Lappeenranta
University of Technology

Jonna Laaksonen

TACTILE-PROPRIOCEPTIVE ROBOTIC GRASPING

Thesis for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium 1382 at Lappeenranta University of Technology, Lappeenranta, Finland on the 18th of June, 2012, at noon.

Acta Universitatis
Lappeenrantaensis **476**

Supervisor Professor Ville Kyrki

Machine Vision and Pattern Recognition Laboratory
Department of Information Technology
Faculty of Technology Management
Lappeenranta University of Technology
Finland

Reviewers Professor Peter K. Allen
Department of Computer Science
Columbia University
United States of America

Professor Norbert Krüger
Mærsk Mc-Kinney Møller Instituttet
University of Southern Denmark
Denmark

Opponents Professor Norbert Krüger
Mærsk Mc-Kinney Møller Instituttet
University of Southern Denmark
Denmark

Professor Arto Visala
Department of Automation and Systems Technology
Aalto University
Finland

ISBN 978-952-265-246-1
ISBN 978-952-265-247-8 (PDF)
ISSN 1456-4491

Lappeenrannan teknillinen yliopisto
Digipaino 2012

Preface

This thesis was born from the GRASP EU project that was funded by the European Commission during 2008-2012. The project would have not happened without the hard work of all of the PIs involved in the project. The project had partners from all over Europe and the project and its members offered unforgettable experiences during the official and unofficial meetings in the last four years.

I want to thank my thesis advisor and one of the PIs in the GRASP project, professor Ville Kyrki, for his valuable advice and time during the last six years I've spent at LUT. I would also like to thank all my co-authors at Universitat Jaume I, Royal Institute of Technology, Karlsruhe Institute of Technology and at University of Southern Denmark. Especially, I want to thank professor Danica Kragic who hosted me at the Computer Vision and Active Perception lab at the Royal Institute of Technology for two weeks.

Of course, one of the key factors that made this thesis possible, is the Machine Vision and Pattern Recognition lab. I would like to thank all the people who worked there during my stay and especially Arto, Behdad, Ekaterina, Heikki, Ilmari, Jani, Jarmo, Joni, Jukka, Lasse, Leena, Nataliya, Olli, Pekka, Risto, the two Teemus, Tarja, Tomi and Tuomas, for making life easier and more interesting at the lab. I also want to thank Sami Moisio from the CEID-centre at LUT for introducing me to the world of simulation during the GRASP project.

Last, but not least, I would like to thank my parents and my brother and sister for showing interest in my work and being there for me.

The journey to complete this thesis has been long and some major changes have occurred during this journey. I would like to thank all who have been understanding towards these changes.

Lappeenranta, 2012

Jonna Laaksonen

Abstract

Jonna Laaksonen
Tactile-Proprioceptive Robotic Grasping
Lappeenranta, 2012
107 p.

Acta Universitatis Lappeenrantaensis 476
Diss. Lappeenranta University of Technology

ISBN 978-952-265-246-1
ISBN 978-952-265-247-8 (PDF)
ISSN 1456-4491

Robotic grasping has been studied increasingly for a few decades. While progress has been made in this field, robotic hands are still nowhere near the capability of human hands. However, in the past few years, the increase in computational power and the availability of commercial tactile sensors have made it easier to develop techniques that exploit the feedback from the hand itself, the sense of touch. The focus of this thesis lies in the use of this sense.

The work described in this thesis focuses on robotic grasping from two different viewpoints: robotic systems and data-driven grasping. The robotic systems viewpoint describes a complete architecture for the act of grasping and, to a lesser extent, more general manipulation. Two central claims that the architecture was designed for are hardware independence and the use of sensors during grasping. These properties enables the use of multiple different robotic platforms within the architecture.

Secondly, new data-driven methods are proposed that can be incorporated into the grasping process. The first of these methods is a novel way of learning grasp stability from the tactile and haptic feedback of the hand instead of analytically solving the stability from a set of known contacts between the hand and the object. By learning from the data directly, there is no need to know the properties of the hand, such as kinematics, enabling the method to be utilized with complex hands. The second novel method, probabilistic grasping, combines the fields of tactile exploration and grasp planning. By employing well-known statistical methods and pre-existing knowledge of an object, object properties, such as pose, can be inferred with related uncertainty. This uncertainty is utilized by a grasp planning process which plans for stable grasps under the inferred uncertainty.

Keywords: robotic grasping, robot architectures, tactile sensing, grasp planning

UDC 004.896:621.865.8

Symbols and abbreviations

DOF	Degrees Of Freedom
XML	eXtensible Markup Language
ASM	Abstract State Machine
CSM	Concrete State Machine
SVM	Support Vector Machine
AdaBoost	Adaptive Boosting
RBF	Radial Basis Function
k-NN	k-Nearest Neighbor
GMM	Gaussian Mixture Model
PCA	Principal Component Analysis
LBP	Local Binary Pattern
SDH	Schunk Dextrous Hand
PSO	Particle Swarm Optimization
GP	Gaussian Process
GPR	Gaussian Process Regression
C	Soft-margin parameter for support vector machine
$K(x_i, x_j)$	Support vector machine kernel
$I(x, y)$	Function of sensor taxels
$m_{p,q}$	Raw image moment
O	Object attributes
G	Grasp attributes
S	Stability of a grasp
$m(x)$	Gaussian process mean function
$k(x, x')$	Gaussian process covariance function

Contents

Chapter I

Introduction

While early robots were built for industrial environments and repetive tasks, recent developments in robotics have allowed robots to move into homes and home-like environments such as hospitals. This field of robotics has grown in the past decades into service robotics. This transition was possible only as the sensor capabilities of robots increased, so that collisions could be avoided with obstacles in dynamic environments, including people. However, the current sensor capabilities of commercially available robots are still quite rudimentary compared to, for example, humans. This fact, and the lack of integrated sensor solutions, has limited the use of robots – most popular service robots today are mainly mobile platforms, such as cleaning robots [1] or robots that can transport goods.

One of the most essential skills of humans, the ability to manipulate objects, is still quite far in terms of everyday use in the robotics field. This thesis is a study into robotic grasping, an integral part of the whole field of robotic manipulation. Grasping is important in the sense that it allows the further study of objects; for example, when an object is in the grasp, it is possible to determine the weight of the object or examine the object from views that would not be possible to obtain otherwise. Grasping is also a gateway to many other manipulation actions that require a certain type of a hold on the object, for example tool use, pouring, writing and handing over objects. These actions would not be possible without the preceding grasp.

The focus of this thesis lies in the use of the sense of touch to grasp objects. This focus diverges from a popular direction of research into robotic grasping where mainly vision is used to estimate poses of objects and the robot hand is guided using the information gained only from vision. However, one can imagine the importance of the sense of touch when grasping objects. In humans, the sense of touch in hands is one the most developed senses, and with accompanying experience of the world, allows the grasping of objects even without any visual information. If one were to lose the sense of touch and the proprioceptive sensing, i.e. the ability to sense the joint positions of the fingers, it would significantly degrade the achieved performance in manipulation and grasping and it would also require constant visual attention to the objects and the hand [2]. It is clear that both vision and sense of touch are desirable in a system focused on manipulation and grasping.

Nevertheless, these abilities are complementary and function best at different stages when considering the whole manipulation sequence. One example can be first locating an object, moving to a position to grasp the object, grasping and transporting the object and then placing the object somewhere else. In this type of a sequence, vision plays an important role in localizing both the object and any obstacles, and relating the movement of the arm and the hand with the whole environment. The role of proprioception and the sense of touch is focused on the action of grasping and keeping the object in hand during the object transportation.

In the scenario described above, the advantage of vision is that it is global and, thus, able to provide a large quantity of data from the surrounding world from a distance. This is ideal for object recognition and pose estimation. However, vision is hampered by occlusions and, for example, translucent or transparent objects can be difficult to detect by vision. On the other hand, using the sense of touch, contacts can be detected quite accurately. Through the physical contact, the ambiguities and unknowns in the environment, such as occlusions, can be solved effectively.

Still, there are some issues in robotics that must be solved before the sense of touch can be used widely across different applications and platforms. This problem is the variance in the design and hardware of robots, especially when considering the service robotics field. This general problem is also present in robotic grasping. For example, the number of fingers between robot hands can be different, the size of the hand can vary and the sensors can differ. In this thesis, one of the goals was to minimize the effect of different designs and capabilities that robot hands might have. The methods developed in the thesis can be used in a wide variety of hands, although, a certain set of basic sensor capabilities is required from the hand, most importantly hand proprioception and tactile sensors.

1.1 Objectives

This thesis was started with the question: "How can sensor-based information be used to help an agent cope in an uncertain world?". As the main body of the work was planned to be completed under the GRASP project [3], which concentrated on robotic grasping, this quickly focused the work on robotic hands and the sensors that are available in that space, in addition to grasping itself.

The main objectives were influenced by the plan of the GRASP project, which called for two separate goals:

- Create a system capable of producing manipulation actions on several platforms.
- Utilize low level data gained during grasping with the robotic hand.

The first item on the list calls for the development of a system capable of working with multiple different robotic platforms with different types of embodiments, especially taking into consideration the capabilities of the robotic hands. The objective, regarding the first item, is to advance the existing architectures that have been developed with similar goals in mind.

The second item, on the other hand, requires the development of methods that can transform the low level data coming from the hand to some meaningful information. This process is called symbol anchoring [4]. The objective, in this case, focused quite closely on the stability of the grasp; that is, how the stability can be determined from proprioception and tactile sensing with all types of robotic hands. One can argue that detecting or measuring the extent of the stability when grasping an object is the most useful skill to have, especially if the object will be manipulated in some way after the grasp.

1.2 Contribution and publications

The contributions[1] made in this thesis can be roughly divided into three parts:

- The development of a manipulation architecture for robotic arms and hands.
- A method for estimating the stability of a grasp, through the use of learning algorithms.
- A framework for grasping objects in uncertain environments.

Each part has its own published articles that have appeared in scientific conferences and journals. In addition, initial work on the framework was published in a workshop on manipulation under uncertainty. One of the articles is currently submitted to an international conference.

The first contribution, discussed in chapter 3, is the development of an architecture that is especially suitable for manipulation and grasping [5]. One of the most fundamental goals set for this architecture was embodiment independence, i.e. the architecture should be able to function with any robotic manipulation platform. This goal was achieved by using a two-tier design, where the action is described using a platform independent language which is then translated into executable code, which is specific to individual platforms. The embodiment independendent action was demonstrated on different hardware platforms, making the architecture unique in the demonstrated capability. The architecture also relies heavily on interfaces to actuators and sensors that can be implemented with reasonable effort for different platforms. This feature is also very useful in terms of validating the action in simulation, where the interfaces to the actuators and sensors differ from the real robot, but the produced action should be the same.

The second contribution, discussed in chapter 4, is a novel learning based approach that can estimate the stability of a grasp only using the sense of touch [6, 7, 8]. In this case, proprioception and tactile sensors were used as an input. Because the stability is learned, there is no need to know, for example, the kinematics of the robot end effector, contrary to the traditional grasp analysis methods. This feature enables the approach to be used with any robotic hand – only the relevant sensor data needs to recorded. The underlying methods in this work were well-known machine learning tools, mainly support vector machine and adaptive boosting. However, the performance of these classifiers with

[1]The scientific articles have been written under the name Janne Laaksonen.

different feature representations and different data is evaluated, which gives valuable information on the applicability of the methods to grasping.

The third contribution [9, 10], discussed in chapter 5, is a novel approach to grasping from the point of view of the sense of touch. Given that in many real world environments the poses or other properties of objects remain uncertain even with vision, it is useful not to dismiss this uncertainty. Based on this idea, a framework was developed that can utilize the information from a series of grasps to refine the uncertain initial knowledge of object properties. After each grasp, some uncertainty is still left. The framework enables the use of the whole uncertainty in grasp planning, which is novel compared to most of the state of the art methods developed for grasping under uncertainty. Each grasp in the sequence is also planned according to the estimated uncertainty, and thus, the term "Probabilistic Grasping" used in [9, 10], contrary to other grasp planning approaches that do not take the uncertainty into account. The approch developed in the contribution is suitable to be used with a variety of robotic hands, and with multiple different sensors found in the hands. In the contribution itself, only proprioception of the hand is used.

Concerning the contributions that the author of this thesis made, in [5, 7, 9] the author was the first author and the primary contributor and was responsible for most of the writing, development and experiments. In the article [6], the author of this thesis was responsible for the so-called one-shot recognition, one of the two methods presented for grasp stability estimation. In addition, the author was also involved in the design of experiments and developing the data collection procedure. Preliminary results were published also in [11]. In the article [8], accepted to 2012 International Conference on Biomedical Robotics and Biomechatronics, the author of this thesis was responsible for the theoretical background and applying the learning method to the collected data. The author of this thesis was also responsible for most of the writing, implementation and experiments in the article [10] which is currently submitted to 2012 International Conference on Intelligent Robots and Systems.

1.3 Outline of the thesis

The rest of the thesis is split into six chapters. Chapter 2 is an overview of robotic grasping in general which largely excludes works that are closely related to the the central contributions made in this thesis. The more closely related work will be discussed within the chapters that describe the contributions in detail. These chapters are: chapter 3, where robotic architectures and the contributions to robotic grasping architectures are discussed, chapter 4 describing how the grasp stability can be learned through grasping, and chapter 5, where the use of tactile and proprioceptive sensors is explored in the context of probabilistic grasping. Chapter 6 discusses the contributions in general and possible further research in the area of this thesis. Finally, chapter 7 is a summary of the entire thesis.

Chapter II

Robotic grasping

There are two distinct lines present in robotic grasping. One line is present in current industry, where the manipulator, i.e. the robotic arm, and the end effector, i.e. the robotic hand, of the robot are designed so that an object, for example a product package, can always be "grasped" when the object is present in some preset pose. Grasping in this sense does not resemble human grasping – thus the quotation marks. Instead the end effector can be of any shape and utilize, for example, suction cups. However, the end effectors can be so specific that they may only grasp a single object from a predetermined pose.

The second line, and more recent development, is the rise of more anthropomorphic end effectors, for example [12, 13, 14], which resemble the human hand, but with various numbers of fingers and degrees of freedoms (DOFs). Due to the complex mechanics required for the full 27 DOFs, most robotic hands have fewer degrees of freedom than the human hand, but these end effectors enable more general grasping. Just like the human hand, the end effectors are able to grasp a large variety of objects. The range of objects is, of course, dependent on the end effector design. These types of end effectors enable grasping in more service oriented scenarios, for example, moving groceries from a shopping bag to refrigerator, compared to the industrial end effectors. The downside to the more general end effectors is that there are no guarantees that the object can be grasped successfully each time. For this reason, many types of sensors are usually used to reduce the uncertainty present in dynamic, unstructured environments.

This thesis focuses on the latter line and on the problems that unstructured environments with general end effectors bring. The rest of this chapter reviews the problems and solutions that have been proposed in the context of robotic grasping to give an overview of commonly accepted techniques and methods applied to robotic grasping.

2.1 Grasp analysis and planning

Grasp analysis, at its core, is a geometrical study of hand-object interaction. The purpose of this study is to find the stability of a grasp using accurate models of a robot hand and an object. The stability of the grasp can be determined through contacts between the hand and the object and can be analytically solved for these known contacts. However, some assumptions and approximations about the contacts must be used to simplify the contact modeling.

Different contact models have been developed and the sophistication of the models has increased in the past few decades, as grasping is a complex action which relies on deformable bodies with varying surface properties, which is especially true in human grasping. One of the simplest models used in robotic grasping is the point contact model [15], which assumes that all contacts are represented by a point and that objects are rigid, i.e. non-deformable. In addition, Coulomb friction is assumed at the contact. The Coulomb friction model can be represented as a cone [16], as shown in Figure 2.1, where $\alpha = \tan^{-1} \mu$. The product of μ, the friction, and f_n, the normal force, dictates how much tangential force in relation to the contact can be applied before the contact slips. Instead of a point contact, a soft finger model can also be used [16]. The advantage of the soft finger model is that it is able to model the resistance to torque that is exerted on the contact, contrary to a pure point contact. The taxonomy of contacts, which was first developed by Salisbury [17], includes more contact types in addition to the point and soft finger contacts. More accurate, but more complex models, are the compliant contact models [18, 19] that assume that the fingers of the hand, the object or both can deform under contact.

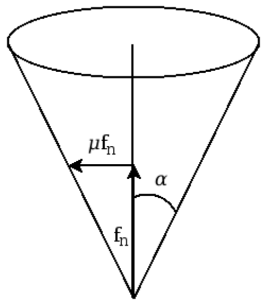


Figure 2.1: Friction cone. Adapted from [16].

Apart from the contact models, the stability of the grasp is analyzed from the perspective of form and force closure. Both of these closures build on the concept of contacts and both describe a set of conditions for the contacts that must be fulfilled. These closure properties are used also outside of robotics; the first study on these properties was published in the 1800s by Reuleaux [20]. For example, form closure is useful in designing static fixtures. Verbally, form closure constrains the object so that the object is fully immobilized by the structure of the hand configuration which requires no force to be exerted by the hand to the object. Force closure, on the other hand, requires force to be applied by the hand and does not guarantee the full immobilization of the object if an external force affects the object in the grasp. Due to this property, force closure grasps also require friction between the contacts. Examples of both grasps are shown in Figure 2.2. Note that in the case of force closure, the normal forces, f_n, and friction generate the friction cones, shown in Figure 2.1, which resist the external forces. More recent work on form and force closures can be found, for example, in [21], where it is proved that force closure does not always imply a stable grasp; [22], showing how many point contacts are needed for form closures of generic 2D and 3D objects. The first study of force and form closure in the context of robotic grasping was introduced in [17], and more surveys and book chapters on the general framework used in this context (contacts and closures) can be found in [23, 24, 25].

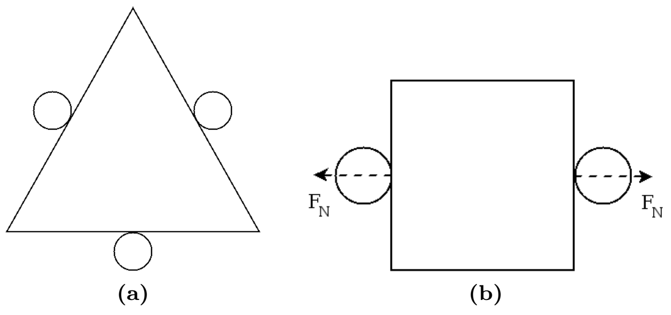


Figure 2.2: (a): Planar form closure grasp using three fingers; (b): Planar force closure grasp using two fingers.

While the above describes the analysis of grasping, no insight was given on how to find stable grasps. The area of grasp planning (or grasp synthesis [24]) has been under intensifying focus for the past few decades. Grasp planning has diverged into two different directions with different assumptions. The first is "traditional" grasp planning, where the object is considered fully known. For example, an accurate 3D-model of the object is available, thus enabling the use of the grasp analysis methods in simulation. To compare different grasps, especially when force closure properties are fulfilled, a criterion is needed to rank the grasps. This criterion is known as the grasp quality measure, and the first grasp quality measures were developed in [26, 27]. These methods estimate the quality through the contacts by estimating how the grasp and the resulting contacts can resist external forces and torques.

The computational load in this case falls to object and pose recognition and grasp planning. The object recognition task can be solved with (stereo) vision or other sensors, such as laser scanners [28], that can directly return point clouds of the object and surrounding scene. If such point clouds cannot be constructed, for example, in case a monocular vision system is used, the object can also be recognized by only its appearance from visual data. Otherwise, as the object is known, the point cloud can be matched to either a single object [29] or to a set of objects [30, 28] with pose estimation.

Grasp planning can be one of the most time consuming tasks in this context, as selecting a force closure grasp from all of the possible grasps in the 6 DOF space is time consuming. To tackle this problem, a variety of techniques have been developed which add some heuristics, for example from human beings, that are based on the object shape. Still, due to the large number of grasps that have to be evaluated, grasping simulators, such as GraspIt! [31], OpenGRASP [32] and OpenRAVE [33], have been in development for the past ten years.

One common technique in grasp planning is to approximate the object model with simpler shapes, such as boxes [34], superquadrics [35] or other primitives [36]. The object can then be grasped based on the simpler shapes instead of the potentially complex shape of the original. Other approaches can, for example, analyze the 3D shape of the object [37] or apply heuristics observed in humans [38] to find good grasps. The grasp analysis and planning is usually done off-line and the results are saved on a database, and a suitable grasp can then be fetched from the database, with reachability and other affordances taken into consideration. Grasp acquisition [24] is then performed according to the chosen grasp. Usually, the actual grasping is open-loop, which requires robust and accurate object pose estimation methods.

The second direction for grasp planning targets unknown or novel objects that the robot has not seen before. In most cases, this assumption implies that, instead of analyzing the grasp per se, an approach vector and/or grasping point is computed, for example, from an image with possible depth information given. Discarding the use of the grasp planning and analysis methods presented previously means that systems must be either trained beforehand or that the system has some intrinsic heuristics on how to grasp objects to account for the uncertainty that rises from the unknown objects. For example, one can detect good points to grasp based on the image and depth data of the object [39, 40, 41]. Based on a similar idea, an autonomous system that can learn good approach directions from visual data was developed in [42].

In this case, feedback from the hand sensors (haptic, tactile, proprioceptive) or vision can also be used to limit the effect of the uncertainty. Thus, the initial approach vector can be thought of as the starting point for further refinement of the grasp using the sense of touch. Research in this area is presented in section 2.2.

There are few approaches that combine both the known and unknown object model approaches by generalizing from the known objects to the unknown objects. One such approach [43] assumes that an unknown object can be represented by a deformable known object, although, in this case, the problem of grasp synthesis on the deformed object is still open. Another approach [44] uses a precomputed database of good grasps on known objects and tries to find the object that best matches with a given unknown object. The approach can work also with incomplete geometric models [45].

It is also important to note the study of underactuated hands. In the case of underactuated hands, the hand might have many degrees of freedom, but some or most of the DOFs are either passive or coupled together, in contrast to a fully actuated hand where each DOF can be individually controlled. Recently, progress has been made using simulations that allow changing of parameters of the hand [46]. In the study, the parameters included the tendon routing of the hand and the stiffness of the joints. Simulation allows the hand to grasp a set of objects and selection of the parameters based on the results of the grasp attempts. The parameters can then be used on a hand that is actually built. Underactuated hands [46, 47] sidestep the grasp planning problem by moving the problem on to the design of the hand that tries to maximize the stability across objects. This property makes underactuated hands more suitable for the latter of the two described directions of grasping. One can also forego the design of the hand completely and choose a fully compliant hand, such as the universal gripper [48], which works by jamming granular matter. However, in general, underactuated hands lose dexterity, i.e. in-hand manipulation capability, compared to fully actuated hands.

The grasp planning methods described above determine only the robot hand pose in relation to the target object. To be able to complete the grasp, the hand must first be moved to the selected pose. This is the field of motion planning. Depending on the type of the robot and the indicated pose of the hand, the robot might even need to move before attempting the grasp, which is usually referred to as path planning. However, due to the topic of the thesis, a short introduction is given only for arm motion planning.

In many approaches, when planning a motion for a robot arm, accurate simulation of the robot environment (consisting of the target object, other objects and obstacles) and the robot itself is required. While the kinematics and dynamics of most industrial robot arms and hands are known, the environment requires more effort to be accurately realized in a simulator. One straightforward method is to use a point cloud, obtained with stereo vision or laser scanners. The point cloud can then be triangularized into a mesh [49, 50], which can be represented in a simulator. However, with only one view of the whole scene, the point clouds do not represent the whole environment. Some additional techniques which attempt to model the uncertainty of the environment in these cases [51] have been presented, and a method has been developed to estimate the symmetry found in objects to generate the missing backsides of objects [52]. A common approach is also to model the principal environment of the robot [53, 54], such as a kitchen. Any unknown or known objects in the environment are then easier to separate from the known environment.

Once the environment, including any obstacles, is modeled, then a suitable path or motion for the robot arm and hand can be computed. While many algorithms exist for path planning, rapidly-exploring random tree (RRT) algorithms [55] and probabilistic roadmap (PRM) algorithms [56] are often used due to their applicability to motion and path planning problems where the space of possible motions is large. As the names suggest, both techniques rely on probability to find a solution to the motion planning problem. Furthermore, RRT algorithms explore the whole search space due to the space filling property of the search. Compared to other search algorithms without this property, RRT guarantees that a solution can be found, if a solution exists. RRTs can also incorporate dynamical states, such as velocity, in the search space. Many improvements have been made to these algorithms in the recent years, for example [57], which eliminates parameter tuning from rapidly-exploring dense trees (RDT), a generalization of RRT, T-RRT [58], which incorporates automatic parameter tuning and works in continuous spaces, and GradienT-RRT [59], which improves the T-RRT further.

2.2 Sensor-based grasping and manipulation control

Section 2.1 discussed the use of sensors in the process of grasp planning. However, the different sensor modalities can also be used during the grasp acquisition itself or in other manipulation tasks. This type of sensor use is especially useful when the objects that are being manipulated are either completely unknown or only partially known, for example when the mass distribution of the object is unknown. The sensors allow the use of feedback from the manipulation action and correction of the action based on the received feedback. However, an on-line response is required to gain benefit from the sensor feedback.

The development in this area is fairly recent, although research has been conducted already in the 1990s [60, 61]. This has been due to a lack of suitable sensors and computing power. The increase in computing power, especially general-purpose computation on graphics hardware (GPGPU) [62], has made it possible to use computationally expensive methods for tracking objects and hands during manipulation or grasping in real time.

Research on tactile sensors has also been on the rise in the past decade, which has resulted in the availability of commercial tactile sensors suitable for robotics [63, 64, 65] in the past five to ten years. Although the goal of a tactile sensor is to sense or measure the force affecting the sensor area, many different types of techniques are utilized in practice. These techniques can be categorized into different paradigms: resistive, capacitative, piezoelectric and optical. These are commonly used but also other types of techniques can be found. Resistive sensors usually employ some type of conductive material that can be compressed. The sensor itself then measures the resistance, which decreases with more compression. An example of these types of sensors are the Weiss tactile sensors [64, 66]. Capacitative tactile sensors measure the change of capacitance between two surfaces that change their relative position under force. For example, the RoboSkin EU project has developed these types of sensors [67]. Piezoelectric sensors are based on the piezoelectric effect; in other words, pressure applied to a piezoelectric material creates an electrical field. Examples of these sensors can also be found within the RoboSkin project [68]. Finally, the optical tactile sensors use some type of vision to determine the deformation of some surface to measure the force affecting that surface. One example of an optical tactile sensor can be found in [69]. For a more complete review of the tactile sensing technologies, see [70].

The use of multimodal sensors in robotic manipulation and grasping has been studied, for example, in [71, 72]. These doctoral theses explore sensor fusion for robot control in manipulation tasks, fusing proprioception, force and vision sensing. Numerous works also exists somewhere in between, combining a few modalities or just using one sensor modality.

For tracking objects during manipulation using vision, particle filter based methods [73, 74] have shown to be useful, as they can track the object on-line as well as off-line, enabling further refinement of the grasp. On the other hand, there are also methods that can track the end effector itself [75, 76]. A more advanced method, using particle swarm optimization (PSO) instead of a particle filter, is able to track both the human hand and a parametrizable model on-line [77]. Ideally, this approach can also be extended to robotic hands, as robotic hands are currently built to be rigid compared to human hand models. Thus, the model used for representing the end effector is easier to build.

The methods described in the previous paragraph are principally based on vision. Methods that utilize mainly the sense of touch will be discussed later, in the context of the contributions of this thesis in chapters 4 and 5, but there are methods that combine the sense of touch with vision. For example, [78] extends the work presented in contribution [6] to include vision in the estimation of the stability of a grasp.

Chapter III

Action Abstraction

The purpose of this chapter is to describe an architecture that can abstract an action to achieve the actual goal of the action itself. Why is this abstraction needed? To elaborate the problem, some examples of grippers and more complex end effectors are shown in Figure 3.1. From the images, one can see the problem that exists within the robotics community – non-standard hardware and equipment. For example, the capabilities of the end effectors are different due to the number of fingers, and the same applies to the manipulators; some have seven or more DOF while some have only six DOFs or even less. The different hardware lead to different actions, even if the task is the same.

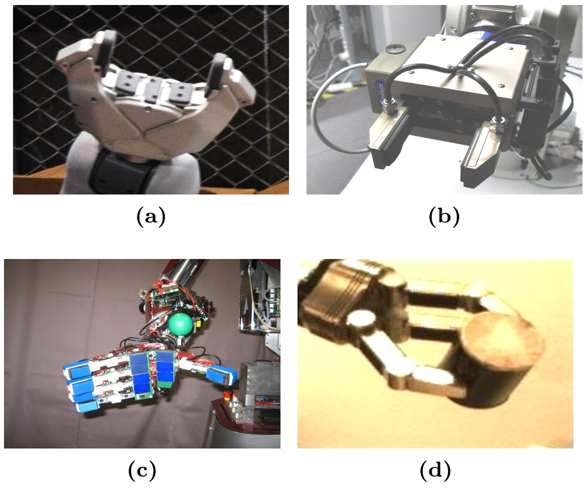


Figure 3.1: (a) Gripper of the Willow Garage PR2 robot (image by Timothy Vollmer); (b) WRT-102 gripper used in the experiments; (c) ARMAR-IIIb end effector; (d) Schunk Dextrous Hand.

Moreover, even if the structure of the robot, i.e. the physical construct, is the same, the robot may have different sensors embedded into it. For example, the robot might have a wrist mounted force sensor, and the tactile sensors embedded in the end effector can be obtained from multiple manufacturers. Therefore, especially in a research environment, a single robot could be a unique piece of equipment not found anywhere else.

These problems present a dilemma for manipulation and grasping. On the one hand, there is a need to access and use the sensors of the robot to be able to operate in uncertain and unstructured environments, but on the other hand, the sensor configurations might be different between robots that try to accomplish a specific action. To tackle this problem, abstraction is needed to separate the robot platform, also referred to as robot embodiment, from the action itself. This can be called knowledge transfer, as the robot can be interchangeable but the action itself is not. One can also consider a human as a robot in this scenario; thus, it should be possible to learn an action by creating a description of a manipulation or grasping action and then transfer the action to any robot embodiment.

3.1 Robot architectures

A number of robot architectures have been presented previously, for both manipulation and for more general use. According to the topic of the thesis, the focus is on the manipulation architectures. The concept of primitive skills is central in many of the works. Similarly, the use of discrete states to divide a manipulation action into parts or subactions is seen in many architectures.

