Visuo-Haptic Grasping of Unknown Objects through Exploration and Learning on Humanoid Robots

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Simon Ottenhaus

Tag der mündlichen Prüfung:	9.12.2019
Referent:	Prof. Dr.-Ing. Tamim Asfour
Korreferent:	Prof. Dr. Gordon Cheng

Zusammenfassung

Die humanoide Robotik ist ein Teilgebiet der Robotik mit der Zielsetzung Roboter zu schaffen, die Menschen sowohl im Aussehen, als auch in den Fähigkeiten ähnlich sind. Das Greifen ist dabei eine zentrale Fähigkeit humanoider Roboter, da es eine Voraussetzung der Objektmanipulation ist. Studien in den Neurowissenschaften zeigen, dass taktiles Feedback für Menschen zentral ist, um Objekte erfolgreich zu greifen. Tatsächlich ist ein großer Bereich des Gehirns für die Verarbeitung von Sinneswahrnehmungen der Hand zuständig. Im Kontext der Robotik wird das Greifen vorwiegend basierend auf visueller Information realisiert. Jedoch kann visuelle Wahrnehmung nur eine Teilansicht des Objekts liefern und wird zusätzlich durch Dunkelheit oder reflektierende Oberflächen beeinträchtigt. Das Greifen von unbekannten Objekte stellt aufgrund von fehlendem a priori Wissen nach wie vor ein ungelöstes Problem in der Robotik dar, welches bei humanoiden Robotern mit einer großen Anzahl an Bewegungsfreiheitsgraden besonders ausgeprägt ist. Daher wird in dieser Arbeit das Ziel verfolgt, ergänzende Objektinformation durch eine haptische Exploration zu gewinnen, um z.B. unbekannte Objekte zu greifen. Hierzu werden haptische und visuelle Information zu einem Objektmodell fusioniert, auf dessen Basis Griffe generiert werden können. Zentrale Herausforderungen hierbei ergeben sich aus Sensorrauschen, ungenauer Objektmodellschätzung bzw. einer unpräzisen Kalibrierung und Griffausführung.

Ziel dieser Arbeit ist es, einen Beitrag zum Greifen unbekannter Objekte durch humanoide Roboter zu leisten. Dazu werden visuelle Informationen mit haptischer Exploration kombiniert, um Greifhypothesen zu erzeugen. Basierend auf simulierten Trainingsdaten wird außerdem eine Greifmetrik gelernt, welche die Erfolgswahrscheinlichkeit der Greifhypothesen bewertet und die mit der größten geschätzten Erfolgswahrscheinlichkeit auswählt. Diese wird verwendet, um Objekte mit Hilfe einer reaktiven Kontrollstrategie zu greifen. Die zwei Kernbeiträge der Arbeit sind zum einen die haptische Exploration von unbekannten Objekten und zum anderen das Greifen von unbekannten Objekten mit Hilfe einer neuartigen datengetriebenen Greifmetrik.

Haptische Exploration für das Greifen unbekannter Objekte

Inspiriert von menschlichen Greif- und Explorationsstrategien werden unbekannte Objekte vom Roboter abgetastet und die gesammelten Kontaktpunkte in einem geschätzten Oberflächenmodell zusammengefasst. In dieser Arbeit wird *Gaussian Process Implicit Surfaces (GPIS)* als dateneffizientes Modell zur Oberflächenschätzung eingesetzt. Zusätzlich wird *GPIS* um Oberflächennormalen ergänzt, um den Informationsgehalt pro Kontaktpunkt zu erhöhen. Um diese Information zu erfassen wurde ein neuer taktiler Sensor entwickelt, bestehend aus einer Trägheitsmesseinheit *(inertial measurement unit, IMU)* und einem taktilen Drucksensor. Der Sensor misst die lokale Oberflächenorientierung mit der *IMU* und detektiert Kontakt mit dem Drucksensor. Durch das Hinzufügen der Oberflächennormalen konnte der Oberflächenschätzungsfehler im Mittel um $50\,\%$ reduziert werden. Das geschätzte Oberflächenmodell bildet die Grundlage für die Generierung von Griffen mit Hilfe eines konventionellen Greifplaners. Dieser Greifplaner optimiert eine gegebene Greifmetrik auf dem geschätzten Oberflächenmodell des Objekts und synthetisiert daraus Greifhypothesen.

Datengetriebenes Greifen unbekannter Objekte durch Ergänzung visueller Information

Im zweiten Teil der Arbeit wird die haptische Exploration durch visuelle Tiefenbilder erweitert und mit *GPIS* in einem geschätzten Oberflächenmodell fusioniert, auf dessen Basis Greifhypothesen durch einen Greifplaner synthetisiert werden. Dazu wird ein unbekanntes Objekt wird von einer aktiven Kamera in Form einer Punktwolke aufgenommen. Die haptische Exploration ergänzt die visuelle Information durch Kontaktpunkte auf der Rückseite des Objekts. Basierend auf den visuellen und haptischen Daten werden Greifhypothesen synthetisiert. Die Greifhypothese mit der höchsten Erfolgswahrscheinlichkeit wird mit einer neu entwickelten datengetriebenen Greifmetrik prädiziert. Hierfür wird in einer Simulationsumgebung mit virtuellen Kameras und simulierter haptischer Exploration ein neuronales Netz trainiert. Für den Transfer auf den humanoiden Roboter ARMAR-6 wurde eine neuartige Greifstrategie entwickelt, die für die unteraktuierten Hände des Roboters optimiert ist. Schließlich greift der Roboter verschiedene unbekannte Objekte durch die Kombination der Greifmetrik und der Greifstrategie.

Acknowledgement

This thesis is the result of my reseach work at the High Performance Humanoid Technologies Lab (H^2T) of the Institute for Anthropomatics and Robotics (IAR), Karlsruhe Institute of Technology (KIT).

First of all, I would like to thank my doctoral supervisor Prof. Tamim Asfour for the confidence placed in me and the opportunity to work on this important and exciting topic. Thanks to his great commitment to robotics, he has succeeded in creating excellent conditions for research in this field.

I would also like to thank the great team at H^2T for the scientific exchange, discussions and mutual support. I would like to thank my office mate Felix Hundhausen for the harmonious office atmosphere and the regular care of our „little“ office tree. I want to thank Niko Vahrenkamp and Mirko Wächter, who have always supported me in my entry into robotics and were always available for questions. Furthermore, I would like to thank Lukas Kaul for his great moral support in challenging phases of my doctoral studies. I would also like to thank David Schiebener, Fabian Paus and Markus Grotz for the many conversations in which we were able to reflect on scientific approaches. I would also like to thank all my colleagues with whom I went climbing and bouldering together. The evenings spent together in the climbing hall were a welcome change.

During my time in the Humanoids Group I had the pleasure to supervise several student theses. In particular, I would like to thank Martin Miller and Daniel Renninghoff for their work, leading to valuable contributions to my PhD.

Above all I want to most warmly thank my parents Marianne and Dietrich for their unconditional support and my girlfriend Uta for her patience, motivation and support, especially in the stressful phases.

Karlsruhe, Dezember 2019 *Simon Ottenhaus*

Contents

1 Introduction

Humanoid robotics is a field of robotics that strives to create robots that have a human-like appearance and human-like capabilities. The field has already made great progress in numerous areas, but many basic capabilities are still considered unsolved. One of these capabilities is grasping objects that the robot has never seen before.

Grasping is the basis of many other, more complex activities, especially object manipulation e.g. the transport of objects. Grasping unknown objects is a difficult task for most robots, since most objects are not optimized for robots, but for humans. For example, most robots need accurate environment and object models to be able to grasp any object. In practice, creating such models is complex and requires expert knowledge. In order for humanoid robots to become suitable for everyday use, they must be able to act independently in unknown environments and grasp unknown objects.

1.1 Motivation and Problem Statement

Grasping is a central skill for humanoid robots, as it is a prerequisite for object manipulation. Studies by Johansson and Flanagan (2009) in neuroscience show that tactile feedback is essential for human grasping. In the context of robotics, grasping is primarily approached using visual information. However, visual perception can only yield a partial view of the object and can be impaired by darkness or reflections. Grasping unknown objects poses an unsolved problem in robotics, due to the lack of prior knowledge. A central challenge is the generation of robust grasp candidates based on partial information about the object.

The lack of prior knowledge can be addressed in several ways. In this work, two approaches are pursued:

- *Acquiring additional information about the object through exploration*: The robot haptically explores the object before trying to grasp it. From the collected

contact information a surface model is estimated, which is used for grasp planning.

- *Building prior knowledge through learning*: Prior to grasping the object, the robot simulates grasping many different objects and collects the results in form of grasp successes and failures. These results are then combined in a unified data-driven model through learning. When the robot then has to grasp an unknown object, it uses this learned model to fall back on previous experience and selects a grasp candidate, that has a high predicted success rate.

This thesis contributes to grasping of unknown objects by humanoid robots by implementing strategies for the two aforementioned approaches: *Haptic exploration for grasping* and *grasping unknown objects through learning*.

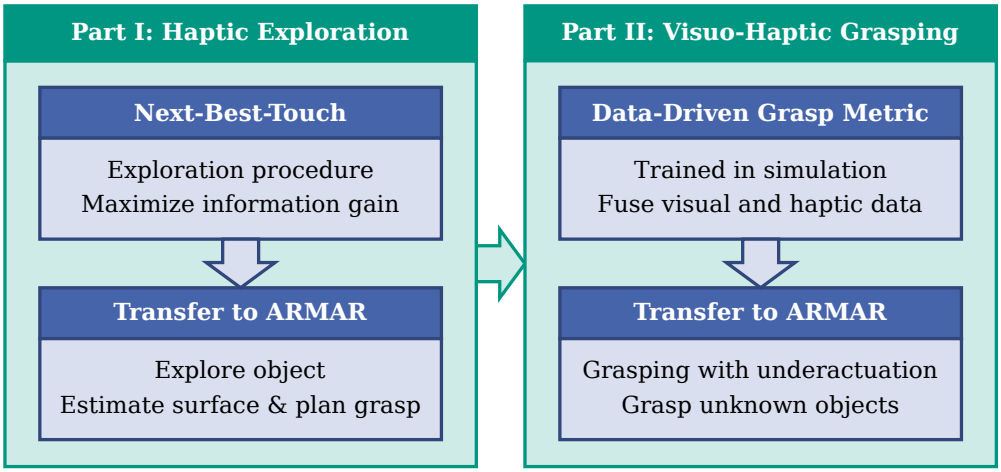


Figure 1.1: Contributions of this thesis, structured into two parts.

1.2 Contributions

This thesis presents a novel approach for humanoid robots to grasp unknown objects through exploration and learning. The contributions are structured into two parts, as shown in Figure 1.1. The first contribution , and can be summarized as follows:

Haptic Exploration for Grasping: Next-Best-Touch

Inspired by human grasping and exploration strategies a novel method is presented that allows the robot to find the next-best-touch while performing a tactile scan of an unknown object. The presented method extends existing state

of the art methods in three ways. Firstly, the exploration focuses not only on the pure information gain per exploration action, but also takes the expected costs into account. The developed method maximizes the information gain per cost. Second, *Gaussian Process Implicit Surfaces (GPIS)* is used as a data-efficient surface modeling technique. The standard formulation of GPIS covers only the position of contact points, but not the local surface orientation at the contact points. In the context of this work, GPIS is extended by normal information to take into account the orientation of the surface. Third, a tactile sensor is introduced that allows the robot to directly measure the local surface orientation. The sensor combines an inertial measurement unit *(IMU)* and a tactile pressure sensor to estimate the local surface orientation and to detect contacts. The combination of the next-best-touch method with the extended formulation of GPIS and the tactile sensor allows the robot to explore unknown objects and to approximate the surface of the object with a mesh model. This mesh model enables a conventional grasp planner to synthesize grasp candidates by optimizing a given grasp metric using the estimated object model and the hand model. The evaluation of the developed next-best-touch exploration method shows that the consideration of path costs for each exploration action leads to a reduction of the total path costs. The next-best-touch algorithm is evaluated using over 100 object models originating from two object model sets. A second evaluation examines the addition of contact normals. It is shown that these additional contact normals can significantly reduce the surface estimation error both in simulation and in experiments with the humanoid robots ARMAR-III and ARMAR-6. The developed methods and evaluation results were published in Ottenhaus et al. (2018a) and Ottenhaus et al. (2018b).

Data-Driven Grasping of Unknown Objects through added Visual Information

In the second part of this thesis, a method is presented that enables the humanoid robot ARMAR-6 to grasp unknown objects. A complete grasping pipeline was developed that combines visual perception with tactile exploration. Exploration follows the next-best-touch method, while a depth camera in the head of the robot performs visual perception. The visual and tactile information is fused into a surface estimate using *GPIS*. Two conventional grasp planners use the surface estimation to synthesize possible grasp candidates. The first grasp planner uses topological information of the object in form of a mean curvature skeleton while the second grasp planner generates uniformly distributed

grasps through random sampling. The grasp candidate with the highest success rate is predicted using a newly developed data-driven grasp metric. To this end, a neural network is trained in a simulated environment using virtual cameras and simulated haptic exploration. This learned grasp metric enables the robot ARMAR-6 to select a suitable grasp candidate for grasp execution. Finally, the robot approaches the best-rated candidate with its under-actuated hand. The closing of the fingers is coordinated with the wrist rotation and interactions with the environment are exploited to shape the under-actuated fingers around the object. During the evaluation of the grasping pipeline in simulation the data-driven grasp metric is compared to a baseline that uses conventional grasp metrics. The new data-driven approach generates grasp candidates with a higher grasp success rate than the conventional approach. The data-driven grasp metric is also validated on the humanoid robot ARMAR-6 by first generating grasp candidates for various unknown objects. Thereafter, the objects are grasped using the best-rated grasp candidate. The grasp pipeline, the evaluation and the validation were published in Ottenhaus et al. (2019).

1.3 Outline

The remainder of this thesis is structured as follows: chapter 2 gives an overview on prior publications related to the two contributions of this thesis, namely haptic exploration for grasping and data-driven grasping of unknown objects. In chapter 3 the developed next-best-touch exploration algorithm is described. The algorithm maximizes the information gain while taking the cost of exploration actions into account. The acquired contact points are used to estimate a surface model of the object, which forms the basis of subsequent grasp planning. In chapter 4 the next-best-touch approach is extended by an initial visual view. Grasp planners generate grasp candidates using a surface estimate based on the fused visual and tactile information. A data-driven grasp metric is then developed to determine the grasp candidate with the highest success rate. A grasping pipeline combines the contributions of this thesis, namely the initial visual view, the next-best-touch exploration, surface estimation, grasp candidate generation, candidate selection with the data-driven grasp metric and grasp execution. This grasping pipeline is first evaluated in simulation and then transferred to the humanoid robot ARMAR-6 to enable the grasping of unknown objects. Finally, chapter 5 concludes the thesis with a summary and discusses possible future work.

2 State of the Art

The goal of this thesis is to enable humanoid robots to grasp previously unseen objects through the combination of haptic exploration, visual perception and learning. This chapter gives an overview over the wide field of research that is humanoid robotic grasping. The overview is structured into two parts, aligned with the two main contributions of this thesis, i.e. haptic exploration for grasping and grasping unknown objects through learning.

In section 2.1, works from the scientific literature on haptic exploration are presented that address the problem of choosing exploration targets efficiently, estimating the surface of the object and synthesizing grasp candidates based on the explored data. These methods are based on either tactile exploration alone or the combination of visual and tactile perception. Grasp synthesis is mostly performed using conventional grasp planners that work on the estimated object surface. Here, the term conventional grasp planners refers to grasp planning algorithms that optimize a grasp with respect to a given grasp metric.

In contrast to exploration-based approaches, section 2.2 summarizes data-driven approaches that work on visual input. The focus here lies on acquiring inherent prior knowledge during a learning phase that can be applied to unknown objects during execution.

2.1 Haptic Exploration for Grasping

The goal of haptic exploration is to explore the surface of an unknown object, to collect a set of contact points. These contact points are then used to estimate a surface of the object. To this end, the robot's hand is equipped with tactile sensors. The robot moves its hand to establish contact between the hand and the object. The tactile sensors are used to determine the location of the contact. The robot then uses its kinematic model to calculate the position of the hand in 3D space and in turn to calculate the position of the tactile sensor at the time of contact. After each contact, the robot creates an estimation of the object and

selects a new goal for the next exploration target. The hand is then moved to the next selected exploration target, until contact occurs. Through this exploration process, the robot touches the object from different sides. All collected contacts are stored in one contact set that is used to finally create a surface estimation of the object. This estimated surface can then be used for grasp planning.

In practice, several questions arise during haptic exploration:

1. How to collect contact information by efficiently selecting the next best touch on the object surface?
2. How to control the robot's hand to move to the next exploration target?
3. How to generate object shape models based on the acquired sparse tactile data?

An extensive body of work has been conducted in the literature to address these questions. First subsection 2.1.1 defines the term haptic perception. Thereafter, the following sections give an overview on the different approaches to haptic exploration.

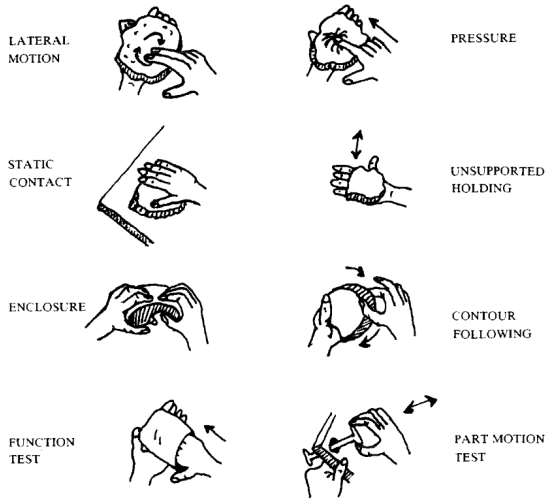


Figure 2.1: Exploration procedures introduced by Lederman and Klatzky (1987), © 1987 Elsevier.

2.1.1 Robotic Haptic Perception and Exploration

Human haptic perception is comprised of proprioception and tactile perception. Proprioception refers to the sense of self-movement and body position (Tuthill and Azim, 2018). This includes limb location, limb configuration, self induced motion, internal or external forces and torques. The sensing nerves are

located within the body in muscles, tendons and joints. Tactile sensing refers to the sensing that arises from the skin and includes temperature, texture, pressure, vibration, slip, and pain.

In the context of robotics proprioception is realized by sensors that are places within the joints and the links of the robot. Examples are the sensing of forces and torques in the joints of the robot or the sensing of linear and angular acceleration. However, robotic proprioception can also be realized by sensing modalities, that have no direct mapping to human sensing, e.g. measuring the motor current to estimate the exerted torque by the joint. Robotic tactile perception can by realized by dedicated tactile sensors that are placed at the fingertips of the robot's hand. Another option is to replace the fingertip by an integrated sensing element that offers sensing of different modalities. Furthermore, the robot can be covered with a robotic skin that offers tactile sensing not only at the fingertips, but on a larger scale. Similar to the robotic proprioception, robotic tactile perception can differ from human tactile perception in terms of available sensing modalities and sensing range.

Humans use a combination of proprioception and tactile perception during object exploration. Lederman and Klatzky (1987) proposed different exploration procedures, that are common during haptic object recognition, see Figure 2.1. These exploration procedures include:

- Lateral motion: Exploration of texture.
- Unsupported holding: Exploration of weight.
- Pressure: Exploration of object hardness/stiffness.
- Enclosure: Estimation of global shape and volume.
- Static contact: Sensing of the temperature.
- Contour following: Exploration of exact shape.
- Function test: Try to use the object as a tool.
- Part motion test: Try to move one part of the object with one hand, while holding the object with the other hand.

Here the *lateral motion* relies mainly on tactile perception to sense the texture of the surface, while the *unsupported holding* uses proprioceptive sensing to estimate the mass of the object. In the context of robotic exploration some of these modalities can be either explored by tactile perception or by proprioception. For example the stiffness of an object was explored using proprioceptive sensing in the form of joint torques by Do et al. (2014) and was also explored using

tactile sensing in the form of normal forces by Kaboli et al. (2018). The ability of robots to sense a modality either through proprioception or tactile perception is also used in this work. In the first part (chapter 3) a dedicated tactile sensor is used to measure contact and surface orientation, while in the second part (chapter 4) the contact between hand and object is inferred using the proprioceptive force torque sensor, mounted in the wrist of the robot.

2.1.2 Potential Field based Exploration

Bierbaum et al. (2008) propose to combine the touch-point selection, path planning and exploration motion generation in one unified approach using harmonic potential fields. The potential field is the sum of attractive potentials and repellent potentials, which are in turn defined as a sum of individual potential sources.This potential field is used to guide the robot hand during exploration. The velocity vector of the hand is calculated to be aligned with the derivative of the potential field.This definition achieves a hand movement that is similar to the motion of a charged particle within an electrical field.

The exploration procedure is initialized by creating attractive potential sources in a grid inside the estimated bounding box of the unknown object. This leads to a hand motion towards the object center. As the hand moves, the potential field is updated. The region covered by the hand motion is marked as explored and the attractive potentials are removed. When the hand makes contact with the object, repellent potentials are added to guide the hand away from known regions. In order to overcome local minima within the potential field, resulting in stalled exploration, the approach uses a small and a large reconfiguration mechanism. During the small reconfiguration, all attractive potentials are inverted, leaving the potential field in a complete repulsive state. After a short while, this reversion is removed and the exploration continues normally. Sometimes the small reconfiguration remains in a local minimum of the potential field. In this case, the hand is retracted completely from the object and approached from a different direction.

During the exploration, all contact points and contact normals are collected in one set of oriented contacts. Based on this set, a four stage grasping pipeline is executed. In the first stage, parallel faces are extracted from the oriented point cloud. In the second stage, the minimum face size is asserted. In the third stage, mutual visibility of face pairs is assured and in the final stage, the distance of the faces is constrained to be within given margins. Face sets that pass

all four pipeline stages are considered grasp candidates. An exemplary exploration with face extraction is shown in Figure 2.2.

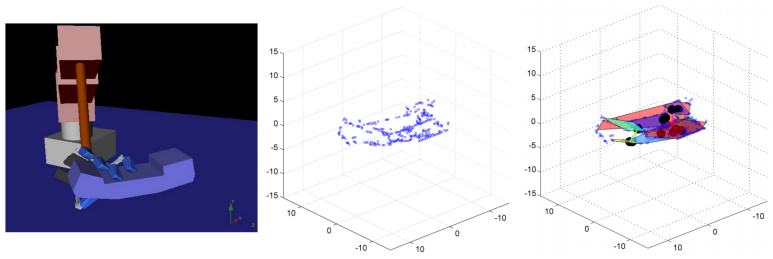


Figure 2.2: Potential field based exploration: Simulation environment (left), oriented point cloud (middle) and resulting parallel faces (right), extracted for grasping, (taken from Bierbaum et al., 2009a, ©2009 IEEE).

2.1.3 Model Uncertainty based Exploration

Another approach to exploring unknown objects is to divide the exploration problem into different parts. Several approaches follow the flowchart shown in Figure 2.3. The exploration process is initialized by either visual observation or the first contact of the hand with the object. Thereafter, the surface of the model is estimated. The next best goal for the exploration is chosen to be the point where the model estimation uncertainty is highest. Then a path is planned from the current hand position to the chosen target and the exploration action is executed. When the hand makes contact with the object again, the model is updated and the process is stared over at the beginning. When the estimation uncertainty has dropped below a predefined threshold, the exploration is stopped and the estimated surface is passed to a grasp planner. Over the years, many approaches for shape estimation have been proposed. Popular approaches are shown in Figure 2.4. Early approaches opted to employ parametric object models for surface estimation. Superquadrics and decompositions of multiple superquadrics have been applied to estimate the surface of the object (Barr, 1981; Solina and Bajcsy, 1987; Allen and Roberts, 1989; Leonardis et al., 1997; Zha et al., 1998; Bierbaum et al., 2007; Biegelbauer and Vincze, 2007; Duncan et al., 2013). Other approaches use a composition of geometric primitives for the representation of the object (Huebner et al., 2008; Huebner and Kragic, 2008; Huebner et al., 2009; Przybylski et al., 2010; Gorges et al., 2010; Marton et al., 2011). For object modeling based on vision data, estimating the back of the object is of special interest and often performed by leveraging assumed object symmetries (Thrun and Wegbreit, 2005; Bohg et al., 2011; Quispe

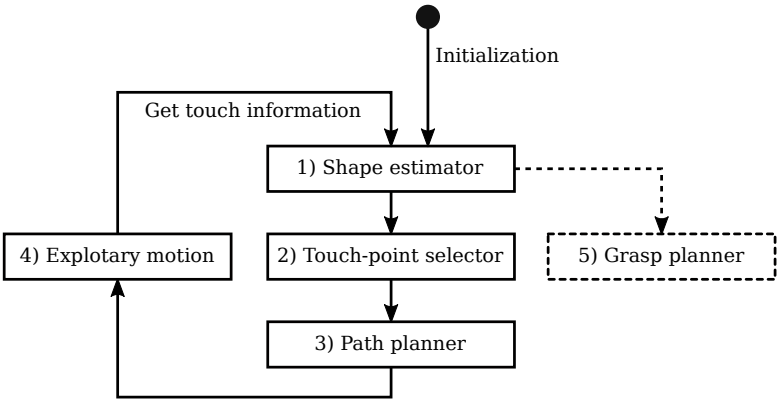


Figure 2.3: Common architecture of haptic exploration approaches. Grasp planning is not present in all approaches and therefore displayed with dashed lines.

et al., 2015; Schiebener et al., 2016). However, when dealing with unknown objects, no prior shape knowledge or symmetries can be assumed and more general surface estimation methods are required. In the field of haptic exploration the use of *Gaussian Processes Implicit Surfaces (GPIS)* (Williams and Fitzgibbon, 2007, see section 3.2) became popular to overcome the limitations of parametric models, and have often been applied (Dragiev et al., 2011; Bjorkman et al., 2013; Sommer et al., 2014; Mahler et al., 2015; Yi et al., 2016; Yang et al., 2016; Martens et al., 2017; Matsubara and Shibata, 2017; Rosales et al., 2018).

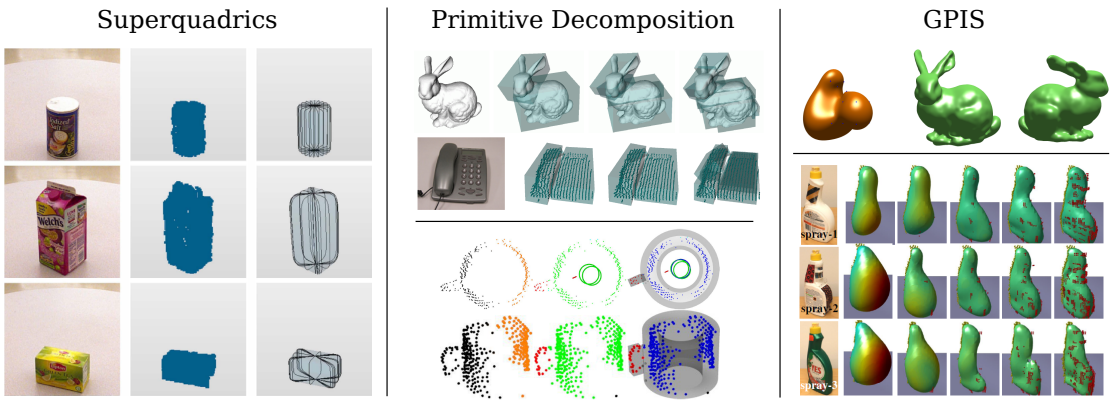


Figure 2.4: Shape estimation methods: Superquadrics (Duncan et al., 2013, © 2013 IEEE, left side), decomposition into primitives (Huebner et al., 2008, © 2008 IEEE, middle top; Marton et al., 2011, © 2011 Springer Nature, middle bottom), Gaussian Process Implicit Surfaces (Williams and Fitzgibbon, 2007, right top; Bjorkman et al., 2013, © 2013 IEEE, right bottom).

GPIS has two properties that benefit haptic exploration in particular:

- *GPIS* does not assume a global shape prior, but combines local Gaussian shape priors to fit all observed contact points.
- *GPIS* yields not only the estimated surface of the object but also provides an uncertainty measure of the estimate.

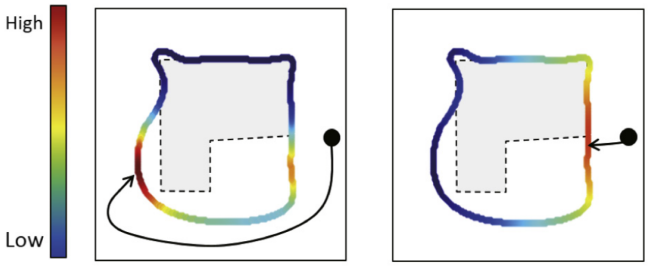
Using these two properties the first two steps of the exploration architecture can be covered. *GPIS* yields a shape estimation, based on the explored point set. Then many potential exploration target candidates x are sampled from the estimated surface S. For each candidate the model estimation uncertainty σ, provided by *GPIS*, is evaluated. The candidate with the highest uncertainty is selected as the next target $\hat{x}$.

$$\hat{x} = \underset{x \in S}{\operatorname{argmax}}\, \sigma(x) \tag{2.1}$$

For path planning and hand motion generation different approaches have been proposed in the literature. Some approaches retract the hand of the robot completely after each exploration action and approach the object again, using inverse kinematics planning. Bierbaum et al. (2009a) use a dual approach where the hand is guided using the derivative of the harmonic potential field. If the hand gets stuck in a local minimum a small reconfiguration is performed. The harmonic potential field is modified so that the field is completely repulsive. This state is kept for a while and the the field is returned to the normal state. If the hand returns to the same location after several of these small reconfigurations the hand is retracted completely in a large reconfiguration. The hand is returned to the initial position and thereafter approaches the object again. Bierbaum et al. (2009a) use *Virtual Model Control*, introduced by Pratt et al. (1996). The approach by Matsubara and Shibata (2017) uses Rapidly-exploring random trees (RRTs) to plan to reach the next exploration action.

