1}$, $k_r = 10N.rad^{-1}$, $d_r = 3N.s.rad^{-1}$

REFERENCE POSITION

Thanks to our algorithm, the reference positions described as p_f in Appendix A.1 do not need to lie on the surface since the controller navigates to the closest point on the surface. Therefore, they can be randomly distributed around the estimated position of the area to explore. We define a list of these reference positions spread around the object to explore. When the end-effector reaches within a threshold of the orthogonal projection of the current reference position on the surface's estimated tangential plane, the next reference position in the list is tracked. This way, the end-effector goes sequentially through all the positions in the list. There are more informed ways to choose the reference positions, for instance using entropy and information gain about the surface being reconstructed, but the target of this experiment is to demonstrate the possibility to be in contact with an unknown surface and to navigate smoothly around it, not the search process itself.

Appendix B

Appendices for Chapter 5

B.1 Likelihood computation for the particle filter's measurement step

Two problems arise from comparing two contact sets. The first one is the conversion of the measurement errors into a likelihood, and the second comes from potentially different dimensionality between the measured contacts and the virtual ones from a particle. Indeed, the number of contacts do not necessarily match. In order to compute the likelihood used in the measurement step of the particle filter, we use the following Algorithm 5.

Algorithm 5: Likelihood computation in the measurement step of the particle filter

```
Data: The measurement $y = \{c_i\}_{i \in N_c}$,
a potential object's pose $\xi$,
a weighting diagonal matrix $D \in \mathbb{R}^{7\times7}$ to compute the distance
between two contact points,
the mapping functions $f, d : \mathbb{R} \to \mathbb{R}$ to convert contact distance and
intensity to likelihood.[1]
Result: Likelihood $L$ that a measurement $y$ was generated by object
        pose $\xi$.
L = 1;
/* Generate contacts from object pose.                              */
$\hat{y} = \{\hat{c}_i\}_{i \in \hat{N}_c} = G(\xi)$
for $i \leftarrow 1$ to $N_c$ do
    if $\hat{N}_c \neq 0$ then
        $L \mathrel{*}= f(\min_{j \in \hat{N}_c} \|c_i, \hat{c}_j\|_D)$
    else
        $L \mathrel{*}= L_{penalty}$
/* Penalize for high contact intensity                              */
for $j \leftarrow 1$ to $\hat{N}_c$ do
    $L \mathrel{*}= d(\hat{c}_j)$
return $L$;
```

---

[1] The functions f and d both have the form $f(x) = \max(\exp(\frac{-(x-x_0)^2}{w}), f_{\min})$. They are

B.2 Proofs of stability for EMDS

These proofs are adapted from Kronander et al. (2015) for externally modulated LMDS.

Proposition 1 (Equilibrium points). *If $M(x,s)$ has full rank for all x and s, the reshaped dynamics has the same equilibrium point(s) as the original dynamics.*

If $M(x,s)$ has full rank, it has an empty null-space, and hence Equation 5.2.3 is zero iff $f(x)=0$.

Proposition 2 (Boundedness). *Assume that the original dynamics is bounded (See. Def 2). Assume further that $M(x,s)$ is locally active on x in a compact subset $\chi \subset \mathbb{R}^N$ (See Def. 1b). Then, the reshaped dynamics is bounded.*

Let B_R be a ball centered at the origin of radius R in $\mathbb{R}^N$. Let R be chosen such that χ lies entirely in B_R. Since χ is a compact set in $\mathbb{R}^N$, it is always possible to find such a R. For each $\delta > 0$, let $\epsilon(\delta)$ be an associated boundary for the original dynamics (refer to Def. 2). Define $\epsilon'(\delta)$ as a boundary for the reshaped dynamics as follows: $\epsilon' = \epsilon(R)$ for $\delta < R$ and $\epsilon' = \epsilon(\delta)$ for $\delta \geq R$. Boundedness follows from Def. 2.