Milighetti et al. [79] presented an architecture which uses primitive skills. The primitive skills combine into skills, which in turn form a complete task. Each primitive skill is selected by heuristic selection out of many possible primitive skills, based on the sensor signals. A neural network is used to detect the change between the skills. Each primitive skill is based on a separate controller. While the basic idea of hierarchical decomposition is similar to the architecture presented in this paper, in the approach of Milighetti et al. there is no possibility to adapt the primitive skills themselves.

Haidacher et al. [80] have introduced an architecture for the DLR Hand II. The architecture is based on different levels of complexity, which handle different aspects of the control. Again, the concept of hierarchical decomposition is central, but the architecture is limited to a single hand and the adaptiveness of the architecture has to be implemented at the highest level, as the lower levels are statically defined.

Han et al. [81] present a control architecture for multi-fingered manipulation. As previously, the architecture is based on different levels that handle control from planning to actual joint control. The problem with the architecture is the lack of adaptation, as the architecture shows that only predetermined architectural components, such as low level controllers, are available to use. In addition, the architecture does not consider the robotic arm, only the hand.

Hybrid discrete-continuous control architectures for manipulation, such as [82] and [83], separate the control phases according to the state of the manipulator. This is achieved by using discrete events to classify the manipulation configuration and using continuous states to control the dynamic behavior in different configurations. This type of architecture is suitable for both low-level control [83] and for a complete control architecture [82]. Petersson et al. [82] demonstrate a control architecture for a mobile manipulator based on behaviors. The actual manipulator behavior is modelled as a sequence of configurable primitive actions, which can be freely defined. These primitives can be chained together using a hybrid automaton to form an action. Although the architecture has some elements desired from a service robotics architecture, such as hardware independence, it lacks the sensor-based approach required to cope in uncertain environments. For example, there is no mention of failure detection using the available sensors.

Another mobile manipulator architecture by Chang and Fu [84], is also based on hybrid discrete-continuous control architecture. However, the architecture is more limited than in [82], only consisting of a pre-determined set of states, which control the manipulator. These states can be configured for different manipulation tasks. Aramaki et al. [85] have also used automata to control a humanoid robot at a low level.

For a non-mobile manipulator, Prats et al. [86], presented a comprehensive system for controlling manipulation. The system also uses automata to control the progress of actions, by separating the primitive actions into the states of the automata. One of the defining features of the architecture is that each state of the automata can be a primitive action or an automaton. This feature can be used to create complex actions. However, the problem of hardware independence is not discussed.

One can also consider communication architectures, such as ROS (Robot Operating System) [87] or YARP (Yet Another Robot Platform) [88], but these only provide a communication framework for the robot platforms and can encompass more than just manipulation. Although these architectures provide a unified way to relay information between communication nodes, the architectures themselves are low level and require additional work to build working systems.

Most of the described manipulation architectures have one common element, the use of automata in determining the current state of the control. This approach is also used in the proposed approach which is described in the following sections. Note that most of the architectures do not discuss hardware independency or how it would affect the described architectures, which was one of the central goals of the abstraction architecture that will be presented next.

3.2 Abstraction architecture

The design of the proposed abstraction architecture is based on the principle of sensor use in uncertain environments with multiple robot embodiments in manipulation scenarios. To abstract the action from the robot embodiment, the architecture is divided according to Figure 3.2. The figure shows how the abstract information is separated from the embodiment specific information which facilitates the use of multiple embodiments for abstract actions and still allows the use of embodiment specific sensors within the action. The two components residing at the edges in the architecture are divided into abstract and concrete actions to denote the effect of the embodiment. On the abstract side, everything is abstracted, and thus, the embodiment has no effect there. The concrete action, on the other hand, is native to the embodiment and the cannot be executed in any other embodiment or platform. The middle component is responsible of handling the change from the abstracted action to the concrete action. This component will be referred to in the next sections as the translator.

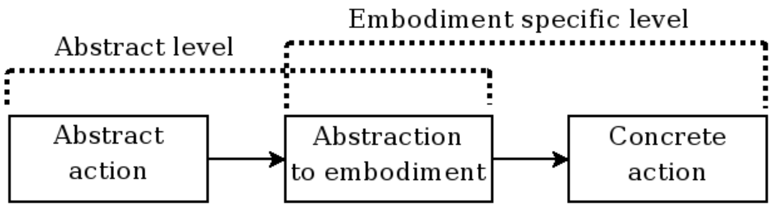


Figure 3.2: Levels of the abstraction architecture.

Before going into the details of the proposed approach, the terminology used in the following sections is explained. The term *abstraction architecture* describes the entire proposed approach. This should not be confused with the control architecture, which is a part of the abstraction architecture related to the actual execution of the actions. Figure 3.3 shows a general hierarchical decomposition of planning and control. The hierarchy consists of three levels: task, action and primitive. The *task* is the highest level of abstraction, representing a semantically meaningful task such as emptying a shopping bag. The task comprises of a sequence of *actions*, which represent subtasks, such as moving an object from one location to another. Actions consist of *primitives*, or primitive actions, which are the lowest level of control in the proposed architecture. A more accurate definition for a primitive action used in the proposed architecture is that each primitive action is implemented using a single low-level controller, which is responsible for the actual control of robot hardware.

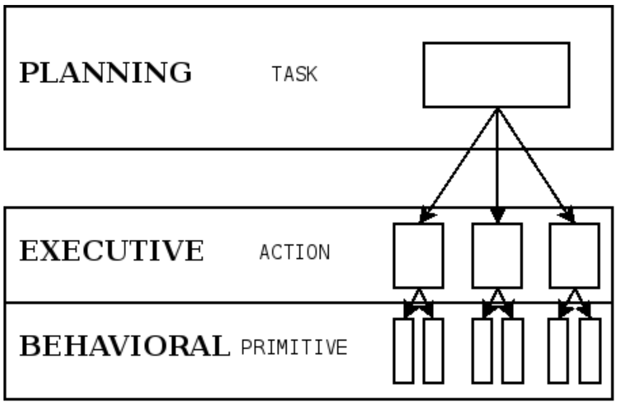


Figure 3.3: Planning and control levels.

The abstraction architecture presented in this thesis will focus only on the action and primitive levels shown in Figure 3.3, that is, the actual on-line part of control instead of task planning, which might be performed off-line. The architecture itself has elements of both behavioral and executive levels discussed in [89]. It should be noted that the behavioral control is not considered in the Brooksian sense [90]; instead, the behavioral level considers primitive actions which can be executed with traditional control theory. It is possible to adapt to different tasks and different hardware at the two lower levels using a set of attributes that are implemented in the actions and in the primitive actions of the abstraction architecture.

3.2.1 Describing an action

As was characterized in section 3.1, many have utilized state machines or automata to describe and control the actions of a robot. This approach is also adopted in this thesis in the form of a finite-state machine (FSM), also known as a finite-state automaton or a deterministic finite automaton which is described, for example, in [91]. This basic structure is used for both the abstract action and the concrete action and the corresponding state machines are called the abstract state machine (ASM) and concrete state machine (CSM). Note that ASM can also refer to the theory of abstract state machines, a generalization of FSM. However, here it is only used to denote the finite-state machine of the abstract action.

As the whole abstraction architecture focuses on grasping, a set of related *attributes* was chosen for the ASM:

- **success:** The success end state of the state machine.
- **failure:** The failure end state of the state machine.
- **move:** Moving the manipulator without an object.
- **transport:** Moving the manipulator with an object.
- **grasp:** Grasp the object.
- **release:** Release the object.

These attributes define the primitives for each state and consequently the type of controllers for the CSM. For example, in a move state, a controller for the end effector may not be needed. While the attribute list itself is short, the attributes are able to represent the core functionality required for actions that require grasping, such as pick-and-place actions. In addition, each state can carry multiple *properties*. These properties further specify the semantic meaning of each state and, thus, give more information for the translation into CSM. Each transition has also a set of properties defining when the transition will trigger. These properties are listed in Table 3.1 and Table 3.2 with their respective descriptions.

Utilizing the properties found in Table 3.1, each state can be adapted to some degree. For example, it is easy to define multiple different grasp types using the *hand_ shape* property, such as cylindrical grasp or pinch grasp, or define a movement type for the manipulator. The advantage of this adaptability is that the structure of the state machine itself can remain unchanged through multiple actions, even if, for example, the object or the hand changes between executions of the action.

Table 3.1: State properties.

State	Description
movement	Describes the desired type of movement
hand_shape	Describes the desired hand shape
path	Describes a path for the end effector

Table 3.2: Transition properties.

Transition	Description
success	Controllers have reached a target value
grasp_stable	Grasp is stable
grasp_lost	Grasp is lost, i.e. the grasp is unstable
finger_contact	Contact is detected in a finger
finger_contact_lost	Contact is not detected in a finger
timeout	Preset time has passed in a state
collision	Collision is detected
hardware_failure	Hardware failure is detected

Figure 3.4 gives a simple example of both ASM and CSM. The two state machines corresponds to each other. The ASM describes a simple two state FSM, the first state being a move state which then transitions to a grasp state, given that the move is successful. The figure also shows how the abstract states are translated into actual controllers that are used in the concrete action. This type of structure found in the concrete action is called hybrid discrete-continuous control architecture, as was discussed in section 3.1. Each state can comprise one or more controllers that are run continuously until the transition condition or conditions are triggered.

While the graphical representation is useful to visualize the FSM, in the abstraction architecture, the abstract action is represented in eXtensible Markup Language (XML). XML was chosen for this task because it is a standard and many tools have been already developed to parse and generate XML. In addition, XML is human readable, as can be seen from Figure 3.5. This example XML description contains all of the required elements for an abstract action, but is shortened by removing some common transitions, for example, transitions to failure due to hardware failure or time limits. One of the key definitions, in addition to the states and transitions, is the object definition. The object definition can include several types of information, such as the pose, weight and even a known geometrical model if one exists. The ASM can also include information about the environment, for example, possible obstacles in the vicinity of the object, but this information was not used in the experiments.

3.2.2 Translating from abstract state machine to concrete state machine

The translation process is what combines the abstract state machine and the embodiment specific state machine. The translation takes the abstract state machine as an input, and

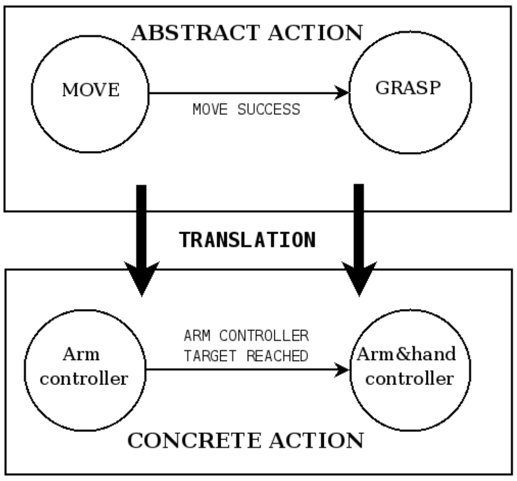


Figure 3.4: A simple example of the abstract and concrete state machines.

translates the abstract state machine into a concrete state machine. The translation process is depicted in Figure 3.6.

As can be seen from Figure 3.6, the translation component needs the specified inputs. The inputs define the target configuration of the translation process, i.e. the target platform. Furthermore, any platform specific transitions and primitive controllers have to be made available to the translation component at this stage to enable full use of the hardware platform. The translator itself is a software component that has logic based on the state and transition attributes and properties. Furthermore, the object definition can be used during the translation, for example, to estimate the required grasping force based on the object weight. The structure of the implemented translator is described in section 3.3.

The benefit of this arrangement is that the only hardware dependent blocks shown in the figure are the primitive controllers and transitions that are platform specific. A set of universal primitive controllers and transitions can be utilized in addition to the hardware specific components. Also the critical requirement of real-time operation for sensor-based control is fulfilled, as the concrete state machine can be run in the platform's native environment without any additional overheads from maintaining hardware independence.

3.3 Implementation of the abstraction architecture

The abstraction architecture is built on the *control architecture* which is responsible for the execution of actions. The control architecture is depicted in Figure 3.7. One of the main features of the control architecture design is the inclusion of two communication interfaces; on-line and off-line. These interfaces serve different purposes and are used during different phases of the action execution. The off-line communication interface handles the communication of the abstract action to the high-level controller which includes the translation component. The result of the translation, the concrete action or the concrete state machine, is then used internally by the high-level control to execute

```
<statemachine>
    <object>
        <pose>-1 0 0 0.2 0 0 1 0.7 0 1 0 0.20 0 0 0 1</pose>
        <weight>0.5</weight>
        <shape>cup.mdl</shape>
        <class>cylinder</class>
    </object>

    <state name="approach" type="move">
        <movement>free</movement>
        <hand_shape>open</hand_shape>
        <path>
            <pose>-1 0 0 0.2 0 0 1 0.65 0 1 0 0.25 0 0 0 1</pose>
        </path>
    </state>
    <state name="preshape_hand" type="move">
        <movement>guarded</movement>
        <hand_shape>pinch_grasp_preshape</hand_shape>
    </state>
    <state name="grasp_object" type="grasp">
        <movement>guarded</movement>
        <hand_shape>pinch_grasp</hand_shape>
    </state>
    <state name="lift_object" type="transport">
        <movement>free</movement>
        <hand_shape>pinch_grasp</hand_shape>
        <path>
            <pose>-1 0 0 0.2 0 0 1 0.7 0 1 0 0.25 0 0 0 1</pose>
            <pose>0 0 1 0.7 1 0 0 0.1 0 1 0 0.25 0 0 0 1</pose>
            <pose>1 0 0 0.2 0 0 -1 -0.7 0 1 0 0.35 0 0 0 1</pose>
        </path>
    </state>
    <state name="put_object" type="transport">
        <movement>guarded</movement>
        <hand_shape>pinch_grasp</hand_shape>
        <path>
            <pose>1 0 0 0.2 0 0 -1 -0.8 0 1 0 0.35 0 0 0 1</pose>
        </path>
    </state>
    <state name="release_object" type="release">
        <movement>guarded</movement>
        <hand_shape>open</hand_shape>
    </state>
    <state name="move_from_object" type="move">
        <movement>guarded</movement>
        <hand_shape>open</hand_shape>
        <path>
            <pose>1 0 0 0.2 0 0 -1 -0.60 0 1 0 0.35 0 0 0 1</pose>
        </path>
    </state>
    <state name="success_end" type="success">
    </state>
    <state name="fail_end" type="failure">
    </state>
    <transition origin="approach" destination="preshape_hand">
        <success/>
    </transition>
    <transition origin="preshape_hand" destination="grasp_object">
        <success/>
    </transition>
    <transition origin="grasp_object" destination="lift_object">
        <success/>
        <grasp_stable/>
    </transition>
    <transition origin="lift_object" destination="fail_end">
        <grasp_lost/>
    </transition>
    <transition origin="lift_object" destination="put_object">
        <success/>
        <grasp_stable/>
    </transition>
    <transition origin="put_object" destination="fail_end">
        <grasp_lost/>
    </transition>
    <transition origin="put_object" destination="release_object">
        <success/>
        <grasp_stable/>
    </transition>
    <transition origin="release_object" destination="move_from_object">
        <success/>
    </transition>
    <transition origin="move_from_object" destination="success_end">
        <success/>
    </transition>
</statemachine>
```

Figure 3.5: An abstract state machine.

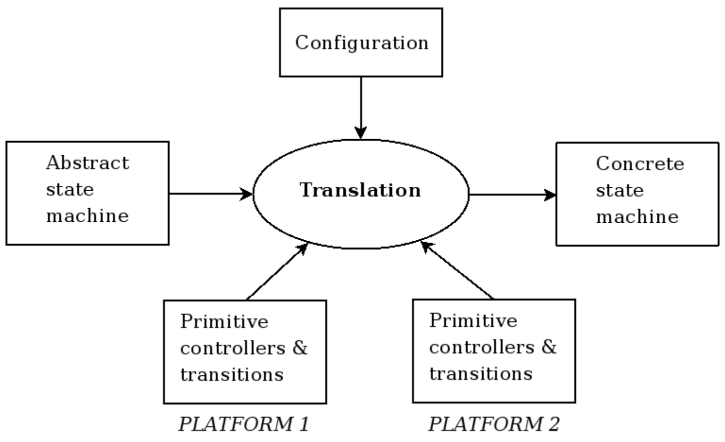


Figure 3.6: Translation process.

the action accordingly. The on-line interface, on the other hand, is used during the execution to transfer sensor data from all of the sensors that are available to a particular platform. The interface is also used to transmit control data to the actuators of the platform. Note, that all of the components that need sensor data, primitive controllers and the transitions, can access all of the sensor data.

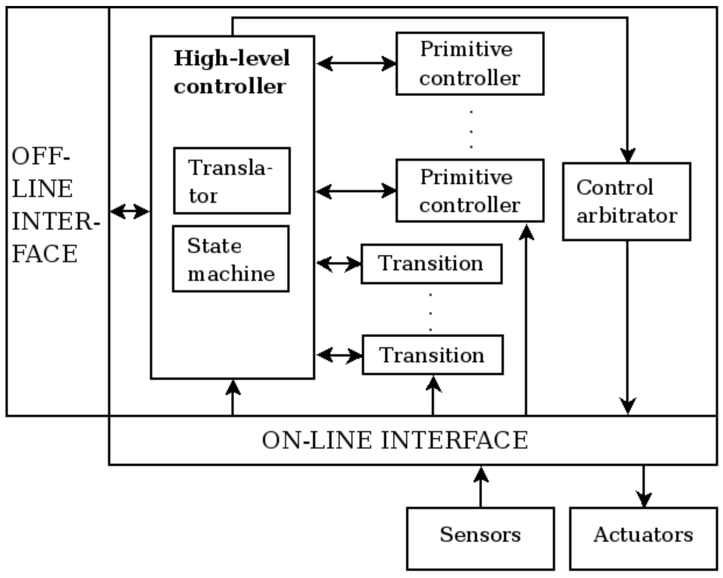


Figure 3.7: Control architecture design.

The second main feature of the control architecture is the separation of the primitive controllers from the high-level controller which separates the execution of the action from the logic of the concrete state machine. The high-level controller acts as a mediator between the components and is responsible for syncronizing the execution of all the primitive controllers and the transitions.

In the beginning of the development of the abstraction architecture, it was decided that the architecture would be built on C++ and a Linux operating system due to the prevalence of both in the GRASP project. Thus, the environment of the abstract architecture is based on these assumptions. Further along in the development, OpenRAVE [92] was adopted in the GRASP project as the simulaton engine which lead to the integration of the abstraction architecture to the OpenRAVE as one of the controller plugins. Although the extra overheads caused some problems, such as maintaining compatibility with new versions, abstraction architecture is able to use the interfaces defined in OpenRAVE, such as the sensor interfaces. In addition, the abstraction architecture can be controlled from Python scripts or Matlab using the standard OpenRAVE plugin interfaces. Next, a detailed description for all of the components and their purpose in the control architecture is given. The most important components have attached Unified Modeling Language (UML) class diagrams to demonstrate the implementation.

High-level controller

The high-level controller is the most complex component in the system as, it includes the translation component and the state machine. This component takes the abstract state machine as input and utilizes the integrated translation component to generate an executable concrete state machine, consisting of the primitive controllers and transitions. After the concrete state machine is generated, the high-level controller initiates the state machine. When an end state is reached – either success or failure – the high-level controller can report back the states that were traversed during the execution of the CSM. The high-level controller was implemented on top of the OpenRAVE controller plugin structure.

Primitive controller

The framework for the primitive controllers is simple, consisting mainly of a method to return control output from the primitive controller based on the sensor data and methods for setting and getting attributes of the primitive controller. Additionally, a success method is defined that enables transitions to query the primitive controllers if their target has been reached successfully. The framework leaves much room to implement any type of control scheme that is desired. The attributes can be used to simply define a target for a controller, for example, a pose for a controller working in the Cartesian space, but the attributes can be used to enable more general use of the primitive controller; for example, a variable number of joints can be controlled using just one primitive controller. The control output is returned to the high-level controller in a structure containing the control output, the priority of the control output for arbitration, the type of the control output, for example, Cartesian manipulator control, and a control mask describing which DOF's of the robot manipulator are controlled by the primitive controller. Each state of the concrete state machine can run multiple primitive controllers. However, in a normal situation, one primitive controller is used for the manipulator and one for the end effector. The UML class diagram for both the primitive controller and the control data structure is shown in Figure 3.8.

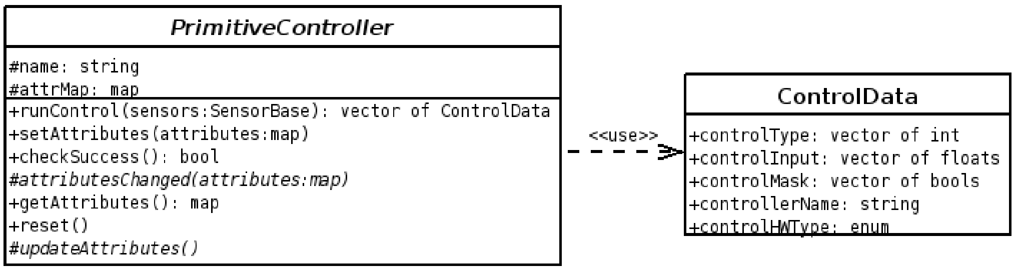

PrimitiveController
#name: string
#attrMap: map
+runControl(sensors:SensorBase): vector of ControlData
+setAttributes(attributes:map)
+checkSuccess(): bool
#attributesChanged(attributes:map)
+getAttributes(): map
+reset()
#updateAttributes()
<<use>>
ControlData
+controlType: vector of int
+controlInput: vector of floats
+controlMask: vector of bools
+controllerName: string
+controlHWType: enum

Figure 3.8: Primitive controller and control data design.

Transition

As the name suggests, transitions represent each transition found in the abstract state machine. When the transitions are translated into the concrete state machine, each transition can comprise many transition conditions defined in the abstract state machine. The framework for these transition conditions is very similar to the primitive controllers; instead of control output, each transition condition must return whether the condition is true or not. A transition can only be made when all of the conditions in it are true. The implementations for both the transitions and transition conditions are shown in Figure 3.9.

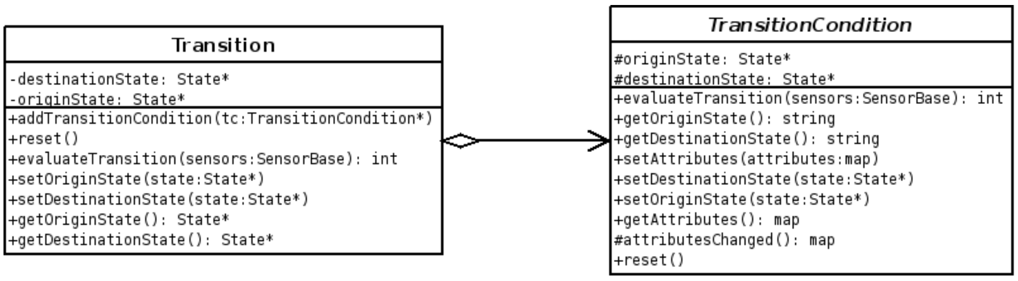

Transition
-destinationState: State*
-originState: State*
+addTransitionCondition(tc:TransitionCondition*)
+reset()
+evaluateTransition(sensors:SensorBase): int
+setOriginState(state:State*)
+setDestinationState(state:State*)
+getOriginState(): State*
+getDestinationState(): State*
TransitionCondition
#originState: State*
#destinationState: State*
+evaluateTransition(sensors:SensorBase): int
+getOriginState(): string
+getDestinationState(): string
+setAttributes(attributes:map)
+setDestinationState(state:State*)
+setOriginState(state:State*)
+getAttributes(): map
#attributesChanged(): map
+reset()

Figure 3.9: Transition and transition condition UML class diagrams.

Control Arbitrator

The control arbitrator can be used in cases where multiple control outputs from the primitive controllers is expected; for example, one controller gives output only if a collision is detected. The control arbitrator can use the data in the control output to decide how to control the DOFs available to the platform. Again, in a normal situation, the arbitrator can be implemented to just pass the output from the primitive controllers to the actuators.

Translator

The translator is embedded within the high-level controller so that it has access to both the abstract state machine through the off-line communication interface, and the concrete state machine. Currently, the translator is implemented for grasping and moving objects in a pick-and-place scenario. While the translator component itself is platform independent, the translator utilizes the abstract factory design pattern to query platfrom dependent primitive controllers and transition conditions. Each platform is represented by an abstract factory, and the factory contains the logic that finds corresponding primitive controllers and transition conditions with the abstract state machine states and transitions. This arrangement enables each platform to be fully exploited and at the same time offers a fall back for platform deficiencies; for example, if a cylindrical hand shape is desired for the grasp in the abstract state machine, a 1 DOF gripper can fall back to just a default grasp. Although the abstract factories are used for individual primitive controllers and transition conditions, the structure of the concrete state machine that is output to the high-level controller is always defined by the translator component itself. This ensures that a minimal amount of work has to be done to implement each new platform, as only a limited set of states and transitions were defined in section 3.2.1, even if the space of possible abstract state machines is large. The translator component with the abstract state machine component is given in Figure 3.10. The abstract state machine in the figure is a representation of the XML abstract state machine and includes all of the information from the XML file.

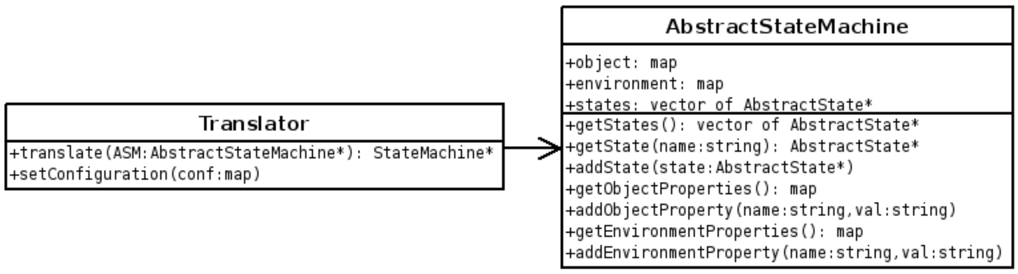

Translator
+translate(ASM:AbstractStateMachine*): StateMachine*
+setConfiguration(conf:map)
AbstractStateMachine
+object: map
+environment: map
+states: vector of AbstractState*
+getStates(): vector of AbstractState*
+getState(name:string): AbstractState*
+addState(state:AbstractState*)
+getObjectProperties(): map
+addObjectProperty(name:string,val:string)
+getEnvironmentProperties(): map
+addEnvironmentProperty(name:string,val:string)

Figure 3.10: Translator and abstract state machine UML class diagrams.

State machine

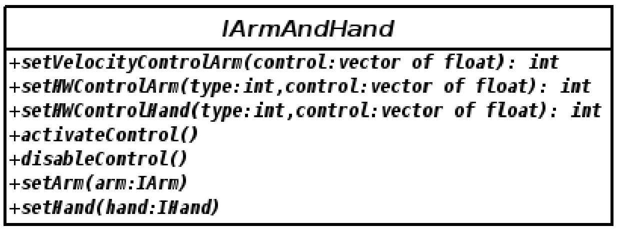
This component represents the concrete state machine. The state machine has access to all of the code, i.e. the primitive controllers and transitions, that is needed to run the whole state machine from the start to the end states. The state machine component handles all of the logic of a finite state machine.

Sensors

Sensors are used by the primitive controllers for control and by the transition conditions to check whether the condition has been fulfilled. The sensors are based on the OpenRAVE definition of sensors, although some modifications were made to maximize the usefulness of the sensor interfaces to both the primitive controllers and transition conditions by placing a type variable for each sensor. This allows the controllers and conditions to check for a certain type of sensor within all sensors made available. For example, the implementation of a time sensor, i.e. a clock, can be different between platforms: consider, for example, a real-time clock versus a clock in simulation. As long as the type and the data are similar between the platforms, the controllers and transitions can be adapted to different platforms.

Actuators

The path to the actual actuators that control the robot is defined through the on-line interface, specifically, the actuator interface. The interface currently consists only of the end effector or the hand of the robot and the arm of the robot. Both of these can be separately defined, allowing a number of combinations with different hands and arms. Simulated actuators were made available to the OpenRAVE by the OpenGRASP project [32] to simulate the dynamic behavior of the real actuators. These actuators were integrated into the control architecture through the actuator interface. The actuator interface is given in Figure 3.11



```
IArmAndHand
+setVelocityControlArm(control:vector of float): int
+setHWControlArm(type:int,control:vector of float): int
+setHWControlHand(type:int,control:vector of float): int
+activateControl()
+disableControl()
+setArm(arm:IArm)
+setHand(hand:IHand)
```

Figure 3.11: Actuator UML class diagrams.

3.4 Experiments

The abstraction architecture was demonstrated on two platforms which differ in their kinematics, control, and sensory capabilities. The first platform is a Melfa RV-3SB robot arm with a Weiss WRT-102 parallel jaw gripper [64]. The WRT-102 is built from the Schunck PG70 gripper. The WRT-102 enables the sense of touch by adding two tactile sensors to the gripper's two fingers. The arm has 6 DOFs and the gripper has 1 DOF which controls the width of the grip. The grasping force is controlled by the feedback from the tactile sensors. Also the stability of the grasp is determined from the tactile sensor feedback.

The second platform consists of a Mitsubishi PA-10 arm with 7 DOF mounted on an Active Media PowerBot mobile robot. The manipulator is is equipped with a three-fingered Barrett hand and a JR3 force/torque and acceleration sensor mounted on the wrist. The Barrett hand has been improved by adding to the fingertips arrays of tactile sensors. Each finger of the hand has a built-in strain sensor. The JR3 is a 12 DOF sensor that measures force, torque and acceleration in each direction of the space. The finger-force sensors are used to stop closing the fingers when the object is touched. Javier Felip (one of the co-authors of [5]) developed the platform specific controllers and transitions for this platform.

3.4.1 Demonstration of platform independence

One of the key challenges that the abstraction architecture addresses is how to combine the need to have sensor based information which is highly coupled to the embodiment and abstraction of the action. To show that that abstraction architecture is able to cope with this problem, sensor-based grasping of objects is demonstrated on the two platforms described above. Using the same abstract instructions, i.e. , the abstract state machine, the same action was executed on the two platforms, and the embodiment specific sensors were utilized to grasp the object.