2.1.4 Inclusion of Visual Perception

To execute exploration motions the robot arm has to move and in some cases has to reconfigure completely, e.g., when the joint limits are reached or the arm has to reach around the object. Therefore acquiring a dense point cloud by haptic exploration takes a long time and is infeasible in real-world grasping tasks. Bierbaum et al. (2009a) use a stereo camera system to estimate the position, orientation and the dimensions of the object. From this initial estimate they initialize the potential field to contain only attractive sources. In a alter work Bjorkman et al. (2013) use a depth camera to capture an initial point cloud of



(a) Uncertainty. (b) Travel cost (sign inverted).

Figure 2.5: Taking path cost into consideration, (taken from Matsubara and Shibata, 2017, © 2017 Elsevier): Model uncertainty (left) and travel cost in terms of path length (right). This example illustrates that an exploring a location with high uncertainty can lead to long travel paths. However, choosing the exploration action with the shortest path does not yield a lot of new information.

the object. This point cloud contains the front view of the object. Then an initial surfaces estimate using *GPIS* is created. Based on this initial estimate the exploration pipeline is started. The first exploration target is chosen using the model uncertainty. This first exploration target often lies on the back of the object, since the back is hidden from the viewpoint of the camera, hence no points can be observed. The robot touches the object several times and the estimated surface is refined, as shown in Figure 2.4 (right side). The inclusion of visual perception can greatly reduce the amount of required exploration actions. In their work, Bjorkman et al. (2013) mention that about ten exploration actions per object were sufficient to successfully classify the objects, based on features extracted from the estimated surface.

2.1.5 Inclusion of Path Cost

Each executed exploration action comes with a cost in terms of robot arm movements. When exploration actions are chosen, based on the predicted model uncertainty alone it can happen that exploration targets are chose that lead to high exploration costs. To reduce the expected cost of exploration actions, resulting from arm movements of the robot, Matsubara and Shibata (2017) proposed to include the path cost during the evaluation of possible exploration candidates. In their work, the authors show that in some cases exploring based on uncertainty alone will lead to overall higher exploration costs than taking smaller exploration steps that consider path costs. A 2D example taken from their work can be seen in Figure 2.5. The object is shown as a gray area while the estimated

surface is displayed as a thick, color-coded line. On the left side, the estimation uncertainty is shown. In this case, the uncertainty is highest at a location far away from the current position, marked as a big dot. The resulting exploration path, shown as an arrow, reaches around the object to reach the area of high model uncertainty. On the right side, the path cost is shown, for each potential target on the estimate surface. The resulting path is much shorter, however not much new information will be gained. This example demonstrates that a trade-off must be found between minimizing uncertainty and minimizing path costs. In their work, the authors propose to find this trade-off by subtracting the path cost ν from the estimated information gain. The selected exploration target $\hat{x}$ is described by maximizing this difference E, see Equation 2.2 and Equation 2.3.

$$\hat{x} = \underset{x}{\operatorname{argmax}}\, E(x) \tag{2.2}$$

$$E(x) = \frac{\sigma(x)}{\max_{x' \in S} \sigma(x')} - \theta \frac{\nu(x)}{\max_{x' \in S} \nu(x')} \tag{2.3}$$

Here x is a potential exploration target, $\sigma(x)$ is the variance at location x, $\nu(x)$ is the path cost to location x and S is the estimated surface. The authors state that θ is needed to balance the two parts of the difference, and object specific tuning of θ is necessary.

2.1.6 Exploration of Object Properties

When exploring unknown objects, various modalities can be explored. In the previous sections, approaches were presented that focus on the contact modality. These approaches deal with the collection of contact points with the goal of surface estimation. In addition to the object surface, there are other properties that can be explored. The exploration can be divided into proprioceptive exploration, where forces, accelerations and torques within the robot's joints are measured and tactile exploration where tactile sensors are mounted at the fingertips of the robot's hand.

One property that can be explored using proprioception is the center of mass of an object. Atkeson et al. (1985) implement the proprioception in form of a force torque sensor to estimate the mass, center of mass and moments of inertia of the object. They use the dynamic response of the object under acceleration in conjunction with the measurements from the force torque sensor and the dynamic model of the robot. The stiffness of an unknown object can also be estimated using the proprioception of robotic hands. The pneumatic actuated

hand (Bierbaum et al., 2009b) of the humanoid robot ARMAR-III (Asfour et al., 2006) offers torque sensing capabilities in the form of air pressure measurement. Do et al. (2014) use this proprioception to estimate the softness of an object in the hand.

Object properties that can be explored by tactile sensing include texture and thermal conductivity. To estimate these parameters Kaboli et al. (2017) use the multi-modal robotic HEX-O-SKIN, first introduced by Mittendorfer and Cheng (2011). The authors use the skin to first explore an unknown workspace in order to find objects. After objects have been found they use the skin to actively explore the physical properties surface texture, stiffness, and thermal conductivity. Using these explored object properties the authors train a Gaussian Process Classifier (GPC) for object discrimination. In a later work, Kaboli et al. (2018) use OptoForce sensors (Tar and Cserey, 2011) which can measure contact forces subdivided into normal forces and shear forces. Kaboli et al. (2018) slide the sensor across the objects' surfaces to explore the surface texture of the objects. Furthermore, the stiffness and the center of mass of the objects is explored.

Another approach to explore the surface texture of objects was proposed by Fishel and Loeb (2012). They use the multi-modal BioTac sensor that integrates sensing of force, vibration, and thermal conductivity (Fishel et al., 2008; Wettels et al., 2008; Lin et al., 2009; Wettels et al., 2014). Fishel and Loeb (2012) use the vibration sensing of the BioTac sensor while sliding the sensor over the surface of different materials. They use exploration motions that are inspired by human exploration motions, that were first analyzed by Lederman et al. (1982) and later formalized as exploration procedures by Lederman and Klatzky (1987).

2.1.7 Tactile Sensing and Robotic Skin

In order to be able to execute exploratory motions with the robotic system end acquire contact information from the object, the robotic hand has to be equipped with some kind of tactile or force sensing element. However, the goal of achieving human-level performance in tactile perception has been studied for several decades (Harmon, 1980, 1982; Esrom, 1989) but remains a major challenge in the field of robotics (Bartolozzi et al., 2016). It is therefore not surprising that a great variety of research work has been dedicated to developing tactile sensors. This section gives a brief overview over the different sensor technologies that have been applied for exploration purposes.

In the field of robotics, many tactile sensors have been proposed to measure contact force, relying on different measurement principles. Tactile sensors have

to be placed at the positions of contact between the hand and the object. This location is often at the fingertips, leading to size constraints on the sensor. Since it is difficult to measure interaction force directly in a confined space, most sensors leverage some kind of electrical measuring principle. In most cases, the force applied to the sensor is transferred to a soft material, leading to a property change of that material, which in turn can be measured. The literature contains a large number of different tactile sensors. In the following, exemplary sensors are presented for some measuring principles.

Barometric Sensors

Tenzer et al. (2012) propose to use a barometric pressure sensor as a tactile sensor. The pressure sensor is housed in a sensor chip mounted on a standard printed circuit board. The chips integrate signal amplifiers, analog-to-digital converters, pressure and temperature sensors and digital communication capabilities, allowing several sensors to be connected to one physical bus. The sensors are covered using an elastic polyurethane to provide good friction for grasping tasks. External forces applied to the polyurethane cover are transferred as a pressure to the underlying sensor. The sensor is able to detect these pressure changes, allowing for accurate measurement of the applied force, which is linear to the digital sensor output.

Resitive Sensors

Another sensing principle relies on the change of resistance under pressure. Weiß and Worn (2005) propose to use resistive tactile sensor cells, comprised of a conductive polymer and electrodes. When the conductive polymer is compressed, the electrical resistance changes according to the applied force. The approach allows the sensor to be built in a matrix arrangement, resulting in a tactile sensing array. This measurement principle was extended to 3D surfaces and applied to a robotic hand fingertip (Koiva et al., 2013). In their work, the authors use laser-structuring technology to apply conductive tracks to curved surfaces, arguing that this enables the manufacturing of 3D-shaped tactile sensors. The signal processing electronics is placed on the backside of an artificial layer of skin within the fingertip. The sensor is applied to the Shadow Robot Hand, resulting in 12 tactile sensor regions. The embedded microcontroller can capture the force patterns with a sample rate of 1 kHz, enabling slip detection of objects.

Capacitive Sensors

Tactile sensing is not only relevant for the fingertips, but may also be implemented as a sensorized skin for humanoid robots. Cannata et al. (2008) developed such a skin for use on the iCub robot (Metta et al., 2008). Continuing this work, large parts of the iCub robot were covered with a sensitive skin (Schmitz et al., 2011), which can be used for kinematic self-calibration (Roncone et al., 2014). Distributed tactile perception capabilities in combination with vision have been used to learn visuo-tactile associations for peripersonal space representation (Roncone et al., 2016). Furthermore, the fingertip of the iCub has also been sensorized based on a capacitive sensor (Schmitz et al., 2010; Jamali et al., 2015).

Optical Sensors

Another measuring principle for measuring forces is to use the force acting on the sensor to elastically deform the structure of the sensor. Reflecting surfaces or small mirrors are mounted inside the structure of the sensor. A light beam is sent onto these surfaces, which is usually generated by one or more LEDs. The applied force deforms the structure elastically, deflecting the light beam. The change in light intensity is then converted into electrical signals by a phototransistor. This measuring method is used in the OptoForce sensor (Tar and Cserey, 2011). The sensor consists of a hemisphere whose inside is reflective. In the middle there is an infrared LED, whose emitted light is reflected by the reflecting inner side. Phototransistors measure the light intensity at several points within the hemisphere. The normal force and the shear forces can then be calculated from the measured light intensity.

Another approach is to use a small camera rather than individual photo transistors. This is used for example in the sensors FingerVision (Yamaguchi and Atkeson, 2016) and GelSight (Yuan et al., 2015, 2017). Small black dots are marked on the surface of the material. A camera inside the sensor looks at these points. The positions of the dots in the camera image are stored in the unloaded state. External forces on the sensor lead to a deformation of the sensor material. This also causes the points on the sensor surface to shift. This displacement is again tracked by the camera. After calibration of the camera and material parameters, tangential forces and torques can be calculated from the relative displacement of the points.

Multi Modal Robotic Skin

With the HEX-O-SKIN, Mittendorfer and Cheng (2011) introduced a hexagonal, modular and multi-modal skin. Each cell of the skin has a hexagonal shape, implemented as a printed circuit board, equipped with sensors for temperature, acceleration and proximity. The goal is to emulate the human sense of temperature, vibration and light touch. To achieve fast results the authors opted to use off-the shelf sensors, instead of developing the sensors themselves, leading to a faster development cycle. The skin was later evaluated by exploring unknown workspaces and by identification of different objects (Kaboli et al., 2017; Kaboli and Cheng, 2018).

Multi Modal Robotic Fingertips

With the BioTac sensor, a multi modal sensor was introduced, that integrates sensing of force, vibration, and thermal conductivity. First Fishel et al. (2008) introduced a sensor concept that allowed the sensing of micro-vibrations, when the sensor was slided across a textured surface. In the same year Wettels et al. (2008) described the placing of electrodes within the fingertip of the sensor that allow the sensing of contact forces. Later, Lin et al. (2009) extended the sensor with signal processing that enabled the sensing of contact forces, micro-vibrations and thermal flux. Finally, Wettels et al. (2014) proposed to extend the sensing modalities by processing the raw signals further using Artificial Neural Networks (ANNs) and calibration procedures.

Capacitive Proximity Sensors

Tactile perception is not limited to contacts between the robot and the environment. Leveraging the unique capabilities of contactless perception with capacitive sensor technology, Navarro et al. (2013) can assure safe human-robot-interaction. The sensor is arranged in a matrix and can be used to detect events in the near proximity, to enable near field perception. The authors use the sensor for objects tracking, including human hands. This allows for safe human robot interaction, as touch events can be detected before contact is made. This sensor has meanwhile been improved and can operate in proximity sensing mode and tactile sensing mode (Alagi et al., 2016). The signal processing is integrated in the sensor prototype, resulting in a modular design. In their work, the authors use the sensor to measure the change in capacity for different materials, including wood, plastic and metal surfaces.

Inertial Measurement Units

An interesting approach to haptic perception is the usage of inertial and orientation sensors. Following this idea, the underactuated Pisa/IIT SoftHand was equipped with IMU sensors to estimate the pose of the hand (Santaera et al., 2015). Traditionally the configuration of the kinematic structure is performed using relative or absolute encoders in the joints. However, when dealing with joints in the fingers of humanoid hands, size and structural constraints often make sensor placement a difficult task. Therefore, the authors opt to place IMU sensors not in the joints of the fingers, but on the fingers. They develop a method to estimate the kinematic configuration state of the hand, based on the orientations provided by the IMUs. Besides reducing the constraints of sensor placement, the choice of IMU sensors also allows for non-rigid kinematic structures, such as soft hands. This enables the approach to be adopted for other sensing tasks, such as sensorizing gloves, to estimate the configuration of human hands during grasping studies.

Achieving human-level tactile sensing capabilities is still an unsolved problem and active field of study. Comprehensive summaries of recent tactile sensor developments can be found in the extensive review papers by Kappassov et al. (2015), Dahiya et al. (2010) and Yousef et al. (2011).

2.1.8 Summary

The previous sections presented different approaches to haptic exploration for grasping. The first approach introduced by Bierbaum et al. (2008) combines the choice of the next exploration target, the motion generation and the robot control in one unified approach using potential fields. The robot's hand is guided by a potential field, following the gradient of the field. The potentials within the field are updated when contact occurs to drive the hand to unexplored regions. The approach was extended to avoid local minima in the potential field.

More recent works in haptic exploration follow a different approach, where the choice of the next exploration target, the robot control and the surface estimation are separated. All approaches use the same surface estimation technique called *Gaussian Process Implicit Surfaces (GPIS)*, introduced by Williams and Fitzgibbon (2007). The reason for this is that GPIS is very well suited for surface estimation when only a few contact points are available. In addition, various modalities may be combined, such as an initial visual view followed by tactile exploration. GPIS is based on Gaussian processes and thus, in addition

to the surface estimation provides the variance of the estimate. This variance is then used to select the nearest exploration point on the estimated surface. In addition to purely optimizing the information gain, Matsubara and Shibata (2017) introduced consideration of the expected exploration costs.

In addition to the exploration strategy, the measurement of contact between robot hand and object is an important topic. To achieve this, a variety of different tactile sensors has been developed. A variety of measurement principles are used, including capacitive sensors, resistive sensors, optical sensors, Hall effect based sensors, pressure sensors, and sensors that combine different measurement principles. The diversity of the different sensors and the large variance of the measuring principles indicate that the reproduction of the human sense of touch is a difficult task. Therefore, some sensors specialize in measuring a single modality, such as normal forces or shear forces. These specialized sensors often do not have the variety of modalities like the human sense of touch. A task-specific specialization, however, allows specific exploration tasks to be performed.

2.2 Data-driven Grasping of Unknown Objects with Humanoid Robots

In industrial and household settings, humanoid robots have to work in an unstructured and partially unknown environment, where unknown objects are present. For many manipulation tasks, the robots need to be able to grasp these objects. However, finding a suitable hand position and approach direction for grasping is a difficult task. In the case of unknown objects no precise objects models are available. In addition, there is no prior knowledge about the objects. The robots need to be able to plan and execute grasps based on the available sensor data. Bohg et al. (2014) describe the problem of finding a suitable grasp in their survey paper:

> Given an object, grasp synthesis refers to the problem of finding a grasp configuration that satisfies a set of criteria relevant for the grasping task. Finding a suitable grasp among the infinite set of candidates is a challenging problem and has been addressed frequently in the robotics community, resulting in an abundance of approaches.

A common approach is to split the grasping problem into two parts. First one or multiple grasp candidates are generated, based on the visual and haptic data.

Following the definition by Morales et al. (2006) and Bohg et al. (2014), these grasp candidates can be described using the following parameters:

- *Grasping point on the object*: relative position of the *tool center point (TCP)* to the object.
- *Approach direction*: vector that the hand should follow, to reach the grasping point.
- *Hand orientation*: rotation of the hand during grasping.
- *Pre-shape*: initial finger configuration.

Thereafter, one of the generated candidates is chosen for execution and the robot's hand is moved to the grasp location. When the hand reaches the target location, the fingers can be closed around the object. However, neither the object perception, nor the execution by the robot is perfect in practice. Therefore, sensor feedback can be incorporated to mitigate uncertainties during grasping and reactively adapt the grasp position. Finally, when the fingers are firmly closed around the object, the object can be lifted and the grasp is complete.

2.2.1 Structure of Data-Driven Grasping Approaches

Following the definition of Bohg et al. (2014), data-driven approaches for grasping unknown objects can be split into two phases, called *offline* and *online*, as shown in Figure 2.6.

In the *offline* learning phase, a set of labeled training examples is available. Each training sample consists of an object, derived object features, such as visual representations, and associated grasping points. The goal of the training process is to build a model that can generate grasp candidates, based on the derived object features, without having access to the underlying object model. Prior to deep learning based approaches, the features used for learning were created using hand crafted algorithms. The model can for example be implemented as a support vector machine (SVM) (Pelossof et al., 2004; Jiang et al., 2011).

In the *online* phase, the robot is confronted with a previously unseen scene, containing unknown objects. First, the scene is segmented into background and objects. Then, one object is chosen for grasping. For this object, features are extracted. Using these derived features, the model can be queried and possible grasp candidates can be predicted.

The paper by Bohg et al. was published in 2014, which is right before deep learning was applied widely for grasping unknown objects by robots. Therefore, numerous non-deep learning approaches are listed and categorized in the

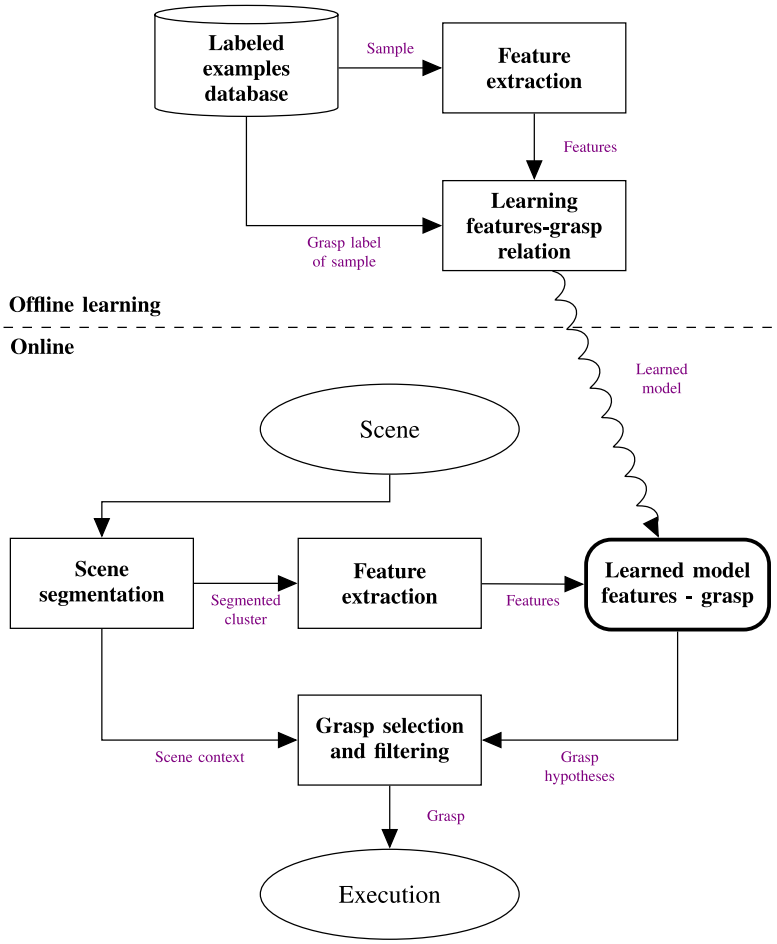


Figure 2.6: Typical flow-chart of data-driven grasping approaches, as defined by Bohg et al. (2014), © 2014 IEEE.

survey paper. However, the introduction of deep learning in grasping has led to a significant performance increase in grasp detection and generation accuracy (see Table 2.2). This trend has been stated explicitly in the computer vision community by Bütepage et al. (2018): „Recently, deep learning has seen unprecedented developments within the computer vision community and is therefore a promising candidate for learning-based approaches.“ Therefore, the related work covered in this section will focus deep learning based approaches, published after 2014. The structure given in the flow-chart still remains valid. The only changed part is that in the case of deep learning the feature extraction is now mostly part of the model, with the exception of simple data preprocessing. When using deep learning usually no hand crafted feature extractors are needed, as this feature extraction process is part of the training process as well.

As stated in Bohg et al. (2014), „finding a suitable grasp among the infinite set of

candidates is a challenging problem“. The approaches to find a suitable grasp can be categorized into five groups, as shown in Figure 2.7.

(A) *Conventional approaches* first extract features, using handcrafted techniques. Based on these features a classifier is trained to rate different possible grasp candidates.

(B) *Generative approaches* leverage the potential of deep learning by using a direct regression approach to output the best grasp pose directly, for any given input.

(C) *Shape completion* circumvents the direct generation of grasp candidates by instead estimating the 3D shape of the object, based on the input data. Then a conventional grasp planner is used to plan grasps, based on this estimated model.

(D) *Heat-map based approaches* transfer the learning problem to a pure image-to-image mapping problem. The idea here is to leverage the advances in deep learning based image processing, without dealing with encoding of grasps directly. For each input image, an associated heat map is created, where good grasping regions are marked.

(E) *Discriminative approaches* take as input the perceived scene and a grasp candidate and output the expected grasp success probability. These approaches function by learning a data-driven grasp metric, which can deal with incomplete information, however a grasp candidate generator is necessary.

An overview of data-driven grasping approaches is listed in Table 2.1.

In the following, possible sources of training data for the offline training phase are presented, followed by a detailed description of the five different categories.

2.2.2 Sources of Training Data

All learning based, data-driven approaches need labeled training data during the offline training phase. Compared to pure computer vision tasks, where labeled training data is abundant, and comes in a standardized form, i.e. images, acquiring training data for robotic grasping is challenging, as a recent survey paper has stated.

> While the performance of deep learning is promising, these approaches are data hungry. [...] Although deep learning proved successful for

Publication	Year	Input	Hand	Objects	Method	Lift success
Lin and Sun	2015	Mesh	Barrett hand	Own set	LfD[1)]	79%[3)]
Nguyen et al.	2016	RGB-D	WALK-MAN	Own set	Heat map (2D image)	92.2%[3)]
Kopicki et al.	2016	RGB-D[4)]	DLR-HIT2	Own set	LfD[1)]	77.8%[1)]
Varley et al.	2017	RGB-D	Barrett hand	YCB subset	SC[6)] (3D grid)	93.3%[2)]
Varley et al.	2018	RGB-D & tactile	Barrett hand	YCB subset	SC[6)] (3D grid)	87.5%[1)]
Schmidt et al.	2018	RGB-D	ARMAR-III	YCB & KIT	Gen[5)] (6D pose)	55%[2)]
Liu et al.	2019	RGB-D[4)]	Shadow hand	BigBIRD, YCB, KIT, Grasp DB	Gen[5)] (Hand config.)	-
Lundell et al.	2019	RGB-D	Barrett hand	YCB, Grasp DB	SC[6)] (3D grid)	59%[1)]

Table 2.1: Data-driven grasping approaches with humanoid hands.
1) Lift success during robot experiments
2) Lift success in simulation
3) LfD: Learning from demonstration
4) Multiple RGB-D images (color and depth images) were captured per grasp
5) Generative
6) Shape completion, resulting in a 3D voxel grid

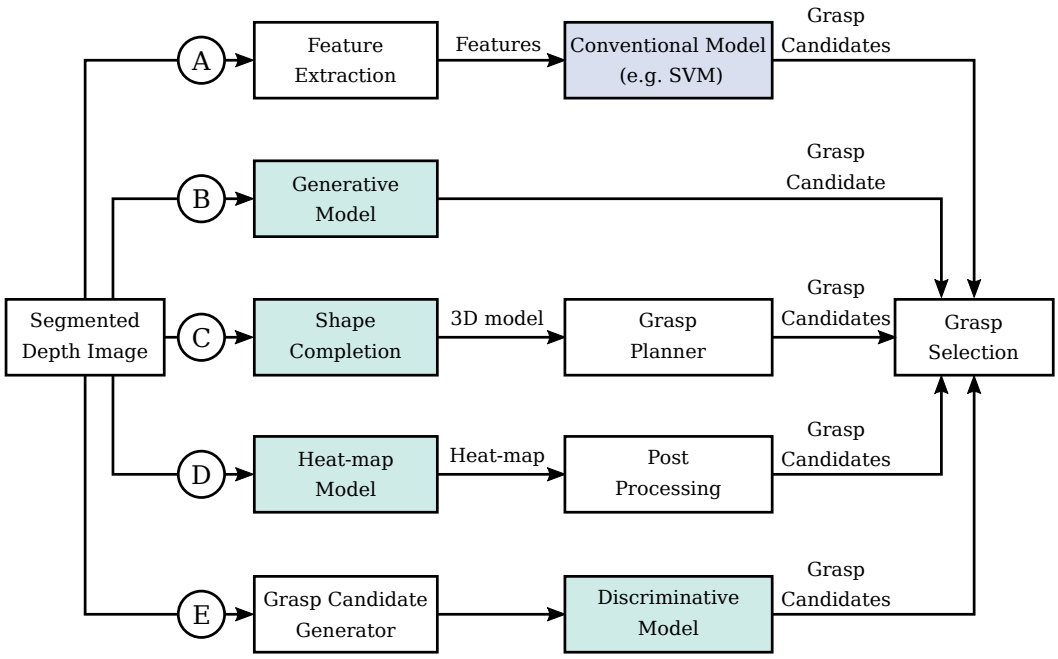


Figure 2.7: The five different categories of approaches to data-driven grasp synthesis.

> images or videos, where large amounts of data are available, collecting data for robotic grasping requires tedious human labeling, as in the above-mentioned approach, or hours of execution time on a real physical system (Bütepage et al., 2018)

In the robotics community, four major approaches exist to generate labeled training data for real robotic system.

1. A grasping dataset is created by humans through *hand labeling* thousands of training samples.
2. A self-supervised approach is executed directly *on the target system*. The robot learns from trial and error and the associated model is incrementally updated.
3. A human teacher demonstrates the grasping process by guiding the robot directly or the robot observes the human grasping process. The robot *learns from these demonstrations*.
4. A dataset can be created by *training data generation in simulation*. Here the robot is placed in a simulated environment with virtual objects.

All aforementioned approaches have strengths and weaknesses, which will be outlined in the following.

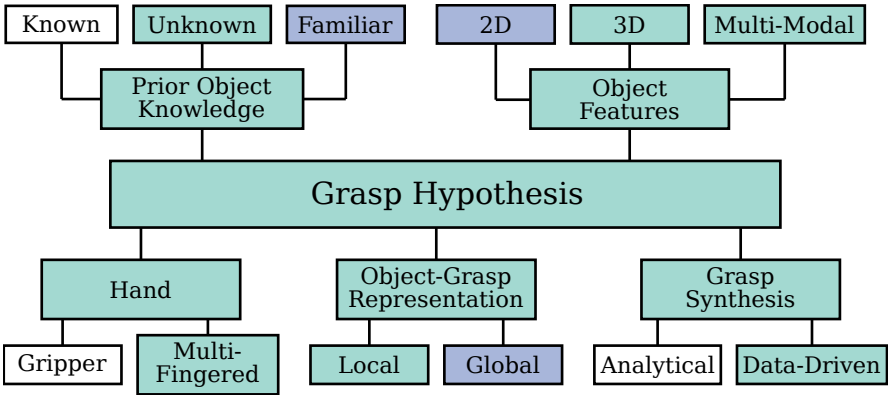


Figure 2.8: Aspects of grasp candidate generation, as defined by Bohg et al. (2014). The aspects covered in this work are highlighted green, while further aspects covered in related work are marked blue. Not covered aspects are displayed in white.

Hand Labeled Datasets

The main advantage of hand labeled data for learning is that the resulting dataset is of high quality. The human knowledge of the problem can be used fully and no special algorithms have to be developed for the generation of the training data. In addition, the data format and the format of the labels can be optimized, so that the learning model can perform optimally.