Proposition 3 (Lyapunov stability). *Consider a system $\dot{x} = f(x)$ that has a single equilibrium point. Without loss of generalizaty; let this equilibrium point be placed at the origin. Assume further that the equilibrium point is stable. Assume that the criteria for Propositions 1 and 2 are satisfied. If in addition, χ does not include the origin, the reshaped system is stable at the origin.*

According to Proposition 1, the reshaped dynamics has a single equilibrium point at the origin. let B_r be a ball centered at the origin with a radius r small enough that B_r does not include any point in χ. Hence, inside B_r, we have $g(x) = f(x)$. By the stability of f, there exists for all $0 < \epsilon < r$ a $\delta(\epsilon)$ such that $||x(0)|| < \delta(\epsilon) \Rightarrow ||x(t)|| < \epsilon, \forall t > 0$. For any $\epsilon > r$, let $\delta(\epsilon) = \delta(r)$. Then, by the stability of f, $||x(0)|| < \delta(\epsilon) = \delta(r) \Rightarrow ||x(t)|| < r < \epsilon$.

Proposition 4 (Local asymptotic stability). *Consider a system $\dot{x} = f(x)$ that has a single equilibrium point. Assume that the conditions of Proposition 1, 2 and 3 are satisfied. then, the reshaped system is locally asymptotically stable at the origin.*

The original dynamics are globally asymptotically stable, which implies the

---

Gaussian functions of amplitude 1, with a minimum threshold to avoid setting the likelihood to 0. The function f is centered on 0 ($x_0 = 0$): a contact distance close to 0 does not decrease the likelihood. The widths of the Gaussian functions are chosen according the desired level of penalization for the measurements' mismatch.

existence of a Lyapunov function $V\colon \mathbb{R}^N \to \mathbb{R}^+$ such that:

$$V(x) > 0, \forall x \neq 0 \text{ and } V(0) = 0 \tag{B.2.1}$$

$$\dot{V} = \frac{\partial V}{\partial x} f(x) > 0, \forall x \neq 0 \text{ and } \dot{V}(0) = 0 \tag{B.2.2}$$

Let B_r be defined as in the proof of Proposition 3. Let $M \subset B_r$ denote the largest level set of V that lies entirely inside B_r. For any $x_0 \in M$, the rehsaped dynamics is exactly equal to the original dynamics $\dot{x} = f(x)$. Hence, $V(x) > 0$ and $\dot{V}(x) < 0$ holds for all $x \in M$, which proves that the system is locally asymptotically stable at the origin with the region of attraction given by M.

REFERENCES

P.K. Allen and P. Michelman. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Transactions on Robotics and Automation*, 6(4):397–404, August 1990. ISSN 1042-296X. doi: 10.1109/70.59353.

D. Anghinolfi, G. Cannata, F. Mastrogiovanni, C. Nattero, and M. Paolucci. On the Problem of the Automated Design of Large-Scale Robot Skin. *IEEE Transactions on Automation Science and Engineering*, 10(4):1087–1100, October 2013. ISSN 1545-5955. doi: 10.1109/TASE.2013.2252617.

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, May 2009. ISSN 0921-8890. doi: 10.1016/j.robot.2008.10.024. URL `http://www.sciencedirect.com/science/article/pii/S0921889008001772`.

Alan D. Berger and Pradeep K. Khosla. Using Tactile Data for Real-Time Feedback. *The International Journal of Robotics Research*, 10(2):88–102, April 1991. ISSN 0278-3649, 1741-3176. doi: 10.1177/027836499101000202. URL `http://ijr.sagepub.com/content/10/2/88`.

Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992. ISSN 0162-8828. doi: 10.1109/34.121791. URL `http://dx.doi.org/10.1109/34.121791`.

A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, volume 1, pages 348–353 vol.1, 2000. doi: 10.1109/ROBOT.2000.844081.

Antonio Bicchi, Marco Gabiccini, and Marco Santello. Modelling natural and artificial hands with synergies. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 366(1581):3153–3161, November 2011. ISSN 1471-2970. doi: 10.1098/rstb.2011.0152.