Figure 3.13, shows snapshots of an action being executed on both platforms. The abstract action contains seven primitive actions: approach, preshape, grasp, lift and move, move down, release and move away. The two objects used in the demonstrations are normal household items: a detergent bottle and a salt container. Both objects have the same mass, 0.5 kg, and are shown in Figure 3.12. In addition to the objects shown, the sensor-based grasping has been demonstrated in the systems with several other similar household objects. As shown in Figure 3.13, using the same abstract state machine for both platforms shows that the abstraction can be effectively employed across robot embodiments.

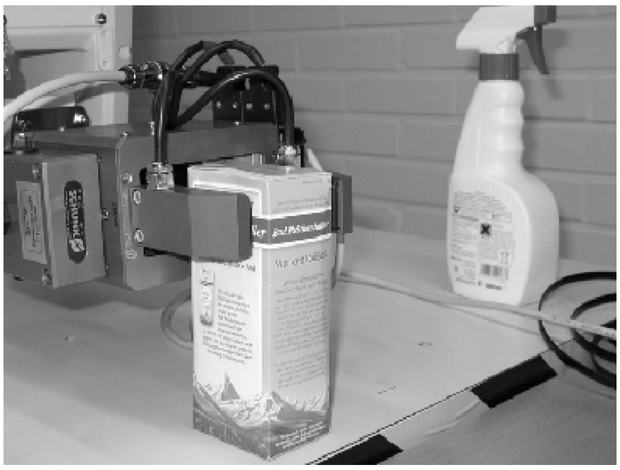


Figure 3.12: Grasped objects, a salt container and a detergent bottle.

In the context of the demonstration, the same controllers could be used for both arms, but the abilities of the hands are too different in terms of kinematics and sensors. As a results of this, both hands had their own controllers. Also the transitions for grasp stability or instability are customized for each of the platforms in order to use the different sensors on the platforms.

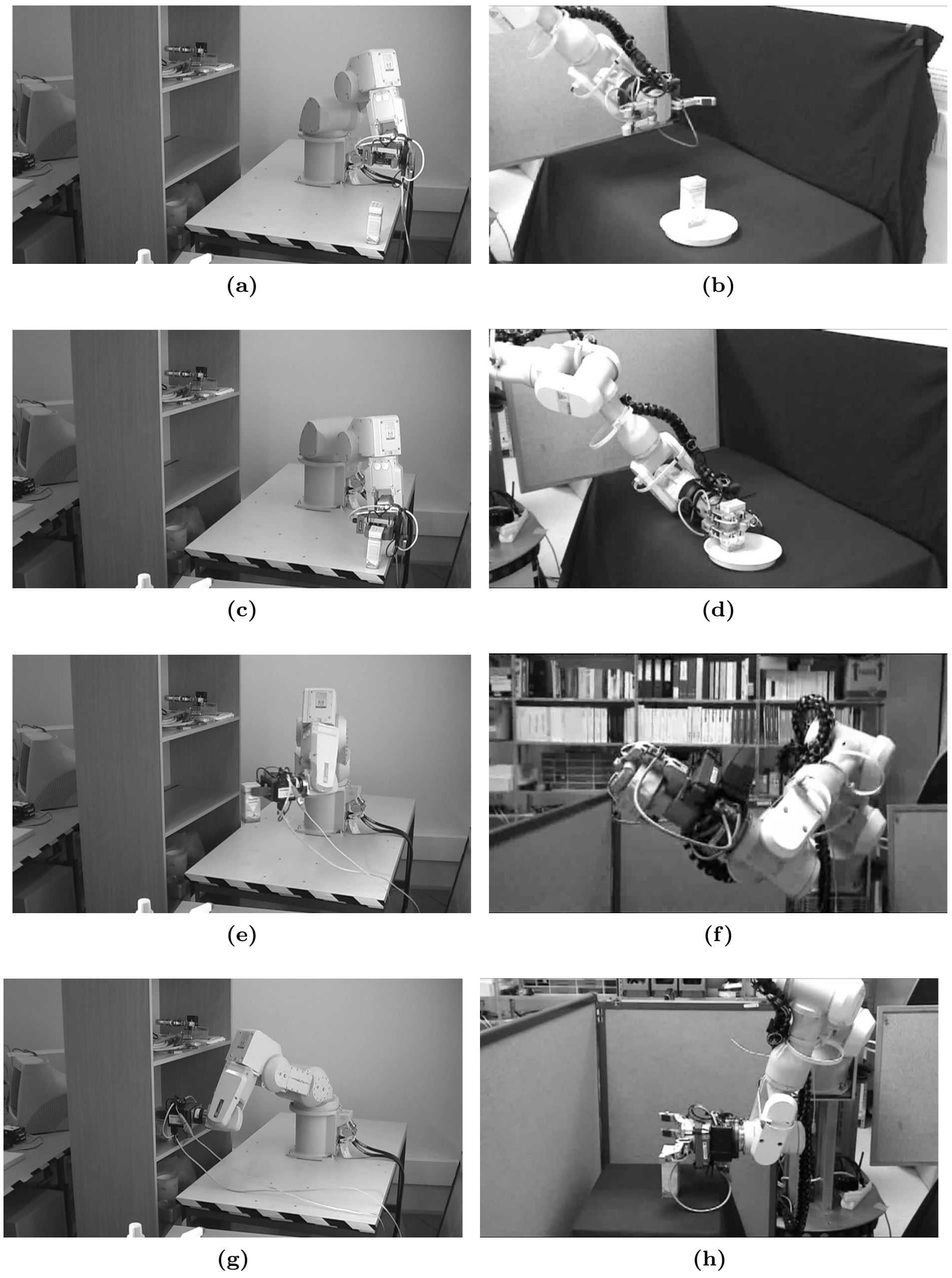


Figure 3.13: Action execution on both platforms: (a) Approach; (b) Grasp; (c) Move; (d) Release; (e) Approach; (f) Grasp; (g) Move; (h) Release.

3.4.2 Detecting failures

Failure detection is an important factor in the proposed architecture. Failure detection can be used to detect surprise and to learn. As the control architecture is focused towards sensor-based control, all available sensors can be used for failure detection. Failure is also explicitly included in the abstract state machine as one of the end states, and the sequence of states that lead to the failure can be recorded to learn from the failure event.

To demonstrate failure detection, the same abstract state machine was used as before with the demonstration platforms. However, the object mass was artificially increased, but this was not reflected on the abstract state machine. Sensor use is critical in detecting failures which can be seen in Figure 3.14, showing the result of the failure detection on the first platform. The figure depicts the total force affecting the tactile sensors and the state changes of the state machine as vertical lines. As can be seen from the figure, the state machine was executed normally until the lift and move primitive failed, and the state machine moved into failure state, halting the execution of the state machine. Note that the transition condition used here is a simple evaluation of the total force that affects the tactile sensors. The failure mode was demonstrated on the platform consisting of RV-3SB and WRT-102. However, both platforms have the capability of failure detection, using their platform specific sensors.

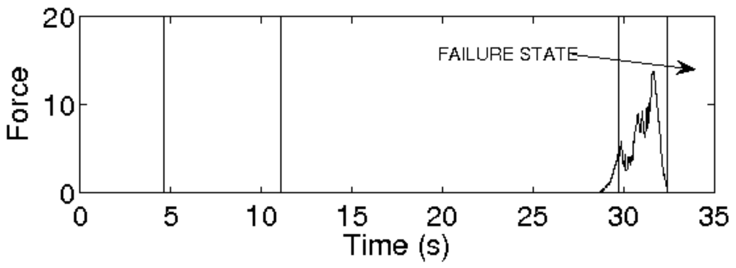


Figure 3.14: Measured force during a failed action.

3.5 Summary

This chapter presented an approach, an abstraction architecture, for handling embodiment independent knowledge and transferring that knowledge to a more embodiment specific representation which can be used to control a single robotic platform. The approach specifically addresses embodiment independence, the use of sensors as an integral part of control, and the modelling of actions as automata of adaptive primitive actions. The embodiment independent knowledge is modelled as a state machine which is then translated to suit each embodiment and its external and internal sensors. The architecture design follows some of the previously published architectures, such as [82], quite closely, but extends them to allow fully abstracted actions. The hardware independence was demonstrated on two different platforms, which consisted of different robotic arms and hands.

The architecture itself consisted of three main components: the abstract state machine, translator and concrete state machine. The abstract state machine was described in XML. A number of different attributes and properties could be used to adapt each individual state to different objects and different hardware platforms. The translation component handled the change from the abstract state machine to the concrete state machine. The translator was implemented to include its own internal logic which queried each platform for individual states, instead of the whole state machines. This reduced the amount of work required to implement the abstraction architecture on different platforms.

The concrete state machine represented the abstract state machine, but in a form that could be executed on a specific hardware platform. The concrete state machine was formed from a set of controllers, controlling an arm and a hand and a set of transitions governing the state changes. Each controller and transition can access all of the sensors found in the hardware platform, enabling the use of complex controllers.

While the experiments were basic, they showed that the abstraction architecture is able to work on different platforms despite a large gap in the capabilities of those platforms. In addition, the experiment on failure detection demonstrated how the sensor capability of the platform can be utilized to detect surprises.

Chapter IV

Learning Grasp Stability

Grasp stability, in an analytical sense, is well defined, as explored in section 2.1, and can be readily computed in simulation where enough data of the grasp is available, i.e. all contacts between the robotic hand and the object. Additionally, using a force closure metric for grasp stability, one can compute a grasp that sufficiently resists outside forces, such as gravity, thus allowing the robot to manipulate objects, for example by lifting them. However, the assumption that all necessary information is available, is not generally true when using real hardware. The tactile sensor data is imperfect, both in the sense of detecting contacts and in the sense of determining the actual contact forces and normals. In some cases, the proprioceptive information, i.e. joint configuration, is also difficult to determine accurately, thus, causing uncertainty in ascertaining the kinematic configuration of the hand which affects the assumed contact locations. All of these phenomena pave the difficult road for computing the grasp stability analytically with real hands and real objects.

This chapter focuses on methods for supervised learning of the grasp stability instead of analytically solving it. Compared to the analytical methods, supervised learning requires data and the labels for the data, which need to be collected beforehand. This can be seen as one of the disadvantages compared to the analytical methods. Nevertheless the training data can be any pertinent data that can be collected from a robotic hand. The work presented in this chapter relies on input from tactile sensors and the hand finger configuration. It is also important to notice that the raw sensor data can be used in the learning; for example, there is no need to know the kinematics of the hand which would be required to compute the true locations of the contacts when analytically solving the grasp stability. This feature is an advantage, as it allows the grasp stability to be learned for many different robotic hands within the same framework with minimal changes.

The work described in this chapter is based on classifiers – a well known field in machine learning – which enable the use of supervised learning. Although the classifiers used in this work are well known, the objective is to study which classifiers and which feature representations work best in the specific problem of determining grasp stability. In addition, the effect of object knowledge on the grasp stability determination is studied.

4.1 Utilizing haptic data in manipulation

Haptic and tactile data has been used for many purposes in the past few years, which has only been possible through the development of affordable and usable tactile sensors. This highlights the fact that the sense of touch can be and is used for applications that either cannot be accomplished using other sensor modalities, such as vision, or that complement information from other sensors. While the haptic data received during grasping can be used for computationally solving the grasp stability, the focus for the methods presented here is on the use of haptic data to estimate different object properties.

Research on the use of tactile and other sensors in the grasping context has increased in the past few years. Felip and Morales [93] have developed a blind grasp primitive, in which no vision is used. It aims to find a suitable grasp for an unknown object after a few initial grasp attempts. However, only finger force sensors were used in the study. In [94], also vision was used to detect good starting points for the grasp primitive.

Apart from using tactile information as a feedback for low level control [95], tactile sensors can be used to detect or identify object properties. Jiméneza et al. [96] use the tactile sensor feedback to determine what kind of a surface the object has and then determine a suitable grasp for the object. Petrovskaya et al. [97], on the other hand, use tactile information to reduce the uncertainty of the object pose upon initial contact with the object. In their work, a particle filter is used to estimate the object's pose, but the tactile sensor used to detect contact with the object is a probe, not a gripper, which makes it easier to extract information about contact normals, for example. This work was extended in [98, 99] to include novel statistical methods to infer the object pose.

Object identification has been studied by Schneider et al. [100] and Schöpfer et al. [101]. Schneider et al. show that it is possible to identify an object using tactile sensors on a parallel jaw gripper. The approach is similar to object recognition from images and the object must be grasped several times before accurate recognition is achieved. Schöpfer et al. use a tactile sensor pad fixed to a probe instead of a gripper or a hand. They also study different temporal features which can be used to recognize objects. Similar object recognition systems have been presented in [102] where the object state is recognized, for example, whether a bottle is full of liquid or empty. In [103], an approach similar to [100] is employed and the method is demonstrated on an antropomorphic hand instead of a simple gripper. Object recognition during manipulation and grasping has been studied earlier, as well. In [104], a set of measurements is approximated by a superquadric and the parameters of the superquadric are compared to a database of existing superquadrics to recognize the object. Geometric representation of the objects is also used in [105], but a neural network is trained to recognize objects based on features computed from the geometric representation.

The approach described in this thesis was published in [11, 7, 6]. A similar approach was proposed in [106]. Also in, [107] support vector regression was used to predict the grasp stability of superquadric objects. Rodriguez et al. [108] have proposed a similar learning approach to detect the presence of an object or objects, in this case marker pens, given a simple hand. The time series approach for learning grasp stability has been published separately in [109] and is not a part of this thesis. This approach has also been extended to include vision [78]. The time series approach utilizes the hidden Markov model (HMM) to determine the end state of the grasp. While more information can be used from a single grasp, the method cannot make a decision during grasping. Instead, a grasp must be fully completed before classifying the grasp as stable or unstable.

4.2 Supervised learning of instantaneous grasp stability

The learning of the grasp stability is based on data. Compared to the analytical approach, a set of advantages and disadvantages can be found. Perhaps the most important of these advantages is that the data from the end effector can be directly used in the learning. However, on the other hand, data is required for all different end effectors. Consequently each end effector requires a set of observations, i.e. grasp attempts from one or more object, that can be used to train the system to learn the grasp stability for the set of objects used in the training. The notation of observations follows the one presented in [6]:

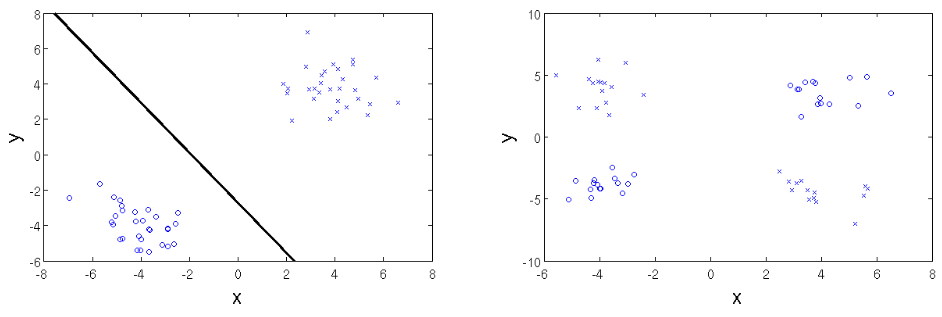
- $D = [o_i], i = 1, \ldots, N$ denotes a data set with N observation sequences.
- $o_i = [x_t^i], t = 1, \ldots, T$ is an observation sequence with T samples.
- $x_t^i = [\mathbf{f}_t^i \; \mathbf{j}_t^i]$, each sample consists of $\mathbf{f}$, the features extracted from tactile sensors and $\mathbf{j}$, the joint configuration.
- In addition, $\mathbf{L} = [l_1, \ldots, l_N]$, denotes the labels (stable, unstable) corresponding to the N observations.

Looking at the notation, a set of features $\mathbf{x}$ and the corresponding labels from $\mathbf{L}$ are available to be used for training. Moreover, the labels are binary: the observed grasp was either stable or unstable. The stability labels should correspond to the final sample x_T^i. This type of formulation is suitable for the classifier algorithms. One type of these classifiers is called linear classifiers [110], which can take the form of

$$y(\mathbf{x}) \; = \; f(\mathbf{w}^T\mathbf{x} + w_0) \tag{4.1}$$

which is called a generalized linear model [111]. When $y(\mathbf{x}) =$ constant, the model forms a *decision surface*. The decision surface divides the space of possible features into two in the case of binary labels. Each of these partitions in the feature space corresponds to a certain label or class. An example of a decision surface in 2D is shown in Figure 4.1(a) with a variety of features labeled into two classes. The problem of choosing this decision surface is what the classification algorithms try to solve with different underlying theories. Figure 4.1(b) shows the classical XOR problem that cannot be solved with a single linear classifier. In this thesis, due to the problems that the linear classifiers face, a set of non-linear classifiers are presented and consequently applied to the problem of learning grasp stability.

The experiments, presented in section 4.5, were conducted using four separate classifiers to study the differences of the results that the classifiers produce and to find out which of the classifiers is the most suitable for learning grasp stability. The classifiers presented here are the support vector machine (SVM), adaptive boosting (AdaBoost), the Gaussian



(a) (b)

Figure 4.1: (a): An example of a decision surface in 2D; (b): XOR problem.

mixture model (GMM) classifier and k-nearest neighbor classifier (k-NN). Each of these classifiers have a different approach to the problem of classification, and thus can provide different insights into learning grasp stability. These classifiers can be divided into two groups [112], discriminative models, including SVM, AdaBoost and k-NN, and generative models, including GMM.

While both models aim to learn the mapping $g : X \mapsto Y$, where X are the features and Y the labels, generative models aim to model the inputs and the outputs. Thus, one can generate new input samples from the generative model. Discriminative models, on the other hand, model only the output, i.e. what the output Y is given an input X. In a statistical sense, generative models model the posterior probability $P(Y|X)$ through $P(X|Y)P(Y)$ which can be obtained from Bayes' theorem. Discriminative models model the $P(Y|X)$ directly. [110]

Support Vector Machine

The support vector machine (SVM) [113, 114] is a maximum margin classifier, i.e. the classifier fits the decision surface so that a maximum margin between the classes is achieved. The support vector machine is named after the fact that only a part of the training data or vectors is used to generate this maximum margin decision surface, such vectors are known as support vectors. Figure 4.2 shows an example of SVM decision surfaces.

Another feature of the SVM is the ability to use non-linear classifiers instead of the original linear hyper-plane classifier. Non-linearity is achieved by applying the kernel trick [110] which allows the use of non-linear kernels. In this thesis, only the radial basis function (RBF) is used as the kernel for SVM:

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \quad \text{for} \quad \gamma > 0\,. \tag{4.2}$$

In addition to the parameter γ, constant C, which is the penalty applied to incorrectly classified training samples, i.e. the soft margin, [113], needs to be set. The parameters can be found by empirically searching the parameter space to find optimal values. An extension to the basic two-class SVM, probabilistic outputs for SVM by Platt [115], is also used to analyze the results given by the SVM. The implementation of the SVM is by Chang and Lin [116].

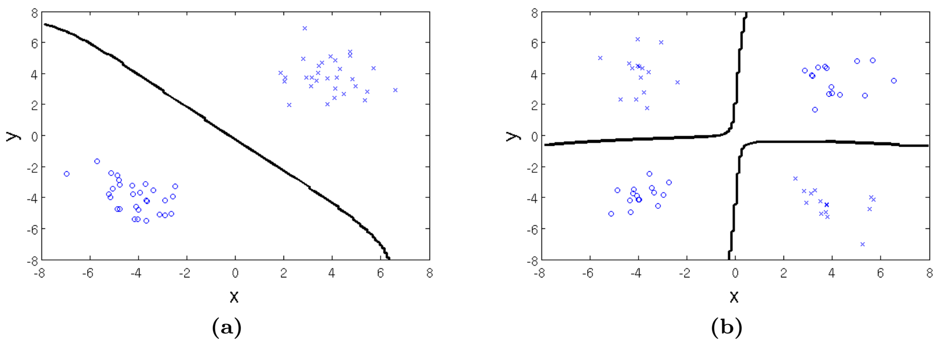


Figure 4.2: (a): SVM decision surface; (b): XOR problem solved by SVM.

Adaptive Boosting

AdaBoost or adaptive boosting is a meta-algorithm for learning which was developed by Freund and Schapire [117]. AdaBoost uses multiple weak classifiers, such as linear hyperplane classifiers, to classify the given training data. AdaBoost has a good generalization ability, but it is not effective when outliers are present in the training data, although more recent algorithms attempt to remedy these problems [118]. AdaBoost has been successfully applied to, for example, face detection [119]. In Figure 4.3, the AdaBoost decision surface for both the linear and XOR problem is shown.

The AdaBoost algorithm that is used in this thesis is based on a decision tree classifier with a variable branching factor. With a branching factor of 1, the tree classifier represents a linear hyperplane classifier. The branching factor was evaluated empirically, which showed that increasing the branching factor was not beneficial. The implementation is by Vezhnevets [120].

k-Nearest Neighbor

The k-nearest neighbor [121] classifier is a simple algorithm to implement. This classifier requires no training phase; instead, during the classification phase, the test samples are compared to all given training samples. The test sample is classified as the class with the most neighboring, i.e. the closest, training samples. The k denotes the number neighboring training samples that are used in the classification phase. However, the classification time complexity grows with the number of training samples. In Figure 4.4, the 1-NN decision surface for the linear and XOR problem is shown, although, in the experiments 3-NN is used. The k-nearest neighbor also has a proven [121] error rate that is no worse than two times the Bayes error rate, i.e. the optimal error given the distribution of data, when the amount of data approaches infinity.

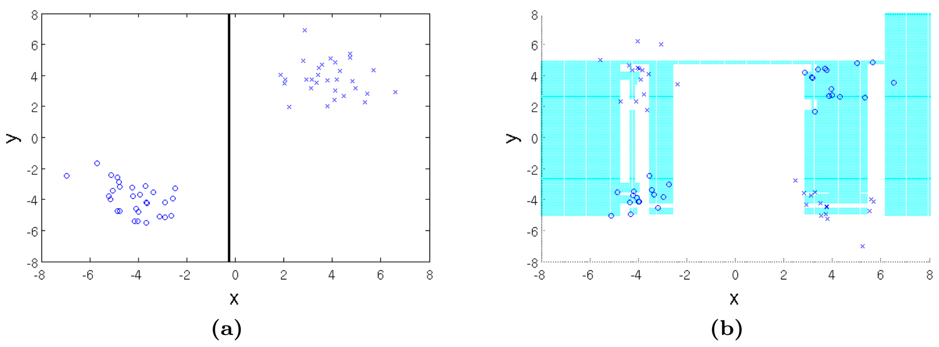


Figure 4.3: (a): AdaBoost decision surface; (b): XOR problem solved by AdaBoost, white denotes 'x' and light color (cyan) denotes 'o'.

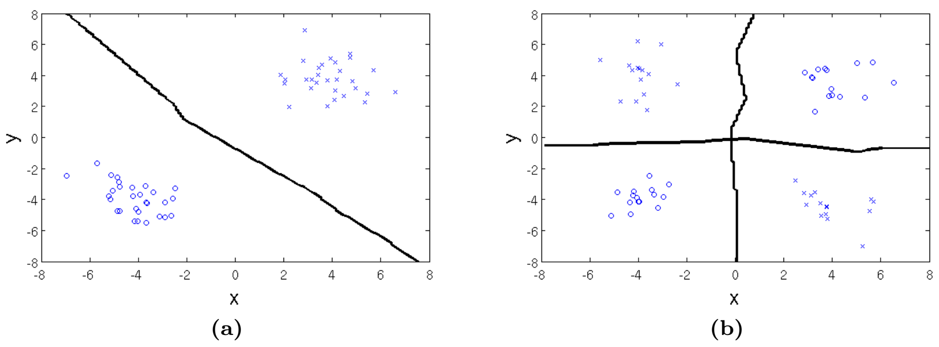


Figure 4.4: (a): 1-NN decision surface; (b): XOR problem solved by 1-NN.

Gaussian Mixture Model

As the naive Bayes classifier assumes that the data is distributed according to some modelable distribution, such as the Gaussian distribution, it is not optimal in cases where this assumption is not true. For example, when multi-modal distributions are present in the training data, Gaussian distribution would fail to represent the data adequately. The haptic data is distributed according to an unknown distribution, and thus it is reasonable to use a Gaussian mixture model (GMM) statistical classifier. In Figure 4.5, the GMM decision surface for the linear and XOR problem is shown.

While GMM methods assume a Gaussian distribution, they use multiple Gaussian distributions to model the data, which enables the methods to model multi-modal and more complex data. The implementation used in the experiments is by Paalanen and Kämäräinen [122].

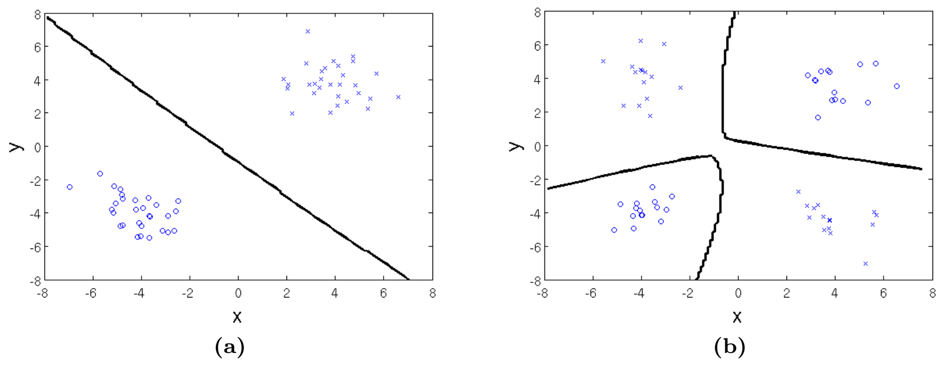


Figure 4.5: (a): GMM decision surface; (b): XOR problem solved by GMM.

4.3 Learning continuous grasp stability

In [6], the temporal information collected during a grasp is used in conjunction with a hidden Markov model (HMM) to decide whether the grasp is stable or not. For the method to be able to decide, the grasp must be completed. The approach proposed in this section is to extend the instantaneous method presented in section 4.2 by applying the learned stability model on-line to each sample $x_1, \ldots, x_T$ that is obtained during a grasp. This extension allows quicker decision making on the grasp stability in the case of a stable grasp.

As the method described in section 4.2 does not remember any of the previous time instances and does not consider the whole grasp sequence from $t = 1, \ldots, t = T$, the classification result over time may oscillate. One pathological example is shown in Figure 4.6. Through the use of filtering and thresholding, the oscillations can be effectively removed, as seen in the figure. One additional problem that exists is the lack of training data from the beginning of the grasp if one strictly follows section 4.2. However, if the whole grasp sequences have been recorded, i.e. from $t = 1$, the grasp configuration at the beginning of the grasp sequences can be used to train the classifiers. The data can be safely labeled as unstable, as the hand has not achieved contact with the object at that time instance. Using this procedure ensures that grasps that are not in contact with the object are classified as unstable, enabling the classifier to be run continuously from the beginning of the grasp until the end of the grasp.

Two different filter types are employed: a mean filter and an exponential filter. The results of the experiments with the filters are shown in section 4.5.4. The input for the filters is the result from the classifier, either 0 or 1, denoting either an unstable or stable grasp, respectively. The mean filter can be defined as a sliding window with the window size w. The mean of the data in the window is then calculated, and this result is the output of the filter. The exponential filter is described by

$$y(t) \;=\; (1-\alpha)\cdot y(t-1) + \alpha \cdot x(t)\;. \tag{4.3}$$

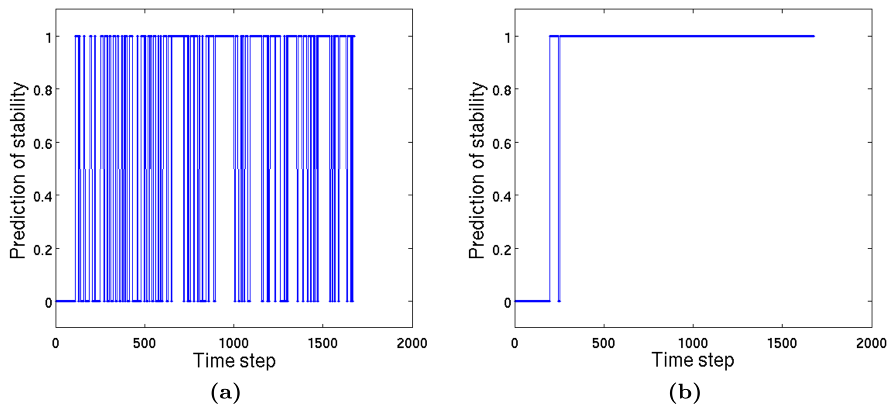


Figure 4.6: (a): Each time instance of a stable grasp classified with a SVM classifier; (b): The classification result filtered with an exponential filter and thresholded.

Equation 4.3 consists of $y(t)$ and $y(t-1)$, filter output at time instances t and $t-1$, of $x(t)$, the binary stability at time t and of α which is a weighting factor. Examples of both filters are shown in Figure 4.7, which depicts the same sequence as in Figure 4.6.

Introducing the filters requires setting more parameters in addition to the parameters for SVM. These include w for the mean filter window width, and α for the exponential filter. In addition, both require the threshold thr for the binary decision of stability. After the threshold has been crossed, the grasp is deemed stable. Again, the values for these were found empirically by varying the parameters and choosing the parameter pairs that produced the best classification results.

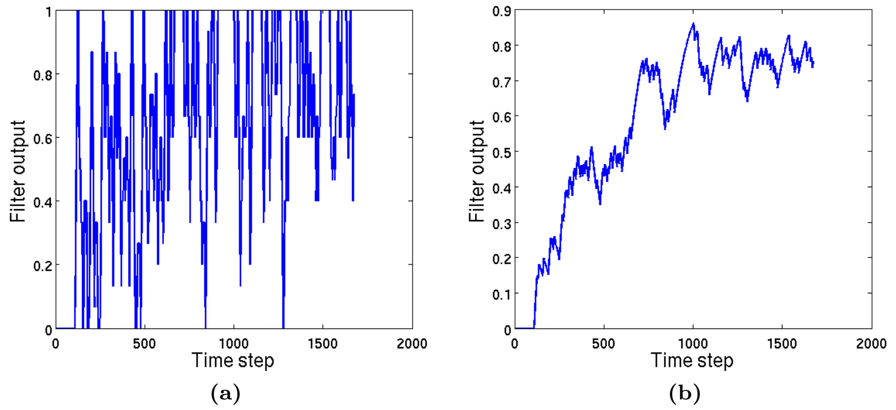


Figure 4.7: (a): Filter output of a mean filter; (b): Filter output of an exponential filter.