The main disadvantage lies in the work required to create such a dataset. For current deep learning approaches to function properly, large amounts of labeled training data is required.

In the field of robotic grasping, only few manually labeled datasets have been created. The most prominent dataset is the Cornell Grasping Dataset, first used by Jiang et al. (2011). One main aspect of this dataset is that it is designed for grippers. Each training sample of this dataset consists of a top down RGB-D camera image of an object and grasping locations encoded as rectangles. This description is well suited for grasping with grippers and has led to a large amount of publications, which work either solely on the dataset or evaluate their approach using the dataset (see Table 2.2). To show the impact of a large hand-labeled dataset, the increase of grasp detection accuracy is shown in Table 2.2.

In the field of grasping with humanoid hands, hand labeled datasets are not widely used. This is due to two main reasons:

- When grasping with a gripper the grasp execution can often be reduced to a simple procedure. The necessary grasp aperture is applied and the grip-

Model	Year	Approach	Input Size	Accuracy	
				Image-wise	Object-wise
Jiang et al.	2011	SVM[1)]	227 x 227	60.5 %	58.3 %
Lenz et al.	2015	Discrim.	227 x 227	73.9 %	75.6 %
Redmon and Angelova	2015	Generative	224 x 224	84.4 %	84.9 %
Redmon and Angelova	2015	Anchor boxes	224 x 224	88.0 %	87.1 %
Wang et al.	2016	Discrim.	96 x 96	81.8 %	69.0 %
Zhang et al.	2017	Anchor boxes	224 x 224	88.9 %	88.2 %
Kumra and Kanan	2017	Generative	224 x 224	89.2 %	89.0 %
Mahler et al.	2017	Discrim.	32 x 32	93.0 %	-
Guo et al.	2017	Heatmap	108 x 108	93.2 %	89.1 %
Morrison et al.	2018	Heatmap	300 x 300	78.6 %	-
Park and Chun	2018	Anchors[2)]	400 x 400	89.6 %	-
Asif et al.	2018	Heatmap	244 x 244	90.6 %	90.2 %
Chu et al.	2018	Anchor boxes	227 x 227	96.0 %	96.1 %
Park et al.	2018	Anchor boxes	360 x 360	96.6 %	95.4 %
Zhou et al.	2018	Anchor boxes	320 x 320	**97.7 %**	**96.6 %**
Wang et al.	2019	Heatmap	400 x 400	94.4 %	91.0 %
Gariépy et al.	2019	Generative	224 x 224	92.4 %	-

Table 2.2: Comparison of detection accuracy on the Cornell dataset.
1) Jiang et al. (2011) is the only non-deep learning approach in this comparison.
2) Park and Chun (2018) use a CNN to generate anchor candidates instead of scanning a full anchor grid. This speeds up the grasp candidate generation, as fewer grasp candidates have to be considered.

per is moved to the grasp pose. Then the gripper is closed. Morales et al. (2006) have introduced a unified grasp description for humanoid hands, in the form of grasp type, grasp starting point, approach direction and hand orientation. The challenge here lies in finding an encoding for the grasp type, that allows the transfer of a grasp from one humanoid hand to another. This is difficult, since humanoid hands are different in size, shape and actuation. E.g. some hands are fully actuated and others are underactuated. Transferring a grasp for grippers is easier, since grippers are very similar to each other.

- In the case of bin-picking applications with grippers, a top-down grasp can be described by the position, rotation and aperture of the gripper, leading to a representation, which can be drawn as a rectangle in a 2D image, e.g. the representation found in the Cornell dataset. In the case of humanoid hands such a top-down representation cannot describe all

grasps that the robot should be able to execute. For example, grasping tall objects from the side is difficult to encode in a top-down description. Furthermore, most humanoid robots do not have access to a top-down view of the scene during grasp execution, as the camera is mounted in the head of the robot, in most cases. Therefore, the process of creating suitable grasp labels for humanoid hands is more complex than labeling grasps directly in the image plane.

As the performance increase in the case of grasping with grippers has shown, ideally, a hand labeled dataset should be used for grasping, however due to the two mentioned reasons, so far, no hand labeled dataset for humanoid hands exists.

Training Data Generation on the Target System

In recent years reinforcement learning based approaches have been shown to be effective in mastering board games such as „Go“ (Silver et al., 2017b) and chess (Silver et al., 2017a), as well as computer games such as Dota 2 (OpenAI, 2018) and StarCraft 2 (Vinyals et al., 2019). All these games are considered challenging to play by artificial intelligence (AI) and are considered milestones in AI research. However, all of these approaches are still very data hungry, e.g. the approach by OpenAI to play Dota 2 has been trained, playing the game for the equivalent of over $40\,000\,\mathrm{h}$uman years. In the setting of board games of computer games, generating this experience during training does not imply any physical cost, since the games can be fully simulated or are designed to be run on a computer in the first place. However, in the case of robotics, training directly on the target system is often infeasible, due to the long run time.

Figure 2.9: Robot setup used for training and testing by Levine et al. (2018), © 2018 SAGE Publications.

Nevertheless, to a certain extent, this fully data-driven deep reinforcement learning approach has been applied to grasping with robots equipped with grippers

by Levine et al. (2018). In their work, the authors use a reinforcement learning approach, training on 14 robotic manipulators in parallel. The robots try to execute grasps, based on the current learned policy. Based on the outcome of the grasping attempt, the policy is updated and refined iteratively. Overall, they perform over $800\,000\,\mathrm{g}$rasping attempts.

Another approach is to pre-train the model in simulations, used by Andrychowicz et al. (2018). Using domain randomization in the simulation, the authors were able to build a model that was able to perform in-hand manipulation to rotate a colored cube, using the shadow hand. The training time in simulation was equivalent to about $100\,\mathrm{y}$ears of experience. Running the simulation in parallel on 384 worker machines, each equipped with 16 CPU cores, allowed for generation of 2 years of simulated experience per hour.

The main advantage of reinforcement learning is that the learned model can deal with complex tasks in theory. However, these approaches are still very data hungry and either require a full and accurate simulation of the target system, or a lot of training time on multiple instances of the target system. Furthermore, pre-training in randomized simulated environments is, at the moment, not accessible to us, as it requires large amount of computation power.

Learning from Demonstration

The main idea behind Learning from Demonstration (LfD) is to enable the robot to learn skills from human demonstrations (Billard et al., 2008; Argall et al., 2009). In the context of grasping, a human teacher demonstrates how to grasp different objects and the LfD approach can represent this in an abstract skill, incorporating the demonstrator's intention (Lin and Sun, 2015) as well as transferring the demonstration from the human hand to the robotic hand. The advantage of LfD is that the full knowledge of the human teacher can be used in learning the task. The human teacher does not necessarily have to be an expert and many demonstrations can be performed quickly using different objects and grasping motions. One of the main challenges in LfD approaches lies in the transfer from human demonstrations to the robot's kinematics and dynamics. E.g., a direct transfer of joint trajectories is infeasible when dealing with robotic hands, since most hands have less degrees of freedom than the human hand. Therefore, other mappings between human demonstration and robot motion have to be found (Romero et al., 2010). Lin and Sun (2015) focus on the grasp type, thumb placement and approach direction. They thereby reduce the complexity of the problem to factors, which can be mapped to most robotic hands.

In fact, in their work, the authors transfer human five-finger grasping motions to a three finger robotic hand. Kopicki et al. (2016) presented another approach where they used kinesthetic teaching to guide the robot's arm directly during training, thereby circumventing the challenge of transferring the demonstrations from humans to the robot. In order to record accurate finger trajectories the authors used the active compliance of the DLR-HIT2 hand.

Training Data Generation in Simulation

The main idea of training data generation in simulation is to create a virtual environment in which grasping can be simulated. In this environment, 3D models of different objects can be loaded and presented to a robot for grasping. The robot can repeatedly perform grasping experiments and the results can be recorded, resulting in an abundance of training data. The main advantage of generating training data in simulation is that this generation does not include operating a real robot or labeling data by hand. However, the main challenge is that no robotics simulation is perfect, especially when dealing with contacts. In grasp planning the stability of grasps has been studied extensively. One focus point are the contact points between hand and object, which are used to assess the quality of the grasp. E.g., the term *force closure* is used to describe that a wrench in any direction can be generated by the grasp. In grasp analysis this *force closure* is considered fundamental, since it is a simple measure to determine grasp stability. However, a recent work by Weisz and Allen (2012) has shown that relying on analytical grasp analysis is prone to misclassification of grasps, leading to low transferability of simulated grasps to the real robotic system. In their work, the authors propose a domain randomization process for grasp analysis that calculates the probability of force closure for a given grasp.

2.2.3 Deep Learning for Image Understanding

Deep learning methods have led to a significant speed up in image processing research. This includes general tasks, such as classification of images (Guo et al., 2016) as well as medical image analysis (Litjens et al., 2017). As deep learning is a widely used method, this section will only briefly introduce the aspects of deep learning models, relevant to this thesis. A comprehensive review of deep learning methods for image processing and classification can be found in the

thesis of Guo et al. (2016) as well as in the review of Alom et al. (2018) and the book by Goodfellow et al. (2016).

Convolutional Neural Networks

Prior to deep learning approaches, image processing was primarily done by feature extraction and subsequent classification, based on these features. The feature extraction process was done by manually implementing feature extractors, such as histograms or gradients. These features were then presented to a conventional classifier, e.g. a Support Vector Machine (SVM). One implementation of such a conventional approach for grasping unknown objects can be found in Jiang et al. (2011).

With the popularization of deep learning most of the image classification approaches moved to using Convolutional Neural Networks (CNNs). One of the first and often cited approaches, relying on a CNN, is AlexNet, introduced by Krizhevsky et al. (2012). In their work, the authors state: „Our results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning.“ AlexNet was one of the milestones in deep learning for image classification. Many more publications followed, where each approach extended the original concept. The main idea of CNNs is to not implement the feature extraction using human intuition, but to use a pure data-driven approach with supervised learning. To this end, a neural network is used as a model and trained. In the case of CNNs, one common network architecture is to use a set of convolutional layers, followed by a set of fully connected layers. These CNNs can be used for 2D images, but the concept is not limited to two dimensions and can be extended to handle 3D voxel grids, as shown in Maturana and Scherer (2015).

2.2.4 Generative Approaches

Motivated by the success of convolutional neural networks in image classification tasks, similar techniques can be applied for robotic grasping. Instead of classifying an image, the neural network is used to predict a feasible grasp pose, given an input image. Here deep learning is applied in a straightforward fashion to the grasping problem, as a deep neural network is trained to predict grasp poses based on the available robot's sensor data, without any pre- or post-processing.

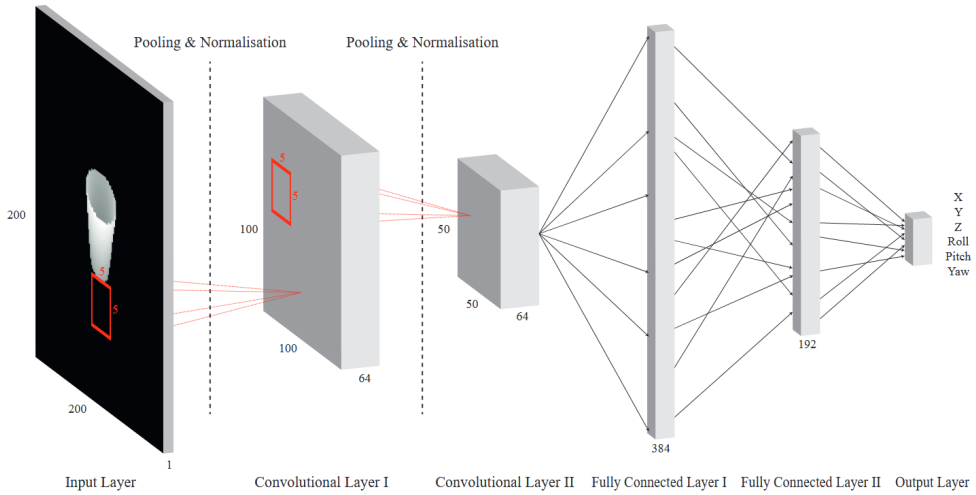


Figure 2.10: Generative network architecture for grasp prediction by Schmidt et al. (2018), © 2018 IEEE.

Following this idea, Schmidt et al. (2018) use a depth camera to capture a depth image of an object. They segment the object from the background and use the depth information of the image as input for a neural network, as shown in Figure 2.10. The network is comprised of two convolutional layers, followed by two fully connected layers. The network is trained to predict the grasp pose, encoded as the position of the tool center point and the orientation of the robot's hand. To train the network the authors use precomputed grasps, which are generated by a skeleton based grasp planner (Vahrenkamp et al., 2018), combined with rendered depth images of the objects.

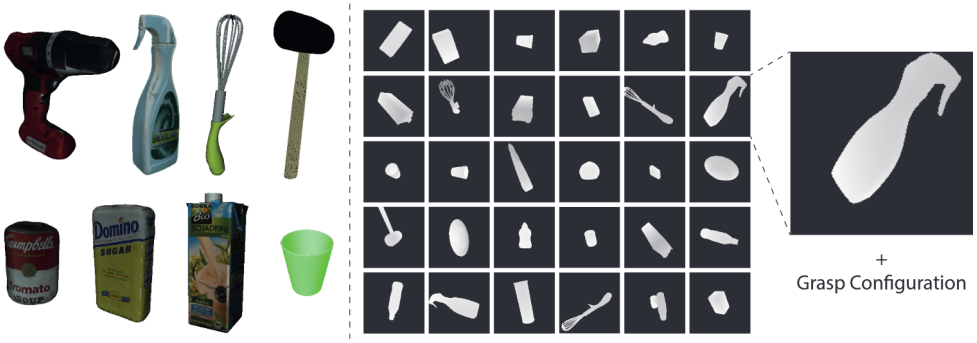


Figure 2.11: Training data set used by Schmidt et al. (2018), © 2018 IEEE.

The necessary training set is generated in simulation. The models taken from the KIT and YCB object databases are placed into a simulation environment. A simulated camera in the robot's head observes the objects, as they are rotated randomly in front of the robot. For each orientation of the object, one suitable grasp is chosen from all available grasps to reduce planning complexity and

to achieve a one-to-one match between an input image and the corresponding grasp pose.

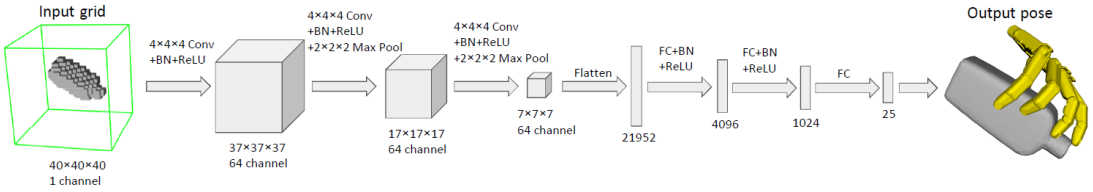


Figure 2.12: Generative network architecture for grasp configuration prediction by Liu et al. (2019).

In a recent work, Liu et al. (2019) use a generative network architecture to predict not only the pose of the robot's hand for grasping but also the full configuration of the hand, i.e. the finger joint angles. Instead of taking a 2D depth image as an input, they use a 3D occupancy voxel grid as input for their neural network. In this voxel grid, each voxel is encoded binary: having a value of one, if a point lies in the voxel and zero otherwise. In the first part of the network, the authors use 3D convolutional layers to reduce the size of the voxel grid. In the second part of the network, the voxel grid is flattened to a one-dimensional vector and processed by several fully connected layers, finally resulting in the joint configuration of the hand.

Generative approaches can be powerful for predicting grasping poses, as the underlying structure of the neural network is straightforward. However, the models rely on direct regression from the input to the desired output. In an effort to predict grasping poses for non-humanoid robot arms equipped with grippers, Redmon and Angelova (2015) show that direct regression approaches can suffer from inaccuracies when multiple grasp candidates are possible for a given input. A regression model tries to minimize the error between the predicted output and the ground truth value. If multiple ground truth values are present, the regression minimizes the average error to all ground truth labels. This results in a prediction that averages over all presented labels. If this average is a valid label then this is not a problem, however if the average of all ground truth labels is not included in the set of valid outputs this averaging can lead to incorrect predictions, as the three examples show in Figure 2.13. The correct grasp poses are displayed as red and blue rectangles in the figure. In case of the plate on the left all these correct grasp poses lie on the border of the plate. When a regression model is trained, using these ground truth labels, the resulting average output of the regression model is the rectangle drawn with green and yellow lines. This predicted grasp pose minimizes the average

error to all ground truth poses, but is in itself not a valid grasp pose.

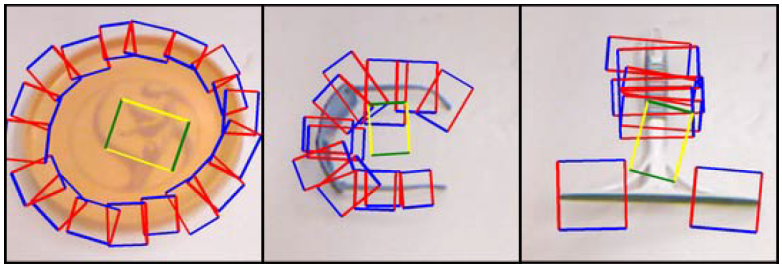


Figure 2.13: Averaging effect in direct regression approaches, (taken from Redmon and Angelova, 2015, © 2015 IEEE). The red and blue rectangles represent the ground truth, while the predicted grasp pose is shown as a green and yellow rectangle.

2.2.5 Shape Completion Approaches

When an object is observed by a camera, only the front of the object is visible. The point cloud calculated from the camera image is therefore always a partial representation of the object. The idea of shape completion is to estimate the shape of the object in occluded areas, i.e. the back, the bottom and the sides. When the model is completed, a traditional grasp planner can be used to plan grasps.

Figure 2.14: Shape completion based on object symmetries by Schiebener et al. (2016), © 2016 IEEE.

An approach by Bohg et al. (2011) performs shape completion by exploiting object symmetries. A set of symmetry plane hypotheses are generated and the most likely symmetry plane is chosen. Then all points are mirrored at the symmetry plane. In a later publication, Schiebener et al. (2016) extend this approach by addressing the remaining gaps after mirroring. To fill in the remaining gaps in the point cloud, additional points at the estimated supporting surface of the

object are added and the sides are filled in, using linear interpolation. An exemplary shape completion of a spray bottle can be seen in Figure 2.14. Using the completed point cloud the object's surface can be triangulated to form a surface mesh. A grasp planner is used to plan grasps, based on the completed surface mesh.

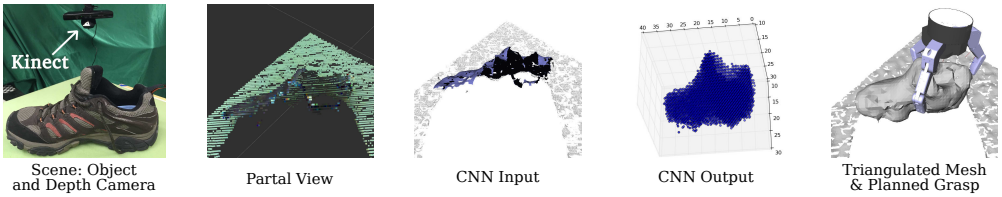


Figure 2.15: Shape completion using CNNs by Varley et al. (2017), © 2017 IEEE.

Another approach by Varley et al. (2017) utilizes deep learning for shape completion. An object is observed by a depth camera, resulting in a partial observation of the objects, encoded as a point cloud. This point cloud is then converted to an occupancy voxel grid, where each voxel has a binary value of one, if the voxel contains a point or zero if the voxel is empty. This voxel grid is the input to a 3D CNN. The output of the CNN is the completed shape, encoded as an occupancy voxel grid, which is transformed to a mesh using triangulation. Finally, grasps can be planned using the completed mesh, as shown in Figure 2.15.

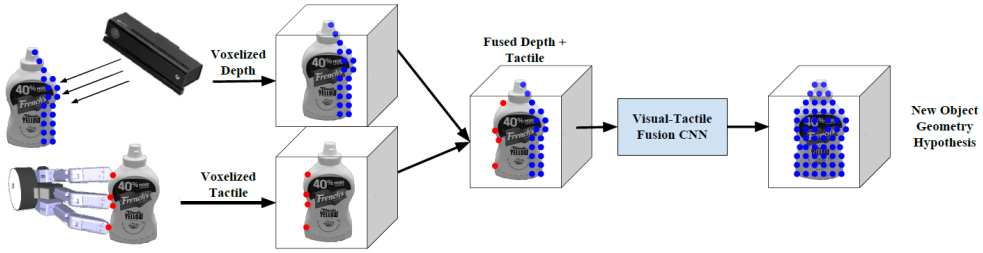


Figure 2.16: Shape completion, combining visual and tactile information by Varley et al. (2018), © 2019 IEEE.

In a later work, Varley et al. (2018) extend their method to include tactile information. They present a method that combines depth data from a Kinect camera with tactile information. The depth data from the camera is converted to a point cloud and fused with the tactile points, as shown in Figure 2.16. This combined point cloud is presented to a 3D CNN that used the same network architecture, as in the previous work. In the offline training phase, the network is presented with depth and tactile data and trained to predict the full model of the object. During online runtime, the network is tasked with shape completion, based on

an incomplete point cloud extended with tactile data. The authors argue that the inclusion of tactile data increases the accuracy of the completed shapes significantly.

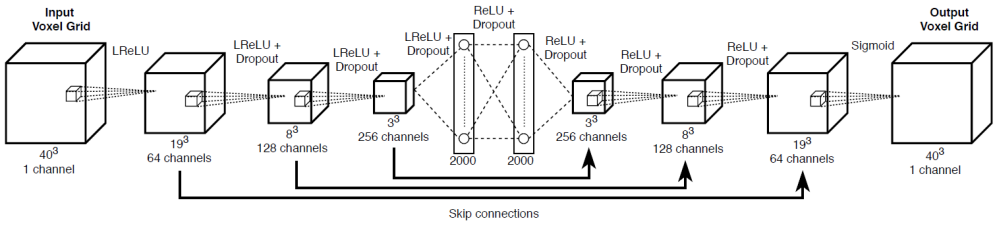


Figure 2.17: Shape completion with uncertainty by Lundell et al. (2019), © 2019 IEEE.

In a recent work, Lundell et al. (2019) extend the work by Varley et al. (2017) to incorporate shape uncertainty. They train a deep neural network to take a partial view of an object as input and output the completed shape of the object, encoded as a voxel grid, see Figure 2.17. The neural network takes an occupancy voxel grid as input. Over several convolutions, the size of the 3D grid is reduced, while the number of features is increased. In the inner most layers of the network all information is combined using a fully connected layer. Then 3D deconvolutions are applied to reduce the number of features and increase the size of the grid, until the dimensions of the input grid are matched. The main novelty of the approach is to include dropout layers in the network, which are enabled during training and during runtime to generate a set of shape candidates, representing the shape uncertainty. They generate grasp candidates on the mean of all shape candidates. Each grasp candidate is then evaluated on all shape candidates in terms of analytical grasp metrics. In their work, the authors show that the inclusion of shape uncertainty during grasp planning increases the grasp success rate, especially when dealing with unknown objects.

2.2.6 Heat Map Approaches

Another approach for detecting grasp candidates in images is to transfer the problem completely to an image-to-image mapping problem. Following this idea, Nguyen et al. (2016) use a network architecture that takes the depth image of a scene as an input and outputs several affordances, encoded as different images.

The authors emphasize the importance of encoding the input and output of the neural network in order to enable the network to train well. The input RGB-D

image is split into six channels. Three channels contain the red, green and blue values of the image. The depth channel is split into three channels, where one contains the horizontal distance to the camera, the second channel contains the height above the ground and the third channel encodes the angle between the surface normal and the gravity vector.

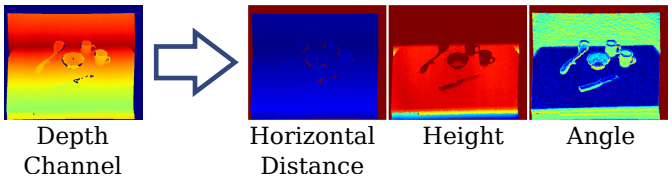


Figure 2.18: Encoding of the depth channel by Nguyen et al. (2016), © 2016 IEEE.

The output of the network are heat map images, where each image corresponds to one affordance. In regions where the affordance is present, the heat map contains a large value and in regions where the affordance is not present, the corresponding heat map has a low value. The authors use a fully convolutional encoder-decoder-network (see Figure 2.19) that was first introduced by Noh et al. (2015).

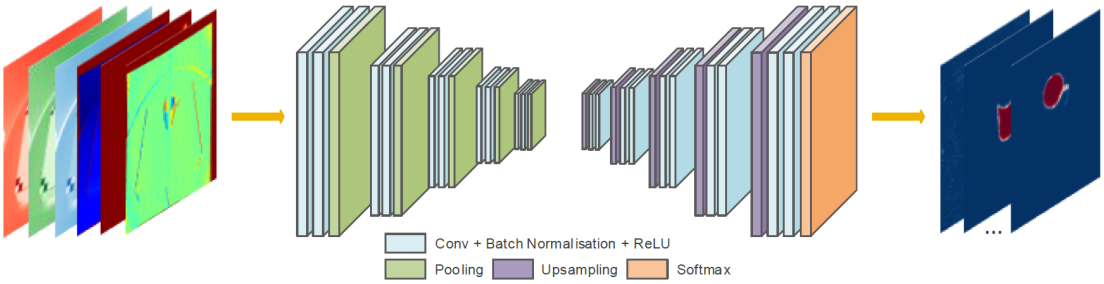


Figure 2.19: Fully convolutional encoder-decoder-network architecture by Nguyen et al. (2016), © 2016 IEEE. From left to right: perceived scene, heat map of the grasp affordance, extracted grasp rectangle.

Using this architecture enables the network to predict multiple affordances at the same time and to predict the occurrence of one affordance multiple times in one image. Examples for detecting the grasp affordance in different scenes are shown in Figure 2.20. In the left row, the scenes are shown, while the center row denotes the heat map of the grasp affordance. Using classical image processing techniques the authors then detect rectangles in the heat map image and use these rectangles as grasp candidates.

In a later work, Nguyen et al. (2017) replace this rectangle description of grasp candidates. They first transform all points matching to one occurrence of a

Figure 2.20: Encoding of grasp affordances as heat maps by Nguyen et al. (2016), © 2016 IEEE.

grasp affordance from 2D to 3D using a calibrated depth camera. Then they perform a principal component analysis on the 3D points, extracting the major axis of the grasp affordance. The hand of the robot is then aligned with this axis during grasp execution.

One major advantage of using heat maps over generative approaches is that heat maps allow the encoding of multiple occurrences of grasps in one image. Therefore, heat map based approaches overcome the limitation of averaging that is present in generative approaches. However, heat map based approaches are similar to pixelwise segmentation approaches, where a class is assigned to each pixel. These techniques operate solely in the 2D image plane. In order to extract grasp candidates from the segmented images, further post processing steps are necessary.

2.2.7 Discriminative Approaches

The idea behind discriminative approaches is not to generate grasp candidates directly from the available sensor input, but to rate grasp candidates. In a way, a discriminative approach can be seen as a data-driven grasp metric that operates on the available sensor data. For each grasp candidate the discriminator is evaluated and produces a predicted grasp success probability.

To the best of the author's knowledge, discriminative approaches have not yet been applied to humanoid grasping. The method will therefore be presented at

the example of a robot equipped with a gripper. Thereafter, the transferability to humanoid robots will be outlined.

In their work, Mahler et al. (2017) use a data-driven discriminator to rate possible grasp poses, given a depth image of a scene containing unknown objects. This discriminator is implemented as a neural network and called „Grasp Quality Convolutional Neural Network“. The discriminator predicts the probability of grasp success from depth images. Grasp candidates are encoded as planar position, angle, and depth of a gripper relative to the depth camera, mounted directly above the scene. The experimental setup is shown in Figure 2.21.

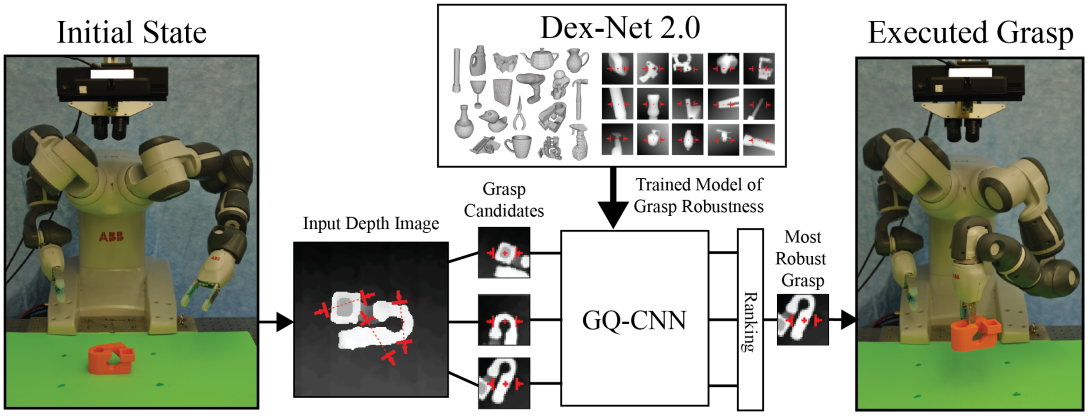


Figure 2.21: Discriminative approach for grasp candidate rating by Mahler et al. (2017).