Alexander Bierbaum, Matthias Rambow, Tamim Asfour, and Rüdiger Dillmann. A potential field approach to dexterous tactile exploration of unknown objects. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 360–366, 2008.

Aude Billard and Gillian M. Hayes. DRAMA, a Connectionist Architecture for

Control and Learning in Autonomous Robots. *Adapt. Behav.*, 7(1):35–63, December 1999. ISSN 1059-7123. doi: 10.1177/105971239900700103. URL http://dx.doi.org/10.1177/105971239900700103.

Aude G. Billard, Sylvain Calinon, and Rüdiger Dillmann. Learning from Humans. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 1995–2014. Springer International Publishing, 2016. ISBN 978-3-319-32550-7 978-3-319-32552-1. URL http://link.springer.com/chapter/10.1007/978-3-319-32552-1_74. DOI: 10.1007/978-3-319-32552-1_74.

Gereon H. Büscher, Risto Kõiva, Carsten Schürmann, Robert Haschke, and Helge J. Ritter. Flexible and stretchable fabric-based tactile sensor. *Robotics and Autonomous Systems*, 63, Part 3:244–252, January 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2014.09.007. URL http://www.sciencedirect.com/science/article/pii/S0921889014001821.

M.G. Catalano, G. Grioli, A. Serio, E. Farnioli, C. Piazza, and A. Bicchi. Adaptive synergies for a humanoid robot hand. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 7–14, November 2012. doi: 10.1109/HUMANOIDS.2012.6651492.

N. Chen, H. Zhang, and R. Rink. Edge tracking using tactile servo. In *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings*, volume 2, pages 84–89 vol.2, August 1995. doi: 10.1109/IROS.1995.526143.

Z. Chen, T. Wimböck, M. A. Roa, B. Pleintinger, M. Neves, C. Ott, C. Borst, and N. Y. Lii. An adaptive compliant multi-finger approach-to-grasp strategy for objects with position uncertainties. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4911–4918, May 2015. doi: 10.1109/ICRA.2015.7139881.

Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dexterous Grasping via Eigengrasps: A Low-Dimensional Approach to a High-Complexity Problem. *Robotics: Science and Systems - Robot Manipulation: Sensing and Adapting to the Real World*, 2007. URL http://www.coreygoldfeder.com/papers/RSS07.pdf.

Matei Ciocarlie, Kaijen Hsiao, Edward Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan A. Şucan. Towards Reliable Grasping and Manipulation in Household Environments. In Oussama Khatib, Vijay Kumar, and Gaurav Sukhatme, editors, *Experimental Robotics*, number 79 in Springer Tracts in Advanced Robotics, pages 241–252. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-28571-4 978-3-642-28572-1. DOI: 10.1007/978-3-642-28572-1_17.

William Cleveland. {LOWESS}: {A} {P}rogram for {S}moothing {S}catterplots by {R}obust {L}ocally {W}eighted {R}egression. *The American Statistician*, 35(1), 1981. ISSN 00031305. doi: 10.2307/2683591. URL http://dx.doi.org/10.2307/2683591.

Ravin de Souza, Sahar El-Khoury, José Santos-Victor, and Aude Billard. Recognizing the grasp intention from human demonstration. *Robotics and Autonomous Systems*, 74, Part A:108–121, December 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2015.07.006. URL `http://www.sciencedirect.com/science/article/pii/S0921889015001505`.

Ravin de Souza, Luis. *Grasping for the Task*. PhD thesis, 2016. URL `https://infoscience.epfl.ch/record/217897?ln=fr`.

Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, June 2014. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364914521306. URL `http://ijr.sagepub.com/content/33/7/1006`.

Jeremy A. Fishel and Gerald E. Loeb. Bayesian exploration for intelligent identification of textures. *Frontiers in Neurorobotics*, 6:4, 2012. doi: 10.3389/fnbot.2012.00004. URL `http://journal.frontiersin.org/article/10.3389/fnbot.2012.00004/full`.