4.4 Features for learning grasp stability

To learn grasp stability, the training data is collected from a series of grasps, noted by the observation sequences o_i. Then, from each observation sequence the last sample x_T^i is used for the training. Additionally, in the continuous stability learning, the first sample x_T^i can be used as training data, as well. Time instant T captures the instant on which the decision of a stable or unstable grasp must be based, for example to lift the object. Both unstable and stable grasps must be included in the training data so that sufficient data is available to discern the stable grasps from the unstable grasps.

The actual features are the readings from the tactile sensors, $\mathbf{f}$ and the joint configuration $\mathbf{j}$. An example of the tactile sensor output is given in Figure 4.8, which shows the output from the six tactile sensors of the ARMAR-IIIb end effector. The images consists of *taxels*, each of which can measure the pressure that affects the taxel. The end effectors used in the experiments will be described in detail in section 4.5, but the hardware ranged from two tactile sensors and one DOF to six tactile sensors and 8 DOFs. Tactile sensor sizes, i.e. the number of taxels, were also different. This range of hardware shows that the presented methods are general and suitable for many types of end effectors.

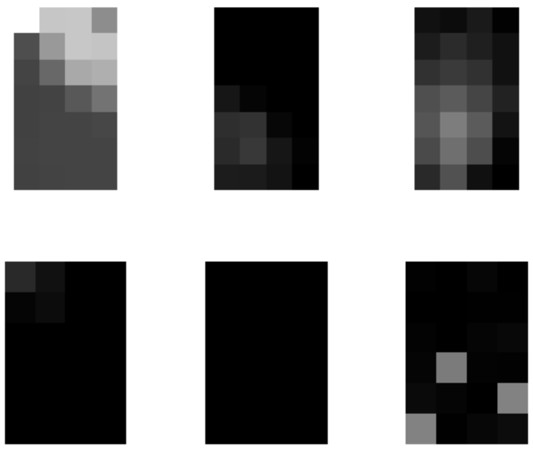


Figure 4.8: Tactile images from ARMAR-IIIb.

Utilizing the raw data, i.e. the tactile images and the joint configuration, is possible with some classifiers, but it is useful also to study different feature representations that can be built from the raw data. The process of building feature representation essentially transforms the raw data into information. The focus is to compute the feature representations from the tactile data that can be very high dimensional (>500 dimensions). Fortunately, the tactile images can be considered as normal images; thus, many feature representations suitable for photographic images are usable in the context of tactile sensors, as well. This fact influenced the choice of some feature representations, such as image moments or local binary patterns. Next, all of the feature representations that were part of the experiments, are presented.

Principal Component Analysis

Principal component analysis (PCA) is a commonly used linear technique for dimensionality reduction. The PCA is computed from the covariance of each tactile sensor matrix. The covariance is then used to solve the eigenvectors and eigenvalues:

$$C = cov(H_T^1, \dots, H_T^N) \ , \tag{4.4}$$

$$V^{-1}CV = D \ . \tag{4.5}$$

V represents the eigenvectors and D the corresponding eigenvalues. H denotes the tactile measurements that are a part of the sample vectors x_T^i. To reduce the dimensionality of the data, it is common to choose a subset of all of the eigenvectors. In this case, only 60 eigenvectors with the largest eigenvalues were chosen. The 60 eigenvectors explained approximately 90% of all of the data.

Image Moments

Raw image moments are defined as

$$m_{p,q} = \sum_x \sum_y x^p y^q I(x, y) \ . \tag{4.6}$$

The moments are computed up to the order of two, that is $(p+q) = o, \ \ o = \{0, 1, 2\}$, the function $I(x, y)$ represents the individual taxels of the tactile sensor. These are related to the total pressure, the mean of the contact area, and the shape of the contact area, indicated by the variance in x- and y-axes. The moments are computed for all tactile sensors individually. Note that each tactile sensor can be represented by the set of six numbers, $\{m_{0,0}, m_{0,1}, m_{1,0}, m_{1,1}, m_{2,0}, m_{0,2}\}$, regardless of the actual dimensions of the sensor itself.

Raw image moments are used in the experiments as more complex image moments did not produce better results. This observation might be due to the fact that, for example, rotation invariant moments are not useful for grasp stability learning, as each grasp uniquely maps to the stability of the grasp.

Histogram

A histogram representation on the tactile data represents binning of the force that affects each cell of the tactile matrix. This operation also removes all spatial information. Thus, the histogram only considers the distribution of the affecting force. In the experiments, 10 bins were used for each tactile sensor.

Spatial Partitioning

Spatial partitioning partitions the area of the sensor matrix and sums up the affecting force in every cell of the sensor matrix in each of these partitions. In essence, this subsamples the tactile image of each sensor matrix, but instead of averaging the taxels, a sum is computed to account for the total force. Partitioning can be thought of as opposite to the histogram operation, as partitioning retains the spatial information but loses some information on the force distribution. In the experiments, a 2x2 grid is used to partition the tactile image on each sensor.

Local Binary Pattern

Local binary patterns (LBPs) [123] are used commonly for texture classification, but also for face recognition. As its name suggests, local binary pattern codes local changes in a binary code. The local changes are found by thresholding the pixel neighbourhood by the value of the center pixel and checking which pixels are above the threshold. These binary codes are then added to a histogram, which is the final feature representing the original data. Images from all sensors are coalesced into one image and the LBP is applied to this image in the experiments.

Row and Column Sums

Row and column sums are another form of spatial feature representation, where the colums and rows are summed up independent of each other. Thus, the resulting dimensionality of the feature representation is the sum of the tactile sensor dimensions $i + j$ for each sensor:

$$sum_{c_j} = \sum_i I(i, j) \ , \tag{4.7}$$

$$sum_{r_i} = \sum_j I(i, j) \ , \tag{4.8}$$

where sum_{c_j} denotes the individual sensor columns, sum_{r_i} denotes the sensor rows and $I(i, j)$ the force reading from individual taxel.

4.5 Experiments

The grasp stability experiments are divided into three subsections, which are constructed along the main topics discussed in sections 4.2, 4.3 and 4.4:

- Experiments on the effect of classifier and feature representation on the classification performance, i.e. the performance experiments.
- Experiments on increasing knowledge of an object when learning the grasp stability of that object, i.e. the knowledge experiments.

- Experiments on learning continouous grasp stability using the ARMAR-IIIb hand, i.e. the continuous grasp stability experiments.

The performance experiments will measure the effect of the classifier and the feature representation on multiple sets of data. These experiments give insight on the performance of each feature representation and classifiers described in sections 4.2 and 4.4, and gives an idea of which classifiers and feature representations to use in the future. The work presented in these experiments was published in [7]. The next experiments, the knowledge experiments, use the findings of the previous experiments and take a look at how much the performance of the classifiers can be improved when more specific knowledge of an object is given. These experiments were published in [6]. The knowledge experiments can therefore be used to evaluate how much information is required for certain classification performance. Finally, the continuous grasp stability experiments are based on the theory presented in section 4.3 and can be used to evaluate how the grasp stability can be evaluated temporally, i.e. across time, instead of just at the end of the grasp. The experiments can also be used to compare the classification performance to the single-shot classification.

4.5.1 Hardware description

In the following experiments, a set of simulated and real hardware is used to show that the grasp stability learning is usable on different types of end effectors that have access to the proprioceptive sense, i.e. the joint configuration sensors and the tactile sensors. Figure 4.9 shows all the end effectors used in the grasp stability experiments. For the performance experiments, two end effectors, the WRT-102 and the Schunk Dextrous Hand, were used, as well as a simulated model of the SDH:

- Simulated Schunk Dextrous Hand, three fingers each with 12x6 tactile elements attached to the distal phalanges.
- Schunk Dextrous Hand, three fingers each with 13x6 tactile elements attached to the distal phalanges .
- Parallel Jaw Gripper, WRT-102, two fingers each with 14x6 tactile elements (Weiss tactile sensors).

Although the SDH is equipped with six tactile sensors, in the performance experiments only the three sensors attached to the distal phalanges were used. The additional sensors attached to the proximal phalanges were utilized in the knowledge experiments. The SDH and its simulated model were the only end effectors used in the knowledge experiments.

In experiments on learning the continuous grasp stability, the ARMAR-IIIb's right hand is used (shown in Figure 4.9(d)). The hand is equipped with joint encoders and pressure sensors. The hand has 1 DOF in the palm, and 2 DOFs in each the thumb, the index finger and the middle finger, which are pneumatically actuated. The hand contains six tactile sensors from Weiss Robotics [124]. One tactile sensor is mounted on the distal phalanges of the thumb, the index finger and the middle finger. Three tactile sensors are mounted in the palm, in the area between the thumb and the index and middle fingers. The tactile sensors have a resolution of 4×7 taxel (phalanges) and 4×6 taxel (palm).

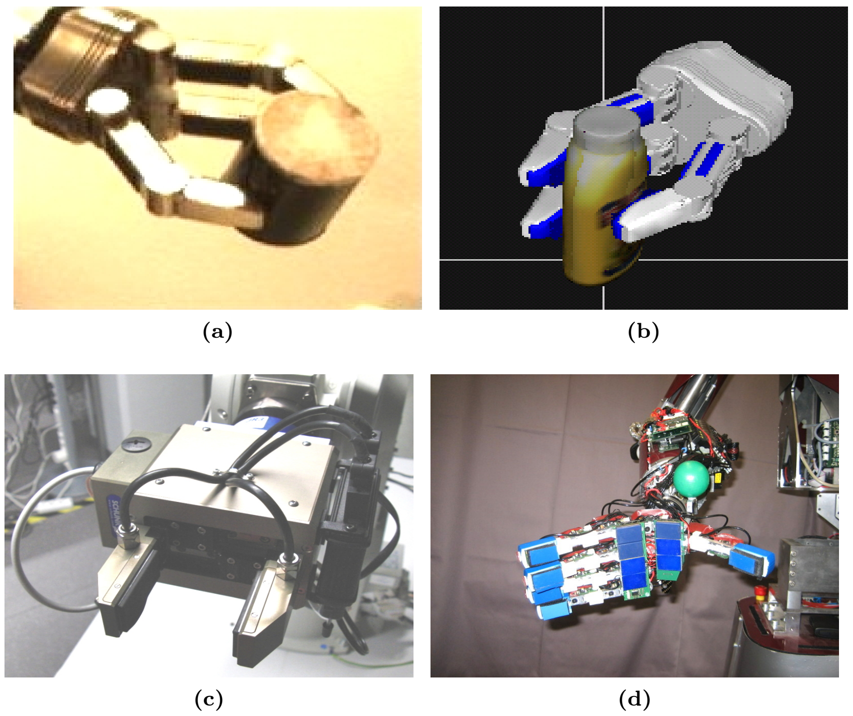


Figure 4.9: Images of the end effectors used in the experiments: (a) SDH; (b) Simulated SDH; (c) WRT-102; (d) Right hand of ARMAR-IIIb.

4.5.2 Performance experiments

The parameters used to obtain the feature representations are shown in Table 4.1. The table shows the selected parameters for each individual feature extraction method, if such are required. The *raw* feature denotes that the tactile readings from the sensors were used as is. Note that in the experiments the feature vector also includes the raw joint configuration measured from the end effector. All of the features were normalized to zero mean and unit standard deviation.

Table 4.2 shows the four classifiers and their parameters that were used in the experiments. The parameters were chosen empirically, i.e. different parameter values were tested and the values that gave the best classification performance were selected. The bias was towards the simplest values; for example, with AdaBoost, similar classification performance was achieved with higher branching factors. However, it is noteworthy that during the empirical evaluation, not all possible feature representations and classifier combinations were tested with all available data sets to determine the best possible parameters. Instead, a few data sets, both from simulated hands and real hands, and some feature representations were used for this task.

The following data sets have been chosen from the simulated data. They were generated by Jimmy Jørgensen (one of the co-authors in [6]) using a simulated SDH hand model

Table 4.1: Table of parameters for features.

Features	**Parameter**	**Parameter**
Raw	-	-
PCA	-	-
Histogram	No. bins: 10	-
LBP	Uniform LBP	Samples: 8,1
Moments	-	-
Partitioning	Grid: 2x2	-
R&C sums	-	-

Table 4.2: Table of parameters for classifiers.

Classifier	**Parameter**	**Parameter**
SVM	C: 0.4	γ: 0.03
GMM	max. clusters: 19	max. error: 0.016
KNN	k: 3	-
AdaBoost	Branch factor: 1	Iterations: 200

in the simulation environment RobWorks, which is described in [125]:

- D_1, a cylinder, grasps sampled from the side
- D_2, a bottle, grasps sampled from the side
- D_3, a bottle, grasps sampled from the top
- D_4, a cylinder, grasps sampled from a sphere
- D_5, a bottle, grasps sampled from a sphere

The data sets $D_{1,2,3}$ represent cases where we know the pose of the object with some accuracy, and can plan for a grasp. The data sets $D_{4,5}$ are simulating situations where the position of the object is known to some extent, but the orientation is highly uncertain. Thus, the grasps are sampled from a sphere around the object. This type of scenario might be possible when only visual data is used to estimate, for example, the center of the mass of an unknown object. The objects corresponding to each data set are shown in Figure 4.10. In the simulated data, the grasp quality measure, i.e. the grasp stability, is based on [126], but instead of one convex hull W, two convex hulls W_f and W_τ are used to separate the wrench space with respect to forces and torques, and additional constraints are placed on W_f, so that

$$\alpha(m \cdot \mathbf{g}) \in W_f \ , \alpha = 1.1 \ . \tag{4.9}$$

This allows the grasp to remain stable even if some additional forces are introduced in addition to gravity. Data sets generated with real hands are the following:

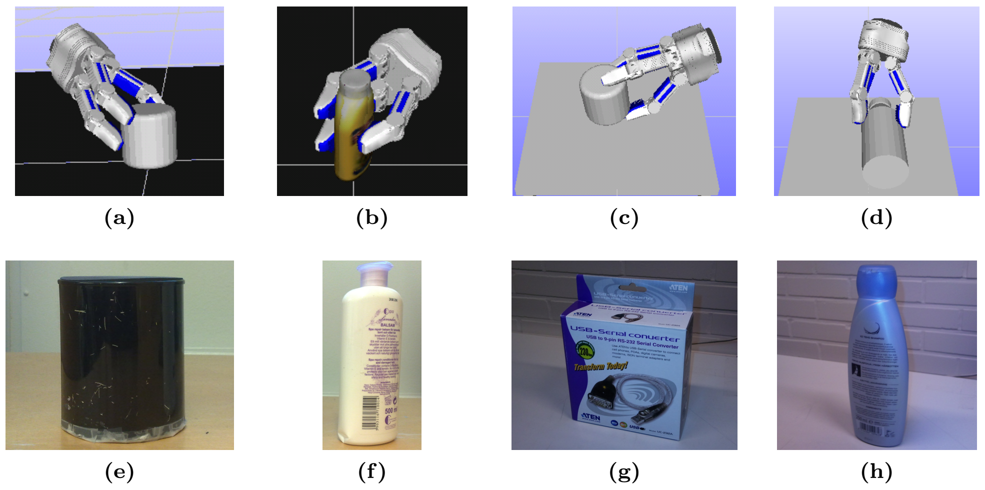


Figure 4.10: Objects used in the data sets: (a) D_1; (b) D_2,D_3; (c) D_4; (d) D_5; (e) D_6; (f) D_7,D_8; (g) D_9; (h) D_{10}, D_{11}.

- D_6, a cylinder, grasps sampled from the side, SDH
- D_7, a bottle, grasps sampled from the side, SDH
- D_8, a bottle, grasps sampled from the top, SDH
- D_9, a box, grasps sampled from the side, WRT-102
- D_{10}, a shampoo bottle, grasps sampled from the side, WRT-102
- D_{11}, a shampoo bottle, grasps sampled from the top, WRT-102

The data sets $D_6, \cdots, D_8$ were collected by Yasemin Bekiroglu (the first author of [6]) and data sets $D_9, \cdots, D_{11}$ were collected by the author of this thesis. Data sets $D_{6,\cdots,11}$ represent cases where an estimate of the object's pose is known, for example, from a vision system. This estimate is commonly noisy, and thus, some noise was added to the hand pose. The objects in data sets $D_{6,7,8,9}$ are rigid and the objects in data sets D_{10} and D_{11} are non-rigid, i.e. the objects are deformable. The grasp stability in these datasets was determined by grasping the object. Again, the objects corresponding to data sets $D_6, \ldots, D_{11}$ are shown in Figure 4.10. In data sets $D_{6,\cdots,8}$ the object was rotated $[-120°, +120°]$ around the approach direction and in data sets $D_{9,\cdots,11}$, the object was lifted and rotated $+90°$ around the X and Y axes and the Z axis was the direction of the lift. If the object moved independently of the hand during the lift or rotations, the grasp was unstable. Otherwise, it was stable.

The performance of the classifiers was evaluated using ten-fold cross validation. The data set sample sizes for each of the given data sets are shown in Table 4.3 with the maximum classification rate summarized from Tables 4.4 and 4.5 with the corresponding classifier and feature representation. The sample size shown in the table is balanced, so

Table 4.3: Data set sample sizes and classification rates.

Data set	**Sample size**	**Max. classification rate**	**Class.+Feat.**
D_1	6400	77.0%	SVM+PCA
D_2	4906	61.4%	SVM+R&C
D_3	4446	62.7%	SVM+Moments
D_4	5302	80.4%	SVM+Part.
D_5	8990	70.5%	SVM+Moments
D_6	140	92.1%	SVM+Moments
D_7	100	92.1%	AdaBoost+Part.
D_8	50	84.6%	KNN+Raw
D_9	148	74.6%	AdaBoost+R&C
D_{10}	148	59.0%	AdaBoost+Hist.
D_{11}	100	64.0%	AdaBoost+Moments

that each data set has an equal amount of stable and unstable grasp samples. All other features were normalized to zero-mean and unit variance, except the raw features which were normalized to the range $[0, 1]$. The normalization parameters were obtained from the training set and applied to both training and test sets.

Result matrices with the described data sets are given in Table 4.4 and Table 4.5. The tables show the classification rate of each data set with the indicated feature and classifier combination. Each row shows the best classifier in **bold** font and the worst in *italic* font. The best and worst classifiers were determined on a 95% confidence interval using the Agresti-Coull interval, which approximates the binomial confidence interval. Multiple classifiers are deemed best if there is no statistically significant difference between them in the classification performance. Some results for GMM are omitted because of the training sample size requirements; thus, the results for datasets $D_{6,\dots,11}$ are not shown. Also the dimensionality of the raw data was too much for the GMM to accurately model the distribution of the data.

The results in Tables 4.4 and 4.5 show that there is a distinctive performance difference between the data sets. The simulated data sets, D_1 and D_4 frquently perform better than the other simulated data sets. This performance gap is caused, at least partially, by the hand configuration, which allows the object to touch other areas of the hand where there are no simulated sensors. This eliminates some of the important information about the object to be used in determining the grasp stability. Thus, it is important to set up the grasp sequence in a way that allows the sensorized part of the hand to grasp the object.

This procedure is evident in the data set gathered from the real hands, especially sets $D_{6,7,8}$, where the classification performance is above 75% in some cases. However, the objects in the data sets were rigid, which is not the case in sets $D_{10,11}$. These sets show mostly poor performance, indicating that possibly more samples should be used to learn the grasp stability. Additionally, the possible configurations of the gripper, used in the data sets $D_{10,11}$, are not as informative as the SDH, which can envelope the object, compared to the WRT-102 gripper which can only grasp the objects in a planar fashion.

The best overall classifier is AdaBoost, which performs the best out of the four classifiers, while SVM is a close second. The worst classifier is GMM, partially due to the extensive

Table 4.4: Classification rates for data sets $D_1, \cdots, D_{11}$.

Data, Feature	**SVM**	**GMM**	**KNN**	**AdaBoost**
D_1	**75.5%**	-	*73.3%*	**76.7%**
D_2	**59.1%**	-	**56.6%**	**58.1%**
D_3	**60.3%**	-	**60.7%**	**62.1%**
D_4	69.7%	-	*63.1%*	**79.3%**
D_5, Raw	65.7%	-	*58.2%*	**68.9%**
D_6	**82.6%**	-	**90.7%**	**91.4%**
D_7	*22.0%*	-	**84.0%**	**91.0%**
D_8	*49.3%*	-	**84.6%**	**80.4%**
D_9	*54.0%*	-	**71.3%**	**71.1%**
D_{10}	**49.3%**	-	**48.6%**	**46.4%**
D_{11}	**50.0%**	-	**54.0%**	**56.0%**
Mean	66.3%	-	*62.6%*	**69.6%**
D_1	**77.0%**	74.1%	*72.5%*	74.5%
D_2	**59.7%**	56.5%	*56.1%*	57.0%
D_3	**61.3%**	**60.4%**	**59.7%**	**60.4%**
D_4	74.0%	68.7%	*67.5%*	**77.6%**
D_5, PCA	**67.6%**	64.5%	*60.4%*	**67.7%**
D_6	**85.7%**	-	*58.6%*	**90.0%**
D_7	**77.0%**	-	*55.0%*	**69.0%**
D_8	50.0%	-	*47.9%*	**78.6%**
D_9	**73.2%**	-	**65.5%**	**71.7%**
D_{10}	**50.0%**	-	**54.0%**	**54.0%**
D_{11}	**46.0%**	-	**55.0%**	**49.0%**
Mean	**68.5%**	65.4%	*63.3%*	**68.1%**
D_1	**76.5%**	*71.1%*	72.5%	**75.9%**
D_2	**61.1%**	*52.7%*	57.4%	**58.6%**
D_3	**62.7%**	*54.0%*	**60.1%**	**62.3%**
D_4	**80.0%**	*61.2%*	72.3%	**79.7%**
D_5, Moments	**70.5%**	*51.8%*	63.6%	**68.9%**
D_6	**92.1%**	-	**93.6%**	**90.7%**
D_7	**91.0%**	-	**86.0%**	**92.0%**
D_8	*27.1%*	-	**67.1%**	**77.9%**
D_9	**64.6%**	-	**69.5%**	**72.7%**
D_{10}	**44.0%**	-	**50.5%**	**44.7%**
D_{11}	**48.0%**	-	**51.0%**	**64.0%**
Mean	**70.6%**	*58.0%*	65.6%	**69.7%**
D_1	**73.1%**	*64.9%*	65.6%	**73.9%**
D_2	**56.0%**	**55.0%**	*52.2%*	**56.4%**
D_3	**62.0%**	*49.9%*	57.6%	**62.1%**
D_4	**79.0%**	71.9%	*69.8%*	**79.4%**
D_5, Histogram	**67.9%**	**66.8%**	*62.1%*	**68.5%**
D_6	**90.0%**	-	**81.4%**	**90.0%**
D_7	**84.0%**	-	**76.0%**	**82.0%**
D_8	**66.1%**	-	**73.6%**	**72.5%**
D_9	**63.0%**	-	**53.8%**	**57.3%**
D_{10}	**57.6%**	-	**49.2%**	**59.0%**
D_{11}	*38.0%*	-	**62.0%**	**57.0%**
Mean	**68.1%**	62.9%	*62.0%*	**68.7%**

Table 4.5: Classification rates for data sets $D_1, \cdots, D_{11}$.

Data, Feature	**SVM**	**GMM**	**KNN**	**AdaBoost**
D_1	**76.1%**	*65.6%*	71.8%	**75.8%**
D_2	**57.3%**	**55.0%**	**56.2%**	**57.3%**
D_3	**62.6%**	*53.2%*	59.0%	**60.8%**
D_4	**80.4%**	*65.7%*	73.0%	**79.7%**
D_5,Partitions	**69.0%**	*60.9%*	64.7%	**68.9%**
D_6	**91.3%**	-	**91.4%**	**92.1%**
D_7	**91.0%**	-	**89.0%**	**89.0%**
D_8	*36.8%*	-	**81.8%**	**67.5%**
D_9	**63.0%**	-	**60.6%**	**64.4%**
D_{10}	**40.8%**	-	**45.5%**	**48.8%**
D_{11}	**56.0%**	-	**48.0%**	**64.0%**
Mean	**69.6%**	*60.6%*	65.5%	**69.2%**
D_1	**75.0%**	*64.4%*	68.6%	**74.9%**
D_2	**54.8%**	52.0%	*51.4%*	**56.4%**
D_3	**60.9%**	*58.0%*	58.1%	**61.3%**
D_4	75.2%	66.4%	*64.0%*	**79.7%**
D_5, LBP	66.4%	58.7%	*57.8%*	**68.4%**
D_6	**84.3%**	-	**79.3%**	**85.0%**
D_7	*26.0%*	-	**68.0%**	**74.0%**
D_8	**47.1%**	-	**68.2%**	**60.0%**
D_9	**61.1%**	-	**69.2%**	**73.4%**
D_{10}	**50.2%**	-	**45.8%**	**48.3%**
D_{11}	**50.0%**	-	**49.0%**	**51.0%**
Mean	66.8%	*60.1%*	60.3%	**68.7%**
D_1	**76.8%**	*63.8%*	72.1%	**76.5%**
D_2	**61.4%**	58.6%	*57.8%*	58.3%
D_3	**62.7%**	**61.5%**	**61.5%**	**60.7%**
D_4	77.3%	*58.9%*	70.2%	**79.6%**
D_5, R&C Sums	**68.7%**	63.4%	*62.6%*	**68.8%**
D_6	**92.1%**	-	**92.1%**	**91.4%**
D_7	**90.0%**	-	**87.0%**	**91.0%**
D_8	*30.7%*	-	**72.1%**	**68.2%**
D_9	**63.5%**	-	**67.5%**	**74.6%**
D_{10}	**55.1%**	-	**50.3%**	**43.8%**
D_{11}	**54.0%**	-	**52.0%**	**64.0%**
Mean	**69.8%**	*61.6%*	65.1%	**69.5%**

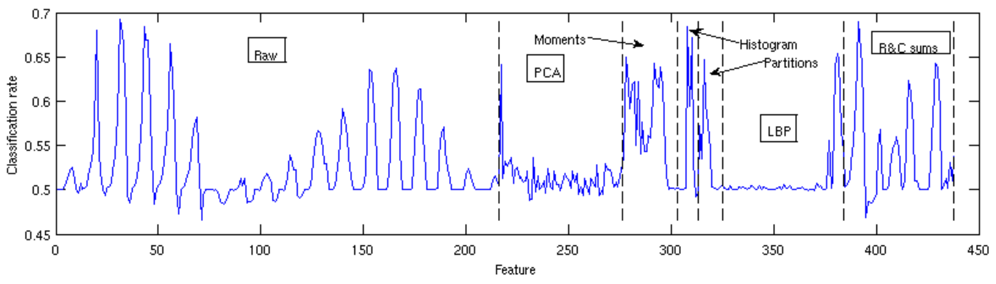


Figure 4.11: Classification rates on individual features.

amount of data needed to train GMM successfully with some of the chosen features. The small amount of data available in data sets $D_{6,\ldots,11}$ makes it difficult to determine the best classifier within the 0.95 confidence interval, but looking at the results, AdaBoost has the highest mean in these cases. SVM has some anomalies, which are suspected to be caused by the parameter and feature combinations, and could be fixed by adjusting the parameters of SVM.

To study the effect of the features on the classification rate, tests with a 3-nearest neighbor classifier were conducted on each dimension of all of the feature representations described in section 4.4 and also on the raw tactile data. The classification rates are shown in Figure 4.11 for the data set D_1.

Classification rates of 0.5 or less in Figure 4.11 are a sign that the feature used is not particularily useful in learning, as it has no correlation with the grasp stability. The figure shows that there are quite many useful features in the set of features that were tested. One interesting observation can be found within the raw data, as these features have multiple spikes which are among the best features for classifying the grasp stability. Especially the taxels corresponding to the thumb, i.e. the single finger opposing the two other fingers, have the highest classification results among all of the features. This indicates that individual cells of the tactile sensors, especially in the thumb, could be used to determine the grasp stability to some extent. Also image moments, the histogram and row and column sums seem to have a number of advantageous features to be used for classification. The experiment was also performed on the real data set D_6, for which the results were similar.

4.5.3 Knowledge experiments

As was discovered in section 4.5.2, the AdaBoost algorithm performed the best, which is the reason behind utilizing it in the knowledge experiments. Nonetheless, the SVM classifier could be expected to perform similarly. The same parameters are applied as in section 4.5.2. An SVM classifier is employed here to show how to utilize the probabilistic outputs. Parameters from section 4.5.2 are used in these experiments, as well.

The knowledge experiments are based on the information hierarchy from [6] shown in Figure 4.12 using the AdaBoost classifier. The information hierarchy is based on levels of increasing knowledge. The first level, also denoted as the root node, is the most general case, representing a set of objects grasped with a general grasp strategy, namely a spherical grasp strategy. The second level contains specific object shape classes, for example, cylindrical objects grasped with a general grasp strategy. The third level comprises of these specific object shape classes grasped with a specific grasp strategy, for example, top grasps. The most specific level is, of course, a single object grasped with a specific grasp strategy. Due to the amount of data required to sample grasps at levels 1-3, this data was generated using simulation. Data on specific objects with a specific grasp strategy, on the other hand, was collected using a real SDH.

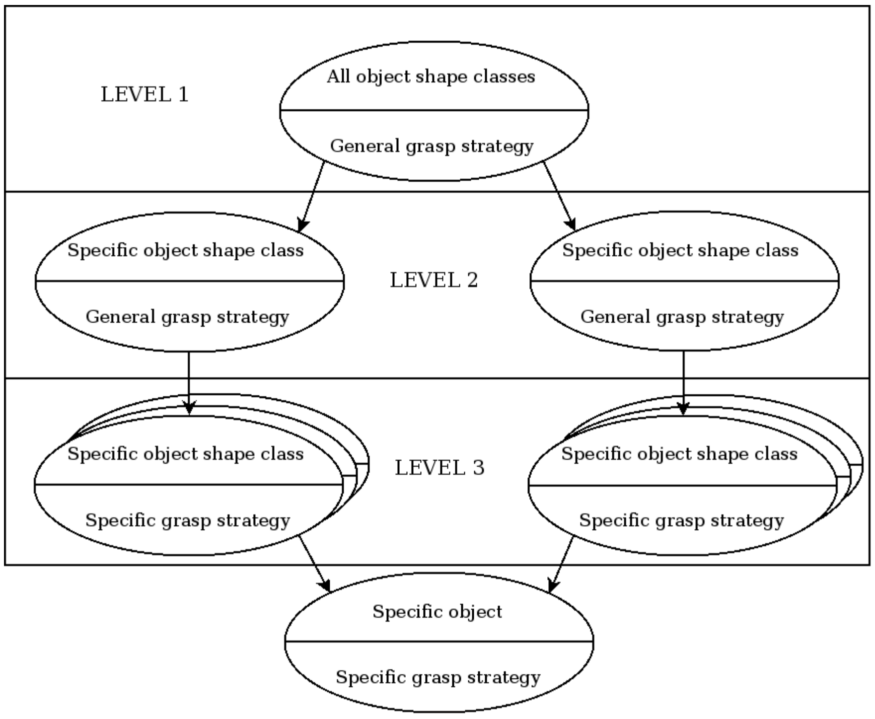


Figure 4.12: Information hierarchy. Adapted from [6].