When a new object is presented to the system, first a depth image of the object is captured from above. Then multiple grasp candidates are generated using antipodal candidate grasps, first proposed by Chen and Burdick (1993). The grasp quality network rates each of the grasp candidates. Finally, the grasp with the highest assigned score is selected for execution.

The authors state that training the grasp quality network requires a „huge number of samples“. They solve this issue by generating the required training data in simulation, rendering depth images of many objects and associating the corresponding analytical grasp scores to grasp candidates within the depth images.

The main advantage of discriminative approaches is the ability to rate multiple grasp candidates within one scene and predict the grasp success probability for each grasp candidate. The challenge in applying a discriminative approach to humanoid robots is that the problem definition of humanoid robotic grasping and grasping with a gripper differs significantly. First, the camera setup is different. In the humanoid case, the camera is not located above the scene, but is

in the head of the robot, observing the objects from a varying view point. Second, the grasp candidate generation technique used by Mahler et al. (2017) is specific for grasping objects from the top, using a parallel jaw gripper and cannot be applied directly to humanoid robots. Therefore, a new grasp candidate generation method has to be developed.

2.2.8 Discussion

This section presented different approaches for data-driven grasping with humanoid robots.

The data-driven grasp generation can be divided into four categories, listed in Table 2.3. The table lists the advantages and disadvantages for each approach. Early approaches focus on direct generation of the best grasp candidate, i.e. they use a learned model that takes an image as an input and generates directly the best grasp candidate. As shown by Redmon and Angelova (2015) this direct generation of grasp candidates can lead to averaging effects, if multiple correct grasp candidates exist for one given input. Other approaches opted to not generate grasp candidates, but to estimate the shape of the object to be grasped by means of shape completion. The learned model takes a partial view of the target object as input and outputs a completed 3D shape, often in the form of a voxel grid. Then a conventional gasp planner generates grasp candidates based on this completed shape. The advantages of shape completion are that existing grasp planners can be reused and that the learned model can be transferred to any robot and hand, if a grasp planner exists for that hand. The main disadvantage is that it is unclear, which of the generated grasp candidates has the highest success rate. Another category are heat map based approaches, where the learning problem is first transferred to an image-to-image mapping problem. This is advantageous, since a lot of research has already been done in the image-to-image learning domain and therefore this knowledge can be used. To this end, first possible grasp candidates are encoded as a 2D heat map, where each pixel denotes if a grasp at that location will be successful or not. The main disadvantage of this approach is that it is currently limited to 2D images. This induces a constraint to the grasp execution where any grasp attempt can only be performed orthogonally to the image plane. Therefore, these kind of approaches can only work on problems that are representable in 2D, but not for full 6D grasp generation, which is necessary for multi fingered, humanoid hands. The last group follows a discriminative approach. Here grasp candidates are not generated directly, but an external candidate generator is

used. The learned model takes a grasp candidate and the perceptual object information as input and predicts the grasp success probability, also called grasp robustness. The advantage of these approaches is that the perception of the object is not constrained to a certain encoding and can be multi-model, allowing e.g. fusion of visual and tactile data. The main disadvantage is that an external grasp candidate generator is necessary.

Method	**Advantage**	**Disadvantage**
Generative	Simple structure	Averaging
Shape completion	Use existing grasp planners	Possible incorrect rating
Heat map	Multiple candidates	Post processing, 2D only
Discriminative	Multiple, rated candidates	Grasp candidate generator required

Table 2.3: Comparison of different grasp candidate generation techniques.

Most of the models used in data-driven grasping are implemented as neural networks. These neural networks require large amounts of raining data. The different possible sources are listed in Table 2.4. The first possible source of training data is to use human labor. A data set of good grasp candidates is created by hand labeling thousands of training examples. These training examples mostly come in the form of images, where grasp candidates are drawn into these images. The advantage of hand labeled data sets is that the quality of the training data is high, since the human perception and reasoning is still far superior to any automated grasp planning approach. The main disadvantage is that this labeling is constrained to 2D image problems. So far, no data set was created aiming at full 6D grasp poses, necessary for humanoid hands. The second option is to generate training data directly on the target system. The robot tries to execute different grasps and determines if the grasp execution was successful or not. This kind of training is called self-supervised, as no human labeling of grasping results is necessary. Levine et al. (2018) demonstrated that learning grasping on the target system is possible and can lead to good grasping results. However, the amount of training samples necessary is still a limiting factor. The robot or multiple robots have to execute thousands of grasp attempts to acquire enough training data. Another approach is to learn grasping from human demonstrations. Here, a human teacher demonstrates how to grasp different objects. The robot observes the grasp execution and builds a model that aims to mimic the human. The advantage of learning from demonstration is that the

training data is of high quality. However, the cost in terms of human labor are high, as many successful grasps have to be recorded to gather enough training data. Furthermore, learning from demonstration poses a challenging domain transfer problem, as the grasping demonstrations have to be mapped from the human to the robot. The last source of training data is to use a simulator. Here, the robot is placed into a virtual environment, where grasping is simulated. The simulator evaluates grasp success based on the geometry of the robot's and the geometry of the target object. The advantage is that the time the robot spends in the simulator comes at almost no cost, i.e. large amounts of training data can be generated. The main disadvantage is that perfect simulators do not exist, especially when it comes to contact models needed for grasping. Therefore, the transfer of the learned model from the simulator to a real robot can be challenging.

Data source	**Cost**	**Domain**
Hand labeled	High	Same
Target system	High	Same
Learning from demonstration	High	Different
Simulation	Low	Different

Table 2.4: Comparison of different training data sources.

3 Next-Best-Touch for Grasping

As humanoid robots move from the laboratory to real world environments, they need to be able to grasp unknown objects. Among others, grasping requires an understanding of the geometric shape of such objects. In order to synthesize grasp candidates using grasp planners, a detailed surface model is required, but not available for unknown objects. While visual information might be the most obvious sensor modality to acquire this surface model, vision based systems can be impaired by either the object properties (e.g. reflecting, translucent, uniformly colored) or the environmental conditions (e.g. poor lighting, smoke, fog, and bright sunshine). To overcome these shortcomings, tactile sensing can be used to gather additional information, which also plays an important role in human grasping, as Johansson and Flanagan (2009) show in their work. In order to enable the robot to grasp unknown objects using tactile exploration four problems have to be solved:

1. How to collect contact information by efficiently selecting the next best touch?
2. How to efficiently generate object shape models based on the acquired sparse tactile data?
3. How to gather as much information per contact as possible?
4. How to plan grasps based on the approximate object model?

This chapter addresses all four challenges. First, section 3.1 outlines the approach of the overall exploration algorithm. Thereafter, section 3.2 presents a data efficient surface model that can operate on sparse and incomplete tactile data. Then section 3.3 explain how the next-best-touch is selected using the estimated surface model. Section 3.4 presents a sensor that enables the robot to not only sense contact with the object, but to also sense the local surface orientation, thus increasing the information per contact. Finally, section 3.5 introduces the chosen grasp planner to synthesize grasps based on the estimated surface. The proposed exploration method is evaluated in simulation and compared against the state of the art in section 3.6. The exploration method, tactile sensor, surface

model and grasp planning approach are then combined and validated in robot experiments.

3.1 Exploration Strategy for the Next-Best-Touch

Haptic exploration of an object can yield detailed surface information. However, acquiring this information is costly, since the robot hand and arm have to be moved after each contact to a new location. Selecting the location of the next-best-touch is therefore an important part of the exploration process. Previous work in this area has mostly focused on selecting this location to yield the most new information per touch action. After each exploration action, the surface of the object is estimated. Thereafter, the location on the surface with the highest estimation uncertainty is determined and selected as the next exploration target. However, this can lead to large travel distances that the robot arm has to cover. This will be illustrated at an imaginary exploration of a 2D plane, that follows the procedure described in Algorithm 1. In this example, the exploration region $\mathbf{A}$ is limited to a $5\,\mathrm{cm}$ by $5\,\mathrm{cm}$ box.

Algorithm 1 Example exploration

1: $\boldsymbol{t} = \mathit{centroid}(\mathbf{A})$ ▷ initial target is in the center
2: **loop**
3: explore at $\boldsymbol{t}$
4: $t = \mathrm{argmax}_{\boldsymbol{x}\in\mathbf{A}}\, \mathit{RateTarget}(\boldsymbol{x})$
5: **end loop**

The first target is predefined and set to be in the center of the exploration area. Then the robot moves to the target and explores the surface of the plane at the target. Thereafter, each possible target $\boldsymbol{x}$ within $\mathbf{A}$ is rated, according to the exploration strategy. The exploration process is executed two times. Once using an exploration strategy that focuses on maximizing the information gain. In the second run, the exploration strategy balances maximizing information gain and minimizing path cost. In the following, the first strategy will be referred to as *greedy* and the second strategy will be labeled *balanced*. Possible results of the two strategies are compared in Figure 3.1. Every five additionally explored points a snapshot is taken. In each snapshot, the explored points are displayed as dots and the covered distance connecting the dots is shown as lines. The background of each snapshot depicts the uncertainty where blue corresponds to low uncertainty and yellow corresponds to high uncertainty. In this example, the uncertainty is calculated to be the distance to the closest explored point.

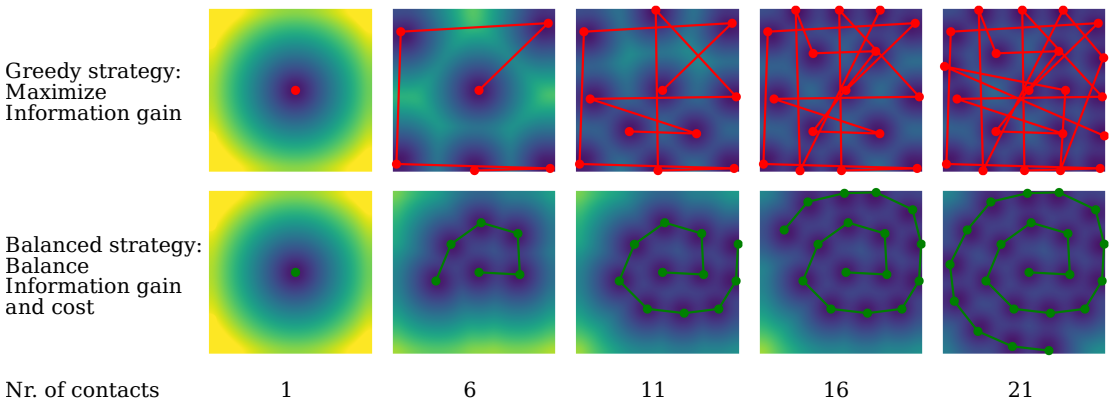


Figure 3.1: Comparison of pure information gain maximization and balancing information gain with cost.

Both exploration procedures start in the center of the exploration area. After the initial contact, the *greedy* strategy jumps into one of the corners of $\boldsymbol{A}$. This is expected, since the points in the corners are furthest away from the already explored point in the center and therefore have the highest uncertainty. After the first corner is explored, the other corners follow next. Then, after all corners have been explored, the *greedy* strategy starts filling in the remaining gaps, see top row of Figure 3.1. The *balanced* strategy starts by selecting the second target close to the initial central point. Then the exploration follows an outward spiral.Visually the second, *balanced* strategy achieves a similar coverage of the exploration area while being more efficient in terms of travel cost. To compare the two strategies the overall travel cost in terms of path length is plotted in Figure 3.2. The cost for exploring the initial point in the center is not included

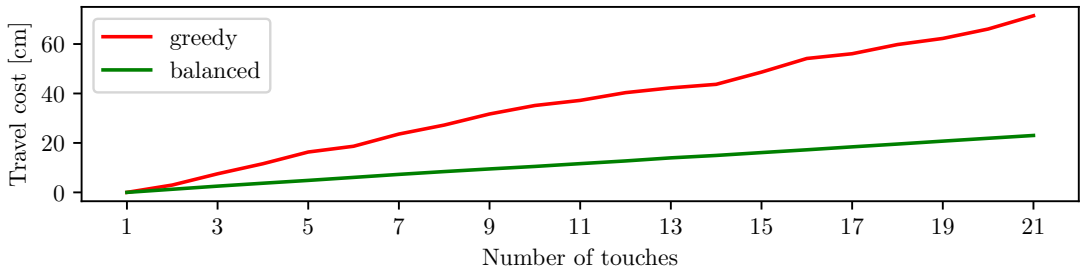


Figure 3.2: Path costs in relation to the number of touches.

in the chart. Therefore, both strategies start with one contact point in the center and zero travel cost. The lines in the chart correspond to the so far, overall covered distance of the respective strategy, for each contact point. The red line in the chart denotes the distance covered of the *greedy* strategy and the green line corresponds to the *balanced strategy*. In this example, the *balanced strategy*

clearly travels less distance per point.

In order to quantify the exploration coverage, a second plot is derived from the exploration progress in Figure 3.3. In this plot, for each point $\boldsymbol{x}$ in the exploration area $\boldsymbol{A}$, the distance to the closest already explored point from the set of all explored points $\mathcal{C}$ is calculated. Then the average over these distances is calculated, according to Equation 3.1.

$$D = \frac{1}{\|\boldsymbol{A}\|} \sum_{\boldsymbol{x} \in \boldsymbol{A}} \min_{\boldsymbol{c} \in \mathcal{C}} \|\boldsymbol{x} - \boldsymbol{c}\| \tag{3.1}$$

In Figure 3.3 this average distance to the closest explored point is plotted for each touched point on the surface. This plot shows that the greedy exploration

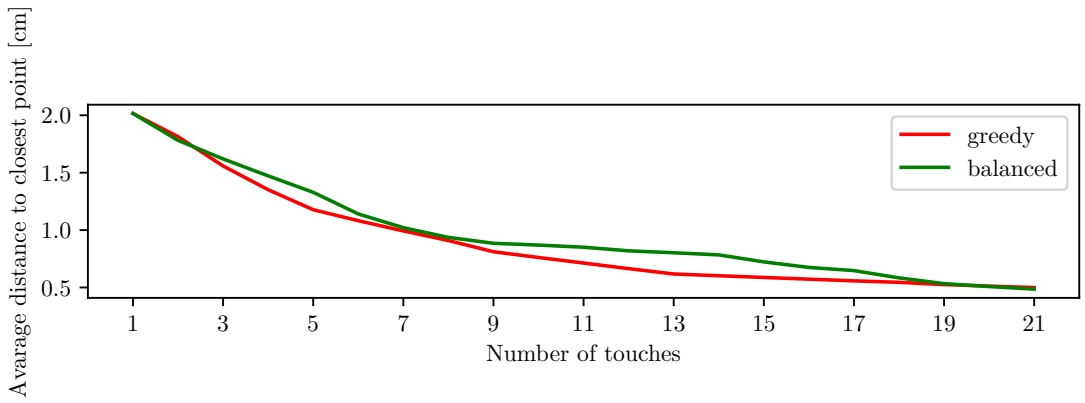


Figure 3.3: Comparison of exploration progress measured by the average distance to the next explored point in terms of the number of touches.

and the balanced strategy explore the plane at a comparable rate per contact, showing that taking small steps can be beneficial in the long run.

Considering these initial observations in the following an algorithm will be developed that follows the design idea of balancing information gain with travel cost, while the baseline strategy will follow a cost agnostic, greedy strategy.

3.1.1 Exploration Algorithm

The exploration strategy is developed in a simulated environment, where the object to be explored is present in an otherwise empty simulation environment. A single robotic fingertip that can freely move in the 3D space performs the exploration. At the start of the simulation, no information about the object is given to the exploration algorithm, with the exception of an initial touch target.

Algorithm 2 Algorithm of the proposed haptic exploration strategy

```
1:  procedure HAPTICEXPLORATION(t_0, strategy)
2:      c_0 ← INITIALIZE(t_0)
3:      C ← {c_0}
4:      i ← 1
5:      loop
6:          S ← ESTIMATESURFACE(C)
7:          t ← RATEANDSELECTTARGET(S, C, strategy)
8:          {T_p(τ), T_R(τ)} ← GENERATETRAJECTORY(t)
9:          c_i ← FOLLOWTRAJECTORY(T_p(τ), T_R(τ))
10:         C ← C ∪ {c_i}                    ▷ add new contact to contact set
11:         i ← i + 1
12:     end loop
13: end procedure
```

This initial target either can be given manually by the operator or can be determined based on visual information. In the context of the simulated exploration, the initial exploration target is chosen to be the center of mass of the object. The robotic fingertip moves towards this initial target, until contact occurs. One this contact occurs, an initial surface can be estimated based on the acquired contact and the exploration process is started.

The exploration procedure can be briefly summarized as follows:

1. Establish an **initial contact** with the object.
2. Add the contact to the set of explored points and **update the surface estimation**.
3. Rate each point on the estimated surface and choose the point with the highest rating as the **next target**.
4. Generate an **exploratory action** to reach the new target.
5. **Move the fingertip** along the trajectory until contact occurs.
6. Go to step 2.

The algorithm that implements this exploration procedure is described in detail in Algorithm 2 while all used symbols are listed in Table 3.1. The fingertip is controlled in velocity mode using a feed forward velocity controller that follows the generated trajectory.

INITIAL CONTACT The exploration algorithm is initialized by setting the start position of the fingertip and the location of the initial target. The initial approach velocity $\boldsymbol{v}_t$ of the fingertip is calculated to approach the initial target,

procedure INITIALIZE($\boldsymbol{t}_0$)
  $\boldsymbol{r} \leftarrow$ *start position*
  $\boldsymbol{t}_p \leftarrow$ *initial target*
  $\boldsymbol{v}_n \leftarrow (\boldsymbol{t}_p - \boldsymbol{r})\|\boldsymbol{t}_p - \boldsymbol{r}\|^{-1}$ ▷ initial approach velocity
  setVelocity($\boldsymbol{v}_n v_0$)
  waitUntilInitialContact()
  $\boldsymbol{c}_{0,p} \leftarrow$ *contact position*
  $\boldsymbol{c}_{0,n} \leftarrow$ *contact normal*
  return $\{\boldsymbol{c}_{0,p}, \boldsymbol{c}_{0,n}\}$
end procedure

procedure ESTIMATESURFACE($\mathcal{C}$)
  $\mathcal{S} \leftarrow$ *GPIS*($\mathcal{C}$)
  return $\mathcal{S}$
end procedure

procedure RATEANDSELECTTARGET($\mathcal{S}, \mathcal{C}, strategy$)
  if strategy **is** GP variance **then**
    $\forall \boldsymbol{s} \in \mathcal{S} : I(\boldsymbol{s}) \leftarrow$ *GP variance*($\boldsymbol{s}$)
  end if
  if strategy **is** Information Gain Estimation Function **then**
    $I(\mathcal{S}) \leftarrow$ *Information Gain Estimation Function*($\mathcal{S}$) ▷ see Algorithm 3
  end if
  $\boldsymbol{t} \leftarrow \mathrm{argmax}_{\boldsymbol{s}}\, I(\boldsymbol{s})$
  return $\boldsymbol{t}$
end procedure

procedure GENERATETRAJECTORY($\boldsymbol{t}$)
  $\boldsymbol{v}_n \leftarrow$ *normalized velocity at contact detection*
  $\boldsymbol{R}_0 \leftarrow$ *current orientation*
  $\boldsymbol{R}_1 \leftarrow \mathrm{getRotation}(\boldsymbol{R}_0, \boldsymbol{t}_n)$
  $\beta \leftarrow 1/3\|\boldsymbol{c}_p - \boldsymbol{t}_p\|$ ▷ control point scaling
  $T_p(\tau) \leftarrow \mathrm{bezier}(\boldsymbol{c}_p, \boldsymbol{c}_p - \beta\boldsymbol{v}_n, \boldsymbol{t}_p + \beta\boldsymbol{t}_n, \boldsymbol{t}_p)$ ▷ trajectory from $\boldsymbol{c}$ to $\boldsymbol{t}$
  $T_R(\tau) \leftarrow \mathrm{lerp}(\boldsymbol{R}_0, \boldsymbol{R}_1)$
  return $\{T_p(\tau), T_R(\tau)\}$
end procedure

procedure FOLLOWTRAJECTORY($T_p(\tau), T_R(\tau)$)
  $\tau \leftarrow 0$
  loop
    $\boldsymbol{r} \leftarrow$ *current position*
    $\boldsymbol{R} \leftarrow$ *current orientation*
    $\tau \leftarrow \text{argmin}_{\tau_\star \in [\tau,1]} \|\boldsymbol{r} - T(\tau_\star)\|$ ▷ find closest point on trajectory
    $v_* \leftarrow \frac{\delta}{\delta\tau} T_p(\tau) + k_{p,pos}(T_p(\tau) - \boldsymbol{r})$
    $v_t \leftarrow v_0 \boldsymbol{v}_* \|\boldsymbol{v}_*\|^{-1}$ ▷ normalize velocity
    setVelocity($\boldsymbol{v}_t$)
    $\boldsymbol{\omega}_t \leftarrow \text{RPY}(\frac{\delta}{\delta\tau} T_R(\tau) + k_{p,ori} \boldsymbol{R}^{-1} T_R(\tau))$ ▷ get angular velocity
    setAngularVelocity($\boldsymbol{\omega}_t$)
    if contact detected **then**
      $\boldsymbol{c}_{i,p} \leftarrow$ *contact position*
      $\boldsymbol{c}_{i,n} \leftarrow$ *contact normal*
      return $\{\boldsymbol{c}_{i,p}, \boldsymbol{c}_{i,n}\}$
    else if $\tau > 1$ **and** $\|\boldsymbol{r} - T(1)\| > \Delta m$ **then** ▷ missed prediction
      $\{T_p(\tau), T_R(\tau)\} \leftarrow$ HANDLENOCONTACT
      $\tau \leftarrow 0$
    end if
  end loop
end procedure

procedure HANDLENOCONTACT
  $\boldsymbol{v}_n \leftarrow$ *current normalized velocity*
  $\beta \leftarrow 1/3 \|\boldsymbol{r} - \boldsymbol{t}_p\|$ ▷ control point scaling
  $T_p(\tau) \leftarrow \text{bezier}(\boldsymbol{r}, \boldsymbol{r} + \beta \boldsymbol{v}_n, \boldsymbol{t}_p - \beta \boldsymbol{t}_n, \boldsymbol{t}_p)$ ▷ trajectory from $\boldsymbol{r}$ to $\boldsymbol{t}$
  $\boldsymbol{R}_0 \leftarrow$ *current orientation*
  $\boldsymbol{R}_1 \leftarrow \text{getRotation}(\boldsymbol{R}_0, -\boldsymbol{t}_n)$ ▷ inverted target approach
  $T_R(\tau) \leftarrow \text{lerp}(\boldsymbol{R}_0, \boldsymbol{R}_1)$
  return $\{T_p(\tau), T_R(\tau)\}$
end procedure

Function / Symbol	Description
$\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{\omega} \in \mathbb{R}^3$	End-effector position, linear velocity and angular velocity
$\boldsymbol{R} \in$ SO(3)	End-effector orientation
$T_p(\tau) : \mathbb{R} \to \mathbb{R}^3$	Trajectory position function
$T_R(\tau) : \mathbb{R} \to$ SO(3)	Trajectory orientation function
$\tau \in [0, 1]$	Argument of the trajectory function $T(\tau = 0)$: trajectory start, $T(\tau = 1)$: trajectory end
$\mathcal{C}$	Set of all contacts
$\boldsymbol{c}_p, \boldsymbol{c}_n \in \mathbb{R}^3$	Position and normal of a contact in $\mathcal{C}$
$\mathcal{S}$	Surface estimation
$\boldsymbol{s}_p, \boldsymbol{s}_n \in \mathbb{R}^3$	Position and normal of a point on $\mathcal{S}$
$\boldsymbol{t}_p, \boldsymbol{t}_n \in \mathbb{R}^3$	Position and normal of the selected target on $\mathcal{S}$
$\Delta m \in \mathbb{R}$	Distance that defines when a target is considered missed
$\sigma_1, \sigma_3, \mu_3, \sigma_\alpha \in \mathbb{R}$	Tuning parameters
$v_t \in \mathbb{R}$	Desired velocity of the end-effector

Table 3.1: Summary of used symbols

and then the velocity is passed to the underlying velocity controller. The initial target is approached until the initial contact occurs. Thereafter, the main exploration loop is started.

SURFACE UPDATE The first step within the exploration loop is to handle the new contact. The contact position is derived from the current position of the fingertip and the contact normal is derived from the local surface orientation of the object, see section 3.4. The approach velocity vector is stored for later use. Then the motion of the fingertip is stopped by setting the velocity to zero. The current contact consisting of the contact position $\boldsymbol{c}_p$ and the contact normal $\boldsymbol{c}_n$ is added to the set of all explored points $\mathcal{C}$. The surface estimate of the object is updated using an extended formulation Gaussian process implicit surfaces, that includes surface normals, see section 3.2.

TARGET SELECTION To choose the next target the developed algorithm can support different target rating strategies, that work on a per point basis. The rating function is evaluated for all points on the estimated surface. Here two different strategies are used and compared later:

1. Variance of the Gaussian process that is the basis of the surface estimation. This variance is a measure, how uncertain the surface estimate is at the query point.
2. The newly developed Information Gain Estimation Function, that bal-

ances information gain and path cost.

After each point of the current surface estimate has been rated by the chosen rating function, the target point with the highest assigned score is selected for exploration.

TRAJECTORY GENERATION For trajectory generation from the current location $\boldsymbol{r}$ to the selected target $\boldsymbol{t}_p$ several requirements have to be considered:

1. The trajectory should start at the current location of the fingertip $\boldsymbol{r}$.
2. The trajectory should end at the planned target.
3. The initial direction of the trajectory should be inverse to the velocity vector just before the contact occurred.
4. The trajectory should approach the selected target in the direction of the estimated normal $\boldsymbol{t}_n$ at the selected target position $\boldsymbol{t}_p$.
5. The trajectory should be smooth, i.e. differentiable.

A simple, yet powerful formulation that meets all these criteria are cubic Bézier curves using four control points. By setting the first control point to be equal to the current position and the last control point to match the target position the first two requirements are already met. How the inner control points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are derived as well as detailed description of the trajectory generation, including orientation, can be found in subsection 3.3.3. The result of the trajectory generation are two trajectory functions: $T_p(\tau)$ encodes the trajectory in 3D Cartesian space and $T_R(\tau)$ denotes the desired 3D orientation at each position.

FINGERTIP CONTROL The motion of the fingertip is controlled using a feed forward velocity controller that follows the desired trajectory and compensates any execution inaccuracies. Besides the initial approach, the FOLLOWTRAJECTORY procedure is the only part of the exploration algorithm where the fingertip is moved. To start the controller the argument of the trajectory τ is initialized to $\mathbf{0}$. Then the control loop is started. In each iteration, first the current position $\boldsymbol{r}$ and current orientation $\boldsymbol{R}$ are queried from the fingertip. The trajectory argument τ is updated by determining the closest point on the trajectory to the current position $\boldsymbol{r}$, so that τ increases. The target velocity $\boldsymbol{v}_*$ is determined using a feed forward velocity controller. The feed forward part is determined using the derivative of the trajectory function $\frac{\delta}{\delta\tau}T_p(\tau)$. To correct inaccuracies during execution a secondary P-controller is employed. The combined controller is given by Equation 3.2.

$$\boldsymbol{v}_* = \frac{\delta}{\delta\tau}T_p(\tau) + k_{p,pos}(T_p(\tau) - \boldsymbol{r}) \tag{3.2}$$

To ensure that the velocity magnitude of the fingertip is constant, the target velocity $\boldsymbol{v}_t$ is normalized to match the desired velocity magnitude given by v_0, see Equation 3.3.

$$\boldsymbol{v}_t = v_0 \frac{\boldsymbol{v}_*}{\|\boldsymbol{v}_*\|} \tag{3.3}$$

The orientation of the fingertip is controlled in a similar way using a feed forward P-controller that follows the desired orientation. To this end, the difference between the current orientation $\boldsymbol{R}$ and the desired orientation is calculated and scaled by the control parameter $k_{p,ori}$. the feed forward part is calculated by the derivative of the orientation trajectory $T_R(\tau)$. Finally, the sum of both parts is converted to a roll pitch yaw rotation velocity and applied to the fingertip, as described in Equation 3.4.

$$\omega_t = RPY(\frac{\delta}{\delta\tau} T_R(\tau) + k_{p,ori} \boldsymbol{R}^{-1} T_R(\tau)) \tag{3.4}$$

The controller follows the trajectory until one of two possible events occurs:

A) A contact between the fingertip and the object is detected.

B) No contact is detected and the end of the trajectory is reached.

If a contact is detected, the contact position and normal are determined, the contact is added to the set of explored points $\mathcal{C}$ and the exploration loop starts over by updating the surface estimate based on the new set of contacts.