F. Flacco, A. De Luca, and O. Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3970–3977, October 2012. doi: 10.1109/IROS.2012.6385619.

Carolina Galleguillos and Serge Belongie. Context Based Object Categorization: A Critical Survey. *Comput. Vis. Image Underst.*, 114(6):712–722, June 2010. ISSN 1077-3142. doi: 10.1016/j.cviu.2010.02.004. URL `http://dx.doi.org/10.1016/j.cviu.2010.02.004`.

A. P. Gerratt, N. Sommer, S. P. Lacour, and A. Billard. Stretchable capacitive tactile skin on humanoid robot fingers; First experiments and results. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 238–245, November 2014. doi: 10.1109/HUMANOIDS.2014.7041366.

P. Giguere and G. Dudek. A Simple Tactile Probe for Surface Identification by Mobile Robots. *Robotics, IEEE Transactions on*, 27(3):534 –544, June 2011. ISSN 1552-3098.

Corey Goldfeder and Peter K. Allen. Data-driven grasping. *Autonomous Robots*, 31(1):1–20, April 2011. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-011-9228-1. URL `http://link.springer.com/article/10.1007/s10514-011-9228-1`.

E. Gribovskaya, S.M. Khansari-Zadeh, and A. Billard. Learning Non-linear Multivariate Dynamics of Motion in Robotic Manipulators. *Int. J. Rob. Res.*, 30(1):80–117, January 2011a. ISSN 0278-3649. doi: 10.1177/0278364910376251. URL `http://dx.doi.org/10.1177/0278364910376251`.

E. Gribovskaya, A. Kheddar, and A. Billard. Motion learning and adaptive impedance for robot control during physical interaction with humans. In

2011 IEEE International Conference on Robotics and Automation (ICRA), pages 4326–4332, May 2011b. doi: 10.1109/ICRA.2011.5980070.

G. Grioli, M. Catalano, E. Silvestro, S. Tono, and A. Bicchi. Adaptive synergies: An approach to the design of under-actuated robotic hands. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1251–1256, October 2012. doi: 10.1109/IROS.2012.6385881.

Leon D. Harmon. Automated Tactile Sensing. *The International Journal of Robotics Research*, 1(2):3–32, June 1982. ISSN 0278-3649, 1741-3176. doi: 10.1177/027836498200100201. URL http://ijr.sagepub.com/content/1/2/3.

G. Heidemann and M. Schopfer. Dynamic tactile sensing for object identification. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 813 – 818 Vol.1, May 2004.

Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, April 2013. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-012-9321-0. URL http://link.springer.com/article/10.1007/s10514-012-9321-0.

Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E. Gil Jones. Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1228–1235. IEEE, 2010.

Rinat Ibrayev and Yan-Bin Jia. Surface Recognition by Registering Data Curves from Touch. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 55 –60, October 2006.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, February 2013. ISSN 1530-888X. doi: 10.1162/NECO_a_00393.

Ioan A. Sucan and Sachin Chitta. MoveIt! URL http://moveit.ros.org.

Herbert Jaeger. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148, 2001. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/EchoStatesTechRep.pdf.

Advait Jain, Marc D. Killpack, Aaron Edsinger, and Charles C. Kemp. Reaching in clutter with whole-arm tactile sensing. *The International Journal of Robotics Research*, 32(4):458–482, April 2013. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364912471865. URL http://ijr.sagepub.com/content/32/4/458.

N. Jamali, M. Maggiali, F. Giovannini, G. Metta, and L. Natale. A new design of a fingertip for the iCub hand. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2705–2710, September 2015. doi: 10.1109/IROS.2015.7353747.

N. Jamali, C. Ciliberto, L. Rosasco, and L. Natale. Active perception: Building objects' models using tactile exploration. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 179–185, November 2016. doi: 10.1109/HUMANOIDS.2016.7803275.