All experiments here are reported as ten-fold cross validation averages, except where otherwise noted. In each case, the data sets used for training and testing the classifiers are balanced, i.e. the data sets contain an equal number of unstable and stable grasps. Image moments are used as the feature representation in all experiments. The joint data in addition to the tactile data is also included in the features, unless otherwise noted.

The real objects used in the demonstration are shown in Figure 4.13. The objects in the figure are denoted in the experiments as follows (from left to right): cylinder, deformable cylinder, cone, orange bottle, shampoo, pitcher, white bottle, blue bottle and box. Each object was sampled for 100 stable grasps and 100 unstable grasps using side grasps. The simulated objects are shown in Figure 4.14. Each object was scaled to three sizes: 0.75, 1.00, 1.25 times the normal size. The objects are referred to (from left to right) as hamburger sauce, bottle, cylinder, box and sphere. Each object is grasped with a specific grasp strategy, referred to as top, side or sph (spherical) in the experiments. The spherical grasp is used to denote a grasp made from a sphere centered on the object, and the approach vector of the grasp points at the object center. Each combination of grasp and object contains 3000 stable and 3000 unstable grasps. This is further divided into 1000 stable and 1000 unstable grasps per object scale. The root node consists of of only the primitive shapes, i.e. cylinder, box and sphere, with a total of 18000 samples.

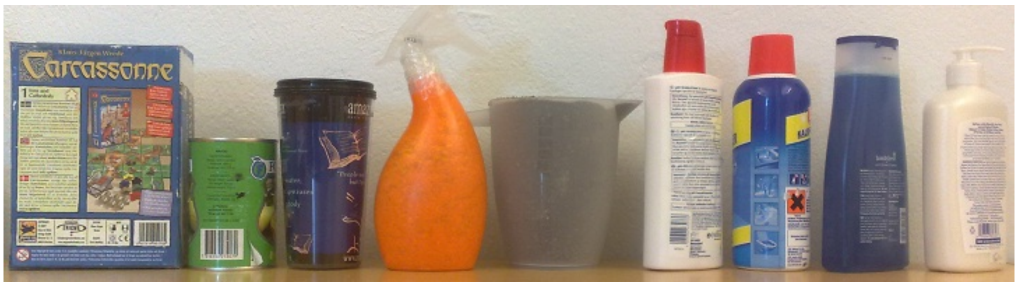


Figure 4.13: Real objects.

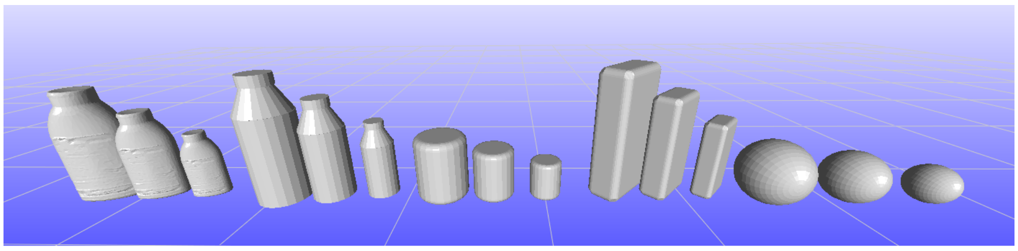


Figure 4.14: Simulated objects.

To study the performance of the grasp stability classification further, a support vector machine classifier is used with image moments to examine the separability of the grasp stability at each level by means of log-likelihood histograms. The effect of the joint configuration data on the classification is also studied by including or excluding it from the feature vector for the classifier when using real data. The training time for the classifiers is less than five minutes, for the reported amount of samples. The AdaBoost training time increases linearly with the amount of samples while the SVM training time increases quadratically. The classification of a single sample takes less than 10 ms with both of the presented classifiers. The SVM classification time increases linearly with the amount of samples used for training.

Real data

The experiments begin by showing results using real data, collected by the first author of [6], Yasemin Bekiroglu. Sampling grasps with a real hand is a slow process and thus the sample size is limited. To study the effect of the amount of samples used for training, a series of tests were run with variable sample sizes. In each case, the same object was used both for training and testing. The results of these tests are shown in Table 4.6, which shows the classification rates for training data sets of difference sizes. The test shows that for a specific grasp on the cylindrical object, 100 samples are already enough to reach classification performance levels achieved with higher amount of samples, and the differences in classification performance above 100 samples are not statistically significant. However, this is the case only when the stable and unstable grasps are distinctive, i.e. a high rate of correctly classified grasps is achieved. In the case of the white bottle data set, where the classification rate is lower, the results show that more than 200 samples could be useful in increasing the classification performance.

Table 4.6: AdaBoost classification rates (as percentages) on data sets with variable amount of samples.

Samples	50	100	150	200
Def. cylinder	74.6%	85.0%	84.8%	89.0%
W. Bottle	64.6%	68.0%	68.5%	75.5%

Classification results as percentages for single object classifiers (known object case) are presented in columns 2 and 3 of Table 4.7. Classification rates are shown both with joint configuration data and without it. The main focus in this experiment is to study the prediction of the grasp stability on known objects that the system has previously learnt. The average classification rate for known objects is 82.5% including joint data, and 81.4% excluding it from the measurements. Thus, the inclusion of joint data seems to have a beneficial effect on the recognition rate. Moreover, the result indicates that at least with known objects, the proposed approach seems to have an adequate recognition rate for practical usefulness.

Table 4.7: AdaBoost classification rates (as percentages) on known and unknown objects with and without joint data.

	Known obj.		Unknown obj.	
	w/j	wo/j	w/j	wo/j
Cylinder	88.9%	90.3%	80.4%	81.9%
Def. cylinder	91.0%	89.0%	76.0%	76.5%
Cone	79.5%	81.0%	73.0%	68.0%
O. Bottle	77.0%	78.5%	72.5%	72.0%
Shampoo	82.5%	76.0%	70.0%	71.5%
Pitcher	84.5%	78.0%	71.0%	66.0%
W. Bottle	76.0%	73.5%	75.0%	76.0%
B. Bottle	74.0%	75.0%	68.5%	69.0%
Box	89.0%	91.0%	78.0%	73.0%

It is also interesting to study how well the trained system can cope with unknown objects, i.e. objects that have not been used to train the system. The results are shown as percentages of correct classification in columns 4 and 5 of the Table 4.7, adjacent to the results with known objects. The results are for a system that has been trained on all of the objects except the object for which the classification rate is shown. The average recognition rate is 73.8% with joint data and 72.7% without it. The results show that while the classification rate is lower than with known objects, it is still possible to predict the grasp stability on unknown objects to some extent. However, this holds true only when similar grasps are applied on unknown objects as were applied to the objects that the system was trained on. In comparison, including grasps from all objects, including the one being tested, for a single classifier yields a result of 78.6% correct classification across all the objects in the real object set.

Two objects of a primitive shape are included in the real data, a box and a cylinder. Table 4.8 shows classification percentages when the classifier is trained only on one of the primitive objects. The classifier is then applied to classify the grasp stability of grasps made on real-world objects of different shapes. Cross validation was not needed in this case, because the training and test sets were, naturally, separate. The average classification rate with the cylinder model is 68.0% and with the box model 66.4%. These results no longer seem adequate for a real system, which again suggests that the variety in the training data is essential.

Table 4.8: Classifier performance (as percentages) when training with a primitive object.

Trained object	**Cylinder**	**Box**
Def. cylinder	76.0%	73.5%
Cone	66.0%	69.5%
O. Bottle	64.5%	61.0%
Shampoo	66.5%	64.0%
Pitcher	71.0%	62.0%
W. Bottle	73.5%	69.5%
B. Bottle	58.5%	65.0%

Simulated data

In contrast to the real data, in simulation, a large number of grasps can be sampled from different objects and using different grasp strategies. The data sets used here were generated by Jimmy Jørgensen, one of the co-authors of [6]. The following classification results were achieved using the simulated data sets described earlier in this section. In Table 4.9, classification percentages are reported for each node in the information hierarchy. The root node (Level 1) was randomly subsampled to 12000 samples due to computational constraints and has classification rate of 75.3%. The average classification for Level 2 (known object, unknown approach vector) is 76.5% and for Level 3 (known object, known grasp) 77.5%. The results are similar to the real object experiments, where increasing the variety of objects, i.e. information, increases the classification performance. However, the results from simulated data sets are poorer, and the increase in performance is not as strong as it was with the real object data sets.

While the primitive shapes used in Table 4.9 are simple and can be used to train the classifier. Then, the classifier can be used to classify grasps sampled from more natural, complex objects. The results are shown as percentages of correct classifications in Table 4.10. Each row corresponds to a tested natural object (hamburger sauce, bottle), while each column corresponds to a combination of a training object and grasp strategy.

Table 4.9: AdaBoost classification rates (as percentages) according to the information hierarchy on simulated data.

Level	**Node**	**Classification rate**
Level 1	Root	75.3%
Level 2	Prim. cylinder sph.	73.5%
	Prim. box sph.	79.2%
	Prim. sphere sph.	77.0%
Level 3	Prim. cylinder side	80.7%
	Prim. cylinder top	67.6%
	Prim. box side	83.5%
	Prim. sphere side	78.5%

A comparison of results when training the classifier with the natural object and corresponding grasping strategy are shown in italic font. The figures in the table show that having data from the correct object has a positive effect on the classification rates. This is again an argument for the variety of training data.

Table 4.10: AdaBoost training with a primitive shape and classifying grasps sampled from a natural object with simulated data.

	Prim. cylinder sph.	**Prim. cylinder side**	**Prim. cylinder top**	**Prim. box sph.**
Hamb. sauce	71.5% *78.7%*	74.0% *83.5%*	62.9% *72.4%*	76.8% *78.7%*
Bottle	68.6% *74.7%*	77.4% *82.0%*	56.2% *65.2%*	72.6% *74.7%*
	Prim. box side	**Prim. sphere sph.**	**Prim. sphere side**	**All classes sph.**
Hamb. sauce	73.6% *82.0%*	61.4% *78.7%*	62.7% *83.5%*	73.4 % *78.7%*
Bottle	76.9% *82.0%*	59.4% *74.7%*	66.9% *82.0%*	69.7% *74.7%*

Using the SVM and its ability to output estimates of the prediction certainty, gives the possibility to examine the performance of the classifier on different data sets in more detail compared to AdaBoost, which supports only the hard decision surface. This comparison can be seen in Fig. 4.15. In the figure, log-likelihood ratios, $\log \frac{1-P(S)}{P(S)}$, calculated from the probabilities for stable and unstable samples are shown in histogram form – red for unstable and light blue for stable. The classification errors are shown in filled color, with the filled area indicating the error probability. Fig. 4.15a-c are from simulated data and Fig. 4.15d is from the real cylinder grasped with the SDH hand. It is evident from the figure that increasing information makes the distributions for stable and unstable grasps more separate, which was also indicated by the earlier results. Moreover, the figure also supports the finding that it seems to be easier to classify the real data than the simulated data. Finally, the figure supports the use of probabilistic approaches for grasp classification, as the ability to measure the uncertainty in classification is important because it can, for example, allow tuning the classification system to give fewer false

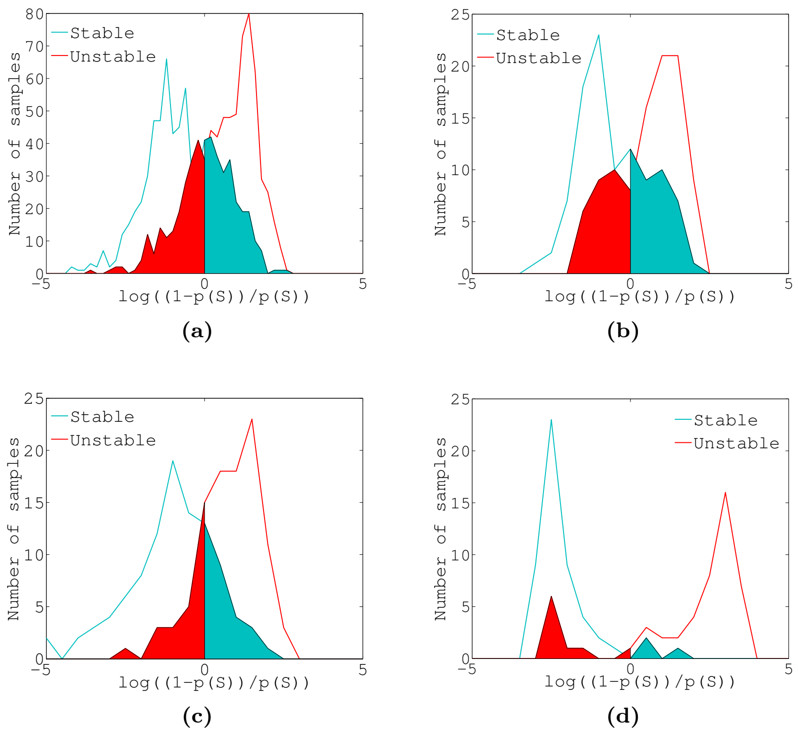


Figure 4.15: Likelihood ratios for comparison of separability: (a) Root node, all objects, random grasp vector; (b) Cylinder, random grasp vector; (c) Cylinder side grasp; (d) Real cylinder side grasps.

positives.

4.5.4 Continuous grasp stability experiments

For the continuous grasp stability experiments, the end effector of the ARMAR-IIIb was employed. The data for the experiments was collected by Markus Przybylski from the Humanoids and Intelligence Systems Lab of the Karlsruhe Institute of Technology located in Karlsruhe, Germany. Markus Przybylski is also one of the co-authors of [8]. Compared to the data sets used in previous experiments, the collection procedure was less rigid. All grasp samples were collected from a set of objects in a basket, which is shown in Figure 4.16). Two different data sets were collected, D_1 and D_2. D_1 contained 71 stable grasps and 94 unstable grasps. D_2 comprised 82 stable grasps and 76 unstable grasps. The two separate sets were collected, partly to study the generalization capability of the SVM classifier between data sets and partly to see whether the sensor hysteresis affects the classification results.

The grasp procedure was somewhat less structured than in previous experiments due to the amount of objects and the relatively small number of grasps. However, the main

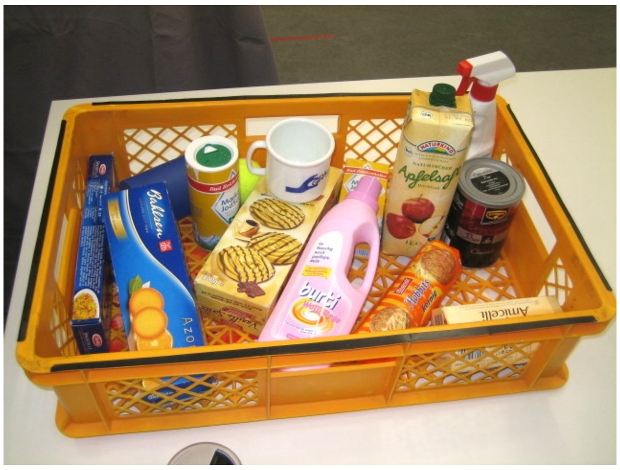


Figure 4.16: The basket with our test objects.

form of grasps were top grasps with varying approach directions. Also the roll angle was varied in relation to the approach direction. Some examples of the grasps are shown in Figure 4.17.

The continuous grasp stability experiments are divided into two parts. The first part consists of synthetic tests, which presents the reliability and accuracy of the classification of the grasp stability and comparisons between the different filter types described in section 4.3. The second part is composed of validation tests. In these tests, the SVM is trained with all of the data from the data set D_1 which is used to classify the grasps in D_2.

In the synthetic tests, both data sets D_1 and D_2 were used. For most experiments, a confusion matrix is presented, showing how the classifier performs in terms of true positives (stable, predicted stable), false positives (unstable, p. stab.), true negatives (unstable, p. unstab.) and false negatives (stable, p. unstab.).

Synthetic tests

In Table 4.11, the SVM was trained with data from a corresponding data set; only the last sample from each observation sequence was classified. The reported results are averages from ten-fold cross validation. The results show that the performance across data sets is similar. These results can be compared with results reported in [6, 106], showing that the ARMAR-IIIb hardware is able to reach performance similar to the Schunk Dexrous Hand (SDH) or the Barrett hand in this task.

Contrary to results in Table 4.11, in Tables 4.12, 4.13 and 4.14 the whole observation sequence was classified using the methodology presented in section 4.3. In Table 4.12, the mean filter was used with a window width of 25 and with a threshold of 0.61, Table 4.13 shows result with an exponential filter with $\alpha = 0.056$ and threshold of 0.61. The parameter values are based on empirical tests on performance while varying the parameters. Results in Table 4.14 were obtained without using a filter.

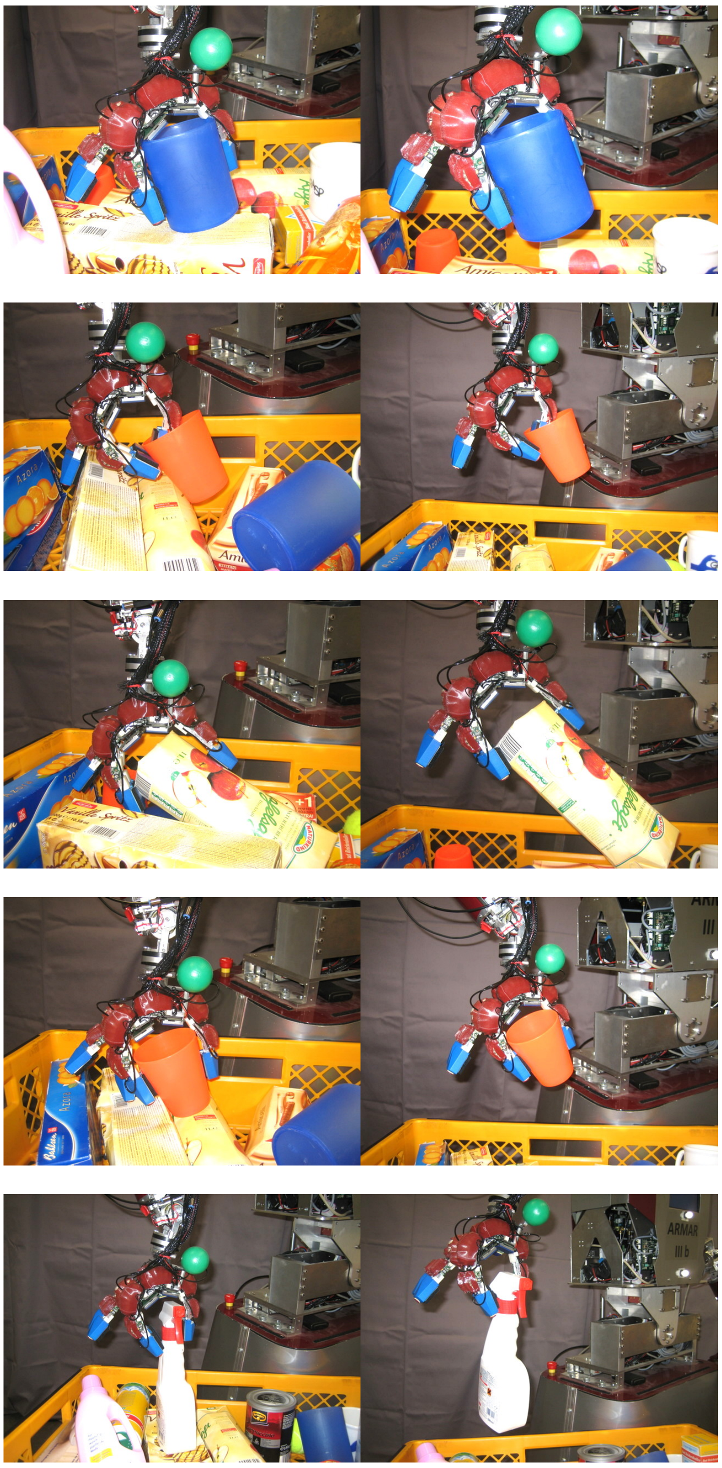


Figure 4.17: Example grasps. The left column shows the grasps and the right column shows the result after lifting the object.

Table 4.11: Confusion matrix for classification rates of grasps when classifying only the last sample, for data sets D_1 and D_2.

D_1	P. Stab.	P. Unstab.	D_2	P. Stab.	P. Unstab.
Stable	0.79	0.21	Stable	0.72	0.28
Unstable	0.28	0.72	Unstable	0.26	0.74

Table 4.12: Confusion matrices for classification rates of grasps using the mean filter ($w = 25$, thr $= 0.61$).

D_1	P. Stab.	P. Unstab.	D_2	P. Stab.	P. Unstab.
Stable	0.77	0.23	Stable	0.74	0.26
Unstable	0.24	0.76	Unstable	0.16	0.84

Overall, when using a filter with the classification, the overall classification rate is similar to the last sample classification, but the classification rate of the unstable grasps is better. This can be explained through the use of the filter which filters out the effect of the last sample, thus leading into a better classification result. In the case where no filters are used, in Table 4.14, the stable grasps are predicted well, but this translates also to falsely predicting that unstable grasps are stable. On average, the filter based classification is better in predicting the stable and unstable grasps.

One interesting possibility that comes with the method described in section 4.3 is that the grasp sequence can be stopped when the classifier decides that a stable grasp has been achieved. Table 4.15 presents the results with different filter types. For example, if a whole grasp sequence is 1000 time steps long, the classification using a mean filter can stop the grasp at time step 686 on average, if the grasp is a stable grasp. Without a filter, the average time decreases as expected, but with at a cost of overall correct classification rate.

Validation tests

To mimic a real world usage scenario, data set D_1 was used to train the SVM classifier. Then, using the trained classifier, data set D_2 was classified. Each observation sequence in the data set was classified with mean and exponential filters and without filtering. The results are show in Table 4.16. Compared to results in Table 4.11, the number of false positives rises. This effect might be due to tactile sensor hysteresis, i.e. the output

Table 4.13: Confusion matrices for classification rates of grasps using the exponential filter ($\alpha = 0.056$, thr $= 0.61$).

D_1	P. Stab.	P. Unstab.	D_2	P. Stab.	P. Unstab.
Stable	0.79	0.21	Stable	0.73	0.27
Unstable	0.23	0.77	Unstable	0.16	0.84

Table 4.14: Confusion matrices for classification rates of grasps without a filter.

D_1	P. Stab.	P. Unstab.	D_2	P. Stab.	P. Unstab.
Stable	0.90	0.10	Stable	0.87	0.13
Unstable	0.46	0.54	Unstable	0.21	0.79

Table 4.15: Average percentage of time steps to reach decision on stable grasp compared to full grasp observation sequence.

D_1	Mean filt.	Exp. filt.	No filt.
Time	68.6%	66.9%	59.6%

from the sensors changes between the collection of data sets, which in turn means that data set D_1 does not represent the data in D_2 and leads to worse results.

4.6 Summary

The focus of this chapter was to present a new methodology for estimating grasp stability. The basis of the methodology is in machine learning, specifically in classification, contrary to the analytical methods that are used widely in the grasping research. The proposed approach has some drawbacks, such as the need for training data, but definitive benefits, as well. A considerable set of data was collected for the experiments using multiple end effectors to estimate the performance and the suitability of the proposed approach in general.

The performance experiments investigated how different machine learning methods and feature representations affect the ability to learn and assess the grasp stability from haptic data. Both simulated and real-world data were used in the performance experiments. The experiments indicated that both AdaBoost and SVM were valid choices for the task of learning grasp stability. Furthermore, the experiments indicated that image moments and row and column sums can be useful as a feature representation for the classifiers. The classification performance varied significantly between different data sets. The results of

Table 4.16: Confusion matrices for validation tests.

Mean filt.	P. Stable	P. Unstable
Stable	0.77	0.23
Unstable	0.39	0.61
Exp. filt.	P. Stable	P. Unstable
Stable	0.76	0.24
Unstable	0.38	0.62
No filter	P. Stable	P. Unstable
Stable	0.90	0.10
Unstable	0.46	0.54

the experiments showed that the grasp stability of deformable objects is more difficult to learn than rigid objects. The data also shows that if the grasped object has contacts with the hand outside of the tactile matrices, the grasp stability cannot be learned effectively.

Using the results of the performance experiments, the effect of object knowledge was studied in the knowledge experiments. These experiments focused on a single end effector which was also implemented in simulation. These experiments showed that while it is possible to classify the grasp stability without knowing the object or the pose of the object beforehand, the performance of the classification will increase if more is known about the object before the actual grasp is performed. The second observation was that the variety of objects used to learn the grasp stability matters, i.e. it is better to learn the grasp stability of multiple objects and then apply this knowledge to new, unseen, objects.

In the continuous learning experiments, the same approach was taken as in previous experiments. However, the SVM based grasp stability classifier was extended with the use of filters which enabled the grasp stability classification to take place throughout the grasp instead of just the end. Additionally, the data for the experiments was collected using the end effector of the ARMAR-IIIb. The experiments showed that similar results were obtained with the ARMAR-IIIb as previously reported on other types of hardware, such as the Schunk Dextrous Hand or Barrett hand. The experiments showed that by using the extended grasp stability classification, the decision can be made earlier in the grasp process.

Chapter V

Probabilistic Sensor-based Grasping

Current grasp planning approaches are usually based on an assumption of perfect knowledge of target objects. While geometric models are good approximations of the objects in the real world, the models are not exactly accurate, especially when speaking of household items. Thus, a difference between expected and realized grasp arises from these approximations, although in many cases the difference is small enough to achieve a stable grasp. However, this discrepancy is usually ignored.

On the other hand, methods using sensor information to grasp by making corrective motions or reacting to the tactile sensor information have been proposed. Contrary to most grasp planners, accurate object models are not usually available in this type of grasping.

This chapter presents a probabilistic framework which unifies some the ideas behind grasp planning and the possibilities of sensor-based grasping. The framework considers the required variables and models for grasping as probability distributions and allows probabilistic representation of the current belief, so that the uncertainty of the knowledge can be modeled. The framework allows interplay between grasp planning and corrective motions – in situations where object attributes, such as pose, are not precisely known – by utilizing sensor information gained during grasping. Such a situation can arise when, for example, visual sensing is used to initially estimate the target objects. The framework is demonstrated in simulation and with real robot platforms.

5.1 Uncertainty in grasping and grasp planning

The proposed approach to finding good grasps, described in section 5.2, is closely related to the field of grasp planning, which was detailed in chapter 2. However, compared to the method presented here, most current grasp planning methods do not account for the uncertainty present in the object or in the object's pose information. Most grasp planners also require a geometric model of the object before grasp planning can be executed on the object. This section gives an overview of current grasp planning techniques and some of the techniques used to reduce uncertainty from tactile and proprioceptive information.

To simplify the grasp planning that is based on the geometric model of the object, many methods employ some form of object decomposition. The goal of the decomposition is to reduce the amount of feasible grasps without trying every grasp on an object. In [34], the object is decomposed to minimum volume bounding boxes, in an effort to understand the underlying shape of the object. The primitive shape is then used to reduce the search space for stable grasps. Instead of boxes, superquadrics are used in [35]. In addition to the construction of the superquadric decomposition, a heuristic is used to define the trial grasps based on the superquadric form of the object which limits the space of possible grasps significantly.

The Columbia Grasp Database [44] takes an approach different from most grasp planners and computes the best grasps for a set of thousands of objects. The grasp planning problem is then transformed to a problem of matching a new object with an object found in the precomputed database of grasps. The work has also been extended to consider partial data [127].

If the object is not known, i.e. a geometric model is not available, the grasp planning methods can still be used if the model of the object can be constructed. The model construction can either be done by vision, laser or tactile exploration. However, the geometric model in this case is usually a mesh or a point cloud, and contains no information about the inherent uncertainty related to the perception. Approaches such as [34] can be applied here, as well, but the results can be worse than in the cases where the full geometric model is known, and the decomposition may fail in cases where large volumes are missing from the perceived object. It is also possible to use simpler approaches to grasp objects instead of grasp planners described above when the geometric model is not known. In [128], unknown objects are grasped by first segmenting a user indicated object and then grasping the object using a parallel jaw gripper. The grasp orientation and location are based on simple rules applied to the segmented point cloud.

Another approach for finding grasps is object affordance modeling. While object affordance is a broader subject, the affordances can also be thought of in the sense of grasp stability. In some grasp related studies, grasp affordances consider the overall stability of the grasp [129], [130] or, for example, the grasp affordance in specific tasks [131].

Learning to find good grasps is another view on the problem. In [129], a real robot learns the grasp affordances of an object. The learning process reduces a vision bootstrapped distribution of grasps to a smaller set containing only good grasps. Reinforcement learning [132] can also be applied to learn a sequence of grasps which will lead to a stable grasp on an object.

The approach presented in this chapter is more related to methods that explicitly assume uncertainty in the object's attributes. Some of these methods are described next. The aim of [133] is to reduce the uncertainty of an object's pose to enable grasping of the object, and in [134], the shape of the object is also uncertain in addition to the pose. In both of the studies, the method is presented with a parallel jaw gripper grasping a 2D object. However, these methods do not utilize sensor information gained during grasping. Also in [135], the authors propose a decision-theoretic controller which minimizes the uncertainty of the object pose using arm trajectories to enable task specific grasps on objects, i.e. a specific, pre-determined, grasps. The pose of the object is inferred with the help of a simulator and tactile sensors that detect contact. The uncertainty of measurements is also considered in [136, 137], where visual and depth information is used to partially observe objects. Then, a set of object representations, derived from the partial data, is generated. Grasp planning is performed on these different object representations in simulation. As the initial data from the object is uncertain and partial, the object representations can give different grasps with varying confidence levels of success. The actual, executed, grasp is then chosen from the possible grasps by means of Bayesian principles.