However, in some cases it is possible that the end of the trajectory is reached without any contact event. The reason is that the target for exploration is chosen based on an estimated surface of the object. Since the goal is exploration of the object the chosen target often lies in a region where the estimated surface is imprecise or incorrect. To compensate this two behaviors are implemented. First, the trajectory is followed further, extending beyond the target point at $\tau = 1$. This helps if the estimated surface is slightly incorrect and the real surface of the object can be reached by extending the trajectory. If the surface estimate is incorrect, this trajectory extension is insufficient to find the target object again. In this case, the procedure **HANDLE NO CONTACT** is called to generate a new trajectory that ensures intersection with the target object.

HANDLE NO CONTACT In the case that the calculated target is reached and no contact event with the object has occurred the fingertip has to be brought back into contact with the object. To this end, a new trajectory is generated that starts at the current position and ends at the last observed contact. However, the target is approached not from the original approach direction, but from the exact opposite direction. This ensures that the trajectory has to pass through

the object at some point. The trajectory is again implemented as a Bézier curve. The effect of this behavior is illustrated in a 2D example in Figure 3.4.

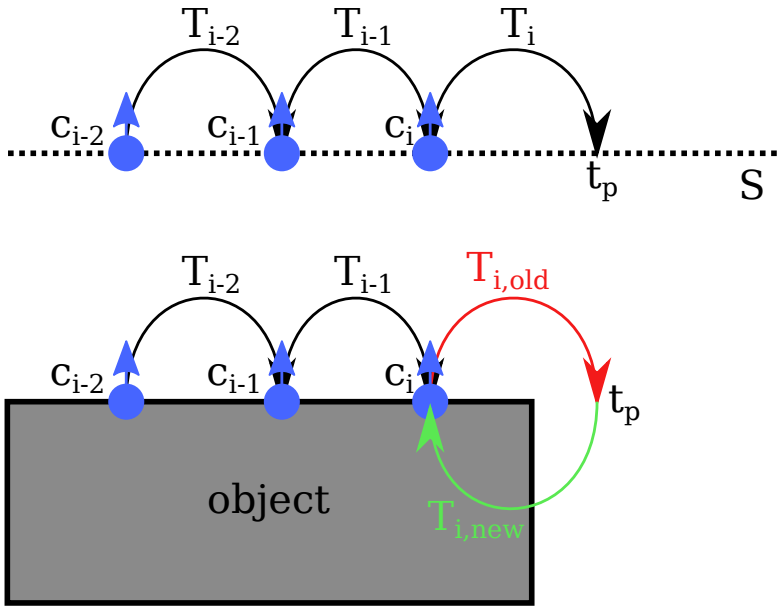


Figure 3.4: Handling of a „no contact event“ at the end of the planned trajectory.

The top row of the image denotes the state of the exploration algorithm. The contacts $\boldsymbol{c}_{i-2}$, $\boldsymbol{c}_{i-1}$ and $\boldsymbol{c}_i$ have been explored. The estimated surface $\mathcal{S}$ resulting from these three contacts is shown as a dotted line. The chosen target position $\boldsymbol{t}_p$ lies right of the last contact. However, the real object has a sharp corner between $\boldsymbol{c}_i$ and $\boldsymbol{t}_p$. Therefore, the trajectory T_i from $\boldsymbol{c}_i$ to $\boldsymbol{t}_p$ will not meet the object. The object is shown in the bottom row of the figure. When the target $\boldsymbol{t}_p$ is reached a new trajectory $T_{i,new}$ is generated, that starts at the current location, which coincides with t_p, and approaches $\boldsymbol{c}_i$ from below. This ensures that the new trajectory intersects the object at some point.

3.2 Data Efficient Surface Model

Touching the object from different sides and at different positions requires the robot arm to move. This movement takes time and therefore the acquisition of many contact points is expensive. During the exploration process, the approximate surface of the object has to be estimated, based on the contact points acquired so far, during the exploration process. When the exploration starts, only few contact points are available. During the process of the exploration, it is possible that the density of observed contact points is inhomogeneous. In

addition, the contact positions are uncertain, since execution and sensor noise exists during exploration on a robotic system.

Therefore, several requirements for the surface model arise.

1. **Sparse data:** The surface model has to be able to generate a surface estimate based on few observed contacts.
2. **Inhomogeneous observations:** Regions of high and low contact observation density can exist. The surface model has to incorporate all contact data, including single observations that are far apart from the rest of the observations.
3. **Sensor noise:** The surface model has to be able to incorporate uncertain observations resulting in contradictory contact data.
4. **Contact normals:** The surface model has to be able to incorporate observed surface normals.

A surface model that meets the first three of the above requirements is Gaussian Process Implicit Surfaces (GPIS), which was first introduced by Williams and Fitzgibbon (2007). Dragiev et al. (2011) first used GPIS in the context of grasping. In the following, the concept of GPIS will be introduced and several examples will illustrate the ability of GPIS to meet the requirements. Thereafter, the GPIS formulation will be extended to include surface normal observations.

3.2.1 Gaussian Process Implicit Surfaces

The basic idea of Gaussian Process Implicit Surfaces (GPIS) is to describe the estimated shape of an object by means of an implicit surface potential (ISP) which in turn is given by a Gaussian process (GP). The ISP function $f(\boldsymbol{x})$ defines for each point in space $\boldsymbol{x}$ if it is on the surface of the object, inside the object or outside of the object.

$$f : \mathbb{R}^d \to \mathbb{R} \begin{cases} = 0, & \boldsymbol{x} \text{ on the surface} \\ < 0, & \boldsymbol{x} \text{ outside} \\ > 0, & \boldsymbol{x} \text{ inside.} \end{cases} \tag{3.5}$$

The estimated surface S can be found by calculating the 0-level set of the ISP:

$$S = \{\boldsymbol{x}, f(\boldsymbol{x}) = 0\} \, . \tag{3.6}$$

Following the original formulation of GPIS by Williams and Fitzgibbon (2007) the estimated ISP f^* is given by a Gaussian process, therefore f^* can be expressed as sum of weighted kernel functions.

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{N} w_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{3.7}$$

In the case of GPIS, radial basis kernels are used. A radial basis kernel is only dependent on the distance between the two arguments $\boldsymbol{x}$ and $\boldsymbol{x}'$, but not on the location of $\boldsymbol{x}$ or $\boldsymbol{x}'$. A Gaussian radial basis function is given in Equation 3.8.

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right) \tag{3.8}$$

The value σ is a scaling value that can be adapted to change the width of the kernel function, as shown in Figure 3.5.

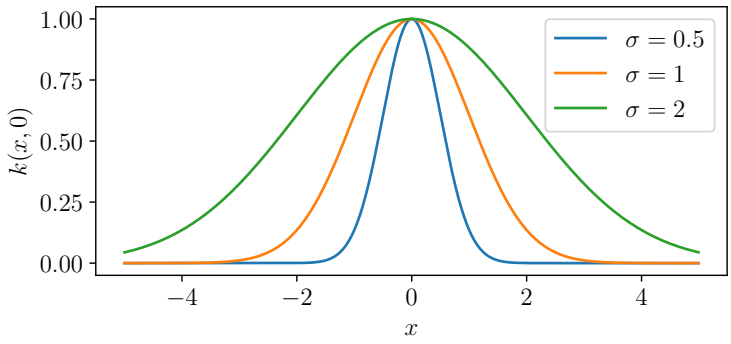


Figure 3.5: Examples of Gaussian radial basis functions with different scaling values σ.

Given a set of observations of $f(\boldsymbol{x})$ in the form of observation location $\boldsymbol{x}_i$ and observation value $f(\boldsymbol{x}) = \boldsymbol{y}_i$ all necessary information is given to determine the weights of the ISP in Equation 3.7.

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{N} w_j \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right) = \sum_{j=1}^{N} w_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{y}_i \quad \forall i \in [1, N] \tag{3.9}$$

Since σ is chosen to be equal in all kernels and the observation locations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are known the values of the kernel functions can be computed. The equation system from Equation 3.9 is written as a linear system, where $\mathbf{K}$ is the symmetric covariance matrix containing the covariances between the samples

i and j given by the kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

$$\mathbf{K}\boldsymbol{\omega} = \boldsymbol{y} \tag{3.10}$$

$$(\mathbf{K})_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.11}$$

Similarly, the ISP function from Equation 3.7 can be written in vector notation using the weight vector $\boldsymbol{\omega}$.

$$f^*(\boldsymbol{x}) = \boldsymbol{k}_*{}^T\boldsymbol{\omega} \tag{3.12}$$

$$(\boldsymbol{k}_*)_i = k(\boldsymbol{x}_i, \boldsymbol{x}) \tag{3.13}$$

By solving the linear system from Equation 3.10 the weight vector $\boldsymbol{\omega}$ can be determined, and in turn, the ISP f can be computed.

$$\boldsymbol{\omega} = \mathbf{K}^{-1}\boldsymbol{y} \tag{3.14}$$

$$f^*(\boldsymbol{x}) = \boldsymbol{k}_*{}^T\mathbf{K}^{-1}\boldsymbol{y} \tag{3.15}$$

This approximate function f is also called the posterior of the Gaussian process. To illustrate the procedure of fitting a Gaussian process to observed values an example is given in Figure 3.6, while the underlying observations are listed in Table 3.2.

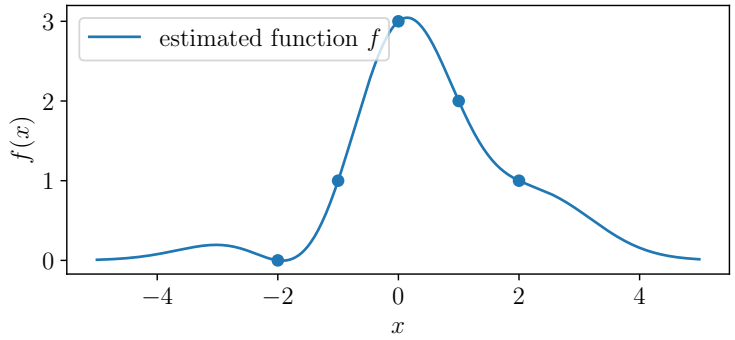


Figure 3.6: Example posterior of a Gaussian process.

i	1	2	3	4	5
x_i	-2	-1	0	1	2
y_i	0	1	3	2	1

Table 3.2: Example observations

If all observed sample locations are separated, this process yields smooth ap-

f_n	$\|\boldsymbol{x}_3 - \boldsymbol{x}_4\|$	ω_1	ω_2	ω_3	ω_4	ω_5
f_1	1	0.75	-2.35	5.1	-1.56	1.28
f_2	0.5	2.21	-6.32	15.38	-10.38	2.36
f_3	0.25	5.2	-15.13	57.84	-48.37	3.8
f_4	0.125	11.09	-33.14	228.07	-208.98	6.53
f_5	0.0625	22.75	-69.24	907.38	-868.77	11.93

Table 3.3: The computed weights get larger when the observed location of sample $\boldsymbol{x}_4$ gets close to another sample $\boldsymbol{x}_3$.

proximations that respect the observations. However, this formulation fails to incorporate contradictory observations. E.g, if two observations with different values are close to each other, but have different observed values the weights $\boldsymbol{\omega}$ get large, leading to invalid ISP estimations. Furthermore, if two observations at the same location yield different values the inverse of $\mathbf{K}^{-1}$ is not defined and $\boldsymbol{\omega}$ cannot be computed.

In an example, the distance between the location of observation $\boldsymbol{x}_4$ and $\boldsymbol{x}_3$ is halved iteratively and for each distance the resulting weights are listed in Table 3.3. The resulting posteriors are given in Figure 3.7.

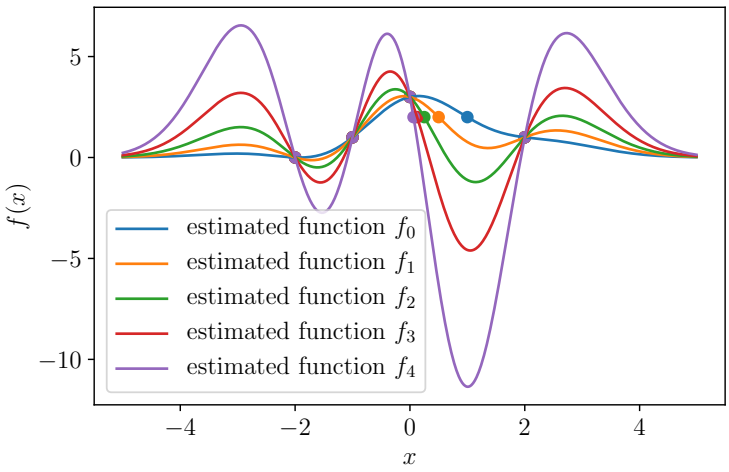


Figure 3.7: Example posteriors for contradictory observations. The sample point at 1 is gradually shifted closer to the observed value at 0. The resulting posteriors assume larger values the closer the contradictory observations come to each other.

To overcome this limitation an observation uncertainty can be added to the covariance matrix along the diagonal axis. The posterior of the Gaussian process

with observation uncertainty is given by Equation 3.16.

$$f^*(\boldsymbol{x}) = \boldsymbol{k}_*{}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\boldsymbol{y} \tag{3.16}$$

Using this formulation allows the Gaussian process to handle the contradicting observations $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$, as shown in Figure 3.8 for different observation noise values. When using small values for σ_n the posterior follows the observations closely, however the shape of the resulting function overshoots between the observations. When using larger values for σ_n the overshoot is mitigated but the resulting function does not pass through the observed values.

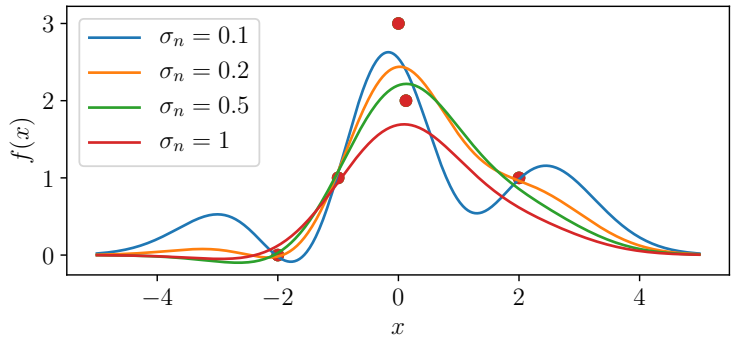


Figure 3.8: Example posteriors with different observation noise for contradictory observations $y_3 = 2$ and $y_4 = 3$ close to $x = 0$.

The Gaussian process formulation from Equation 3.16 can now be used as a basis for the implicit surface potential. To define meaningful observation locations and values to define the ISP Williams and Fitzgibbon (2007) propose to add 0-value observations on the surface of the object and to include additional observations inside and outside of the object's surface:

- At each surface observation, a sample is generated with value zero.
- A sample inside of the object is added with value 1.
- Multiple samples are added outside of the object with value -1.

To illustrate the surface approximation capabilities of GPIS several examples are given below. In these examples, the center point is calculated as the mean value of all contact positions. Several outside points are placed in a ring around the center point. The first example in Figure 3.9 shows that GPIS can deal with sparse contact data while the second example in Figure 3.10 displays inhomogeneous contact density and contradictory observations.

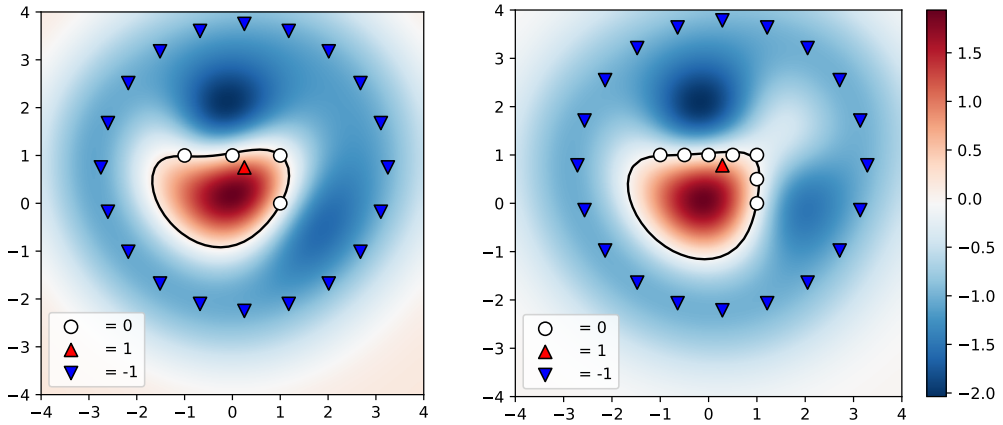


Figure 3.9: GPIS can deal with sparse contacts (left) and dense contacts (right).

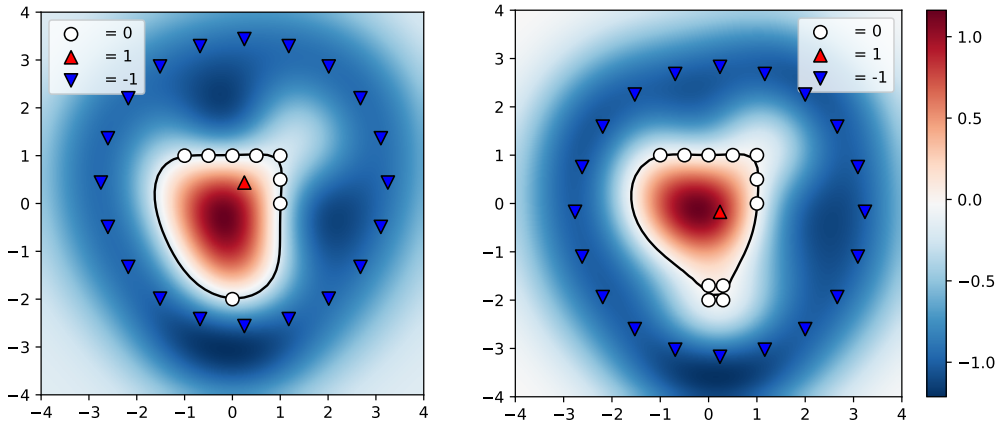


Figure 3.10: GPIS can deal with inhomogeneous contact density (left) and contradictory observations (right).

3.2.2 Extension of GPIS to Include Surface Normals

The original formulation of GPIS by Williams and Fitzgibbon (2007) does not include surface normal observations. However, these surface normals can be of great benefit for the surface estimate, as Martens et al. (2017) and Dragiev et al. (2011) have indicated in their work.

Therefore, an extended formulation of GPIS is required, that can incorporate surface position and surface normal observations. For each contact $\boldsymbol{x}_i$ with the object, three observations can be made:

- The position of the contact $\boldsymbol{x}_i$.
- The local surface orientation, encoded as the surface normal $\boldsymbol{n}_i$.
- The desired ISP value $\boldsymbol{y}_i$.

Since the value of the ISP should be zero on the surface of the object, the desired ISP at each $\boldsymbol{x}_i$ should be zero:

$$f(\boldsymbol{x}_i) = 0 \,. \tag{3.17}$$

The local surface orientation can be incorporated by constraining the derivative of the ISP:

$$\nabla f(\boldsymbol{x}_i) = \boldsymbol{n}_i = (n_{i,1}, n_{i,2}, n_{i,3})^T \,. \tag{3.18}$$

In the original formulation of GPIS, it is necessary to constrain the sign of the GP inside and outside of the object by adding additional points. By adding the normals in terms of derivative information to the GP these inside and outside points are no longer necessary. Furthermore, the magnitude of the derivative observations has only an impact on the overall scale of the GP, but not on the location of the 0-level set. Therefore, the magnitude of the normals is chosen to be 1:

$$\|\boldsymbol{n}_i\| = 1 \,. \tag{3.19}$$

Following the argument in Williams and Rasmussen (2006, p. 191), the GP can be extended to include derivative observations: „Since differentiation is a linear operator, the derivative of a Gaussian process is another Gaussian process. Thus we can use GPs to make predictions about derivatives, and also to make inference based on derivative information. In general, we can make inference based on the joint Gaussian distribution of function values and partial derivatives.“

In a GP without derivative information, the covariance k can be defined on two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Using this insight the covariance can be extended to include mixed covariances between data points and partial derivatives and between partial derivatives:

$$\operatorname{cov}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.20}$$

$$\operatorname{cov}\left(f(\boldsymbol{x}_i), \frac{\partial f(\boldsymbol{x}_j)}{\partial x_{j,m}}\right) = \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{j,m}} \tag{3.21}$$

$$\text{and} \quad \operatorname{cov}\left(\frac{\partial f(\boldsymbol{x}_i)}{\partial x_{i,n}}, \frac{\partial f(\boldsymbol{x}_j)}{\partial x_{j,m}}\right) = \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,n} \partial x_{j,m}} \,. \tag{3.22}$$

Now all components of the GP have to be extended to include the additional derivative information. Following the notation by Martens et al. (2017) introduced all extended formulations will be marked with a plus sign.

For a better overview, the covariance matrices are repeated here:

$$\mathbf{K} \in \mathbb{R}^{N \times N}, \quad (\mathbf{K})_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3.23}$$

$$\boldsymbol{k}^* \in \mathbb{R}^{N \times 1}, \quad (\boldsymbol{k})_i = k(\boldsymbol{x}, \boldsymbol{x}_i). \tag{3.24}$$

Here N denotes the number of observations.

Since at each data point all three partial derivatives are observed the covariance matrices $\mathbf{K}$ and $\boldsymbol{k}$ have to be extended to include $N(1+3)$ observations. One entry for the value observation and three entries for the partial derivatives:

$$\mathbf{K}^+ \in \mathbb{R}^{4N \times 4N} \tag{3.25}$$

$$\boldsymbol{k}^{+*} \in \mathbb{R}^{4N \times 1} \tag{3.26}$$

For a better understanding the matrix $\mathbf{K}^+$ is split into 4×4 sub matrices, where the covariances and mixed covariances are interleaved. The indices i and j denote the respective sample numbers for each observed contact, comprised of observed valued and the three observed partial derivatives while n and m denote the index within the sub matrix.

$$((\mathbf{K}^+)_{i,j})_{m,n} = \begin{cases} \operatorname{cov}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) & m = 0, n = 0 \\ \operatorname{cov}\left(f(\boldsymbol{x}_i), \frac{\partial f(\boldsymbol{x}_j)}{\partial x_{j,m}}\right) & m > 0, n = 0 \\ \operatorname{cov}\left(\frac{\partial f(\boldsymbol{x}_i)}{\partial x_{i,n}}, f(\boldsymbol{x}_j)\right) & m = 0, n > 0 \\ \operatorname{cov}\left(\frac{\partial f(\boldsymbol{x}_i)}{\partial x_{i,n}}, \frac{\partial f(\boldsymbol{x}_j)}{\partial x_{j,m}}\right) & m > 0, n > 0 \end{cases} \tag{3.27}$$

By replacing the covariances with the respective kernels from equations (3.20) through (3.22) the full sub matrix $\mathbf{K}^+_{i,j}$ is given by

$$\mathbf{K}^+_{i,j} = \begin{bmatrix} k(\boldsymbol{x}_i, \boldsymbol{x}_j) & \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{j,1}} & \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{j,2}} & \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{j,3}} \\ \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,1}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,1} \partial x_{j,1}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,1} \partial x_{j,2}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,1} \partial x_{j,3}} \\ \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,2}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,2} \partial x_{j,1}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,2} \partial x_{j,2}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,2} \partial x_{j,3}} \\ \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,3}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,3} \partial x_{j,1}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,3} \partial x_{j,2}} & \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,3} \partial x_{j,3}} \end{bmatrix} \tag{3.28}$$

while the full matrix $\mathbf{K}^+$ can be constructed from the sub matrices

$$\mathbf{K}^+ = \begin{bmatrix} \mathbf{K}^+_{1,1} & \dots & \mathbf{K}^+_{1,N} \\ \vdots & \ddots & \\ \mathbf{K}^+_{N,1} & & \mathbf{K}^+_{N,N} \end{bmatrix} \tag{3.29}$$

Similarly the covariance vector $\boldsymbol{k}^{+*}$ between a test point and the observe samples can be extended, using the same indexing notation:

$$((\boldsymbol{k}^{+*})_i)_m = \begin{cases} \operatorname{cov}(f(\boldsymbol{x}), f(\boldsymbol{x}_j)) & m = 0 \\ \operatorname{cov}\left(f(\boldsymbol{x}), \frac{\partial f(\boldsymbol{x}_j)}{\partial x_{j,m}}\right) & m > 0 \end{cases} \tag{3.30}$$

The extended value vector $\boldsymbol{y}^+$ is given by

$$((\boldsymbol{y}^+)_i)_m = \begin{cases} f(\boldsymbol{x}) & m = 0 \\ \frac{\partial f(\boldsymbol{x}_i)}{\partial x_{i,m}} & m > 0 \end{cases} \tag{3.31}$$

by inserting the 0-value observations and the normal observations from Equation 3.17 and Equation 3.18 the extended value vector can be written as

$$\boldsymbol{y} = (0, n_{1,1}, n_{1,2}, n_{1,3}, 0, n_{2,1}, n_{2,2}, n_{2,3}, \dots)^T . \tag{3.32}$$

By solving the linear system

$$(\mathbf{K}^+ + \sigma_n^2 \mathbf{I})\boldsymbol{w}^+ = \boldsymbol{y}^+ \tag{3.33}$$

$$\boldsymbol{w}^+ = (\mathbf{K}^+ + \sigma_n^2 \mathbf{I})^{-1}\boldsymbol{y}^+ \tag{3.34}$$

the extended weight vector can be obtained, where weights for values and partial derivatives are alternated:

$$\boldsymbol{w} = (w_{1,0}, w_{1,1}, w_{1,2}, w_{1,3}, w_{2,0}, w_{2,1}, w_{2,2}, w_{2,3}, \dots)^T \tag{3.35}$$

Finally, the estimated ISP $f^*(\boldsymbol{x})$ can be computed as:

$$f^*(\boldsymbol{x}) = \boldsymbol{k}^{+*}\boldsymbol{y}^+ = \sum_{i=1}^{N} \left(w_{i,0} k(\boldsymbol{x}, \boldsymbol{x}_i) + \sum_{n=1}^{3} w_{i,n} \frac{\partial k(\boldsymbol{x}, \boldsymbol{x}_i)}{\partial x_n} \right) . \tag{3.36}$$

3.2.3 Covariance Functions

In the literature, two different radial basis kernel functions have been proposed to be used with GPIS: the standard squared exponential covariance function and the thin plate covariance function from the original GPIS publication by Williams and Fitzgibbon (2007). A radial basis function k_{RBF} is a function defined on two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, where the function only depends on the distance between the vectors:

$$k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{RBF}(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \,. \tag{3.37}$$

Therefore, the function is stationary

$$k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{RBF}(\boldsymbol{x}_i + \boldsymbol{a}, \boldsymbol{x}_j + \boldsymbol{a}) \tag{3.38}$$

and symmetric

$$k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{RBF}(\boldsymbol{x}_j, \boldsymbol{x}_i) \,. \tag{3.39}$$

In practice, this can be exploited to reduce the amount of derivatives that have to be computed since

$$\frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,n}} = -\frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{j,n}} \tag{3.40}$$

In the following, the partial derivatives for the **squared exponential** kernel and the **thin plate** kernel are derived.

The **squared exponential** kernel is given by

$$k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right) \,. \tag{3.41}$$

The partial derivatives are

$$\frac{\partial k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,m}} = -\frac{1}{\sigma^2}(x_{i,m} - x_{j,m}) k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.42}$$

$$\frac{\partial^2 k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,n} \partial x_{j,m}} = \begin{cases} \frac{1}{\sigma^4}\left(\sigma^2 + (x_{i,m} - x_{j,m})^2\right) k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) & n = m \\ \frac{1}{\sigma^4}(x_{i,m} - x_{j,m})(x_{i,n} - x_{j,n}) k_{SE}(\boldsymbol{x}_i, \boldsymbol{x}_j) & n \neq m \end{cases} \tag{3.43}$$

The **thin plate** kernel is given by

$$k_{TP}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 2d^3 - 3Rd^2 + R^3 \tag{3.44}$$

with $d = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ and $R = \max_{i,j}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. The partial derivatives are

$$\frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,n}} = 6(x_{i,n} - x_{j,n})(R + d) \tag{3.45}$$

$$\frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{i,n} \partial x_{j,m}} = \begin{cases} 6(R+d) + \frac{6(x_{i,n} - x_{j,n})^2}{d} & n = m \\ \frac{6(x_{i,n} - x_{j,n})(x_{i,m} - x_{j,m})}{d} & n \neq m \end{cases} \tag{3.46}$$

3.2.4 Comparison

The benefits of adding normal observations will be illustrated using 2D and 3D examples. Contacts are sampled from a ground truth model while the surfaces estimates provided by GPIS with and without derivative information are compared. The four ground truth models include:

- a **2D flat surface**,
- a **2D corner**,
- a **2D stair structure** with two steps,
- and a **3D cube**.

In case of the **flat surface** three contacts are sampled on the surface. For the reconstruction using GPIS, an inner point inside of the object is assumed to be below the sampled surface points. The resulting GPIS surface estimate resembles and oval shape and can be seen in Figure 3.11 on the left side of the top row. When the contact normals are included as derivative information of the Gaussian process the surface estimate can match the ground truth, as shown on the right side of the top row in the figure.