R. S. Jamisola, P. Kormushev, A. Bicchi, and D. G. Caldwell. Haptic exploration of unknown surfaces with discontinuities. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1255–1260, September 2014. doi: 10.1109/IROS.2014.6942718.

A. R. Jiménez, A. S. Soembagijo, D. Reynaerts, H. Van Brussel, R. Ceres, and J. L. Pons. Featureless classification of tactile contacts in a gripper using neural networks. *Sensors and Actuators A: Physical*, 62(1–3):488 – 491, 1997. ISSN 0924-4247. Proceedings of Eurosensors X.

R. S. Johansson and G. Westling. Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental Brain Research*, 56(3):550–564, 1984. ISSN 0014-4819.

Roland S. Johansson and J. Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, April 2009. ISSN 1471-003X, 1471-0048. doi: 10.1038/nrn2621. URL `http://www.nature.com/doifinder/10.1038/nrn2621`.

Magnus Johnsson and Christian Balkenius. Experiments with Proprioception in a Self-Organizing System for Haptic Perception. In *[Host publication title missing]*. Towards Autonomous Robotic Systems 2007, University of Wales, Aberystwyth, UK, 239-245., 2007. URL `http://lup.lub.lu.se/record/948984`.

O. Kanoun, F. Lamiraux, and P. B. Wieber. Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, 27(4):785–792, August 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2142450.

Zhanat Kappassov, Juan-Antonio Corrales, and Véronique Perdereau. Tactile sensing in dexterous robot hands — Review. *Robotics and Autonomous Systems*, 74, Part A:195–220, December 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2015.07.015. URL `http://www.sciencedirect.com/science/article/pii/S0921889015001621`.

C. C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [Grand Challenges of Robotics]. *IEEE Robotics Automation Magazine*, 14(1):20–29, March 2007. ISSN 1070-9932. doi: 10.1109/MRA.2007.339604.

S. M. Khansari-Zadeh and A. Billard. Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics*, 27 (5):943–957, October 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2159412.

O. Khatib, L. Sentis, J. Park, and J. Warren. Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 01(01):29–43, March 2004. ISSN 0219-8436. doi: 10.1142/S0219843604000058. URL `http://www.worldscientific.com/doi/abs/10.1142/S0219843604000058`.

Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1):90–98, March 1986. ISSN 0278-3649, 1741-3176. doi: 10.1177/027836498600500106. URL `http://ijr.sagepub.com/content/5/1/90`.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, 1987. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087068`.

Junggon Kim, K. Iwamoto, J.J. Kuffner, Y. Ota, and N.S. Pollard. Physically Based Grasp Quality Evaluation Under Pose Uncertainty. *IEEE Transactions on Robotics*, 29(6):1424–1439, December 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2273846.

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. volume 3, pages 2149–2154. IEEE, 2004. ISBN 978-0-7803-8463-7. doi: 10.1109/IROS.2004.1389727.

O. Kroemer, C. H. Lampert, and J. Peters. Learning Dynamic Tactile Sensing With Robust Vision-Based Training. *IEEE Transactions on Robotics*, 27(3): 545–557, June 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2121130.

K. Kronander and A. Billard. Passive Interaction Control With Dynamical Systems. *IEEE Robotics and Automation Letters*, 1(1):106–113, January 2016. ISSN 2377-3766. doi: 10.1109/LRA.2015.2509025.

K. Kronander, M. Khansari, and A. Billard. Incremental Motion Learning with Locally Modulated Dynamical Systems. *Robot. Auton. Syst.*, 70(C):52–62, August 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2015.03.010. URL `http://dx.doi.org/10.1016/j.robot.2015.03.010`.

Susan J Lederman and Roberta L Klatzky. Hand movements: A window into haptic object recognition. *Cognitive Psychology*, 19(3):342–368, July 1987. ISSN 0010-0285. doi: 10.1016/0010-0285(87)90008-9. URL `http://www.sciencedirect.com/science/article/pii/0010028587900089`.