A new algorithm, Guaranteed Recursive Adaptive Bounding (GRAB), for inference was developed in [98]. The algorithm was also tested in a manipulation environment where the algorithm made an accurate inference of the object's pose in both simulation and real environments. However a geometric model is required. The method was further developed in [99]. Nevertheless, only the problem of object localization was studied, and a force probe was used as the end effector instead of a more complex hand which would be capable of grasping the object.

5.2 Framework for grasping under uncertainty

The probabilistic framework is now presented in a general form. The sensor-based grasping is modeled with the following variables: S denotes the stability of a grasp as a continuous value from 0 to 1, G the grasp attributes (e.g. the pose of the end-effector), O the object attributes (e.g. the pose of the target object) and T represents on-line measurements, for example, proprioceptive information. The variables have different characteristics: G, the grasp attributes, can be controlled, T can be measured for each grasp attempt, while O is not directly observable, that is, we assume we only have an uncertain initial estimate of the object attributes.

Traditional grasp planning algorithms can be interpreted in a probabilistic framework as attempting to maximize the stability S by controlling the grasp attributes G with perfect knowledge of the object attributes O,

$$\arg\max_{G} \; P(S|G,O) \; . \tag{5.1}$$

In the proposed model, O is not assumed to be precisely known, but instead, it is represented as a probability distribution. In addition, the model for the stability (5.1) is not required to be fully accurate. Instead, the model can be uncertain in itself due to approximations and inaccuracies in, for example, simulation when computing grasp quality measures.

In order to perform grasp planning, the distribution of object attributes needs to marginalized over, i.e. the mode of $P(S|G)$ needs to be found. The marginalization can be written as

$$P(S|G) = \int P(S|G,O)P(O)\,\mathrm{d}O \; . \tag{5.2}$$

This is a major difference to traditional grasp planning, where the best single estimate of object attributes, that is, the mode of O, is used instead of marginalization over the whole distribution.

Because the proprioceptive measurements for a grasp attempt is not available before the attempt is performed, the proprioceptive information from the previous grasp attempts can be used only to update the posterior distribution for the object attributes $P(O)$. That is, the model $P(O|G,T)$ can be used to update the posterior distribution of object attributes. Thus, after some sensor information has been collected from sequential grasp attempts, for grasp planning the argument for G corresponding to the maximum of

$$\begin{aligned}&\arg\max_G P(S|G) \approx \\ &\arg\max_G \int P(S|G,O)P(O|G_{t_0:t-1},T_{t_0:t-1})\,\mathrm{d}O\ ,\end{aligned} \tag{5.3}$$

is found. This shows that the stability S can be maximized by finding the best grasp G, when $G_{t_0:t-1}$ and $T_{t_0:t-1}$ are known (subscripts denoting that these are from the previous attempts).

The process described by Equation (5.3) is depicted in Figure 5.1, which shows that the knowledge of the object attributes O is iterated over the time steps, $t_0, \ldots, t-1, t, t+1, \ldots, t_n$. The knowledge of O is refined using information from the known grasp attributes G and the measurements T. One of the drawbacks of the presented framework is that the object must stay as static as possible across the grasp attempts. This may seem unreasonable, but given the advancements in tactile sensing, it should be possible to detect contact without disturbing the object in the future. However, note that the first grasp attempt, which is likely not to be a stable grasp, can of course move the object freely. Furthermore, if the object moves between the grasp attempts, the motions only increases the uncertainty of the estimation as the measurements do not correspond, for example, to a single object pose across the grasps.

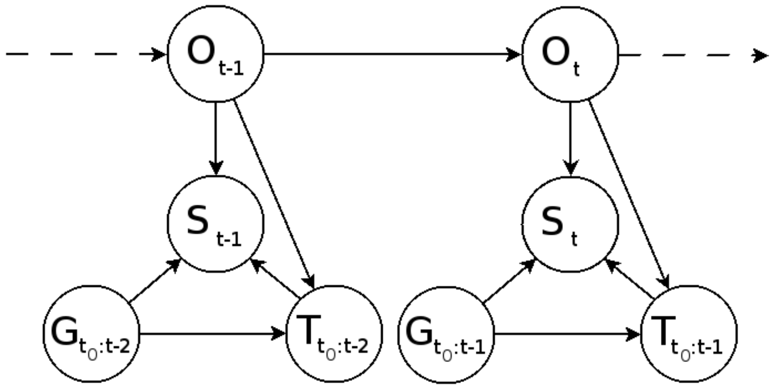


Figure 5.1: Process of refining object knowledge.

To build a working system based on Equation (5.3), two models are needed:

- Model for $T|G,O$, describing the relation between measurements and grasp and object attributes
- Model for $S|G,O$, stability as a function of grasp and object attributes

It should be noted that posterior distribution $P(O|G,T)$ can be obtained from the prediction model of sensor measurements $T|G,O$ using the Bayes formula, where $P(T|G,O)$ is the likelihood for the posterior distribution $P(O|G,T)$. These models are not trivial to build and depend on the object and the manipulator used to grasp the object. Still, there exists models for both cases, see e.g. [138, 99, 135] for a model for $P(O|G,T)$ and [6] for a model for $P(S|G,O,T)$ or [129] for $P(S|G,O)$. The simplest approach to create these models is to utilize a simulator with geometric models of both the object and the end effector. However, this may become unfeasible due to the computational demands of more complex objects and end effectors. The second approach is to generate the models from data by sampling a set of grasps and recording the required measurements T and the stability of each grasp. This procedure also applies to real end effectors. Both of these approaches are presented in section 5.3 to demonstrate the framework. The section also includes experiments with a real robot.

The framework does not place constraints on the actual models, and the attributes G, O and T can be freely chosen. For example, G and O can include the poses of the manipulator and the object. The benefit of the presented probabilistic framework is that throughout the grasping process the uncertainty of the actions arising from Equation (5.3) is known. Also, measurement errors can be accounted for during both grasp planning and on-line grasp stability detection.

5.3 Experiments

As was described in section 5.2, the grasping framework gives a great deal of room to choose the models and possible attributes. For this reason, the framework is demonstrated with a simulation based approach in section 5.3.1 and a data-driven approach in section 5.3.2, where both simulated and real data is used. However, both experiments model a table top scenario where only top grasps are performed. In addition, only three DOFs, translation and rotation on a plane, are used in the experiments.

The simulated approach is based on a simple static simulation of a rectangle and a two finger gripper to keep the computational cost of simulating a single grasp small. The depicted problem is quite basic, but demonstrates the principles of the framework adequately. These results were published in [9] and represent a simpler version of the framework which does not consider a sequence of multiple grasp attempts.

The data-driven model is based purely on data. The models used in these experiments are built with Gaussian Process Regression. The GPR was chosen as it can effortlessly represent the uncertainty in the prediction which can be directly used in the estimation of the posterior distribution $P(O|G,T)$. The GPR can be thought of as a surrogate for the simulation, comparable for example to [107], where SVM was used to learn grasp stability for certain superquadrics. This data-driven approach to the framework has been submitted to 2012 International Conference on Intelligent Robots and Systems [10].

The general approach for both demonstrations is based on the sequence of actions shown in Figure 5.2. It is assumed that some type of initial estimate (with associated uncertainty) of the object pose is obtained in phase 1, for example from vision. Using the estimate, a grasp can be planned with the uncertainty from the initial estimate (phase

2). Then a grasp is performed (phase 3) giving measurement data (at least joint configuration data should be made available). Using the measurement data, a decision on the grasp stability can be made (phase 4). If the grasp is stable, the object can be manipulated, if not, a new grasp can be planned (phase 5) with the new information from the attempted grasp. This loop can then be further iterated until grasp stability conditions are satisfied.

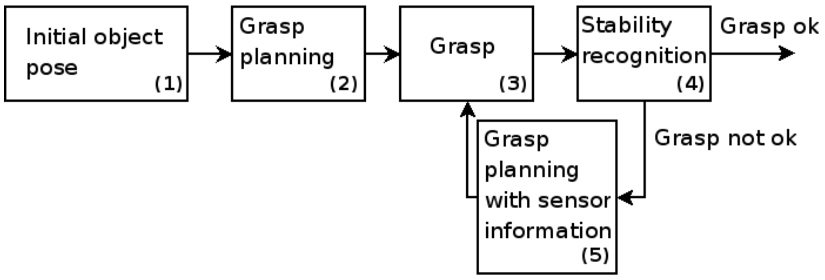


Figure 5.2: Sequence of actions.

Both of the demonstrations are based on the same algorithm, which is construed in Algorithm 5.1. The algorithm is divided into three parts. Algorithm 5.1 describes the sequence of actions shown in Figure 5.2. Algorithms 5.2 and 5.3 are subalgorithms that infer the object properties, given some measurements T^*, and find the best grasp possible given the known uncertainty, respectively. The variables used in the algorithms are marked with a bold font.

Algorithm 5.1 requires the initial estimates of the uncertainty given in $\mu_{\mathbf{init}}$ and $\sigma_{\mathbf{init}}$ for each of the variables (x, z, θ). The particle set $\mathbf{O}$ in Algorithm 5.1 represents the posterior distribution of the object, $P(O|G,T)$. This initial particle set is then given to the grasp planner, Algorithm 5.3, in which the particle filter technique is utilized to find a grasp that is most likely to produce a stable grasp. Although particle filters are usually employed with dynamical systems for tracking, here the particles are guided with a random process to find a single maximum value from the posterior $P(S|G,O)$. Particle swarm optimization (PSO) [139] is also used in the experiments in section 5.3.2. PSO algorithm replaces Algorithm 5.3 in these experiments completely. The result of the algorithm is a single grasp that produces the most stable grasp given the uncertainty present in $\mathbf{O}$.

After a suitable grasp has been found in Algorithm 5.3, it is executed. The grasp configuration is stored in an array **grasp** and the measurements, in this case the joint configuration after the grasp execution, are stored in an array $\mathbf{T}^*$. Note that all measurements and grasp configurations are stored to enable the inference over sequential grasps. In Algorithm 5.2, the measurements and the grasp configuration are used to infer the pose of the object. The particle set $\mathbf{O}$, which describes the posterior distribution of the object pose, is refined using the information from the object model, **object** with the stored grasp configurations and measurements. The goal is to produce a posterior distribution for the object pose which can then be utilized in Algorithm 5.3. The method used to infer the posterior distribution is different in the two experiments and will be detailed further in the corresponding sections.

Given the refined uncertainty **O**, the probability for a stable grasp can be estimated, and given a threshold, **p**, a decision can be made whether to manipulate the object further, in line 10 of Algorithm 5.1. Remember that Algorithm 5.2 has run while the object is still in the grasp, making it possible to decide whether, for example, to lift the object or continue executing Algorithm 5.1.

Relating to Figure 5.2, Algorithm 5.3 takes care of the grasp planning, that is, phases 2 and 5. Algorithm 5.2 handles phase 3, updating the posterior distribution of the object pose, but Algorithm 5.1 handles the execution of a grasp. In lines 9-11 of Algorithm 5.1, the grasp stability probability is computed and corresponds to phase 4 of the action sequence.

Algorithm 5.1 find_stable_grasp(object,n,p,$\boldsymbol{\mu}_{init}$,$\boldsymbol{\Sigma}_{init}$)

```
1:  Generate initial particle set O of size n, according to $\mathcal{N}(\boldsymbol{\mu}_{init}, \boldsymbol{\Sigma}_{init})$
2:  $q \leftarrow 1$
3:  while $q >= 1$ do
4:      $(\mathbf{x}, \mathbf{y}, \theta) \leftarrow$ find_best_grasp(object,O,$\boldsymbol{\Sigma}_{init}$)
5:      grasp(q) $\leftarrow (\mathbf{x}, \mathbf{y}, \theta)$
6:      Move end effector to grasp(q)
7:      Grasp object, measure $\mathbf{T}^*$(q) after grasp is complete
8:      O $\leftarrow$ find_object_attributes(object,grasp,$\mathbf{T}^*$, O)
9:      $G \leftarrow$ grasp(q)
10:     Approximate $P(S|G)$ by $\sum_i \frac{1}{\mathbf{n}} P(S|G, \mathbf{O}_i)$
11:     if $P(S|G) > $ p then
12:         $q \leftarrow 0$
13:     end if
14:     $q \leftarrow q + 1$
15:     Release grasp
16: end while
```

Algorithm 5.2 find_object_attributes(object,grasp,$\mathbf{T}^*$, O)

```
1:  while O is not converged do
2:      for i=1, ..., n do
3:          for j=1, ..., q do
4:              Query the object model, object, for measurements $\mathbf{T}_i$(j) given known $\mathbf{O}_i$ and
                grasp(j)
5:          end for
6:      end for
7:      Estimate posterior distribution $P(O|G,T)$ through the likelihood $P(\mathbf{T}_i|\mathbf{T}^*)$
8:      Choose a new particle set, $\mathbf{O}^{new}$, based on O and a chosen statistical approach
9:      O $\leftarrow \mathbf{O}^{new}$
10: end while
11: return O
```

Algorithm 5.3 find_best_grasp(object,O,$\mathbf{\Sigma}_{init}$)

```
1:  $\boldsymbol{\mu}_O \leftarrow \text{mean}(\mathbf{O})$
2:  $\mathbf{\Sigma}_O \leftarrow \text{cov}(\mathbf{O})$
3:  Generate a particle set, $\mathbf{M}$ according to $\mathcal{N}(\boldsymbol{\mu}_O, 5\mathbf{\Sigma}_O)$
4:  while $\mathbf{M}$ is not converged do
5:      Weigh particles in $\mathbf{M}$ with $\mathbf{w} \propto P(S|G,\mathbf{O})$
6:      $(\mathbf{x}_{max}, \mathbf{y}_{max}, \theta_{max}) \leftarrow \arg\max_G P(S|G,\mathbf{O})$
7:      Apply importance filtering using $\mathbf{w}$
8:      Use $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ as proposal distribution with $\mathbf{\Sigma} \leftarrow 0.2\ \mathbf{\Sigma}_{init}$
9:  end while
10: return $(\mathbf{x}_{max}, \mathbf{y}_{max}, \theta_{max})$
```

5.3.1 Simulation model

The purpose of this demonstration is to demonstrate the theoretical possibilities of the basic idea behind the framework described in Section 5.2. The simulation environment is depicted in Figure 5.3. The environment consists of a parallel jaw gripper with finger width $l_{\texttt{finger}}$ and a rectangular object with side lengths of 6 and 2. Note that in the demonstration, the cases where the gripper fingers would touch the shorter side or the corners of the rectangle are not considered. The angle of the gripper in degrees is denoted by θ, which is zero when the gripper is perpendicular to the long side of the rectangle. The gripper center is denoted with (x, y), which is relative to the object center (x_0, y_0). In the experiment, the object is static. When grasping, the two fingers of the gripper can be closed independently of each other and the distance to contacts from gripper center d_1 and d_2 are used as the measurements, representing the sensor information T. Also, it is assumed that the fingers have the capability to detect when they come into contact with the object and that the fingers can be stopped at those time instants. A 3-tuple (x, y, θ) is used to denote the gripper variables, which are in relation to the object center (x_0, y_0, θ_0). Moreover, (x, y, θ) represent G, the grasp attributes, while (x_0, y_0, θ_0) represent O, object attributes.

Note that because of the symmetry of the setup, there is ambiguity about the orientation of the object, due to the symmetrical fingers of the gripper. Also due to the symmetry of the object, the uncertainty along the x-axis cannot be reduced.

In these experiments, Algorithm 5.2 is implemented with a particle filter to make the computation of posterior distribution $P(O|G,T)$ tractable. Algorithm 5.4 describes the object pose inference used in the experiments. Particle filters have been used in manipulation, for example in [138], to estimate the object pose using tactile sensors. The advantages of this approach is that the particle filter process quickly converges to the maxima of the posterior distribution. However, particle filters do not represent the whole posterior distribution as well as some other methods, such as the Metropolis- algorithm, which is used in the data-driven experiments. Likelihood, $P(T|G,O)$, which are shown in Figure 5.4, were chosen manually for the purposes of this demonstration. Figure 5.4(a) shows how likely a measurement T, in this case d, is a correct measurement in relation to the true measurement T^* or d^*; this model is used for both d_1 and d_2, and represents additive Gaussian noise in the measurement. Figure 5.4(b)-(d) represents how likely a grasp is

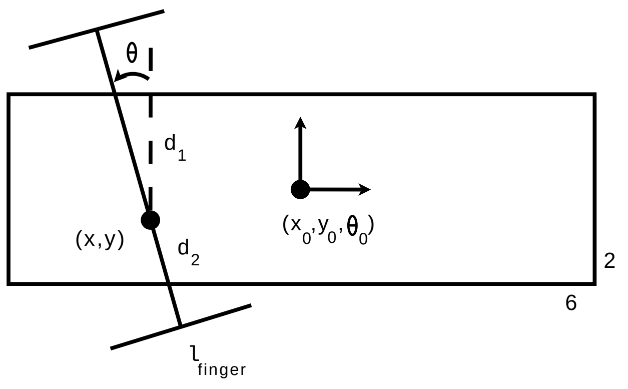


Figure 5.3: Simulation environment.

stable relative to the posterior distribution of object pose O. The stability model defines that a grasp is stable when the grasp is performed close enough to the center of the object. The stability model is further simplified so that the model is factorized into independent factors for different coordinate axes, i.e. , $P(S|G, O) = P(S|x, x_0)P(S|y, y_0)P(S|\theta, \theta_0)$.

Algorithm 5.4 find_object_attributes_pf(object,grasp,T*,O)

```
1:  while $\mathbf{O}$ is not converged do
2:      for i=1, ..., n do
3:          for j=1, ..., q do
4:              Query the object model, object, for measurements $\mathbf{T}_i$(j) given known $\mathbf{O}_i$ and grasp(j)
5:          end for
6:      end for
7:      Assign weights, $w$, for each particle in $\mathbf{O} \propto P(\mathbf{T}_i|\mathbf{T}^*)$
8:      Choose a new set of particles $\mathbf{O}^{new}$ according to importance filtering of $\mathbf{O}$
9:      Apply a proposal distribution to $\mathbf{O}^{new}$
10:     $\mathbf{O} \leftarrow \mathbf{O}^{new}$
11: end while
12: return O
```

Results

Figure 5.5 shows a single example run of Algorithm 5.1. The example was run with the true object pose (x_0, y_0, θ_0) set to (0, −0.3, −15). Figure 5.5(a) shows the initial distribution of O_1, which was initialized around zero with $\sigma_{\mathbf{init}} = [0.3\ 0.3\ 6]$. Particle locations are shown in green, indicating the possible object location, and $\frac{1}{4}$ of the particles are plotted with a blue line, indicating the orientation θ_0 of the object. As the distribution is Gaussian, most of the particles are located near the mean, both in position and orientation. As the initial distribution is zero mean, during the first iteration the grasp planning stage will produce a grasp that would be optimal if the object were at $(0, 0, 0)$.

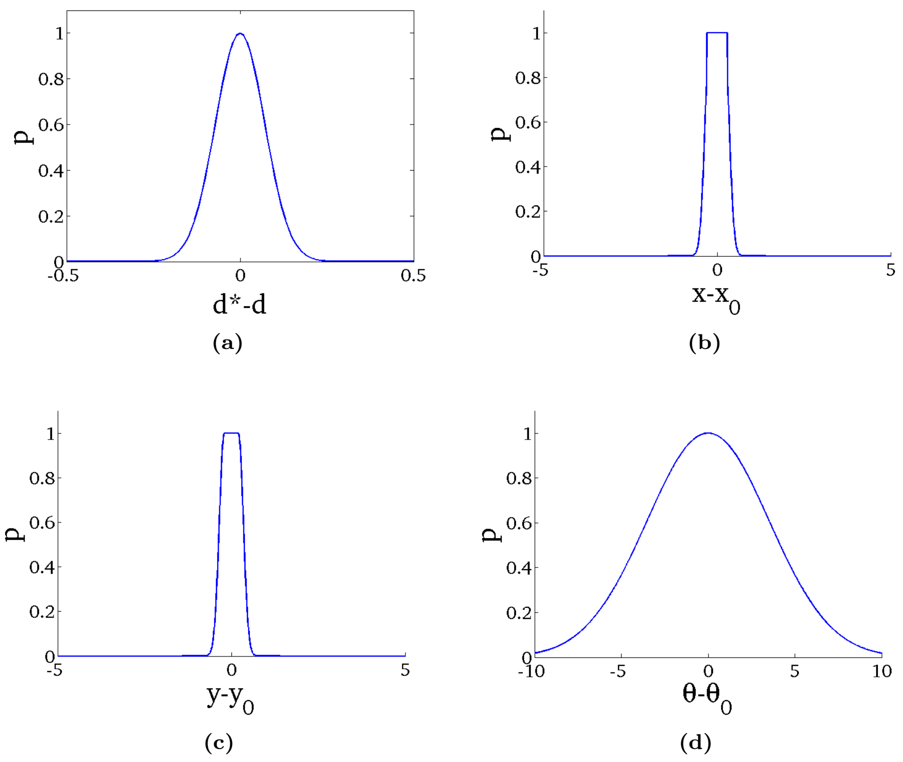
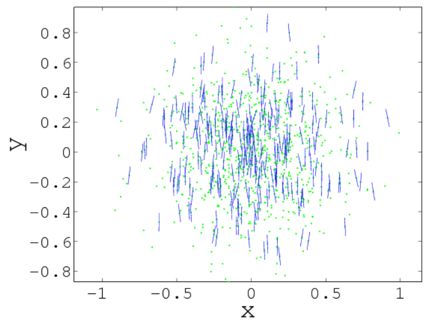


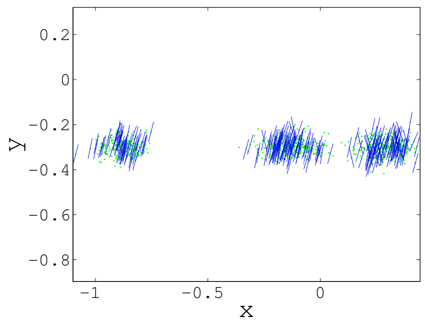
Figure 5.4: Likelihoods for: (a) Measurement model, $p(T|T^*)$; (b) Grasp stability, x; (c) y; (d) θ.

In Figure 5.5(b), the first grasp attempt has been made, and the distribution of object pose changes to account for the measurements, d_1 and d_2. The figure also shows the symmetry of the problem and two modes arising from this symmetry, one for $\theta_0 = 15$, the other for $\theta_0 = -15$. This grasp does not satisfy the threshold of 0.5 for the grasp stability probability. Maximizing $P(S|G,O)$ yields the solution $(0.07, -0.32, -14.3)$ for the grasp G. Figure 5.5(c) shows the posterior distribution of O after the information from the second grasp is used. This grasp is determined as stable as the probability of a stable grasp is greater than the probability of an unstable grasp. The mean of the final posterior distribution O was $(-0.24, -0.32, -14.28)$ compared to the set pose which was $(0, -0.3, -15)$. As can be seen, the method was able to find a corrective motion for the gripper and produce a stable grasp and after the two grasps, the particle cloud converged to near optimal values for the object pose, except for the x-variable for which the uncertainty cannot be reduced.

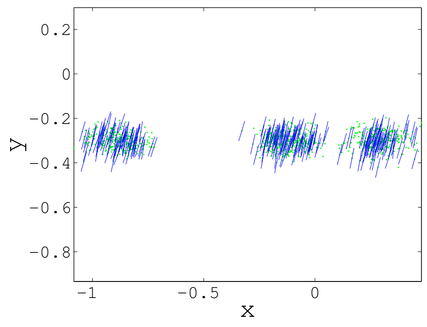
However, as the method is probabilistic, for the second grasp attempt maximizing the probability $P(S|G)$ can produce a motion that is opposite to the correct one in θ. This is the result of the symmetry present in both the gripper and the object. For example, if the widths of the fingers on the gripper were different, the object rotation angle θ_0 could



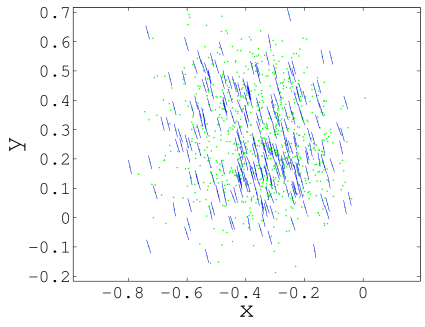
(a)



(b)



(c)



(d)

Figure 5.5: Sequential distributions of particles modeling $P(O|G,T)$, (a)-(c), for a single run of Algorithm 5.1: (a) Initial distribution ; (b) Distribution after the first grasp; (c) Distribution after the second grasp, for which $P(S|G) = 0.503$; (d) An example of the particles used in Algorithm 5.3.

be solved from the first grasp.

One of the benefits of the probabilistic approach is that uncertainty behind the actions is known. By marginalizing over O, the uncertainty is directly taken into account in the stability estimation and grasp planning. Also, if non-contact measurements are preferred, a low probability of a stable grasp in the planning could be used to trigger additional visual measurements to improve the object pose estimate.

5.3.2 Data-driven model

Contrary to the experiments in section 5.3.1, the data-driven models do not require simulation to infer the object pose or the stability of the grasp. The models replace the need for simulation. Of course, simulation is still used in a majority of the experiments due to the need to simulate the actual grasps, but fundamentally, the simulation environment is not required by the data-driven models. Also, compared to the previous experiments, the object pose inference is implemented with the Metropolis algorithm [140]. The Metropolis algorithm was chosen for the object attribute inference because the method allows modeling of the whole distribution without the degeneracy problems occuring with particle filters. Algorithm 5.5 describes the use of Metropolis algorithm in the context of object pose inference. Note that traditionally the Metropolis algorithm has been used to create a single chain across many iterations of the Metropolis algorithm which represents the posterior distribution. Here, instead of using just one chain, a set of chains, as many as there are particles in the set **O**, is created and the chains are iterated fewer times. This choice was made due to practical reasons; many simultaneous chains allowed efficient sampling of the posterior distribution using the GPR models that were used in the experiments. However, this setup should not change the outcome, i.e. the estimate of the object pose posterior distribution.

Algorithm 5.5 find_object_attributes_metro(object,grasp,$\mathbf{T}^*$, O)

```
1:  while O is not converged do
2:    For each particle $\mathbf{O}_i$ in set O, generate a proposal $\mathbf{O}_i^{new}$ using distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$
3:    for i=1, ..., n do
4:      for j=1, ..., q do
5:        Query the object model, object, for measurements $\mathbf{T}_i$(j) given known $\mathbf{O}_i$ and grasp(j)
6:        Query the object model, object, for measurements $\mathbf{T}_i^{new}$(j) given known $\mathbf{O}_i^{new}$ and grasp(j)
7:      end for
8:      Compute acceptance probability $\alpha = \min\left[1, \frac{P(\mathbf{T}_i|\mathbf{T}^*)}{P(\mathbf{T}_i^{new}|\mathbf{T}^*)}\right]$
9:      Replace $\mathbf{O}_i$ with $\mathbf{O}_i^{new}$ with probability of $\alpha$
10:   end for
11: end while
12: return O
```

For the experiments in the simulation, five objects (two mugs, a pitcher, a cylinder and a cube) were chosen. The objects are shown in Figure 5.6 with the end effector, the Barrett hand. The simulation environment was the GraspIt! simulator. The data was collected using a 3D grid in (x, z, θ), where the x and z represent the table plane. The object was grasped at each of the points in the grid. For the cylinder and cube, x and z were discretized from -50 mm to 50 mm at 10 mm intervals and θ was discretized from -180 to 180 degrees at a 15 degree interval. For the two mugs and the pitcher, x and z were discretized from -100 mm to 100 mm at 20 mm intervals and θ was discretized from -180 to 180 degrees at a 15 degree interval. The grid was centered on the object and contained 3025 points for each object. Each grasp was executed using the auto grasp-function in GraspIt!; thus, only one preshape was used when grasping. After each grasp, the quality of the grasp was measured and the finger joint values were recorded with the relative pose between the object and the end effector. For the quality measure, the existing quality measures from GraspIt! were used. In this case, the force-closure quality measure, i.e. the ϵ-measure, was employed. Values above 0.1 were considered fully stable. From these measurements, one can build the models $S|G, O$ and $T|G, O$. Note that the object is modeled through the joint configuration and not as a geometric presentation.

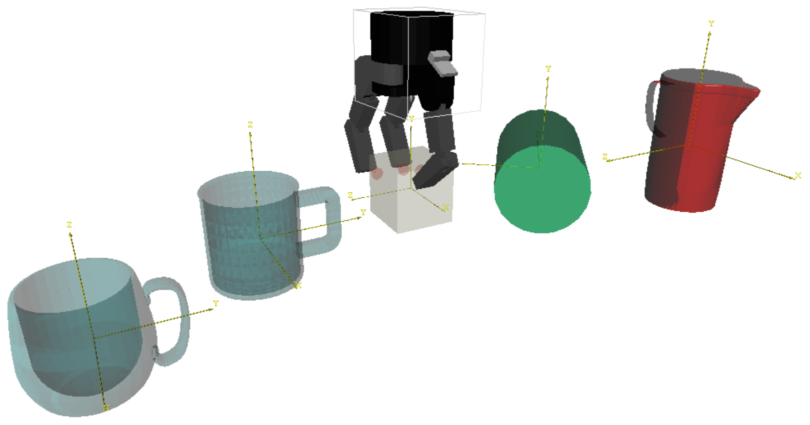


Figure 5.6: Objects used in the experiments, from the left: mug 1, mug 2, cube, cylinder and a pitcher. The simulated Barrett-hand is also shown in the figure grasping the cube.

A similar procedure was performed on two real robotic platforms. The platforms consisted of a Melfa RV-3SB 6-DOF arm and either a Weiss WRT-102 robotic gripper or a Robotiq 3-finger hand. The same approach was used to sample the objects with the platforms, as was used with the simulated objects. The platform with the WRT-102 gripper and the sampled object, a correction roller, is shown in Figure 5.7. The platform with the Robotiq hand and the object, a cardboard box, is shown in Figure 5.8.

The platform with the WRT-102 gripper posed some practical problems due to the single DOF – the width of the grip. To work around this problem reactive grasping was utilized to enable grasping the object. Additionally, the object had to be stationary. This was achieved by suspending the object on three screws that were mounted on the base. The gripper width was also the only measurement T used with this platform. Furthermore, due to the limitations of the gripper, e.g. the maximum gripper width being 70 mm, the DOFs were decreased from three to two, (x, θ). The dimensions of the correction roller were 45 mm across Y and 120 mm across X. A total of 26 samples were collected from the object by grasping from -50 mm to 90 mm in x and from -7 degrees to 7 degrees in θ. The stability of each grasp was determined manually by lifting the object after each grasp and labeling each grasp as stable or unstable, depending on the outcome of the lift. If the object moved in relation to the gripper during the lift, the grasp was unstable, otherwise the grasp was stable.