The **corner** example illustrates a situation that is common during the exploration process. A surface is explored and three contact points have been gathered. However, the surface is not limitless and has a sharp edge, therefore the next acquired contact point has a normal perpendicular to the other contacts. The GPIS estimate without normals underestimates the size of the object, while the GPIS estimate with normal information extrapolates the object's surface, as shown in Figure 3.11 (middle).

In the **stair** example, the GPIS estimate without normals fails to estimate the surface of the stairs, while the GPIS estimate with surface normals can represent the stair structure and extrapolates in unseen regions. While the fine details of the convex and concave stair corners are smoothed over, the local surface

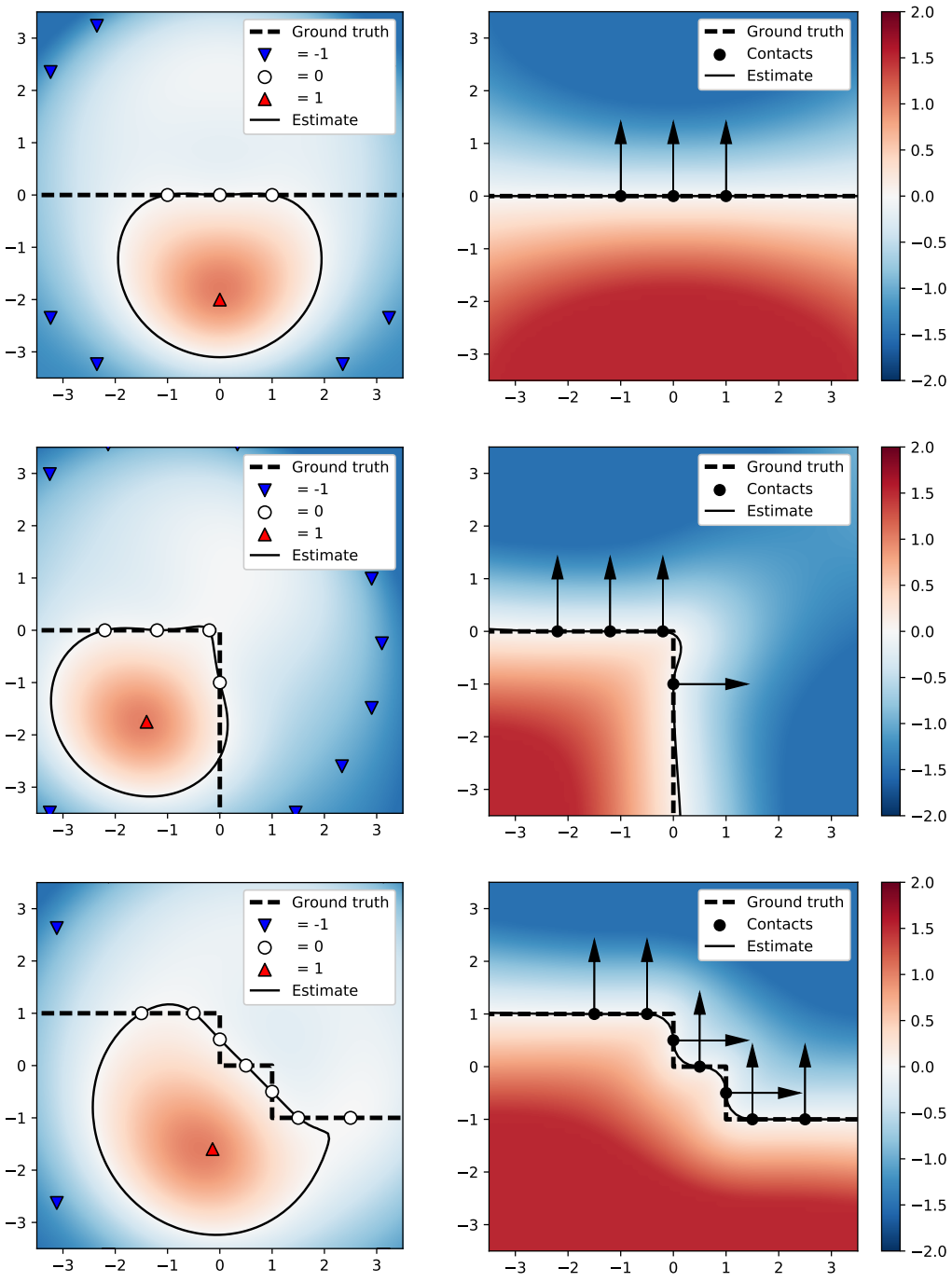
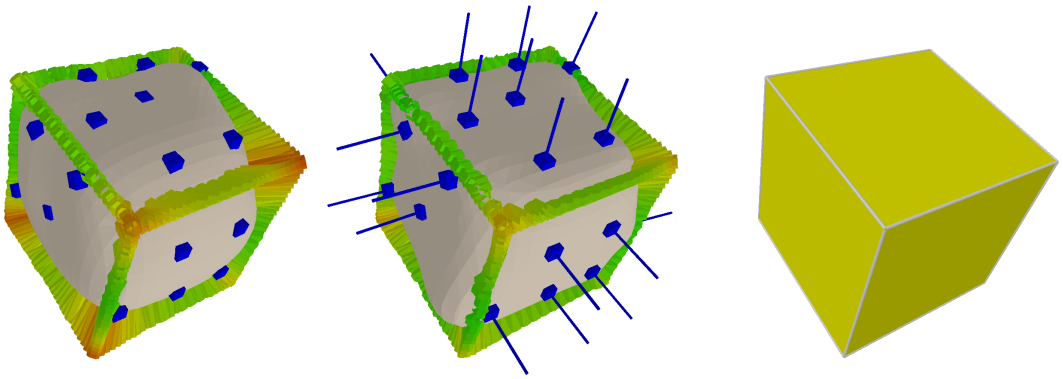


Figure 3.11: Comparison of the original GPIS formulation (left) and the extended formulation including surface normals (right).

orientation at the contact points is reproduced according to the ground truth object, as shown in Figure 3.11 (bottom).

A similar reconstruction behavior can be observed for **3D objects**. In Figure 3.12 the GPIS reconstruction with and without normals is compared at the example

of a 3D cube. When normals are included, the estimated surface extrapolates more accurately in unseen regions leading to smaller reconstruction errors at the edges of the object.



(a) GPIS without normals (b) GPIS with normals (c) Ground truth

Figure 3.12: Comparison of GPIS surface estimation with and without normals. The contacts on the object's surface are displayed in blue. The reconstruction error at the edges of the cube are shown as color-coded lines. Green indicates a small error whereas red corresponds to a large error. Images taken from Ottenhaus et al. (2018b), © 2018 IEEE.

The addition of normals to the GPIS process has several key advantages:

- **Improved surface orientation**: The orientation of the estimated surface conforms to the ground truth surface orientation.
- **Improved interpolation**: Without normals GPIS tends to remove edges by connecting neighboring contact points with a smooth surface. The reconstruction near edges and corners is improved by including normals, see Figure 3.12a and Figure 3.12b.
- **Improved extrapolation**: The addition of normals allows GPIS to extrapolate the surface in unknown regions, see Figure 3.11.
- **Less hyper-parameters**: The original formulation of GPIS requires additional points to constrain the value of the Gaussian process inside and outside of the object. These points have to be chosen carefully to ensure that the resulting GPIS estimate resembles the underlying object. The added normals introduce derivative information, which automatically constrains the value of the Gaussian process, thereby eliminating the need for additional points.

- **Higher information density**: The added surface normals increase the information per contact. Therefore, the object's surface can be estimated to a higher level of detail using the same number of exploration actions.

Adding normal observations to GPIS clearly benefits the surface model in several ways. Therefore, the extended formulation of GPIS including surface normals will be used in the remainder of this work.

3.3 Selection of the Next-Best-Touch

Starting from the requirements an efficient exploration algorithm should maximize the information gained per exploration action while the cost to perform the exploration actions should be minimized. In real world scenarios these goals are often contradictory, i.e. actions with low cost will not acquire much new information while high information gain actions have high execution costs. Therefore, it is desirable to maximize the gained information ($\Delta \mathit{information}, \Delta I$) in relation to the cost ($\Delta \mathit{cost}, \Delta C$), as expressed in Equation 3.47.

$$\mathit{Maximize} \ \frac{\Delta \mathit{information}}{\Delta \mathit{cost}} = \frac{\Delta I}{\Delta C} \tag{3.47}$$

The information gain is defined using the ground truth model and the already acquired contacts. For each point $\boldsymbol{m}$ on the ground truth model $\mathcal{M}$ the state of exploration is defined using the already acquired contact points $\boldsymbol{c}_i \in \mathcal{C}_n$, where $\mathcal{C}_n$ is the set of contacts after n exploration steps. A point $\boldsymbol{m}$ is considered explored if any acquired contact point is within a certain radius r. The state of exploration $e_n(\boldsymbol{m})$ is given by

$$e_n(\boldsymbol{x}) = \begin{cases} 1 & \min_i \|\boldsymbol{x} - \boldsymbol{c}_i\| < r \ , \ \boldsymbol{c}_i \in \mathcal{C}_n \\ 0 & \text{otherwise} \ . \end{cases} \tag{3.48}$$

The overall explored surface E_n after n exploration steps can be calculated as

$$E_n = \int_{\boldsymbol{m} \in \mathcal{M}} e(\boldsymbol{m}) \ . \tag{3.49}$$

The information gain ΔI_n is given by the difference between the explored surface at exploration step n and the explored surface at the previous exploration step $n-1$.

$$\Delta I_n = E_n - E_{n-1} \tag{3.50}$$

The cost for each exploration step is quantified by the distance the robotic fingertip has to cover to execute each exploration action. Let the trajectory of exploration action n from the previous contact $\boldsymbol{c}_{n-1}$ to the current contact $\boldsymbol{c}_n$ be given by $T_n(\tau)$ with $\tau \in [0, 1]$. The trajectory starts at the previous contact and ends at the current contact.

$$T_n(0) = \boldsymbol{c}_{n-i} \tag{3.51}$$

$$T_n(1) = \boldsymbol{c}_n \tag{3.52}$$

The cost ΔC_n can be defined as the path length of the trajectory

$$\Delta C_n = \int_0^1 \left\| \frac{\partial T(\tau)}{\partial \tau} \right\| d\tau \,. \tag{3.53}$$

Maximizing the fraction in Equation 3.47 can be achieved by balancing the amount of information gained with the costs per exploratory action.

3.3.1 Gaussian Variance based Exploration

During the exploration of an unknown object, the ground truth surface of the object is not available. Therefore, possible candidates for exploration can only be chosen based on the current surface estimate of the object. In the context of haptic exploration, GPIS is widely used to estimate the surface of the object (Dragiev et al., 2011; Bjorkman et al., 2013; Sommer et al., 2014; Matsubara and Shibata, 2017; Yi et al., 2016). The GPIS potential is given by the mean value of a Gaussian process $f(x) : \mathbb{R}^3 \to \mathbb{R}$ defining the implicit surface potential (ISP)

$$f(\boldsymbol{x}) = \boldsymbol{k}^*(\boldsymbol{x})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} y \,, \tag{3.54}$$

where $\mathbf{K}$ is the covariance matrix, and $\boldsymbol{k}^*$ is the covariance vector between a query point x and the observed sample locations x_i

$$(\mathbf{K})_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.55}$$

$$(\boldsymbol{k}^*)_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i) \,. \tag{3.56}$$

Besides the mean value f the Gaussian process also yields a variance $\boldsymbol{Q}$, that can be evaluated for any query point $\boldsymbol{x}$.

$$\boldsymbol{Q}(x) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^{*T} (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \boldsymbol{k}^* \tag{3.57}$$

This variance is used by Bjorkman et al. (2013) and Yi et al. (2016) to select the next potential target for exploration.

A variance-based exploration will be illustrated using a synthetic 1D example. The ground truth function $f(x)$ is plotted in Figure 3.13 and given by

$$f(x) = \cos(4x)e^{-\frac{x^2}{3^2}} \ . \tag{3.58}$$

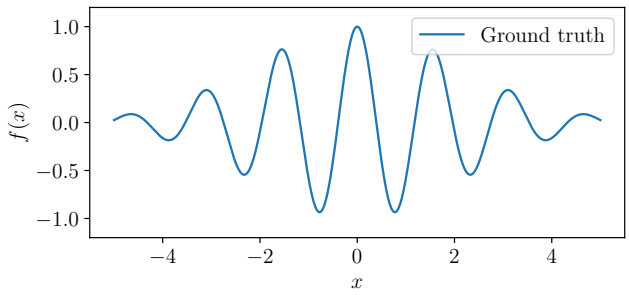


Figure 3.13: Synthetic 1D ground truth function.

The first exploration target t_1 is chosen to be at $x = 0$. The width of the Gaussian kernel is chosen as $\sigma = \frac{1}{2}$. The exploration procedure is given by

1. Sample ground truth function at t_i.
2. Update function estimate using the Gaussian process.
3. Calculate mean and variance.
4. Select next target t_{i+1} to be at the location of the highest variance and repeat.

Following this exploration scheme the algorithm proceeds to explore the ground truth function over 30 steps. The resulting mean and variances after step 1,5,10 and 30 are shown in Figure 3.14, where the variance is displayed above and below the mean. When using the Gaussian kernel function the variance $\boldsymbol{Q}$ is bounded to $[0, 1]$. The variance $\boldsymbol{Q}$ is small when evaluated in proximity of an explored point, while $\boldsymbol{Q}$ approaches 1 when evaluated far away from al observed samples. This can be seen after the initial contact occurs (Figure 3.14, top left): the variance is small around the sample point, but remains high in other areas. The next exploration target is chosen to be as far away from the initial observed point as possible, thereby maximizing the information gain and minimizing the overall variance.

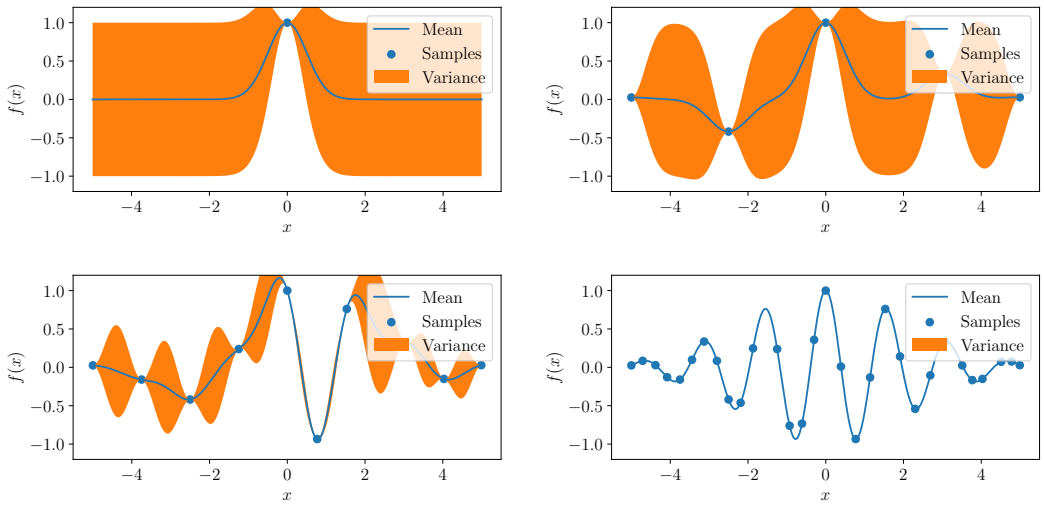


Figure 3.14: Variance-based exploration example: Mean and variance of the Gaussian process estimate during the exploration process after exploration steps 1,5,10 and 30.

For comparison, a simple linear exploration strategy is also be considered. Here the next exploration target is always chosen to be right of the previous sample point, with a constant offset. the resulting exploration covers the ground truth function from left to right, as shown in Figure 3.15.

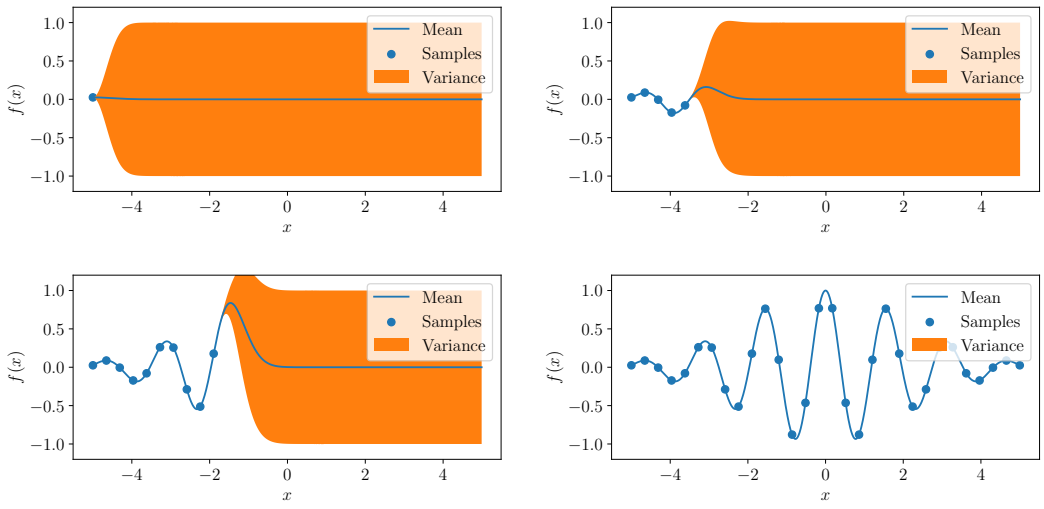


Figure 3.15: Linear exploration example: Mean and variance of the Gaussian process after exploration steps 1,5,10 and 30.

When comparing the function estimates and variances in Figure 3.14 and Figure 3.15 the variance-based strategy seems to decrease the variance of the Gaussian process faster. In addition, the estimated mean approaches the ground

truth faster. This observation can be quantified by comparing the mean variance and the root mean squared error (RMSE) between the estimate and the ground truth for both strategies. The mean variance is given by

$$\overline{\boldsymbol{Q_i}} = \frac{1}{b-a} \int_a^b \boldsymbol{Q_i}(x) dx \tag{3.59}$$

while the RMSE is given by

$$\text{RMSE}_i = \sqrt{\frac{1}{b-a} \int_a^b \|f(x) - g(x)\|^2 dx} \,. \tag{3.60}$$

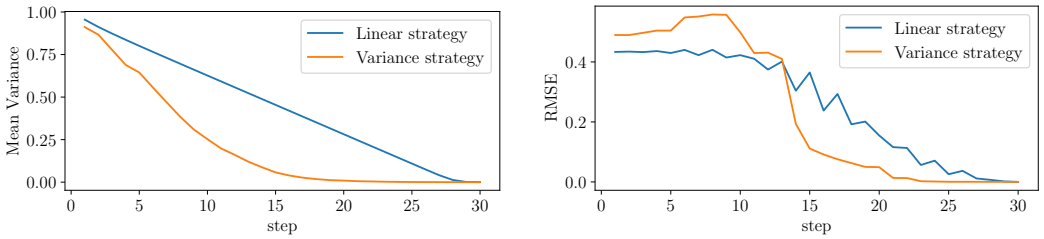


Figure 3.16: Comparison of mean variance and root mean squared error for the variance-based and linear exploration strategy in the 1D case.

In Figure 3.16 the mean variance and the RMSE are given for each exploration step. Using the variance-based exploration strategy, the variance of the Gaussian process drops quickly in the beginning, compared to the linear strategy. This is to be expected, since the variance-based exploration aims to decrease the variance of the GP as quickly as possible. When comparing the estimation error after each exploration step the results are different. In the beginning, the linear strategy has lower RMSE than the variance-based strategy, until it is surpassed after approximately 15 exploration steps. Another interesting observation is that the variance of the GP does not necessarily correlate closely with the RMSE, in the case of the variance-based strategy. While the variance drops quickly below $50\,\%$ after 7 exploration steps, the RMSE remains high until 15 exploration steps have been performed. This illustrates that variance reduction does not always imply RMSE reduction. Furthermore, optimizing only the reduction of variance neglects the path costs, as Matsubara and Shibata (2017) have also stated in their work: „We propose a novel criterion in active touch point selection for fast estimation, which considers both uncertainty of shape estimation and travel cost to touch.“

3.3.2 Information Gain Estimation Function

Following the idea of Matsubara and Shibata (2017) in this work, an approach is developed to explore an unknown object by maximizing the newly gained information while considering execution cost. The estimated information gain is modeled using Gaussian kernels, similar to the Gaussian variance based approaches. During the exploration, possible exploration candidates are generated using the estimated surface of the object. To rate the exploration candidates, four different goals are considered:

1. **Uncertainty:** Each exploratory action should yield the maximum amount of new information.
2. **Locality:** Prefer local targets to distant ones.
3. **Travel cost:** Consider the movement of the fingertip during an exploration action.
4. **Rotation cost:** Minimize the rotation of the fingertip during an exploration action.

The first goal drives the exploration process to unknown regions, to gain as much new information per contact as possible. To counterbalance this greedy information maximization the other goals consider the expected cost of the exploration actions.

Each exploration goal is quantified by a corresponding metric ($\Psi_1 \dots \Psi_4$). Each metric yields a high value if a given candidate should be preferred and returns a low value if the given candidate is of less interest. All metrics are evaluated separately, based on

- the current **fingertip location** $\boldsymbol{r}$,
- the set of all **previous contacts** $\boldsymbol{c}_i \in \mathcal{C}$,
- a **candidate target** $\boldsymbol{s}$ on the estimated surface $\mathcal{S}$ of the object.

Each contact $\boldsymbol{c}_i$ is comprised of a position component $\boldsymbol{c}_{i,p}$ and a surface normal component $\boldsymbol{c}_{i,n}$. In addition, candidate targets are comprised of a position $\boldsymbol{s}_p$ and normal $\boldsymbol{s}_n$. The full set of used symbols can be found in Table 3.1.

The **first metric** Ψ_1 aims to explore the object by preferring distant candidates and by penalizing candidates that are close to already explored contact points. In fact, Ψ_1 is zero when evaluated directly at a previously observed contact point c_i. This metric prefers candidates in previously unseen regions of the surface estimate. Acquiring new contacts in these unseen regions often improves

the surface estimate and reduces the uncertainty of the surface estimate.

$$\Psi_{1,\boldsymbol{c}}(\boldsymbol{s},\boldsymbol{c}) := 1 - \exp\left(-\frac{\|\boldsymbol{s}-\boldsymbol{c}\|^2}{{\sigma_1}^2}\right) . \tag{3.61}$$

The **second metric** Ψ_2 prefers candidates in the neighborhood of already acquired contact points. This is achieved using a Gaussian kernel with non-zero mean μ_3, preferring candidate points that have an approximate distance of μ_3 to previously explored contacts.

$$\Psi_{2,\boldsymbol{c}}(\boldsymbol{s},\boldsymbol{c}) := \exp\left(-\frac{(\|\boldsymbol{s}-\boldsymbol{c}\|-\mu_3)^2}{{\sigma_3}^2}\right) . \tag{3.62}$$

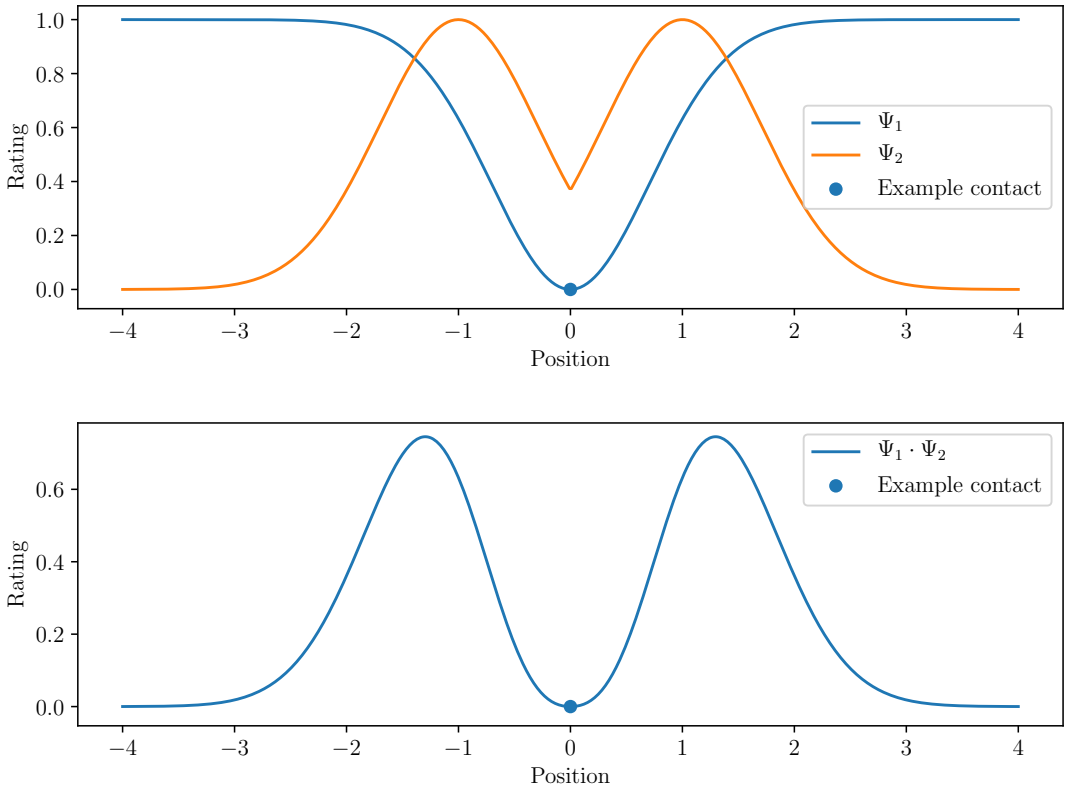


Figure 3.17: Exemplary rating values according to the metrics Ψ_1 and Ψ_2. Here one contact point is assumed at location 0, while the scaling factors σ and μ are set to 1. Top: Individual values of Ψ_1 and Ψ_2. Bottom: Product of Ψ_1 and Ψ_2.

The characteristics of Ψ_1 and Ψ_2 are visualized in Figure 3.17 using one example contact at location 0. The singular use neither Ψ_1 nor Ψ_2 leads to a desirable outcome. The metric Ψ_1 penalized the candidates close to 0, however the maximum value of Ψ_1 will only be reached at infinity. When Ψ_1 is combined with

Ψ_2 via multiplication, the shape of the rating function has the desired shape: Candidate targets at approximately distance μ_3 are preferred while candidates near the explored location 0 are penalized.

So far $\Psi_{1,\boldsymbol{c}}$ and $\Psi_{2,\boldsymbol{c}}$ have been defined per contact. To include all previous contacts into the rating metrics different aggregate functions are employed. In case of $\Psi_{1,\boldsymbol{c}}$, the minimum value is used to find the overall value of Ψ_1

$$\Psi_1(\boldsymbol{s}) = \min_{\boldsymbol{c}\in\mathcal{C}} \Psi_{1,\boldsymbol{c}}(\boldsymbol{c}, \boldsymbol{s})\,. \tag{3.63}$$

The reason is that $\Psi_{1,\boldsymbol{c}}$ primarily penalized candidates near already explored contacts, therefore the minimum as an aggregate penalizes candidates near any already explored contact.

In case of $\Psi_{2,\boldsymbol{c}}$ the overall metric should return a measure that describes if a candidate is within the desired range to several contacts. Therefore, a sum is used as an aggregate function

$$\Psi_2(\boldsymbol{s}) = \sum_{\boldsymbol{c}\in\mathcal{C}} \Psi_{2,\boldsymbol{c}}(\boldsymbol{c}, \boldsymbol{s})\,. \tag{3.64}$$

The **third metric** Ψ_3 implements a travel cost function by measuring the length of the planned trajectory $T_{\boldsymbol{r},\boldsymbol{s}}(\tau)$ from the current position $\boldsymbol{r}$ to the candidate touch point $\boldsymbol{s}$. The argument of the trajectory τ is valid within the range $[0, 1]$ with $T_{\boldsymbol{r},\boldsymbol{s}}(0) = \boldsymbol{r}$ and $T_{\boldsymbol{r},\boldsymbol{s}}(1) = \boldsymbol{s}$. The metric Ψ_3 serves two purposes: it reduces the time spent moving the end-effector from contact to contact and it leads to a local exploration where the estimated surface is not likely to diverge greatly from the actual object.

$$\Psi_2(\boldsymbol{s}) \coloneqq \frac{1}{\mathit{Len}\left(T_{\boldsymbol{r},\boldsymbol{s}}(\tau)\right)}\,, \tag{3.65}$$

where the length of the path $T_{\boldsymbol{r},\boldsymbol{s}}$ from $\boldsymbol{r}$ to $\boldsymbol{s}$ is calculated as the arc length of the curve

$$\mathit{Len}(T_{\boldsymbol{r},\boldsymbol{s}}(\tau)) = \int_0^1 \left\| \frac{\partial T_{\boldsymbol{r},\boldsymbol{s}}}{\partial \tau} \right\| d\tau\,. \tag{3.66}$$

Combining the metrics Ψ_1 through Ψ_3 leads to an exploration scheme that follows roughly an outward spiral, as can be seen in Figure 3.18.