Miao Li, Kaiyu Hang, Danica Kragic, and Aude Billard. Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 75, Part B:352–364, January 2016. ISSN 0921-8890. doi: 10.1016/j.robot.2015.09.008. URL `http://www.sciencedirect.com/science/article/`

pii/S0921889015001967.

H. Maekawa, K. Tanie, and K. Komoriya. Tactile sensor based manipulation of an unknown object by a multifingered hand with rolling contact. In , *1995 IEEE International Conference on Robotics and Automation, 1995. Proceedings*, volume 1, pages 743–750 vol.1, May 1995. doi: 10.1109/ROBOT.1995.525372.

Perla Maiolino, Marco Maggiali, Giorgio Cannata, Giorgio Metta, and Lorenzo Natale. A Flexible and Robust Large Scale Capacitive Tactile System for Robots. *IEEE Sensors Journal*, 13(10):3910–3917, October 2013. ISSN 1530-437X, 1558-1748. doi: 10.1109/JSEN.2013.2258149. URL http://arxiv.org/abs/1411.6837. arXiv: 1411.6837.

N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks. In *International Conference on Advanced Robotics, 2009. ICAR 2009*, pages 1–6, June 2009.

M. Meier, M. Schopfer, R. Haschke, and H. Ritter. A Probabilistic Approach to Tactile Shape Reconstruction. *IEEE Transactions on Robotics*, 27(3):630–635, 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2120830.

Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-293-1. doi: 10.1145/1774674.1774683.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, June 2014. ISSN 0921-8890. doi: 10.1016/j.robot.2014.03.001.

A. Narendiran and B. George. Capacitive tactile sensor with slip detection capabilities for robotic applications. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 464–469, May 2015. doi: 10.1109/I2MTC.2015.7151312.

S.E. Navarro, N. Gorges, H. Worn, J. Schill, T. Asfour, and R. Dillmann. Haptic object recognition for multi-fingered robot hands. In *Haptics Symposium (HAPTICS), 2012 IEEE*, pages 497 –502, March 2012.

A.M. Okamura and M.R. Cutkosky. Haptic exploration of fine surface features. In *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*, volume 4, pages 2930–2936 vol.4, 1999. doi: 10.1109/ROBOT.1999.774042.

A.M. Okamura, M.L. Turner, and M.R. Cutkosky. Haptic exploration of objects with rolling and sliding. In , *1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings*, volume 3, pages 2485–2490 vol.3, April

1997. doi: 10.1109/ROBOT.1997.619334.

Jaeheung Park and Oussama Khatib. Robot multiple contact control. *Robotica*, 26(05):667–677, September 2008. ISSN 1469-8668. doi: 10.1017/S0263574708004281. URL http://journals.cambridge.org/article_S0263574708004281.

P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 365–371, September 2011. doi: 10.1109/IROS.2011.6095059.

Peter Pastor, Mrinal Kalakrishnan, Franziska Meier, Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems*, 61(4):351–361, April 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2012.09.017. URL http://www.sciencedirect.com/science/article/pii/S0921889012001716.

JR. Platt, A. H. Fagg, and R. A. Grupen. Null-Space Grasp Control: Theory and Experiments. *IEEE Transactions on Robotics*, 26(2):282–295, April 2010. ISSN 1552-3098. doi: 10.1109/TRO.2010.2042754.

N.S. Pollard. Synthesizing grasps from generalized prototypes. In *, 1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings*, volume 3, pages 2124–2130 vol.3, April 1996. doi: 10.1109/ROBOT.1996.506184.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

Máximo A. Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, July 2014. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-014-9402-3. URL http://link.springer.com/article/10.1007/s10514-014-9402-3.

C. Rosales, A. Ajoudani, M. Gabiccini, and A. Bicchi. Active gathering of frictional properties from objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 3982–3987, September 2014. doi: 10.1109/IROS.2014.6943122.