The platform with the Robotiq hand had fewer practical issues, and the object could be grasped in the experiments without making the object static. The dimensions of the cardboard box were 55 mm across Y and 240 mm across X. We collected 133 samples from the object, sampled area was -170 mm to 190 mm in x, -20 mm to 20 mm in y and -20 degrees to 20 degrees in θ. The object was kept static during the sampling process. The Robotiq hand is underactuated, each finger has three joints, but each finger is driven only by a single actuator in the base of the hand. We used the measurements from the positions of the three actuators as the measurements T.

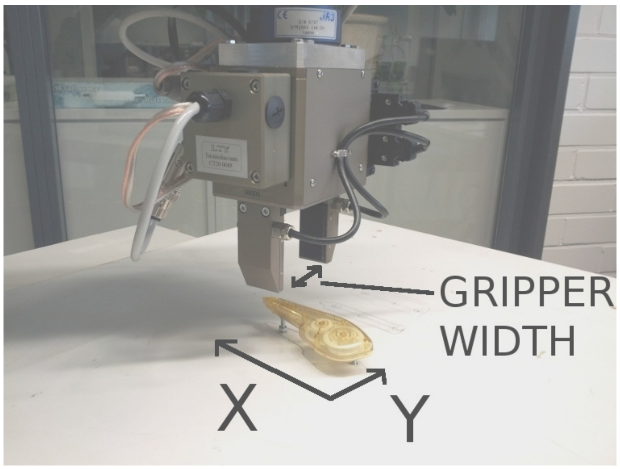


Figure 5.7: The WRT-102 gripper and the object, a correction roller.

To build the required models, Gaussian process regression was employed in this task. Both models, $S|G, O$ and $T|G, O$, were built using the GPR. Next, a brief introduction is given to GPR.

Gaussian process regression for modeling objects

The Gaussian Process (GP) used in GPR is defined as a set of random variables of which any finite number have a joint Gaussian distribution. The GP function is defined as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \ , \tag{5.4}$$

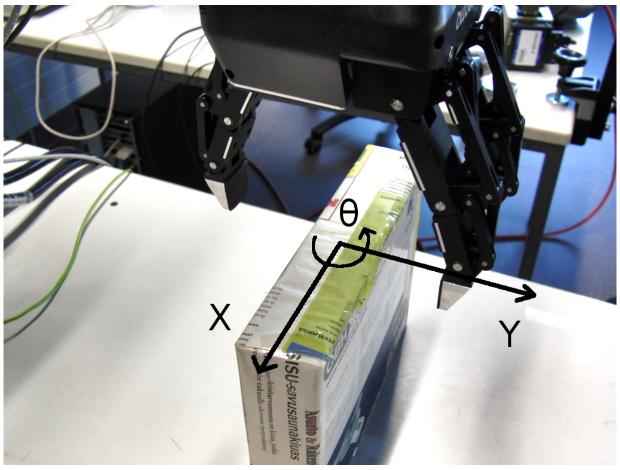


Figure 5.8: The Robotiq hand and the object, a cardboard box.

where

$$m(x) = \mathbb{E}[f(x)] \ , \tag{5.5}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \ . \tag{5.6}$$

The GP is constructed of the mean function $m(x)$ and the covariance function $k(x, x')$. As GP models the data using these functions, the GP is also able to model the underlying uncertainty present in the data, which increases in the gaps of the data and decreases where the data has a high density.

Suitable mean and covariance functions are usually dependent on the form of the data and require some thought. Different covariance functions enable different types of data to be modeled. In the case of grasping, many discontinuities appear due to discrete events, such as a finger missing an object completely during grasping. Due to this phenomenon, the neural network covariance function [141] was chosen:

$$k(x, x') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{x}^T \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^T \Sigma \tilde{x})(1 + 2\tilde{x}'^T \Sigma \tilde{x}')}} \right) \ , \tag{5.7}$$

where $\tilde{x} = (1, x_1, \ldots, x_n)$, which is an augmented input vector. The neural network covariance function is non-stationary and can model discontinuous data better than the more commonly used squared exponential covariance function [142]. Choosing which mean function to use is not as critical to the regression as the covariance function, and therefor a constant mean function was chosen.

Once the model has been selected, GPR finds the most probable function over the training data. GPR can then be used to estimate $f(x)$, given a new x. Here, the input x is the 6 DOF relative pose between the end effector and the object $(X, Y, Z, \alpha, \beta, \gamma)$ while the output, i.e. $f(x)$, is the joint angle of a finger of the end effector in the case of the model

$T|G, O$ and the stability of the grasp in the case of the model $S|G, O$. Thus, each finger of the end effector is separately modeled to form the model $T|G, O$.

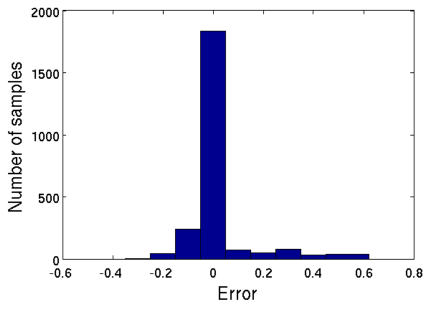
However, to operate properly, GP requires that hyperparameters, i.e. parameters of the mean and covariance functions, are set as well. Finding proper values for the hyperparameters is the most challenging task with GP techniques, but methods exist that use training data to optimize the hyperparameters. Readers interested in GPs can get a deeper understanding from e.g. [142].

The practical implementation used in the experiments was the GPML Matlab toolbox [143]. The reason why this particular implementation was chosen was that it provided a complete toolkit for both Gaussian process regression and classification with a number of ready-made covariance and mean functions and the minimization functions to optimize the hyperparameters for the Gaussian processes. The toolbox also included an approximation technique (fully independent training conditional or FITC) [144] that can be used with large datasets to reduce computational demands. This feature was used in all of the simulated data experiments. However, the approximation also reduces the accuracy of the regression. A set of error histograms are shown in Figure 5.9. These histograms depict the error in the prediction of $T|G, O$, given a separate test set. The GPR computation times for the test set were 3.7 seconds for GPR without FITC, 1.9 seconds for GPR with 33% of data used as inducing inputs and 0.09 seconds for GPR with 5% of the data used as inducing inputs. The times were obtained from the GPML toolbox running on dual core 2.6 GHz Intel Core i5 processor. The computation times show that a significant speed up can be achieved by utilizing the FITC approximation. In the experiments, only 5% of the 3025 samples were used as the inducing inputs for the FITC approximation to enable fast computation of the posterior distributions $P(O|G, T)$ and $P(S|G, O)$.

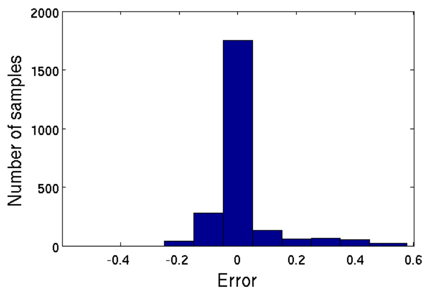
Simulated experiments

In the experiments, a sequence of grasps is executed following the diagram in Figure 5.2. The experiments show that objects can be localized using the Metropolis algorithm, given some uncertain initial estimate. Then, using the computed distribution of the object pose, a stable grasp can be achieved despite the uncertainty still present in the pose and the relatively inaccurate models of the objects. All experiments, except the experiments with the WRT-102, utilized the PSO algorithm to find the maximum of the grasp stability. In the experiments with the WRT-102, Algorithm 5.3 was used instead. The difference between the algorithms was 1-2%, i.e. the grasps that PSO algorithm found improved the grasp stability expectation by 1-2% compared to the particle filter algorithm. In all of the experiments, 1000 particles were used to model the object attribute distribution in (x, z, θ), and 40 particles and 20 iterations were used within the PSO. Furthermore, in the the experiments with the WRT-102, 100 particles were used in the particle filter process.

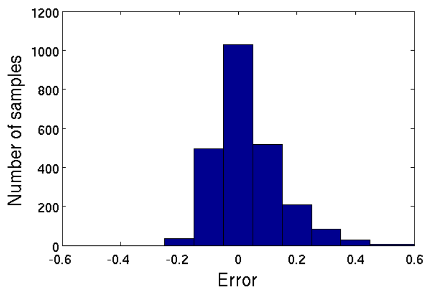
In the experiments, the initial object mean $\mu_o = (0, 0, 0)$ and the initial object standard deviation $\sigma_o = (30, 30, 30)$ were assumed. Furthermore, μ_o and σ_o are given as input to Algorithm 5.1. In the experiments, the number of grasps was limited to four, instead of defining a threshold for the stable grasp probability. In Algorithm 5.5, the object posterior distribution refinement was limited to 50 iterations of the Metropolis algorithm



(a)



(b)



(c)

Figure 5.9: Prediction of $T|G, O$: (a) Error histogram without FITC; (b) Error histogram with FITC (33% of data); (c) Error histogram with FITC (5% of data)

instead of a more sophisticated convergence criterion. However, 50 iterations were found to form the posterior distribution adequately. In total, 500 000 evaluations of the model $T|G, O$ and approximately 3 million evaluations of the model $S|G, O$ are made in each individual experiment. If the GPR models were replaced with a simulation, it would mean that over three million simulated grasps would be needed to be made for each experiment, which is currently infeasible. With the GPR models, each grasp can be planned in less than two minutes on a computer with a 3.0 GHz Pentium D processor released in 2005.

Due to the probabilistic nature of the algorithms, the first results are shown as examples of posterior distributions, which can be seen in Figure 5.10 for two different runs of Algorithm 5.1. Each run in Figure 5.10 shows how the posterior distribution of the object pose evolves after grasp attempts are made and each run consists of four grasps. The real object pose is marked with a red cross in each subfigure. The title of each subfigure in the figure shows the quality measure (QM) computed by GraspIt! and the corresponding stable grasp probability (Stab. Prob). A quality measure of -1 indicates a non-force-closure grasp, i.e. a failed grasp. Note that the orientations of subfigures differ to give a better view of the whole posterior distribution.

The first run in Figure 5.10(a)-(d), shows the result for the cube in Figure 5.6. The distributions show that multiple modes are found near the correct pose. The probability

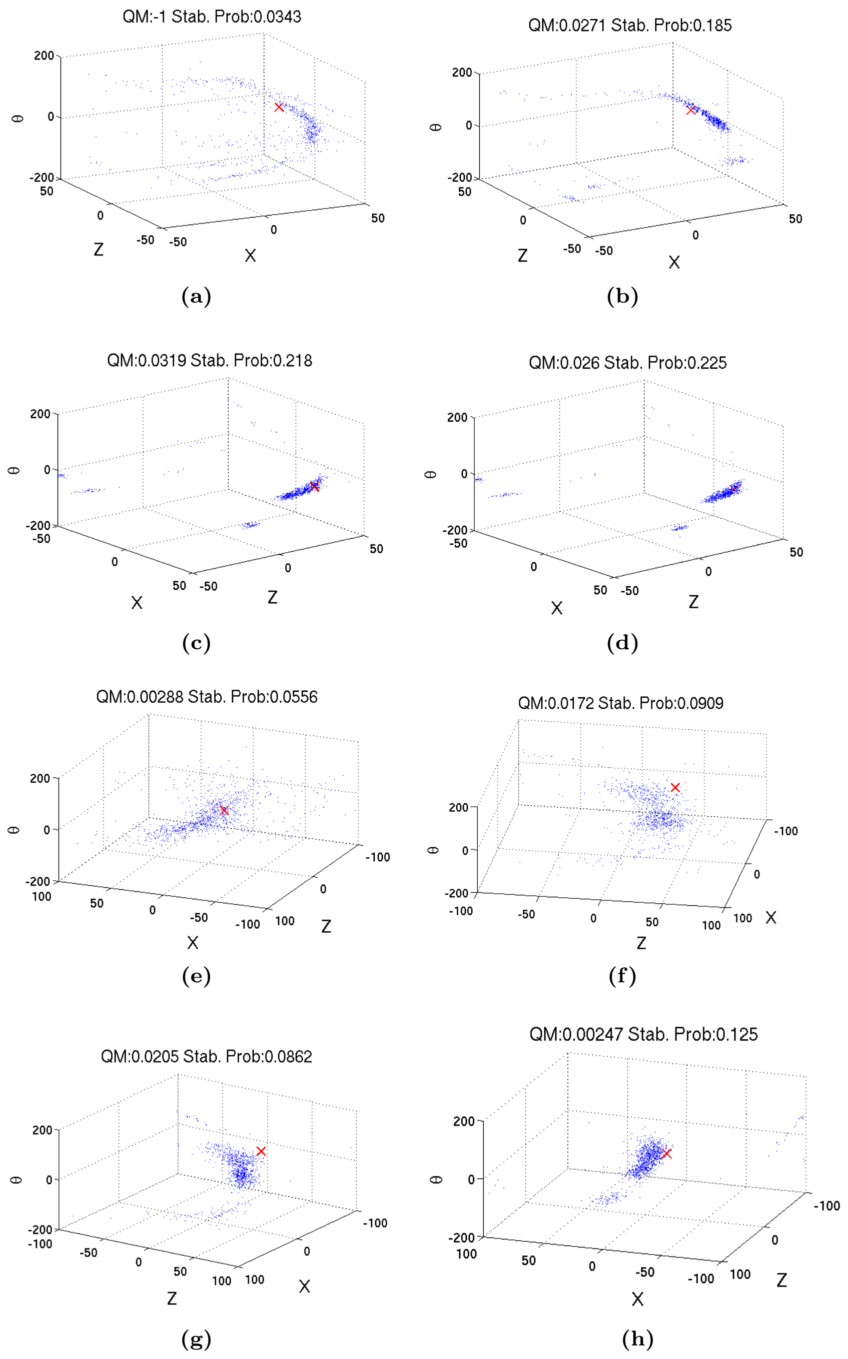


Figure 5.10: (a)-(d): Grasping the cube at pose (45,25,0);(e)-(h): Grasping the mug 2 at pose (-32,38,68).

Table 5.1: Probability of a force closure grasp across 100 runs.

Grasp	Cube	Cyl.	Mug1	Mug2	Pitcher
1	0.43	0.49	0.47	0.27	0.13
2	0.76	0.52	0.82	0.49	0.40
3	0.76	0.56	0.85	0.61	0.46
4	0.72	0.60	0.87	0.72	0.53

of a stable grasp increases after each successive grasp due to the convergence of the object pose posterior. The quality measure also shows that force-closure is achieved during the grasps. In the second run, in Figure 5.10(e)-(h), mug 2, shown in Figure 5.6, is grasped. The posterior distribution reflects the uncertainty in the orientation (vertical axis of the figure) of the mug, as the mug is almost completely symmetric. As before, the grasp stability probability increases after each grasp. The quality measure is also improved. However, the posterior density is not as dense as in Figure 5.10(a)-(d) with remaining uncertainty due to the symmetry. While the quality measure is increased between the grasp attempts in most cases, the quality measure may also decrease due to the imperfect stability model.

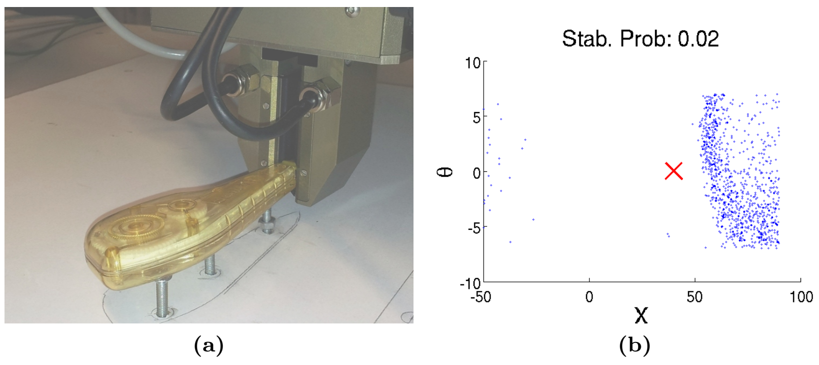
All of the posterior distributions show some outliers at the edges of the parameter space. This is due to the imperfect regression results by GPR. One of the probable reasons behind this phenomenon is the discretization of the sampling grid which is quite coarse especially on the mugs and the pitcher.

To show that the probabilistic grasp planning presented in Algorithm 5.1 actually improves the grasp quality, 100 tests were run on all five objects to find out how each successive grasp reduces unstable grasps by refining the knowledge on the object pose. For each individual run, the object pose was randomly chosen from a uniform distribution within the model boundaries described in 5.3.2. The results are described in Table 5.1, which shows the probability of achieving a stable grasp after the number of grasps shown in the "Grasp" column. From the results, one can see that the framework is able to refine the object pose distribution, and as a result, find a stable grasp more often. These results can be compared to some extent to the results found in [135], although the object set used in [135] is different. The results reported in [135] indicated that four actions are needed to grasp an object correctly with 80% to 90% probability. However, in [135], each action is planned to be as informative as possible. In this thesis, each grasp is planned to produce the maximal stability. Also note that the definition of a successful grasp is different in [135]. This thesis considers any grasp with a force closure property a success, while in [135], the grasp must be within defined bounds.

Experiments on real robotic platforms

To validate the approach proposed in this paper on a real platform, the framework was also implemented on the two robot platforms described earlier and shown in Figures 5.7 and 5.8.

In the experiment with the WRT-102, the object was displaced 40 mm, while setting $\mu_o = (0, 0)$ and $\sigma_o = (40, 4)$. The results of the two grasps executed with these parameters



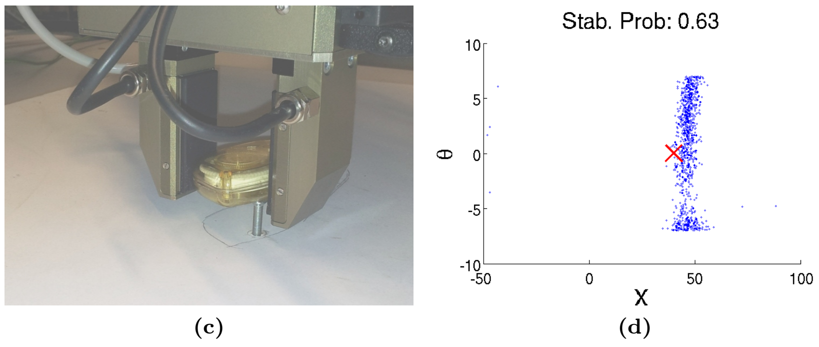
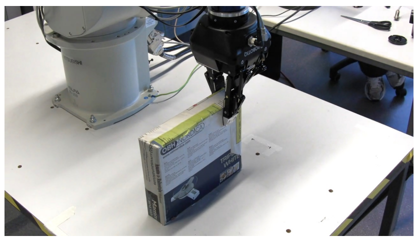


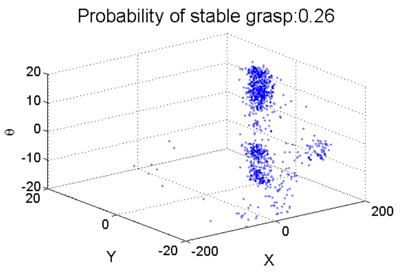
Figure 5.11: (a) First, unstable, grasp in sequence; (b) Posterior distribution after first grasp; (c) Second, stable, grasp; (d) Posterior distribution after second grasp.

are presented in Figure 5.11. As can be seen from the figure, the posterior distribution converges close to the real pose in X. However, θ remains largely uncertain because the measurements are not able to disambiguate the angle of the object. The stability probabilities also reflect reality. After the first grasp, the probability of achieving a stable grasp given by the model is only 2%. The first grasp was an unstable grasp when lifting the object. After the second grasp, the probability had grown to 63%. The second grasp was stable enough to lift the object.

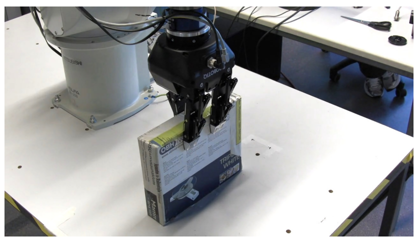
Two experiments are reported for the platform with the Robotiq hand. However, many similar experiments were conducted with this platform with consistent results, and the reported experiments are examples of the expected outcomes of similar experiments. The experiments show that the framework is able to function even when the assumption of a static object is relaxed. In the first experiment the object is intially displaced roughly 120 mm in x and 20 mm in y and the object is also rotated by a small angle from the expected mean. The initial uncertainty is set to $\mu_o = (0, 0, 0)$ and $\sigma_o = (40, 5, 5)$. The results of the two grasps executed with these parameters are presented in Figure 5.11. As can be seen from the figure, the posterior distribution $P(O|G, T)$ converges close to the real pose in X after the first grasp. The first grasp also aligned the object with the hand. However, θ remains uncertain, because the measurements are not able to disambiguate this. The stability probabilities reflect reality well as after the first grasp, the probability



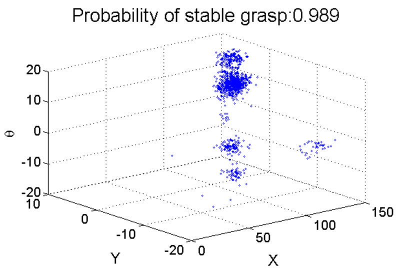
(a)



(b)



(c)



(d)

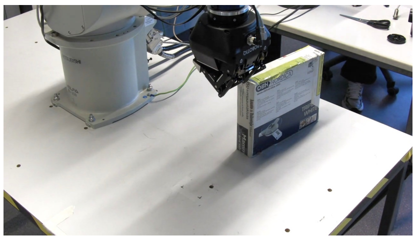
Figure 5.12: (a) First grasp in sequence which is an unstable grasp; (b) $P(O|G,T)$ after the first grasp, note the low probability of stable grasp; (c) Second grasp, which is a stable grasp; (d) $P(O|G,T)$ after the second grasp.

of a stable grasp given by the model is only 26%, which in reality is an unstable grasp, but after the second grasp the probability has grown close to 99%, which was a stable grasp when lifting the object.

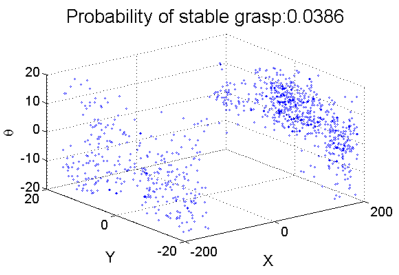
The second experiment shows the implicit capability of the approach to explore the environment. In this experiment, the robot fails to grasp the object during the first attempt (Figure 5.13(a)). In this case the posterior distribution $P(O|G,T)$ has two modes, shown in Figure 5.13(b). This leads to an exploration phase, where both modes are explored. The second grasp (Figure 5.13(c)) fails, as well, as the robot decides to grasp the wrong mode. However, the measurements after the second grasp support only the correct mode (Figure 5.13(d)) and the robot successfully grasps the object on the third attempt (Figure 5.13(e)).

5.4 Summary

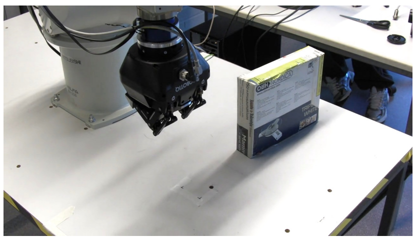
This section presented a novel framework for grasping, which operates in a probabilistic setting. The framework allows grasp planning, measurements, and corrective motions to interact, leading to a system where uncertain object attributes, such as pose, can be estimated and used as a foundation for improving grasp stability. While the proposed framework has some drawbacks – such as the assumption of a static object – the framework and the experiments show how to utilize the uncertainty in grasp planning.



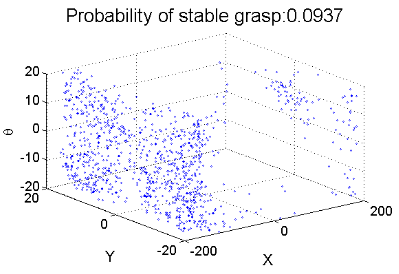
(a)



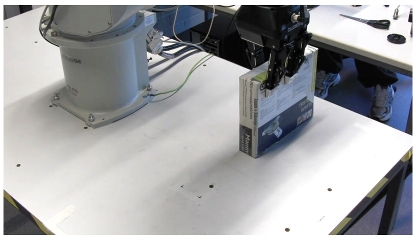
(b)



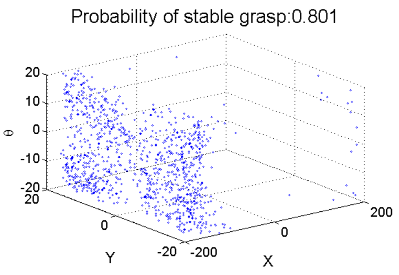
(c)



(d)



(e)



(f)

Figure 5.13: Grasp sequence with the Robotiq hand: (a) First grasp fails completely; (b) Corresponding posterior distribution $P(O|G, T)$; (c) Second grasp; (d) Posterior distribution after the second grasp; (e) Third grasp which succeeds; (f) Posterior distribution after the third grasp.

The framework can be divided into two parts: the object attribute inference and the grasp planning that takes advantage of the object attribute inference. Under the framework, object attributes can be inferred across multiple grasp attempts. The grasp planning part then finds a suitable grasp that takes into account the uncertainty still left after the inference. However, to plan for grasps, a model of the object is required. Both of the presented parts have been separately introduced in earlier works, but the framework combines these parts, producing a novel method for grasp planning under uncertainty.

The experiments were based on both simulated and data-driven versions of the fundamental models required by framework. While the experiment based on the simulated model was to test the applicability of the framework, the data-driven models can be used with real objects, even without the need for a static object. Gaussian process regression was chosen as the base for the data-driven models, as it can provide an estimate of the prediction confidence. The accuracy of the data-driven models was restricted compared to the achievable accuracy to keep the computational costs reasonable. Furthermore, the framework was demonstrated on both simulation and on real robotic platforms. The experiments showed that, using the framework, the probability to acquire a stable grasp can be increased using the information from the grasp attempts.

Chapter VI

Discussion

The presented work dealt mostly with the sense of touch in a robotic grasping context. The topics ranged from robot architectures to statistical methods. All of the proposed approaches have both challenges and future prospects.

The abstraction architecture, discussed in chapter 3, has been designed scalable in the sense of expanding the architecture to new robotic platforms and extending the functionality to new manipulation problems. Furthermore, also other possibilities exist that can be brought into the architecture. For example, one could conceive a graphical user interface for the architecture and enable easy construction of abstract actions which could be transferred across multiple platforms. One of the more interesting additions would be to integrate the methods found in chapters 4 and 5 into the architecture which would, for example, enable platform specific learning of grasp stability instead of just using heuristics to determine the stability. It is also important to note that the control architecture inside the abstraction architecture can be driven dynamically, i.e. the state machine can be built on-line according to some rule set. One example of this type of dynamic actions can be found in [145].

The abstraction architecture can also be used as a basis for further research on the use of sensors as a part of manipulation and especially of manipulation learning. The architecture can be taken to the direction of simulation, as it was integrated into the OpenRAVE environment and some steps have already been taken into this direction. For example, the WRT-102 end effector was already implemented for the OpenRAVE and can be driven through the abstraction architecture. Adding a complete simulation backend for the robot platforms would simplify the design and verification of the abstract state machines as the simulation can be driven with the same controllers as the real platform.

If one wants to consider the whole task execution then one critical piece is still missing, i.e. how to divide the task into individual actions that can be executed within the abstraction architecture. The task decomposition leans more towards the field of artifical intelligence as the task decomposition requires reasoning about the objects and actions that can be executed on them to reach the eventual goal of the whole task. Due to these reasons the task decomposition system was left out from the abstraction architecture itself. To fully utilize the abstraction architecture, however, some type of a system to split the task into suitable actions is required, either manual or autonomous.

The proposed approach to determine grasp stability from training data, described in chapter 4, has some challenges due to its data-driven nature. One of the greatest challenges is sensor hysteresis, particularly on the tactile sensors. This effect was noticed on a few of the end effectors used in the experiments and required new training grasps. One obvious solution is to improve the quality of the sensors or replace them with sensors that are based on more stable materials that can withstand, for example, variance in temperature without changing the sensor output. Another solution would be to implement some type of on-line learning of the grasp stability, but this might be infeasible due to the amount of grasps required to learn the stability adequately.

The second problem is how to choose the grasps that make up the training data. In the experiments, the training data was essentially random grasps. It would be interesting to develop a method that could optimize the grasps so that a fewer number of grasps would be made, similarly to [129]. However, without any previous knowledge of the object, as was assumed in the experiments, developing a method for minimizing the number of grasps is difficult, if not impossible. Introducing some type of geometrical model or other type of object representation ties this method with the probabilistic grasp planning method presented in chapter 5. Both of the presented methods require some kind of sampling of grasps to determine the field of grasp stability around a given object.

The work done to determine the grasp stability from sensor data raises some other, tangential, questions. For example, which locations on the end effector would be the most useful to cover with tactile sensing elements so that as many contacts as possible could be measured during grasping. Answering this question would benefit both the data-driven and the analytical grasp stability algorithms, as both require sensing of the contacts.

Also one direction to take is to add another sensor modality in addition to the tactile and proprioceptive sensing to help determine the grasp stability. One good candidate for this is vision, which allows linking the global pose of the hand to the local sensing. Without knowing the pose of the hand in relation to the object, it is difficult to determine the grasp stability with the method proposed in this thesis, as the mass distribution and the center of mass affects the torque seen by the contacts between the hand and the object. This phenomenon is especially evident with relatively large objects, such as the cardboard box used in the experiment shown in Figure 5.12. There has already been some work in this direction, such as [78], which combine visual and tactile grasp stability models.

Vision could also be used for training the system by automating the grasp synthesis and the consequent labeling of the grasp result. Again, this is very similar setup that is presented in [129], but instead of building the grasp affordance on top of visual models, the tactile and proprioceptive sensor outputs could be used to build the grasp stability model. The autonomous collecting of data would accelerate the learning of the grasp stability models of real objects as the task of manually collecting training data was quite time consuming.