The **fourth metric** Ψ_4 aims to limit the rotation of the fingertip to execute an exploratory action. To this end, a measurement is used that yields a value of 1 if the predicted rotation is zero. The returned value of Ψ_4 drops as the predicted rotation angle increases. The metric is implemented using an angular kernel, as

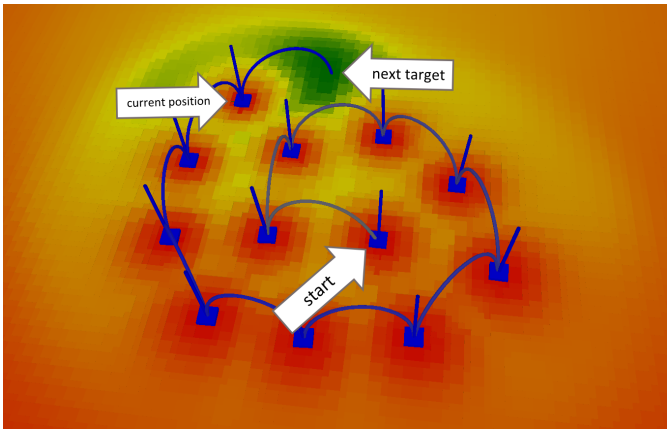


Figure 3.18: Exemplary exploration of a plane. The combination of the exploration goals Ψ_1 through Ψ_3 leads to an exploratory path that follows an outward spiral. In green areas, the predicted information gain is high; in red areas, it is low. Image taken from Ottenhaus et al. (2018a), © 2018 World Scientific Publishing Co Pte Ltd.

described in (Rasmussen and Williams, 2006, chapter 4.2.3).

$$\Psi_4(\boldsymbol{s}) \coloneqq \exp\left(-\frac{2\sin^2\left(\frac{1}{2}\arccos(R_z \cdot \boldsymbol{s}_n)\right)}{{\sigma_\alpha}^2}\right) \tag{3.67}$$

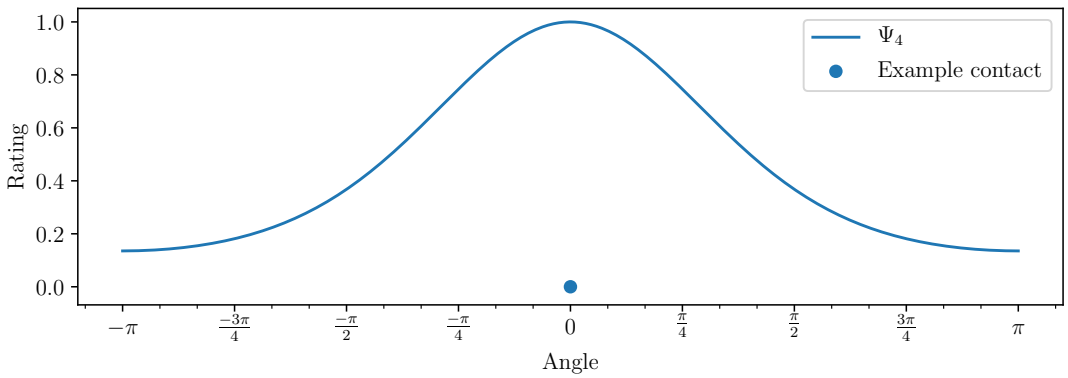


Figure 3.19: The metric Ψ_4 is implemented as an angular kernel.

The metrics Ψ_1 through Ψ_4 have been designed to be combined via multiplication. Therefore, the overall metric is given by

$$\Psi = \prod_{n=1}^{4} \Psi_n \,. \tag{3.68}$$

The algorithm using this overall metric is given in Algorithm 3. The four met-

rics are evaluated for each possible candidate target on the estimated surface $\mathcal{S}$ and the combined rating for each candidate is returned. To reduce the execution time of the rating function candidate targets are only considered within an $R_0 = 5\,\mathrm{cm}$ radius. Furthermore the surface $\mathcal{S}$ is discretized using the marching cubes algorithm by Lorensen and Cline (1987) using a grid length of $5\,\mathrm{mm}$.

Algorithm 3 Calculation of the Information Gain Estimation Function metrics

Input $S, \mathcal{C}, r, v_n$
for $\boldsymbol{s} \in S \wedge \|\boldsymbol{s} - \boldsymbol{r}\| < R_0$ **do**
  $\Psi_1(\boldsymbol{s}) \leftarrow \min_{\boldsymbol{c}} \Psi_1(\boldsymbol{s}, \boldsymbol{c})$ ▷ see Equation 3.61
  $\Psi_2(\boldsymbol{s}) \leftarrow \sum_{\boldsymbol{c}} \Psi_3(\boldsymbol{s}, \boldsymbol{c})$ ▷ see Equation 3.62
  $\Psi_3(\boldsymbol{s}) \leftarrow \mathrm{Len}\left(T_{\boldsymbol{r},\boldsymbol{s}}\right)^{-1}$ ▷ see Equation 3.65
  $\Psi_4(\boldsymbol{s}) \leftarrow 1 - \exp\left(-2\sin^2(\frac{1}{2}\alpha(R_z, \boldsymbol{s}_n))\sigma_\alpha^{-2}\right)$ ▷ see Equation 3.67
end for
return $\Psi_1\Psi_2\Psi_3\Psi_4$ ▷ see Equation 3.68

3.3.3 Trajectory Generation

The trajectory generation should yield a function describing the path from the current fingertip location to a potential exploration candidate. This trajectory function $T_{vecr,\boldsymbol{s}}(\tau)$ is defined for $\tau \in [0, 1]$ and should meet the functional requirements given in subsection 3.1.1. These requirements will be repeated below and translated to constraints. Additionally a qualitative requirement is added that focuses on runtime efficiency.

REQ 1: The trajectory should start at the current location of the fingertip $\boldsymbol{r}$.

$$T_{\boldsymbol{r},\boldsymbol{s}}(0) = \boldsymbol{r} \tag{3.69}$$

REQ 2: The trajectory should end at the candidate target position.

$$T_{\boldsymbol{r},\boldsymbol{s}}(1) = \boldsymbol{s}_p \tag{3.70}$$

REQ 3: The initial direction of the trajectory should be inverse to the velocity vector just before the contact occurred.

$$\frac{\partial T_{\boldsymbol{r},\boldsymbol{s}}(0)}{\partial \tau} = -k\boldsymbol{v}_n \ , k \in \mathbb{R}^+ \tag{3.71}$$

where k is a positive scaling factor that determines how quickly the trajectory can deviate from the initial direction. Small values of k allow a quick deviation

while large values of k require the trajectory to follow the initial velocity vector longer.

REQ 4: The trajectory should approach the candidate $\boldsymbol{s} \in \mathcal{S}$ in the direction of the estimated normal $\boldsymbol{s}_n$ at the selected candidate position $\boldsymbol{s}_p$.

$$\frac{\partial T_{\boldsymbol{r},\boldsymbol{s}}(1)}{\partial \tau} = -k\boldsymbol{s}_n \ , k \in \mathbb{R}^+ \tag{3.72}$$

where k is the same scaling factor as in REQ 3.

REQ 5: The trajectory should be smooth, i.e. differentiable.

$$\frac{\partial T_{\boldsymbol{r},\boldsymbol{s}}(\tau)}{\partial \tau} \text{ exists } \forall \ \tau \in [0,1] \tag{3.73}$$

REQ 6: The generation and evaluation of $T_{\boldsymbol{r},\boldsymbol{s}}$ should be computationally efficient.

As stated previously, a Bezier curve meets requirements REQ 1 through REQ 5. The choice of parameters and the derivation of the Bezier control points will be explained in the following.

The definition of a cubic Bézier is given in Equation 3.74.

$$\boldsymbol{b}(\tau) = (1-\tau)^3\boldsymbol{b}_0 + 3\tau(1-\tau)^2\boldsymbol{b}_1 + 3\tau^2(1-\tau)\boldsymbol{b}_2 + \tau^3\boldsymbol{b}_3 \tag{3.74}$$

The requirements REQ 1 through REQ 4 can be expressed as

$$\begin{pmatrix} \boldsymbol{b}(0) \\ \boldsymbol{b}(1) \\ \frac{\partial \boldsymbol{b}(0)}{\partial \tau} \\ \frac{\partial \boldsymbol{b}(1)}{\partial \tau} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{s}_p \\ -k\boldsymbol{v}_n \\ -k\boldsymbol{s}_n \end{pmatrix} \tag{3.75}$$

Since $\boldsymbol{b}(0) = \boldsymbol{b}_0$ and $\boldsymbol{b}(1) = \boldsymbol{b}_3$ REQ 1 and REQ 2 can be met by setting $\boldsymbol{b}_0 = \boldsymbol{r}$ and $\boldsymbol{b}_3 = \boldsymbol{s}$.

To meet REQ 3 and REQ 4 first $\frac{\partial \boldsymbol{b}}{\partial \tau}$ has to be computed

$$\frac{\partial \boldsymbol{b}}{\partial \tau} = -3\boldsymbol{b}_0(1-x)^2 + 3\boldsymbol{b}_1((1-x)^2 - 2x(1-x)) - 3\boldsymbol{b}_2(x^2 - 2x(1-x)) + 3\boldsymbol{b}_3x^2 \ . \tag{3.76}$$

Evaluating $\frac{\partial \boldsymbol{b}}{\partial \tau}(0)$ and $\frac{\partial \boldsymbol{b}}{\partial \tau}(1)$ yields

$$\frac{\partial \boldsymbol{b}}{\partial \tau}(0) = 3(-\boldsymbol{b}_0 + \boldsymbol{b}_1) \text{ and} \tag{3.77}$$

$$\frac{\partial \boldsymbol{b}}{\partial \tau}(1) = 3(-\boldsymbol{b}_2 + \boldsymbol{b}_3). \tag{3.78}$$

By inserting the calculated values for $\boldsymbol{b}(0), \boldsymbol{b}(1), \frac{\partial \boldsymbol{b}(0)}{\partial \tau}$ and $\frac{\partial \boldsymbol{b}(1)}{\partial \tau}$ into Equation 3.75 all four control points of the Bézier curve are fully constrained

$$\begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_3 \\ 3(-\boldsymbol{b}_0 + \boldsymbol{b}_1) \\ 3(-\boldsymbol{b}_2 + \boldsymbol{b}_3) \end{pmatrix} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{s}_p \\ -k\boldsymbol{v}_n \\ -k\boldsymbol{s}_n \end{pmatrix} \tag{3.79}$$

Solving the equation system results in the control points

$$\begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{r} - \frac{1}{3}k\boldsymbol{v}_n \\ \boldsymbol{s}_p + \frac{1}{3}k\boldsymbol{s}_n \\ \boldsymbol{s}_p \end{pmatrix} \tag{3.80}$$

To scale the shape of the Bézier curve with the distance of $\boldsymbol{r}$ and $\boldsymbol{s}$ the scaling parameter k is chosen to be

$$k = \|\boldsymbol{r} - \boldsymbol{s}_p\| \tag{3.81}$$

By using a short hand notation of

$$T(\tau) = \text{bezier}(\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3) \tag{3.82}$$

the overall trajectory function is then given by

$$T(\tau) = \text{bezier}(\boldsymbol{r}, \boldsymbol{r} - \frac{1}{3}\boldsymbol{v}_n\|\boldsymbol{r} - \boldsymbol{s}_p\|, \boldsymbol{s}_p + \frac{1}{3}\boldsymbol{s}_n\|\boldsymbol{r} - \boldsymbol{s}_p\|, \boldsymbol{s}_p) \tag{3.83}$$

REQ 5 is met, since the derivative of the Bézier equation exists for all τ. Furthermore, REQ 6 is met, since the Bézier equation is given by a simple polynomial function.

The orientation of the fingertip is calculated by a linear interpolation between the start R_s and the target orientation R_t. The target orientation is calculated so

that the fingertip is aligned with the surface normal at the target.

$$T_R(\tau) = (1 - \tau) * R_s + \tau R_t \tag{3.84}$$

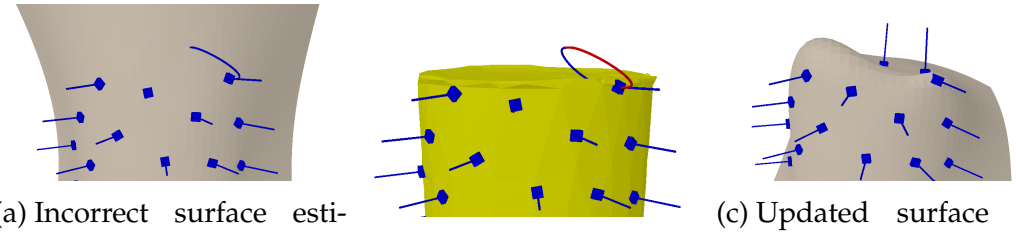
During the exploration, the trajectory is followed to the chosen target. If a contact occurs before the end of the trajectory, the contact is added to the set of contacts and the exploration process continues normally. If no contact occurs, the trajectory is extended by a small amount Δm to compensate for small inaccuracies in surface estimation. However, if the surface estimate deviates strongly from the actual object's surface a different strategy is necessary. In this case, a new trajectory has to be generated that guides the fingertip back to the object surface. This new trajectory starts at the current location $\boldsymbol{r}$ and ends at the previous contact location $\boldsymbol{c}_p$. Furthermore, the requirements for the velocities at the start and target are modified, so that the velocity magnitude and direction of the fingertip is continuous. In addition, the previous contact location $\boldsymbol{c}_p$ is approached in the negative direction of the contact normal $-\boldsymbol{c}_n$. The requirements can be expressed as

$$\begin{pmatrix} T(0) \\ T(1) \\ \frac{\partial T(0)}{\partial \tau} \\ \frac{\partial T(1)}{\partial \tau} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{c}_p \\ \boldsymbol{v}_n \|\boldsymbol{r} - \boldsymbol{c}_p\| \\ -\boldsymbol{c}_n \|\boldsymbol{r} - \boldsymbol{c}_p\| \end{pmatrix} \tag{3.85}$$

$$T(\tau) = \text{bezier}(\boldsymbol{r}, \boldsymbol{r} + \frac{1}{3}\boldsymbol{v}_n \|\boldsymbol{r} - \boldsymbol{c}_p\|, \boldsymbol{c}_p - \frac{1}{3}\boldsymbol{c}_n \|\boldsymbol{r} - \boldsymbol{c}_p\|, \boldsymbol{c}_p) \tag{3.86}$$

The requirements from Equation 3.85 lead to the parametrization of the cubic Bezier in Equation 3.86. The signs of the parameters defining the control points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are flipped compared to the signs in Equation 3.83. This ensures that contact with the object is always re-established, since the target position is the last contact point and the approach direction is inverted, resulting in an approach from the underside of the surface.

An example is given in Figure 3.20 where the already acquired contacts are shown as blue boxes and the contact normals are shown as blue lines. The surface estimate is rendered in gray and the actual object is yellow. The chosen exploration target, sampled from the surface estimate (Figure 3.20a) does not lie on the actual object's surface and thus the calculated trajectory misses the object. The previous trajectory (red line in Figure 3.20b) is continued (blue line in Figure 3.20b) to re-establish contact with the object by approaching the previous contact. After contacts have been established on the top of the object, the surface estimate is updated (Figure 3.20c).



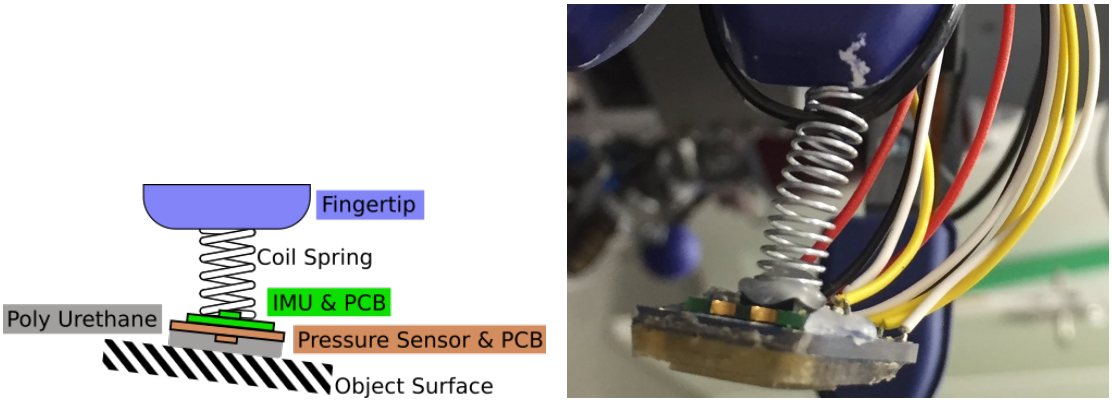
(a) Incorrect surface estimate. (b) Actual object. (c) Updated surface estimate.

Figure 3.20: Example of trajectory continuation to restore contact with the object.

3.4 Maximizing the Information Gain per Contact

In the previous sections, the benefit of adding local surface orientation in the form of normal information to the surface estimation has been shown. This section describes the approach to acquire surface orientation using a newly developed prototype sensor. This sensor allows the direct measurement of surface orientation as well as contact forces. This is achieved by the combination of an Inertial Measurement Unit (IMU) (BNO055, Bosch Sensortec) and a pressure sensor (BMP280, Bosch Sensortec). Both sensing elements are arranged as shown in the schematic drawing in Figure 3.21a. The IMU provides 3D linear acceleration measurement, 3D rotational acceleration measurement and a 3D magnetic sensor, resuling in a 9-axis measurement system. The IMU was originally developed for use in mobile devices and therefore integrates on-board signal processing and signal fusion methods. In particular an absolute 3D orientation is offered, which can be accessed via a digital interface, reducing the necessary signal processing on the host system. The pressure sensor operates using a piezo-resistive sensing element and provides absolute barometric pressure measurements with $\pm 1\,\mathrm{hPa}$ absolute accuracy and $\pm 0.12\,\mathrm{hPa}$ relative accuracy. The measured pressure is also available via a digital interface. Therefore, no external analog signal processing is necessary and both sensors can be connected directly to a common microprocessor.

In order to convert the air pressure sensor to a tactile sensing element the sensor is covered in a layer of flexible material, similar to the approach described by Tenzer et al. (2012). Tenzer et al. used polyurethane (PU) to completely fill in the opening of the pressure sensor, allowing the PU to transmit any external forces directly to the underlying sensing element. However, the original sensor used by Tenzer et al. was no longer available, as it reached the end-of-life. The polyurethane application process had to be adapted, since most recent integrated pressure sensors feature a much smaller opening, which does not allow



(a) Conceptual drawing (b) Sensor at fingertip

Figure 3.21: The sensor system consists of an IMU and a pressure sensor that is covered in polyurethane. The conceptual drawing (left) shows the structure of the sensor. For experimental evaluation, the sensor is mounted at the fingertip of the humanoid robot ARMAR-III (right). Images taken from Ottenhaus et al. (2018b), © 2018 IEEE

to be filled with PU.

While the pressure sensor is placed in a mold, the polyurethane is poured on top until the desired covering height is achieved. Due to surface tension, the polyurethane cannot flow through the opening of the pressure sensor, since this opening has a diameter of only $0.3\,\mathrm{mm}$. A small amount of residual air remains within the pressure sensor as the opening is sealed with polyurethane. While the polyurethane is still in the liquid phase the surrounding air pressure is reduced to about $0.9\,\mathrm{bar}$. The lower surrounding pressure forces the residual air from within the pressure sensor out. As the air expands into the polyurethane a small bubble forms. The polyurethane hardens while the air pressure is kept constant at $0.9\,\mathrm{bar}$. After the polyurethane has cured, a part of the polyurethane was carved out, leaving a small raised section behind as displayed in Figure 3.22.

The operation principle of the sensor is as follows: When an external force is applied, the polyurethane is compressed. This compression leads to an increase in pressure within the air bubble, which is measured by the pressure sensor. When the sensor contacts a surface, the contact force is distributed to the raised section of the polyurethane on the sensor leading to a higher pressure and therefore increasing the sensitivity of the sensor.

The dimensions of the combined sensing element are $18\,\mathrm{mm} \times 18\,\mathrm{mm} \times 8\,\mathrm{mm}$ (width, length, height). The width and length are mainly determined by the size of the pressure sensor circuit board. The height divides out into the IMU

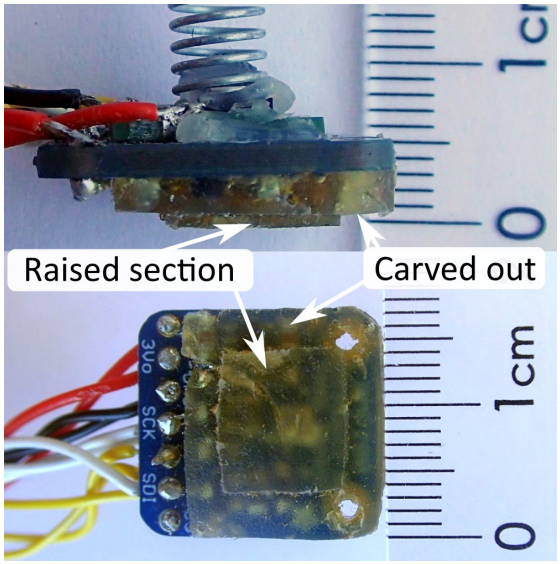


Figure 3.22: Side view (top) and bottom view (bottom) of the combined sensor. The raised section of the polyurethane increases the sensitivity of the sensor.

with circuit board ($2\,\mathrm{mm}$), the pressure sensor with circuit board ($2\,\mathrm{mm}$) and the polyurethane cover ($4\,\mathrm{mm}$). The sensor was attached to the index finger of ARMAR-III using a flexible coil spring as displayed in Figure 3.21b.

When the sensor is pressed against an inclined surface the coil spring allows for self-alignment of the sensor, as the contact force increases. The pressure sensor can then measure this contact force. If a given force threshold is surrpassed, it can be assumed that the sensor has fully aligned with the surface and the absolute orientation of the IMU can be queried to estimate the surface orientation.

3.5 Grasp Synthesis

Using the exploration strategy with the surface orientation sensor from the previous sections a set of oriented contact points can be acquired from an unknown object. From this contact set, the surface of the object can be estimated using Gaussian Process Implicit Surfaces. This surface estimate is converted to a mesh model using the marching cubes algorithm introduced by Lorensen and Cline (1987). For grasp planning two different grasp planners were considered.

The first grasp planner is a surface based grasp planner provided by Simox, developed by Vahrenkamp et al. (2013). The mesh model of the estimated surface is provided to the grasp planner and several grasp candidates are generated.

All generated grasp candidates are considered stable according to the calculated ϵ-metric. However, not all of the calculated candidates will result in a stable grasp when executed by the robot. Especially the second candidate in Figure 3.23 will probably fail, since the fingers will slip off the object.

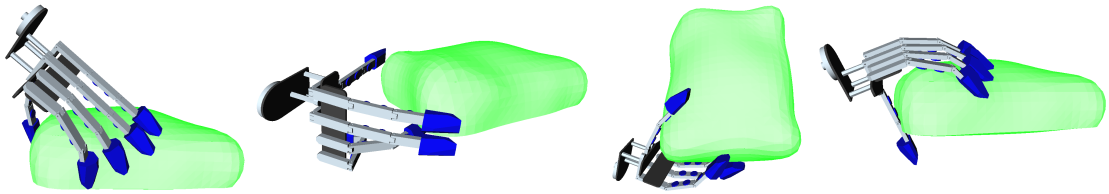


Figure 3.23: Grasp candidates generated by the surface based grasp planner.

The second grasp planner is the skeleton-based grasp planner by Vahrenkamp et al. (2018). This grasp planner first extracts a mean curvature skeleton from the object mesh. Thereafter, several grasp candidates can be generated that are aligned with the generated skeleton. The grasp candidates generated by the skeleton-based grasp planner are stable for the test object, as can be seen in Figure 3.24. Therefore, the grasp synthesis was performed using the skeleton-based grasp planner.

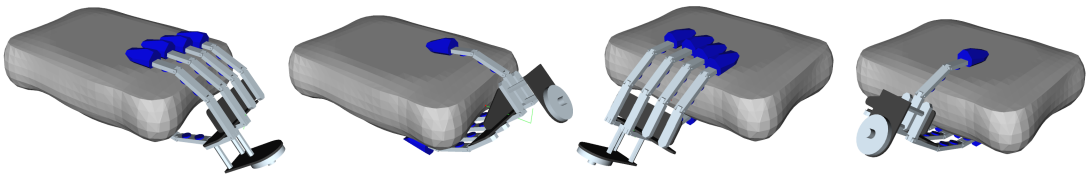


Figure 3.24: Grasp candidates generated by the skeleton-based grasp planner.

The use of a general grasp planner allows to easily exchange the surface estimate mesh model and the hand model, as shown in Figure 3.25. Both surface estimates were acquired via haptic exploration using the tactile sensor and the humanoid robot ARMAR-6. Grasp candidates were then generated for the ARMAR-III hand and the ARMAR-6 hand.

3.6 Evaluation

The evaluation of the tactile exploration is split into different parts. First, the exploration strategy, proposed in section 3.1, will be evaluated in simulation by comparing different next-next-touch approaches. To this end a set of 180 objects, taken from the KIT and YCB object sets (Kasper et al., 2012; Calli et al.,

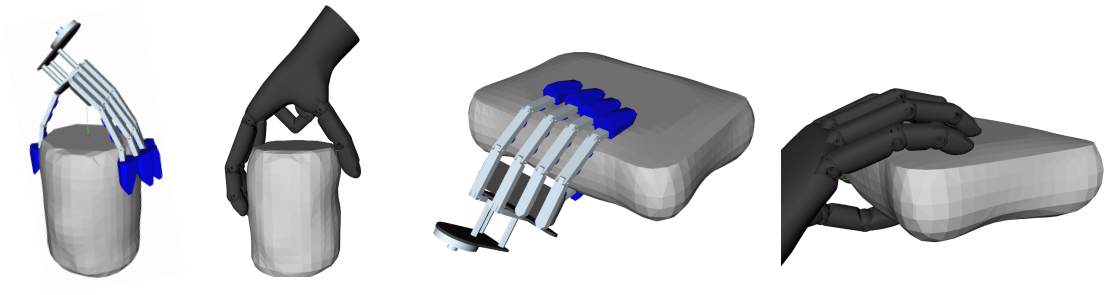


Figure 3.25: Grasp candidates for the estimated surfaces for the ARMAR-III and ARMAR-6 hands.

2015) is explored in a simulated environment. Thereafter, the proposed tactile sensor (see section 3.4) is used to explore objects, while the sensor is attached to the humanoid robots ARMAR-III and ARMAR-6, to gather oriented contact points from the objects. Finally the acquired contact points will be used to estimate the surface of the objects using the extended formulation of Gaussian Process Implicit Surfaces (GPIS), presented in section 3.2. This surface estimate is the basis fo the grasp candidate synthesis (see section 3.5).

3.6.1 Evaluation of the Next-Best-Touch Strategy

To evaluate the next-best-touch strategy a robotic fingertip is placed into a simulated environment, where one of the objects from the object set is present. The fingertip can move freely around the object and is controlled in velocity mode. When the tip of the finger collides with the object, a touch event is simulated by extracting the touch position and surface normal at the contact location. To guide the fingertip the exploration strategy described in section 3.1 is used. Before the exploration starts, one of three strategies for selecting the next-best-touch is chosen:

- Recent publications described in the state-of-the art suggest using the uncertainty of the surface estimate to select the exploration target (Matsubara and Shibata, 2017). To this end the variance of the underlying Gaussian process is evaluated. In the following, this approach will be called *GP-V*.
- The second strategy is the proposed Information Gain Estimation Function to rate the potential next-best-touch locations. This approach will be called *IGEF* in the following.
- To establish a common baseline a random exploration will also be per-

formed, labeled *RND*. Here the next exploration target is sampled randomly from the current estimated surface.

All three strategies have in common that the next exploration target is selected from the current surface estimate, following an update-surface, select-exploration-target and explore cycle, which is common in the haptic exploration literature (Dragiev et al., 2011; Bjorkman et al., 2013; Matsubara and Shibata, 2017; Yi et al., 2016).

Exploration of Simple Shapes

The three strategies will first be compared using simple geometric shapes to give an understanding about the characteristics of each strategy and the whole exploration process. Throughout the evaluation, several 3D views of the exploration will be provided. All views share the same color-coding and markers, which are explained in Table 3.4.

Marker / Color	**Description**
Small blue box & blue line	Explored contact position with normal
Curved blue line	Exploration trajectory
Gray surface	Surface estimate
Yellow object	Ground truth object
Color coded (red to green) boxes	Rating of potential exploration targets by the next-best-touch strategy. Green corresponds to a high rating while red denotes a low rating.

Table 3.4: Overview of used colors and markers

First, a flat plane is explored, where the space from which possible exploration targets are selected is limited. The initial contact is selected to be in the center. Further exploration targets are chosen according to the selected strategy. In Figure 3.26 the exploration is depicted for each next-best-touch strategy. The GP-V approach selects the next target to be far away from already explored contact points, thereby maximizing the information gained per contact. The IGEF based strategy selects the next exploration target to be close to the current position, resulting in a spiraling exploration pattern. The RND strategy chooses the target randomly. The path between the contacts follows the Bézier curves, described in subsection 3.3.3. In this simple example, the path length is closely

related to the distance of the chosen target from the current position of the fingertip. The right column of the figure shows the complete path covered by the exploration. Using the GP-V and the random strategy results in longer path lengths per contact, while the IGEF strategy generates short paths.