L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, and J. Y. Fourquet. Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints. *IEEE Transactions on Robotics*, 29(2):346–362, April 2013. ISSN 1552-3098. doi: 10.1109/TRO.2012.2234351.

Eric L. Sauser, Brenna D. Argall, Giorgio Metta, and Aude G. Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 60(1):55 – 71, 2012. ISSN 0921-8890.

Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):537–547, 2003. URL http://rstb.royalsocietypublishing.org/content/358/1431/537.short.

A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 243 –248, October 2009.

Carsten Schürmann, Matthias Schöpfer, Robert Haschke, and Helge Ritter. A High-Speed Tactile Sensor for Slip Detection. In Erwin Prassler, Marius Zöllner, Rainer Bischoff, Wolfram Burgard, Robert Haschke, Martin Hägele, Gisbert Lawitzky, Bernhard Nebel, Paul Plöger, and Ulrich Reiser, editors, *Towards Service Robots for Everyday Environments*, number 76 in Springer Tracts in Advanced Robotics, pages 403–415. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-25115-3 978-3-642-25116-0. URL http://link.springer.com/chapter/10.1007/978-3-642-25116-0_27. DOI: 10.1007/978-3-642-25116-0_27.

Allen I. Selverston. Are Central Pattern Generators Understandable? *Behavioral and Brain Sciences*, 3(4):535, 1980.

Ashwini Shukla and Aude Billard. Coupled dynamical system based arm–hand grasping model for learning fast adaptation strategies. *Robotics and Autonomous Systems*, 60(3):424–440, March 2012. ISSN 0921-8890. doi: 10.1016/j.robot.2011.07.023.

N. Sommer. *Learning with tactile feedback on a humanoid robot.* Master thesis, INSA de Strasbourg, February 2012. URL http://eprints2.insa-strasbourg.fr/1073/.

N. Sommer and A. Billard. Face classification using touch with a humanoid robot hand. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 120–125, 2012. doi: 10.1109/HUMANOIDS.2012.6651508.

Nicolas Sommer and Aude Billard. Multi-contact haptic exploration and grasping with tactile sensors. *Robotics and Autonomous Systems*, 2016. ISSN 0921-8890. doi: 10.1016/j.robot.2016.08.007. URL http://www.sciencedirect.com/science/article/pii/S0921889016301610.

Nicolas Sommer, Miao Li, and Aude Billard. Bimanual compliant tactile exploration for grasping unknown objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6400–6407. IEEE, 2014.

J. S. Son and R. D. Nowe. Tactile sensing and stiffness control with multifingered hands. In *, 1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings*, volume 4, pages 3228–3233 vol.4, April 1996. doi: 10.1109/ROBOT.1996.509204.

S. A. Stansfield. A haptic system for a multifingered hand. In *, 1991 IEEE In-*

ternational Conference on Robotics and Automation, 1991. Proceedings, pages 658–664 vol.1, April 1991. doi: 10.1109/ROBOT.1991.131658.

Ricarda Steffens, Matthias Nieuwenhuisen, and Sven Behnke. Continuous Motion Planning for Service Robots with Multiresolution in Time. In *Intelligent Autonomous Systems 13*, pages 203–215. Springer, Cham, 2016. URL http://link.springer.com/chapter/10.1007/978-3-319-08338-4_16. DOI: 10.1007/978-3-319-08338-4_16.

Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 297–303, November 2015. doi: 10.1109/HUMANOIDS.2015.7363558.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998. URL http://www.cell.com/trends/cognitive-sciences/pdf/S1364-6613(99)01331-5.pdf.

Pete Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, page 0278364914557874, February 2015. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364914557874. URL http://ijr.sagepub.com/content/early/2015/02/05/0278364914557874.

M. R. Tremblay and M. R. Cutkosky. Estimating friction using incipient slip sensing during a manipulation task. In *, 1993 IEEE International Conference on Robotics and Automation, 1993. Proceedings*, pages 429–434 vol.1, May 1993. doi: 10.1109/ROBOT.1993.292018.