Relating to the data collection, one important question was still left unanswered in this thesis: "How to use simulated grasp stability models with real objects?" While some very preliminary tests were done, the tests showed that the simulated data was not useful when classifying grasps with a real hand. The probable reason behind this is that the output of the simulated sensors did not correspond to the output of the real sensors. Using simulation would immensely speed up the generation of grasp stability models, similarly to the Columbia Grasp Database [44]. However, if the problem lies within the simulation itself, developing a more high quality simulation environment is a challenging task.

Probabilistic grasp planning, presented in chapter 5, takes some cues from grasp stability learning, but extends the problem to include also grasp planning or grasp synthesis. In many ways, this topic is the most interesting, as it brings together the estimation of grasp stability with the estimation of the object attributes. However, one significant challenge is the models required by the described framework. The data-driven models that were used in the experiments were quite inaccurate due to computational constraints and the simulation models, while accurate, cannot be efficiently used within the implemented framework due to their computational demands. Given more accurate models, one could imagine that the framework could be utilized to simultaneously infer, for example, object scale and object identity as well as the pose.

In this thesis, these models were not explored very far. In traditional grasp planning, the emphasis is to find good grasps and save these grasps for further use. However, the models used for the probabilistic grasp planning requires that the grasps represent the whole object from the point of view of the sensors embedded into the robotic hand. Compared to the static grid used in this thesis to sample the grasps on individual objects, a more adaptive sampling approach might be more useful and accurate in capturing the details of the object.

There are also other possibilities that the presented framework is capable of, but were not included in this thesis due to the time constraints on the thesis work itself. Perhaps, the most interesting of these, is the capability to determine the most stable grasp among many possible objects, all of which may have uncertain pose. The premise here is quite similar to [136], but instead of considering the object presentation and its ability to predict good grasps, all object models are considered simultaneously to provide the most stable grasp across all of the given objects. This idea can also be extended to scenarios where multiple physical objects are present but their pose and possibly their identity is uncertain.

It would also be interesting to see how the particle based representation of uncertainty used in chapter 5 could be utilized in other manipulation actions such as pushing. In this type of scenario the object would be represented in a similar fashion to a mobile robot that uses particle filter techniques to represent its own pose. However, the difference is that the robot can use inbuilt sensors to reduce the uncertainty of the robot's pose, while the object must be observed externally. While the pushing action itself could be learned, i.e. what are the effects of the push on the object given known poses of the robot hand and the object, the planning phase requires additional thought and a new approach that can plan multiple actions that move the object to the desired pose under uncertainty.

Some advances can also be made in the proposed grasp planning method. It should be possible to direct the grasp planning process with some other measure than grasp stability. One good example would be the task suitability of the grasp [131]. One can also direct the grasp planning into the direction of informative grasps. These grasps, instead of optimizing the expected grasp stability, are optimized to reduce the uncertainty in the object attributes. Some work has already been published in this area, for example [135].

In addition, it should be noted that improvements can also be made at the sensor measurement level. For example, the work presented in this thesis considered only the proprioceptive sense, i.e. the joint angles of the robot hand, but given access to tactile sensing, the accuracy of the object attribute estimation could be improved. With tactile sensing, the robot could make a distinction between, for example, edges and corners, reducing the ambiguity and the uncertainty of the estimation.

Chapter VII

Conclusion

The abstraction architecture, presented in chapter 3, proposes an approach to solve hardware independence. The abstraction architecture is based on the concepts of abstract and concrete actions. Both are represented by a finite state machine. The abstract action describes a single action, such as picking and placing an object from one location to another, in an abstract form; that is, the robot platform or embodiment has no effect on the description of the abstract action. Furthermore, the abstract action is encoded in XML, which allows both humans and computers to create and modify the abstract actions in a codified manner.

One of the most important components of the abstraction architecture is the translator. The translator translates the XML description of the abstract action into a concrete action. The translator can utilize both platform independent and platform dependent program code to form the concrete action. The concrete action is the hardware dependent version of the abstract action. The concrete action might differ from the abstract action due to deficiencies of the platform itself. However, the aim of the action remains the same across the abstract and the concrete action. The concrete action is executed by the control architecture, which includes all of the program code to run the finite state machine. One goal of the abstraction architecture was to integrate the use of sensors into the architecture. This goal is evident in the control architecture where all of the components that need access to the sensors can access all of the sensors of the platform.

The demonstrations showed that implementing the abstraction architecture across multiple platforms is feasible, and that the abstract action can be successfully translated into concrete action on these platforms even if the capabilities of the hardware are different. In addition, failure detection, which was also one of the design goals of the architecture, was demonstrated. Integrating the failure detection into the architecture allows learning from surprises and feedback to higher level systems utilizing the abstraction architecture.

Chapter 4 proposed an approach to determine the grasp stability from the feedback of the sensors that are embedded into robot hands. As the main input or features, the proposed method used both the prorioceptive and the tactile sense of the hand. From these inputs, the grasp stability can be learned through multiple example grasps that had resulted in either stable or unstable grasps. The methods used to learn the stability were machine learning algorithms called classifiers.

Several different classifiers, such as support vector machine and AdaBoost, and feature representations were experimented on to find a suitable combination for further experiments. Of the different classifiers tested, adaptive boosting and support vector machine classification were the best. Further experiments considered the effect of the object knowledge, that is, whether it helps to have more specific information about the object in terms of grasp stability. The experiments showed, as could be expected, that increasing the knowledge indeed improves the classification results.

The originally proposed method was also extended in this thesis. The original method looks at the very end of the grasp and gives an estimate of the stability for that time instant. The extended method classifies the whole grasp from the beginning until the end and can make decisions on the stability faster in the case of a stable grasp. The experiments on the extended method also showed that the classification performance was similar to the original method.

The work to determine the grasp stability was further extended in chapter 5. Instead of just estimating the stability of any grasp, the proposed method also planned for a stable grasp, given an uncertain pose of an object. A general framework for planning grasps under uncertainty was presented and consequently demonstrated to show the potential of the framework.

The demonstrations were performed in both simulation driven and data-driven environments. While the simulation driven planning was shown to be accurate, the performance of the simulation could not be scaled to the levels required to operate in a real environment. For this reason, the data-driven environment was adopted in further experiments. The data-driven models were based on Gaussian process regression. These experiments showed that the framework is capable of inferring object pose distribution, given information from grasps, and that the framework is then able to plan for stable grasps given the inferred object pose distribution.

Bibliography

[1] iRobot Corporation. iRobot, http://www.irobot.com/, [Accessed 8 December 2011].

[2] G. Robles-De-La-Torre, "The importance of the sense of touch in virtual and real environments," *Multimedia, IEEE*, vol. 13, pp. 24 –30, July-Sept. 2006.

[3] Emergence of Cognitive Grasping through Introspection, Emulation and Surprise, GRASP. http://www.csc.kth.se/grasp/, [Accessed 8 December 2011].

[4] S. Coradeschi and A. Saffiotti, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2-3, pp. 85 – 96, 2003.

[5] J. Laaksonen, J. Felip, A. Morales, and V. Kyrki, "Embodiment independent manipulation through action abstraction," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2113 –2118, May 2010.

[6] Y. Bekiroglu, J. Laaksonen, J. Jorgensen, V. Kyrki, and D. Kragic, "Assessing grasp stability based on learning and haptic data," *Robotics, IEEE Transactions on*, vol. 27, pp. 616 –629, June 2011.

[7] J. Laaksonen, V. Kyrki, and D. Kragic, "Evaluation of feature representation and machine learning methods in grasp stability learning," in *10th IEEE-RAS International Conference on Humanoid Robots*, pp. 112 –117, 2010.

[8] J. Schill, J. Laaksonen, M. Przybylski, V. Kyrki, T. Asfour, and R. Dillmann, "Learning continuous grasp stability for a humanoid robot hand based on tactile sensing," in *Biomedical robotics and Biomechatronics, 2012 IEEE International Conference on*, June 2012. To appear.

[9] J. Laaksonen and V. Kyrki, "Probabilistic approach to sensor-based grasping," in *ICRA 2011 Workshop on Manipulation Under Uncertainty*, 2011.

[10] J. Laaksonen, E. Nikandrova, and V. Kyrki. Probabilistic Sensor-based Grasping, submitted to 2012 International Conference on Intelligent Robots and Systems.

[11] Y. Bekiroglu, J. Laaksonen, J. A. Jorgensen, and V. Kyrki, "Learning grasp stability based on haptic data," in *Robotics: Science and Systems (RSS 2010) Workshop on Representations for Object Grasping and Manipulation in Single and Dual Arm Tasks*, 2010.

[12] Schunk GmbH. Schunk Dextrous Hand, http://www.schunk-modular-robotics.com/left-navigation/service-robotics/components/actuators/robotic-hands/sdh.html, [Accessed 7 November 2011].

[13] Barrett Technology Inc. BarrettHand, http://www.barrett.com/robot/products-hand.htm, [Accessed 7 November 2011].

[14] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An integrated humanoid platform for sensory-motor control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pp. 169–175, 2006.

[15] C. Chunsheng and B. Roth, "On the spatial motion of a rigid body with point contact," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, pp. 686 – 695, Mar 1987.

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation.* Boca Raton, Florida, USA: CRC Press, 1st edition ed., 1994.

[17] J. K. Salisbury, *Kinematics and Force Analysis of Articulated Hands.* PhD thesis, Stanford University, 1982.

[18] M. Ciocarlie, A. Miller, and P. Allen, "Grasp analysis using deformable fingers," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 4122 – 4128, Aug. 2005.

[19] D. Marhefka and D. Orin, "A compliant contact model with nonlinear damping for simulation of robotic systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 29, pp. 566 –572, Nov 1999.

[20] F. Reuleaux, *The Kinematics of Machinery: Outlines of a Theory of Machines.* London:Macmillan, second ed., 1876. Reprinted as The Kinematics of Machinery. New York: Dover, 1963.

[21] W. Howard and V. Kumar, "Modeling and analysis of the compliance and stability of enveloping grasps," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2, pp. 1367 –1372 vol.2, May 1995.

[22] J. Czyzowicz, I. Stojmenovic, and J. Urrutia, "Immobilizing a polytope," in *Algorithms and Data Structures* (F. Dehne, J.-R. Sack, and N. Santoro, eds.), vol. 519 of *Lecture Notes in Computer Science*, pp. 214–227, Springer Berlin / Heidelberg, 1991. 10.1007/BFb0028264.

[23] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, pp. 348 –353 vol.1, 2000.

[24] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), Berlin, Germany: Springer-Verlag, first ed., 2008.

[25] I. Kao, K. Lynch, and J. W. Burdick, "Contact modeling and manipulation," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), Berlin, Germany: Springer-Verlag, first ed., 2008.

[26] D. Kirkpatrick, B. Mishra, and C.-K. Yap, "Quantitative steinitz's theorems with applications to multifingered grasping," *Discrete Comput. Geom.*, vol. 7, pp. 295–318, March 1992.

[27] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pp. 2290 –2295 vol.3, May 1992.

[28] R. Truax, R. Platt, and J. Leonard, "Using prioritized relaxations to locate objects in points clouds for manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2091 –2097, May 2011.

[29] W. Wohlkinger and M. Vincze, "Shape-based depth image to 3d model matching and classification with inter-view similarity," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 4865 –4870, Sept. 2011.

[30] C. Papazov and D. Burschka, "An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes," in *Proceedings of the 10th Asian Conference on Computer Vision (ACCV'10)*, November 2010.

[31] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robot. Automat. Mag.*, vol. 11, pp. 110–122, Dec. 2004.

[32] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann, "Opengrasp: A toolkit for robot grasping simulation," in *Simulation, Modeling, and Programming for Autonomous Robots* (N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. von Stryk, eds.), vol. 6472 of *Lecture Notes in Computer Science*, pp. 109–120, Springer Berlin / Heidelberg, 2010.

[33] R. Diankov, *Automated Construction of Robotic Manipulation Programs.* PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[34] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1628 –1633, May 2008.

[35] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning Via Decomposition Trees," in *IEEE International Conference on Robotics and Automation*, pp. 4679–4684, 2007.

[36] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, pp. 1824 – 1829 vol.2, sept. 2003.

[37] M. Przybylski, T. Asfour, and R. Dillmann, "Planning grasps for robotic hands using a novel object representation based on the medial axis transform," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1781 –1788, sept. 2011.

[38] R. Balasubramanian, L. Xu, P. Brook, J. Smith, and Y. Matsuoka, “Human-guided grasp measures improve grasp robustness on physical robot,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2294 –2301, May 2010.

[39] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[40] Q. Le, D. Kamm, A. Kara, and A. Ng, “Learning to grasp objects with multiple contact points,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5062 –5069, May 2010.

[41] Y. Jiang, S. Moseson, and A. Saxena, “Efficient grasping from rgbd images: Learning using a new rectangle representation,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3304 –3311, May 2011.

[42] D. Kraft, R. Detry, N. Pugeault, E. Başeski, F. Guerin, J. Piater, and N. Krüger, “Development of object and grasping knowledge by robot exploration,” *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 368–383, 2010.

[43] C. Papazov and D. Burschka, “Deformable 3D Shape Registration Based on Local Similarity Transforms,” in *Proceedings of the 9th Eurographics Symposium on Geometry Processing (SGP’11)*, July 2011.

[44] C. Goldfeder, M. Ciocarlie, H. Dang, and P. Allen, “The columbia grasp database,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, pp. 1710 –1716, May 2009.

[45] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. Allen, “Data-driven grasping with partial sensor data,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1278 –1283, oct. 2009.

[46] M. Ciocarlie and P. Allen, “Data-driven optimization for underactuated robotic hands,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1292 –1299, May 2010.

[47] A. M. Dollar and R. D. Howe, “The highly adaptive SDM hand: Design and performance evaluation,” *I. J. Robotic Res.*, vol. 29, no. 5, pp. 585–597, 2010.

[48] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. Zakin, H. Lipson, and H. Jaeger, “Universal robotic gripper based on the jamming of granular material.,” in *Proceedings of the National Academy of Sciences (PNAS)*, vol. 107, 2010.

[49] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” *SIGGRAPH Comput. Graph.*, vol. 26, pp. 71–78, July 1992.

[50] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[51] J. Bohg, M. Johnson-Roberson, M. Björkman, and D. Kragic, "Strategies for multi-modal scene exploration.," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2010.

[52] J. Bohg, M. Johnson-Roberson, B. Leon, J. Felip, X. Gratal, N. Bergstrom, D. Kragic, and A. Morales, "Mind the gap - robotic grasping under incomplete observation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 686 –693, May 2011.

[53] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schröder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 54 – 65, 2008.

[54] L. Chang, S. Srinivasa, and N. Pollard, "Planning pre-grasp manipulation for transport tasks," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2697 –2704, May 2010.

[55] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 473 –479 vol.1, 1999.

[56] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 566 –580, Aug 1996.

[57] N. Vahrenkamp, P. Kaiser, T. Asfour, and R. Dillmann, "RDT+: A parameter-free algorithm for exact motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 715 –722, May 2011.

[58] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based RRT for path planning in continuous cost spaces," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2145 –2150, Sept. 2008.

[59] D. Berenson, T. Simeon, and S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4561 –4568, May 2011.

[60] P. Allen, A. Miller, P. Oh, and B. Leibowitz, "Integration of vision and force sensors for grasping," in *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, pp. 349 –356, Dec 1996.

[61] P. K. Allen, A. T. Miller, P. Y. Oh, and B. S. Leibowitz, "Integration of vision, force and tactile sensing for grasping," *Int. J. Intelligent Machines*, vol. 4, pp. 129–149, 1999.

[62] gpgpu.org. General-Purpose Computation on Graphics Hardware (GPGPU), http://gpgpu.org/, [Accessed 23 November 2011].

[63] Tekscan®. FlexiForce®, http://www.tekscan.com/grip-force-measurement, [Accessed 23 November 2011].

[64] Weiss Robotics. WRT 101 - A Sense of Touch for Service Robots, http://www.weiss-robotics.de/en/component/content/article/76-sensorsystem-wrt-101.html, [Accessed 23 November 2011].

[65] Pressure Profile Systems, Inc. Robotouch - overview, http://www.pressureprofile.com/products-robotouch, [Accessed 30 January 2012].

[66] K. Weiss and H. Worn, "The working principle of resistive tactile sensor cells," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 1, pp. 471 – 476, July-1 Aug. 2005.

[67] A. Schmitz, P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta, "Methods and technologies for the implementation of large-scale robot tactile sensors," *Robotics, IEEE Transactions on*, vol. 27, pp. 389 –400, June 2011.

[68] L. Seminara, M. Capurro, P. Cirillo, G. Cannata, and M. Valle, "Electromechanical characterization of piezoelectric pvdf polymer films for tactile sensors in robotics applications," *Sensors and Actuators A: Physical*, vol. 169, no. 1, pp. 49 – 58, 2011.

[69] M. Ohka, N. Morisawa, H. Suzuki, J. Takata, H. Koboyashi, and H. Yussof, "A robotic finger equipped with an optical three-axis tactile sensor," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3425 –3430, May 2008.

[70] H. Yousef, M. Boukallel, and K. Althoefer, "Tactile sensing for dexterous in-hand manipulation in robotics – a review," *Sensors and Actuators A: Physical*, vol. 167, no. 2, pp. 171 – 187, 2011.

[71] M. Prats, *Robot Physical Interaction through the combination of Vision, Tactile and Force Feedback.* PhD thesis, Universitat Jaume I, 2009.

[72] O. Alkkiomäki, *Sensor fusion of proprioception, force and vision in estimation and robot control.* PhD thesis, Lappeenranta University of Technology, 2010.

[73] P. Azad, D. Munch, T. Asfour, and R. Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5204 –5209, May 2011.

[74] L. E. Zhang and J. C. Trinkle, "Exploring the application of particle filters to grasping acquisition with visual and tactile occlusion," tech. rep., Department of Computer Science, Rensselaer Polytechnic Institute, 2012. Submitted and also presented at ICRA 2011 Shanghai workshop on Uncertainty in Automation.

[75] J. Sorribes, M. Prats, and A. Morales, "Visual tracking of a jaw gripper based on articulated 3d models for grasping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2302 –2307, May 2010.

[76] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Markerless and Efficient 26-DOF Hand Pose Recovery," in *Proceedings of the 10th Asian Conference on Computer Vision, ACCV'2010, Part III, LNCS 6494, Queenstown, New Zealand*, pp. 744 –757, November 2010.

[77] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraint," in *Proceedings of the IEEE International Conference on Computer Vision, ICCV'2011, Barcelona, Spain*, November 2011.

[78] Y. Bekiroglu, R. Detry, and D. Kragic, "Learning tactile characterizations of object- and pose-specific grasps," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1554 –1560, Sept. 2011.

[79] G. Milighetti, H. Kuntze, C. Frey, B. Diestel-Feddersen, and J. Balzer, "On a primitive skill-based supervisory robot control architecture," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pp. 141 –147, July 2005.

[80] S. Haidacher, J. Butterfass, M. Fischer, M. Grebenstein, K. Joehl, K. Kunze, M. Nickl, N. Seitz, and G. Hirzinger, "Dlr hand ii: hard- and software architecture for information processing," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 684 – 689 vol.1, Sept. 2003.

[81] L. Han, Z. Li, J. Trinkle, Z. Qin, and S. Jiang, "The planning and control of robot dextrous manipulation," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, pp. 263 –269 vol.1, 2000.

[82] L. Petersson, M. Egerstedt, and H. Christensen, "A hybrid control architecture for mobile manipulation," in *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3, pp. 1285 –1291 vol.3, 1999.

[83] T. Schlegl and M. Buss, "A discrete-continuous control architecture for dextrous manipulation," in *Proc. IEEE SMC'99*, vol. 2, pp. 860–865, 1999.

[84] C.-F. Chang and L.-C. Fu, "A hybrid system design of a mobile manipulator," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 406 –411, May 2006.

[85] S. Aramaki, H. Shirouzu, and K. Kurashige, "Control program structure of humanoid robot," in *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, vol. 3, pp. 1796 – 1800 vol.3, Nov. 2002.

[86] M. Prats, A. del Pobil, and P. Sanz, "A control architecture for compliant execution of manipulation tasks," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, p. 11, Oct. 2006.

[87] ROS.org. ROS wiki, http://www.ros.org/wiki/, [Accessed 1 February 2012].

[88] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robot. Auton. Syst.*, vol. 56, pp. 29–45, January 2008.

[89] D. Kortenkamp and R. Simmons, "Robotic systems architecture and programming," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), Berlin, Germany: Springer-Verlag, first ed., 2008.

[90] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, pp. 14 – 23, Mar 1986.

[91] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, second ed., 2000.

[92] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, July 2008.

[93] J. Felip and A. Morales, "Robust sensor-based grasp primitive for a three-finger robot hand," in *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Oct. 2009.

[94] J. Felip, J. Bernabe, and A. Morales, "Emptying the box using blind haptic manipulation primitives," in *ICRA 2011 Workshop on Manipulation Under Uncertainty*, 2011.

[95] T. Takahashi, T. Tsuboi, T. Kishida, Y. Kawanami, S. Shimizu, M. Iribe, T. Fukushima, and M. Fujita, "Adaptive grasping by multi fingered hand with tactile sensor based on robust force and position control," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 264 –271, May 2008.

[96] A. Jiménez, A. Soembagijo, D. Reynaerts, H. V. Brussel, R. Ceres, and J. Pons, "Featureless classification of tactile contacts in a gripper using neural networks," *Sensors and Actuators A: Physical*, vol. 62, no. 1-3, pp. 488–491, 1997.

[97] A. Petrovskaya, O. Khatib, S. Thrun, and A. Ng, "Bayesian estimation for autonomous object manipulation based on tactile sensors," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 707 –714, May 2006.

[98] A. Petrovskaya, S. Thrun, D. Koller, and O. Khatib, "Guaranteed inference for global state estimation in human environments," in *RSS 2010 Mobile Manipulation Workshop*, 2010.

[99] A. Petrovskaya and O. Khatib, "Global localization of objects via touch," *Robotics, IEEE Transactions on*, vol. 27, pp. 569 –585, June 2011.

[100] A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard, "Object identification with tactile sensors using bag-of-features," in *In Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2009.

[101] M. Schöpfer, M. Pardowitz, and H. J. Ritter, "Using entropy for dimension reduction of tactile data," in *14th International Conference on Advanced Robotics*, Proceedings of the ICAR 2009, (Munich, Germany), IEEE, 22/06/2009 2009.

[102] S. Chitta, M. Piccoli, and J. Sturm, "Tactile object class and internal state recognition for mobile manipulation," in *IEEE International Conference on Robotics and Automation*, pp. 2342–2348, 2010.

[103] N. Gorges, S. E. Navarro, D. Göger, and H. Wörn, "Haptic object recognition using passive joints and haptic key features," in *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[104] P. Allen and K. Roberts, "Haptic object recognition using a multi-fingered dextrous hand," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 342 –347 vol.1, May 1989.

[105] S. Caselli, C. Magnanini, and F. Zanichelli, "Haptic object recognition with a dextrous hand based on volumetric shape representations," in *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI '94.*, pp. 280 –287, oct 1994.

[106] H. Dang, J. Weisz, and P. K. Allen, "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5917 –5922, May 2011.

[107] R. Pelossof, A. Miller, P. Allen, and T. Jebara, "An SVM learning approach to robotic grasping," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, pp. 3512 – 3518 Vol.4, 26-May 1, 2004.

[108] A. Rodriguez, M. Mason, and S. Srinivasa, "Manipulation capabilities with simple hands," in *12th International Symposium on Experimental Robotics*, 2010.

[109] Y. Bekiroglu, D. Kragic, and V. Kyrki, "Learning grasp stability based on tactile data and HMMs," in *RO-MAN, 2010 IEEE*, pp. 132 –137, Sept. 2010.

[110] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC, 2006.

[111] P. McCullagh and J. A. Nelder, *Generalized Linear Models.* Chapman and Hall, second ed., 1989.

[112] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in Neural Processing Systems (NIPS*2001)*, 2001.

[113] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[114] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[115] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, pp. 61–74, MIT Press, 1999.

[116] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[117] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth Internation Conference on Machine Learning*, pp. 148–156, Morgan Kaufmann, 1996.

[118] A. Vezhnevets and V. Vezhnevets, “Modest adaboost – teaching adaboost to generalize better,” in *Graphicon*, pp. 322–325, 2005.

[119] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004. 10.1023/B:VISI.0000013087.49260.fb.

[120] A. Vezhnevets, *GML AdaBoost Matlab Toolbox*, 2006. Available at `http://graphics.cs.msu.ru/ru/science/research/machinelearning/adabooosttoolbox`.

[121] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[122] P. Paalanen and J.-K. Kämäräinen, *GMMBAYES - Bayesian Classifier and Gaussian Mixture Model ToolBox*, 2004. Available at `http://www2.it.lut.fi/project/gmmbayes/downloads/src/gmmbayestb/`.

[123] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[124] Weiss Robotics, “Tactile sensor module, type: DSA 9335.” http://www.weiss-robotics.de/en/products/tactile-sensing/tactile-transducers/87-dsa-933x.html, [Accessed 19 March 2012].

[125] J. A. Jorgensen and H. G. Petersen, “Usage of simulations to plan stable grasping of unknown objects with a 3-fingered schunk hand,” in *IROS’08 Workshop on Robot Simulators*, (Nice, France), September 2008.

[126] A. Miller and P. Allen, “Examples of 3d grasp quality computations,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1240 –1246, 1999.

[127] C. Goldfeder and P. Allen, “Data-driven grasping,” *Autonomous Robots*, vol. 31, pp. 1–20, 2011. 10.1007/s10514-011-9228-1.

[128] A. Jain and C. C. Kemp, “El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces,” *Auton. Robots*, vol. 28, pp. 45–64, January 2010.

[129] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater, “Learning grasp affordance densities,” *Journal of Behavioral Robotics*, vol. 2, pp. 1–17, 2011.

[130] C. Barck-Holst, M. Ralph, F. Holmar, and D. Kragic, “Learning grasping affordance using probabilistic and ontological approaches,” in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1 –6, June 2009.

[131] D. Song, K. Huebner, V. Kyrki, and D. Kragic, “Learning task constraints for robot grasping using graphical models,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1579 –1585, Oct. 2010.

[132] R. Platt, “Learning grasp strategies composed of contact relative motions,” in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pp. 49 –56, Dec. 2007.

[133] K. Goldberg and M. Mason, “Bayesian grasping,” in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 1264 –1269 vol.2, May 1990.

[134] V. Christopoulos and P. Schrater, “Handling shape and contact location uncertainty in grasping two-dimensional planar objects,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1557 –1563, Nov. 2007.

[135] K. Hsiao, L. Kaelbling, and T. Lozano-Perez, “Task-driven tactile exploration,” in *Proceedings of Robotics: Science and Systems*, (Zaragoza, Spain), June 2010.

[136] P. Brook, M. Ciocarlie, and K. Hsiao, “Collaborative grasp planning with multiple object representations,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2851 –2858, may 2011.

[137] K. Hsiao, M. Ciocarlie, and P. Brook, “Bayesian grasp planning,” in *ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, 2011.

[138] C. Corcoran and R. Platt, “A measurement model for tracking hand-object state during dexterous manipulation,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4302 –4308, May 2010.

[139] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942 –1948, Nov/Dec 1995.

[140] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[141] C. K. I. Williams, “Computation with infinite neural networks,” *Neural Comput.*, vol. 10, pp. 1203–1216, July 1998.

[142] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* the MIT Press, 2006.

[143] C. E. Rasmussen and H. Nickisch, “Gaussian processes for machine learning (GPML) toolbox,” *Journal of Machine Learning*, vol. 11, pp. 3011–3015, 2002.

[144] E. Snelson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” in *Advances in Neural Processing Systems (NIPS*2006)*, 2006.

[145] K. Hsiao, L. Kaelbling, and T. Lozano-Perez, “Grasping pomdps,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4685 –4692, April 2007.

454. HEINIMÖ, JUSSI. Developing markets of energy biomass – local and global perspectives. 2011. Diss.

455. HUJALA, MAIJA. Structural dynamics in global pulp and paper industry. 2011. Diss.

456. KARVONEN, MATTI. Convergence in industry evolution. 2011. Diss.

457. KINNUNEN, TEEMU .Bag-of-features approach to unsupervised visual object categorisation. 2011. Diss.

458. RUUSKANEN, VESA. Design aspects of megawatt-range direct-driven permanent magnet wind generators. 2011. Diss.

459. WINTER, SUSANNA. Network effects: scale development and implications for new product performance. 2011. Diss.

460. JÄÄSKELÄINEN, ANSSI. Integrating user experience into early phases of software development. 2011. Diss.

461. KÄÄRIÄINEN, TOMMI. Polymer surface modification by atomic layer deposition. 2011. Diss.

462. KOCHURA, ALEKSEY. Growth, magnetic and transport properties of InSb and II-IV-As2 semiconductors doped with manganese. 2011. Diss.

463. PUTKIRANTA, ANTERO. Possibilities and challenges of longitudinal studies in operations management. 2011. Diss.

464. HAPPONEN, ARI. Muuttuvaan kysyntään sopeutuva varastonohjausmalli. 2011. Diss.

465. VASAVA, PARITOSH. Application of computational fluid dynamics in modelling blood flow in human thoracic aorta. 2011. Diss.

466. PURO, LIISA. Identification of extractives and polysaccharides as foulants in membrane filtration of pulp and paper mill effluents. 2011. Diss.

467. LAPPALAINEN, PIA. Socially Competent Leadership – predictors, impacts and skilling in engineering. 2012. Diss.

468. PLAMTHOTTATHIL, ANSHY OONNITTAN. Application of electrokinetic Fenton process for the remediation of soil contaminated with HCB. 2012. Diss.

469. EBRAHIMI, FATEMEH. Synthesis of percarboxylic acids in microreactor. 2012. Diss.

470. JANTUNEN, SAMI. Making sense of software product requirements. 2012. Diss.

471. VILKO, JYRI. Approaches to supply chain risk management: identification, analysis and control. 2012. Diss.

472. TANSKANEN, VESA. CDF modelling of direct contact condensation in suppression pools by applying condensation models of separated flow. 2012. Diss.

473. HUHTANEN MIKKO. Software for design of experiments and response modelling of cake filtration applications. 2012. Diss.

474. PARJANEN, SATU. CREATING POSSIBILITIES FOR COLLECTIVE CREATIVITY
Brokerage functions in practice-based innovation. 2012. Diss.

475. KUKKONEN, SAKU. Generalized differential evolution for global multi-objective optimization with constraints. 2012. Diss.