A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard. Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations. *IEEE Transactions on Robotics*, 31(6):1458–1471, December 2015. ISSN 1552-3098. doi: 10.1109/TRO.2015.2495003.

Y. Yamakawa, A. Namiki, M. Ishikawa, and M. Shimojo. One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 703–708, October 2007. doi: 10.1109/IROS.2007.4399379.

CURRICULUM VITAE

NICOLAS SOMMER

Roboticist

@ n.sommer@epfl.ch ☎ +41 78 614 15 38 ✉ Chemin du Bochet, 2 VD 1025 St-Sulpice, Switzerland
github.com/nisommer
Nationality: French **Residence permit:** Permis B

EDUCATION

PhD

“Multi-contact tactile exploration and interaction with unknown objects”

EPFL

December 2012 – March 2017 Lausanne, Switzerland

- Supervision of a student Master thesis: *Proprioception for collision detection*
- In charge of IT administration in the lab

Msc, Microengineering *Robotics and Autonomous Systems*

EPFL

2010 – 2012 Lausanne, Switzerland

- Master project: *Learning with tactile feedback on a humanoid robot*
  I applied Programming by Demonstration (PbD) methods to learn and reproduce manipulation tasks, taking into account tactile signals from the robot’s fingertips
- Double-degree between EPFL (CH) and INSA (FR)

Msc, Mechatronics

INSA

2006 – 2012 Strasbourg, France

- Project: Design of a quadrotor controller
  Model identification of a quadrotor, optimizing controller parameters for flight, programming of the controller on onboard microcontroller (Microchip dsPIC family)

WORK EXPERIENCE

Teaching assistant

EPFL

2011-2015 Lausanne, Switzerland

- Applied Machine Learning class for Master Students
  PCA | ICA | K-means | KNN | GMM/GMR | SVM/SVR | Neural networks | HMM
- Machine Learning class for PhD Students
  CCA | Bagging/boosting | Reinforcement Learning

Summer internship

Siemens

June – July 2009 Haguenau, France

- Study of the precision and accuracy of the instruments in the metrology laboratory (Quality departement)

Summer internship

Nussbaum (automobile lifts)

June – July 2007 Bodersweier, Germany

- First experience in a company. Various tasks: CAD modeling of hardware components, electronic circuit verification, assistance to truck lifts production.

RESEARCH INTERESTS

Machine Learning

Control with Tactile Sensors

Automation
Useful for everyday life

SKILLS

C++ | Python | Matlab | LaTeX

ROS | git | μC programming

CAD (ProE, Solidworks)
Electronic Circuit Design (Altium Designer)

LANGUAGES

French	●●●●●
English	●●●●●
German	●●●○○
Spanish	●●●○○
Portuguese	●○○○○
Turkish	●○○○○

INTERESTS

Volleyball | Piano | Salsa | Diving

DIY projects | Home automation | 🏍

PUBLICATIONS

Journal Articles

- Sommer, Nicolas and Aude Billard (2016). "Multi-contact haptic exploration and grasping with tactile sensors". In: *Robotics and Autonomous Systems*.

Conference Proceedings

- Gerratt, Aaron P., Nicolas Sommer, Stéphanie P. Lacour, and Aude Billard (2014). "Stretchable capacitive tactile skin on humanoid robot fingers—First experiments and results". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE.
- Sommer, Nicolas, Miao Li, and Aude Billard (2014). "Bimanual compliant tactile exploration for grasping unknown objects". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.
- Sommer, Nicolas and Aude Billard (2012). "Face classification using touch with a humanoid robot hand". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

Workshop Presentations

- Nicolas Sommer (2014). *Tactile exploration with the iCub robot*. Presented at the iCub and friends Workshop, ICRA 2014. Hong-Kong.
- Nicolas Sommer (2012). *Face Classification using Touch with a Humanoid Robot*. Presented in the Second Workshop on Advances in tactile sensing and touch-based human-robot interaction, IROS 2012. Villamoura, Portugal.