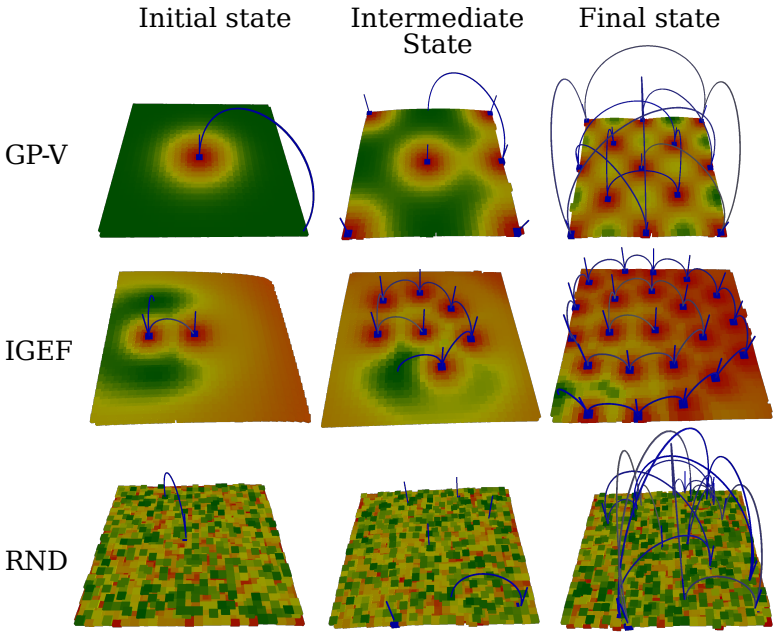


Figure 3.26: Comparison of different next-best-touch strategies at different exploration stages. From left to right the number of explored contacts increases. The color-coding of the plane denotes the rating according to the strategy.

The second example covers the exploration of a sphere. The GP-V approach quickly covers the surface by taking large steps while the IGEF strategy explores the surface in a spiral pattern. The covered exploration path is shown in Figure 3.27.

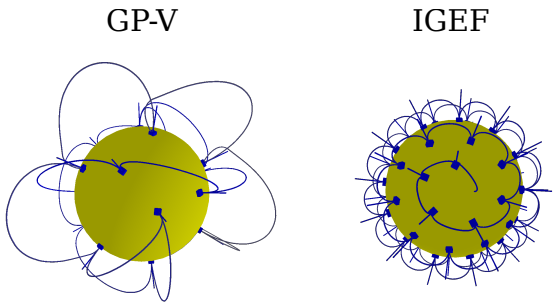


Figure 3.27: Comparison of the exploration of a sphere.

The final example is the exploration of a cube, as shown in Figure 3.28. The top row of the figure shows the ground truth object, acquired contacts and the path

covered by the exploration. The bottom row shows the final surface estimate. For both strategies, a similar amount of contacts was necessary to achieve comparable surface reconstructions, while the path covered by the GP-V approach is longer than the path taken by the IGEF approach.

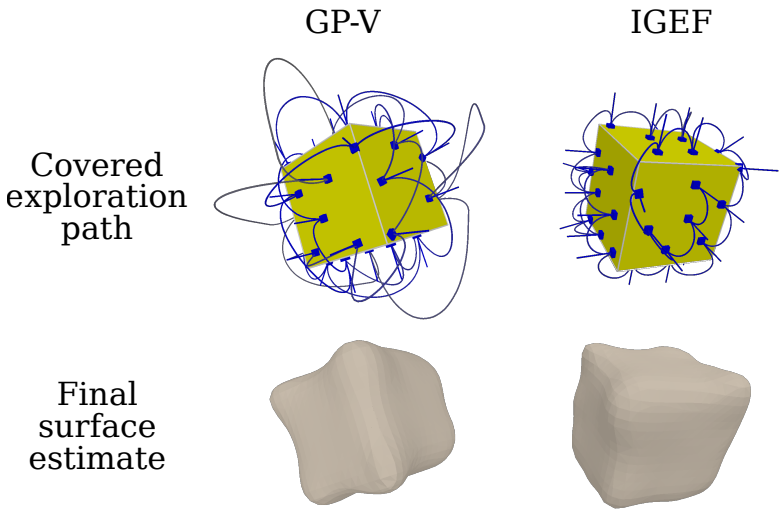


Figure 3.28: Comparison of the exploration of a cube.

Exploration Termination

During the exploration of an unknown object, the exploration procedure has to be stopped at one point, to avoid an endless exploration and refinement of the object's surface. Therefore the exploration process is executed, until one of two exploration termination conditions is met:

1. Over $X\,\%$ of the object's surface is marked as explored.
2. The exploration is stuck and no new information is gathered over 10 successive contacts.

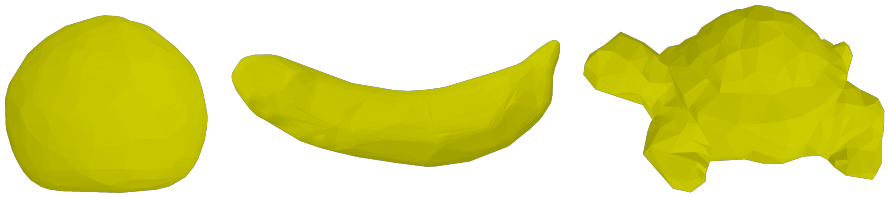
To evaluate the first condition, the state of exploration is determined by marking an area of radius r_E on the surface of the ground truth object as explored around each object. During the evaluation, a value of $r_E = 2\,\mathrm{cm}$ was used. The exploration was terminated when over $80\,\%$ of the object's surface was marked as explored. In this evaluation, $80\,\%$ were chosen instead of $100\,\%$ exploration, since the exploration of the remaining $20\,\%$ can take disproportional amounts of time to the information gained. This is the case if some disjoint areas of the object's surface are still unexplored, while the surface estimate does not cover these areas, i.e. thin or small structures on the object that have not been touched

during exploration and are not present in the surface estimate. Finding these areas can be left to random chance and can thereby distort the exploration results.

The second condition terminates the exploration if the exploration procedure is not able to gather any additional information over 10 successive contacts. In some cases, the chosen exploration target on the estimated surface is far away from the actual object's surface. Then the exploration strategy guides the fingertip back to the object's surface and generates a new contact, which updates the surface estimate. However, this new contact can be close to existing contacts, resulting in no additionally gained information. Thus, the updated surface estimate is similar to the previous surface estimate and the next chosen exploration target is similar to the previous exploration target.

Exemplary Exploration

In this section, the exploration algorithm will be performed for the objects „YCB-Softball", „YCB-Banana" and „YCB-Tortoise". The objects are sorted by surface complexity, starting with the softball, which resembles a sphere.



Softball Banana Tortoise

Figure 3.29: Objects used in the exemplary exploration.

The softball is explored 100 times using each next-best-touch strategy. An example exploration path for the GP-V and IGEF approaches is shown in Figure 3.30. As the softball is similar to a sphere, the exploration path is similar to the sphere example. The GP-V approach takes large steps to maximize the information gain per contact, while the IGEF approach takes smaller steps. The spiral exploration pattern can be seen on the right of Figure 3.30. While Figure 3.30 shows one example exploration, the average performance of the exploration approaches is of interest. Therefore, Figure 3.31 shows the exploration progress for each next-best-touch approach. The explored surface percentage is plotted over the distance covered. The values for the individual exploration runs are shown using light colors, while the median is shown as a thick dashed

line. Here the median was chosen instead of the mean, since some of the exploration runs take much longer to explore the object. In the plot, the exploration is more efficient, if the respective curve rises more quickly, i.e. more surface is explored per distance covered. As can be seen the IGEF approach explored the object more quickly on average than the GP-V or the random approach. In the beginning, the random approach explores the object faster than the GP-V approach, while GP-V surpasses the random approach later. The increase of explored surface is almost linear for the IGEF approach with lower variance than for the other approaches.

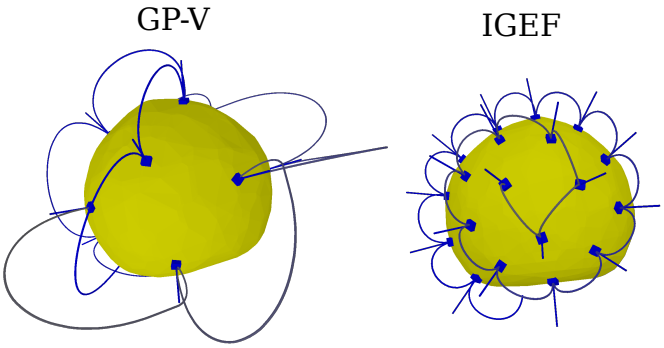


Figure 3.30: Exploration of the object „Softball“.

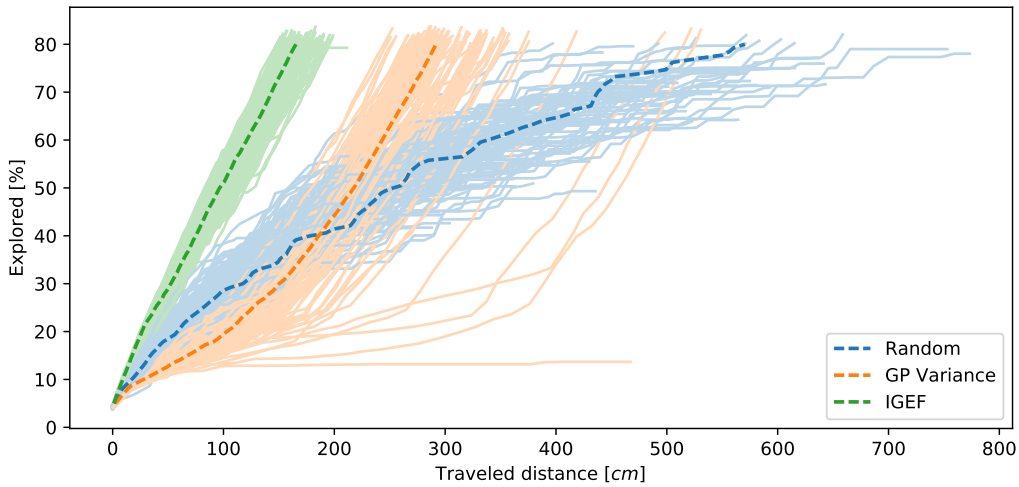


Figure 3.31: Exploration progress for the object „Softball“.

Examples for the exploration patterns for the „Banana“ object are shown in Figure 3.32. The GP-V approach takes large steps while the IGEF approach takes smaller steps. Here the spiral exploration pattern is not as obvious, due to the high curvature of the banana in one direction and the elongation in the other direction. The exploration progress in shown in Figure 3.33. Here the IGEF ap-

proach explores the object quickly; however, the variance is a bit higher. The main difference to the softball is that the GP-V approach is on average slower than the random exploration. However, this is only true for the average, due to the high variance in the results.

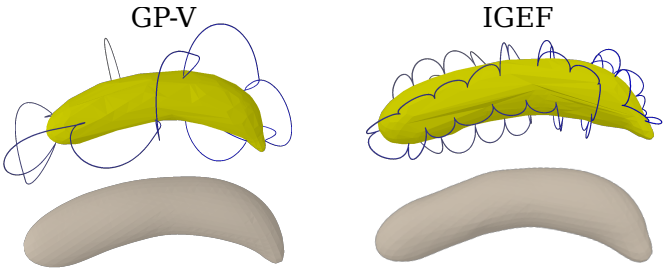


Figure 3.32: Exploration of the object „Banana“.

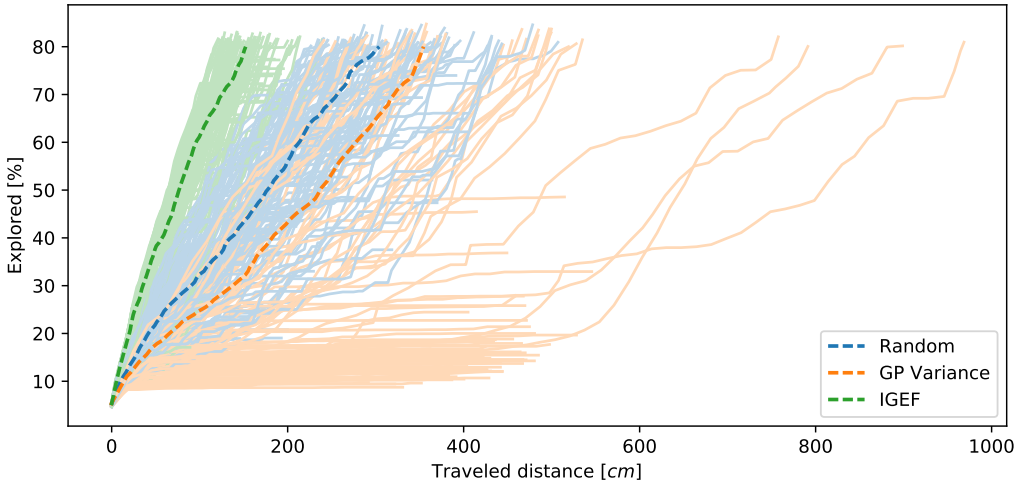


Figure 3.33: Exploration progress for the object „Banana“.

The exploration patterns for the „Tortoise“ are shown in Figure 3.34. Again, the GP-V approach takes larger steps while IGEF takes smaller exploration steps. The resulting exploration progress shown in Figure 3.35 is similar to the exploration progress of the banana, while the overall exploration takes a bit longer, since the tortoise has a larger surface area than the banana. Besides the example exploration path, Figure 3.34 also shows the surface estimate in an intermediate state and after the exploration has finished. The GP-V approach first covers all sides of the object, resulting is a rough initial estimate. The final surface estimate is similar for both exploration approaches.

The question arising from these results is: „Why is the GP-V based exploration slower than the IGEF exploration?“ One possible explanation is that the GP-V

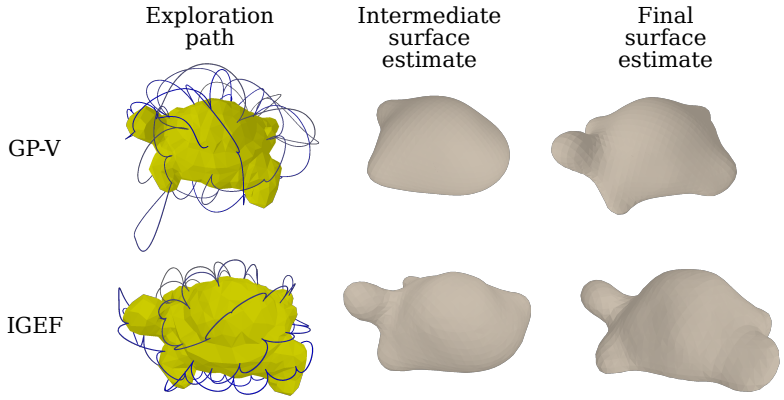


Figure 3.34: Exploration of the object „Tortoise“.

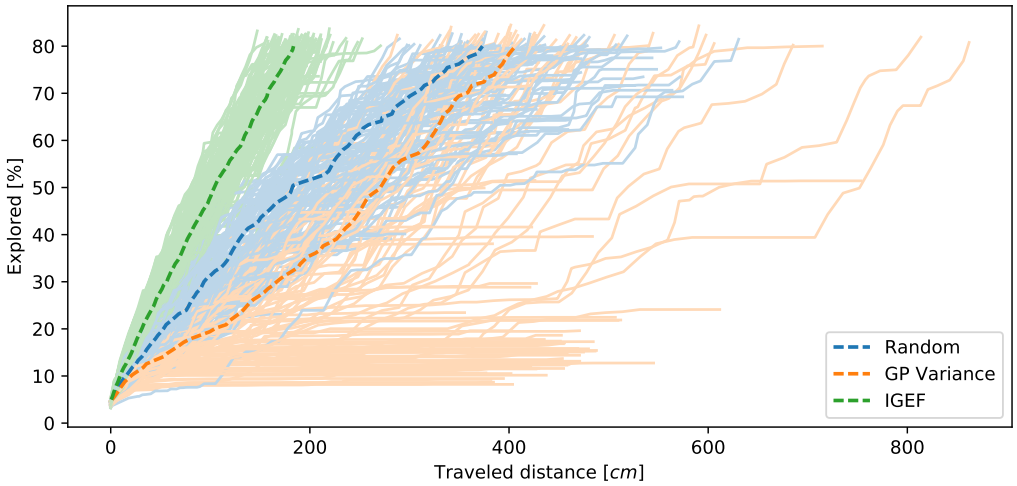


Figure 3.35: Exploration progress for the object „Tortoise“.

approach chooses exploration targets far away from previous contacts. These possible exploration targets are always sampled from the estimated surface. The estimated surface has a high probability of being incorrect if sampled far away from already explored points. To validate this hypothesis an additional metric is evaluated. For each exploration action, the predicted contact location is compared with the actual contact location. The average results for the example objects and the next-best-touch approaches are shown in Figure 3.36. As the figure shows, the distance between the predicted and the actual contact is high for the GP-V approach and low for the random and the IGEF approaches. The targets chosen by the GP-V approach are often far away from the actual surface of the object. Therefore, the exploration has to return to the actual object's surface, resulting in additional travel cost.

As a result of the exploration of the example object several observations can be

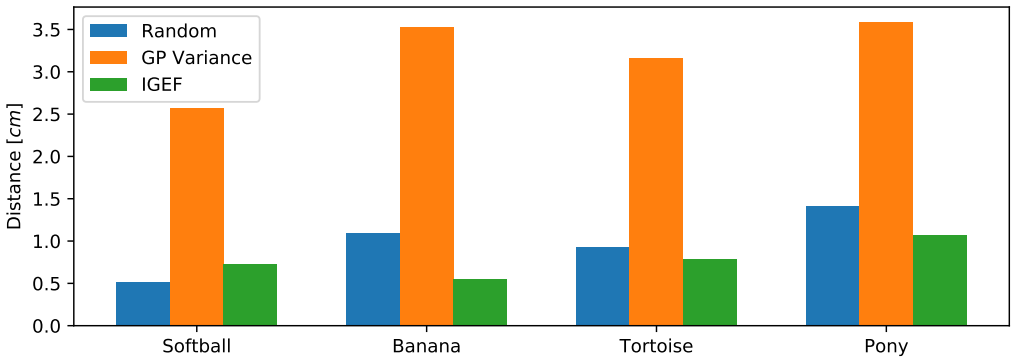


Figure 3.36: Distance between the predicted contact on the estimated surface and the actual contact location.

made:

1. The IGEF approach gains more information per distance covered than the GP-V approach.
2. The final surface estimates are similar for the IGEF and the GP-V approach.
3. The GP-V approach often chooses exploration targets that are far away from the actual object's surface.

In order to evaluate if these observations generalize to all objects from the evaluation set the exploration procedure is repeated for all objects.

Quantitative Evaluation

The exploration of the example objects suggests that the IGEF approach can explore an unknown object more rapidly than the state-of-the-art approach GP-V. In this section, the proposed IGEF strategy is compared with the state-of-the art approach GP-V. Furthermore, a common baseline is established by choosing the exploration target randomly from the current surface estimate. The strategies are compared using objects from the KIT and YCB object sets. Each strategy is executed 100 times per object, using different initial conditions, like starting positions. The initial target for exploration is chosen to be the center of mass of the object. The exploration strategy has no prior knowledge of the object. The ground truth object is only used to calculate surface contact positions and contact normals during exploration and to estimate the exploration process according to the termination conditions. Each object is explored, until one of the aforementioned termination conditions is met. Thereafter, different metrics are evaluated:

1. **Distance covered**: The overall path length of all exploratory actions.
2. **Overall rotation**: The accumulated rotation of the fingertip orientation during all exploratory actions.
3. **Surface RMSE**: The Root-Mean-Square Error (RMSE) between the ground truth mesh and the triangulated estimated surface after the exploration has finished.
4. **Prediction error**: The average distance between the predicted contact based on the estimated surface and the actual contact with the ground truth mesh.

The distance covered is recorded for each exploration run, when the exploration is complete. The object surface area is plotted over this distance in Figure 3.37. For each exploration run, one dot is plotted in the figure. Thereafter, a linear regression is fitted to the data for each strategy. Points on the left of the graph correspond to better results, while points on the right side correspond to larger distances. The figure shows that the IGEF strategy outperforms the GP-V approach in most cases, since the green dots (IGEF) are mostly left of the orange dots (GP-V). Most of the distances covered by the IGEF approach follow the linear regression. Furthermore, the average distance traveled is proportional to the object's surface area. In the case of GP-V, the variance of the data is higher and the average indicates a higher distance covered to explore the objects.

The average results for all evaluated metrics are displayed in Table 3.5. Each row of the table gives the results for one metric, while the columns correspond to the three strategies. The last column compares the IGEF strategy with the GP-V approach. For each metric, a lower value is better. As the table shows, the IGEF strategy outperforms the GP-V strategy in distance covered, overall rotation of the fingertip and the prediction error. The achieved surface reconstruction RMSE is similar for both approaches. The last row gives the average number of contacts, for comparison.

3.6.2 Evaluation of Surface Normal Observations

In the previous sections, the surface estimation was performed using the contact positions and the surface normals. To obtain these surface normals the tactile sensor presented in section 3.4 is used. Two experiments are performed to evaluate the contact detection and the surface normal sensing of the proposed

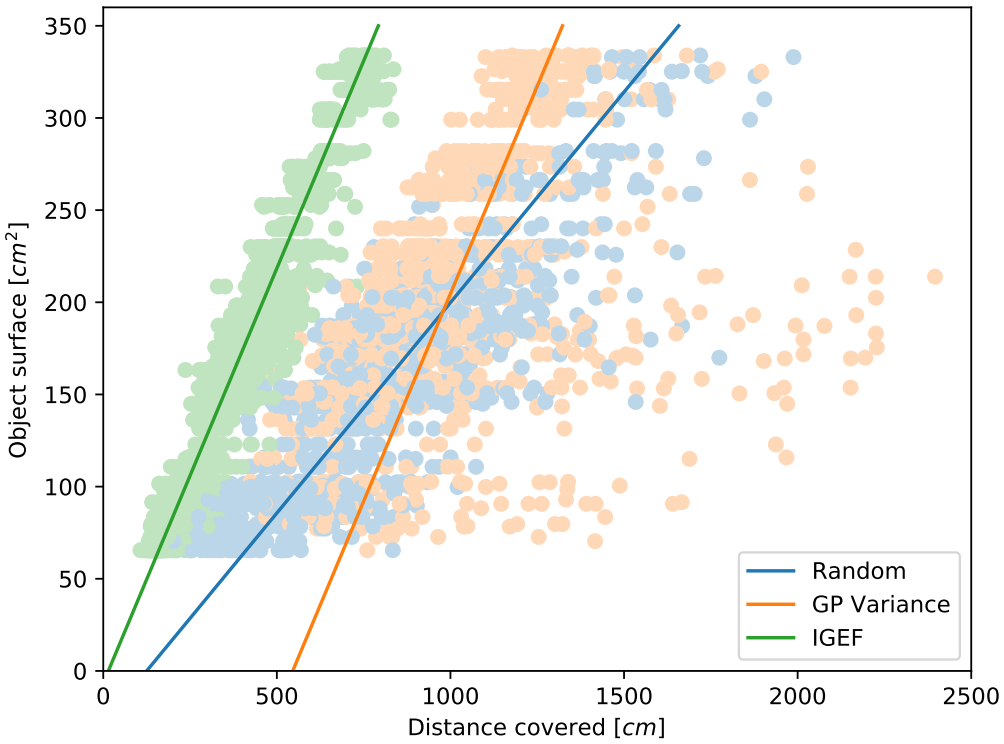


Figure 3.37: Evaluation of the distance covered.

Metric	Random	GP-V	IGEF	Avg. Improvement
Avg. distance covered	243 cm	255 cm	117 cm	54 %
Avg. overall rotation	2680°	1884°	1383°	27 %
Avg. surface RMSE	0.81 mm	0.65 mm	0.53 mm	18 %
Avg. prediction error	4.7 mm	14.9 mm	3.5 mm	77 %
Avg. number of contacts	58.8	38.0	46.1	-

Table 3.5: Results for the KIT and YCB Object Dataset

sensor. Thereafter, the benefit of adding surface normals is evaluated by comparing the surface estimation quality with and without surface normals.

Obtaining Surface Normals using ARMAR-III and ARMAR-6

To evaluate the contact detection and normal sensing capabilities of the tactile sensor presented in section 3.4 the sensor is mounted at the tip of the index finger of ARMAR-III. The sensor is then brought into contact with a surface repeatedly. During the experiment the linear acceleration, the angular deviation from the initial orientation and the pressure is plotted in Figure 3.38. The sensor touches the surface seven times during the experiment. During each contact

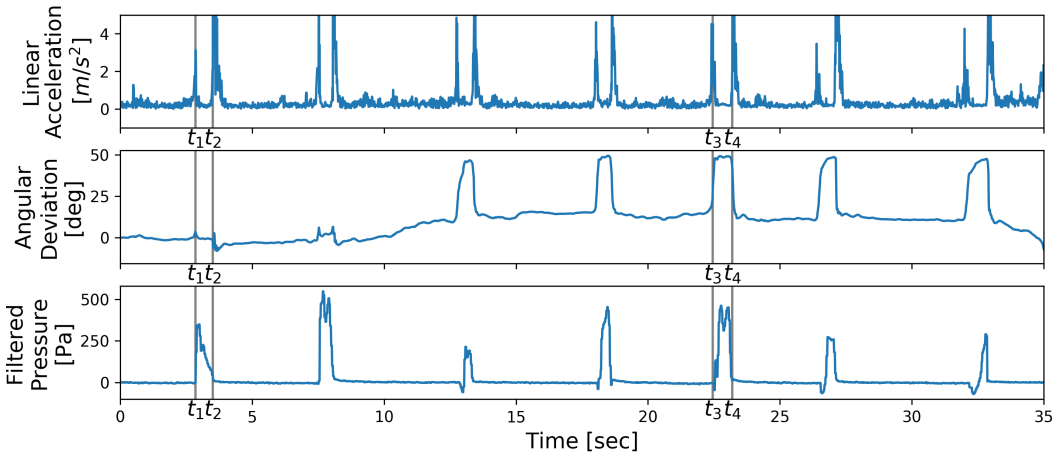


Figure 3.38: Comparison of the sensing modalities of the IMU and the pressure sensor for contact detection. Some contacts can be estimated from the IMU data whereas all contacts can be reliably derived from the drift compensated pressure data. Image taken from Ottenhaus et al. (2018b), © 2018 IEEE.

event one or multiple of the sensing modalities are effected. To detect contacts with the environment, the value of the filtered pressure can be used. This is highlighted for two contact events in the figure, where t_1 marks the begin of the first contact and t_2 marks the end of the first contact, while t_3 and t_4 correspond to the beginning and end of the second highlighted contact. For each contact, the filtered pressure deviates from the zero line and a simple threshold decider can be used to detect contacts. The other two modalities are sometimes effected, but not in all cases, as can be seen between t_1 and t_2, in case of the angular deviation. When using the linear acceleration alone it is difficult to distinguish between establishing contact and breaking contact with the surface. Therefore, the filtered pressure is used to detect contacts.

In a second experiment, the accuracy of the obtained surface normals is evaluated. To obtain the surface orientation the absolute orientation of the IMU within the sensor is used, when a contact is detected. During the experiment, the sensor is brought into contact with a tilted surface. For each contact the ground truth surface tilt and the measured surface tilt is recorded and shown in Figure 3.39. The blue line in the figure denotes the ideal reference line, while each cross corresponds to a measurement. The measured orientation follows the ground truth with an average error of $7.3°$. This orientation error is mostly due to the orientation drift, present within the IMU sensor.

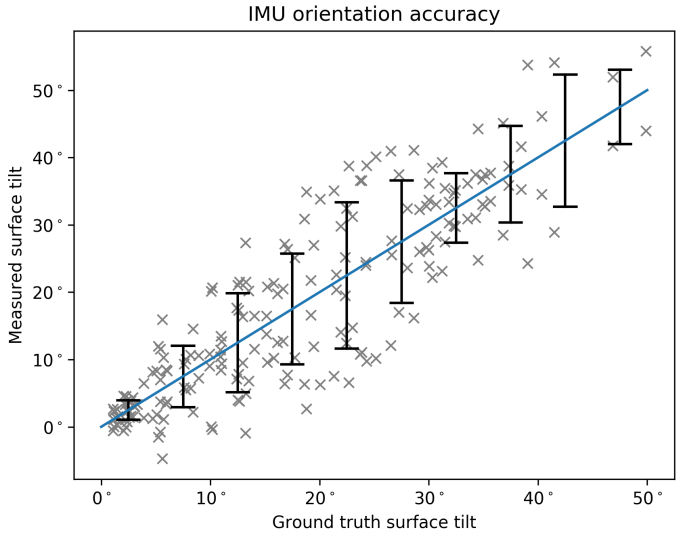


Figure 3.39: The IMU indicates orientation directly when the sensor is pressed against a surface. The relative tilt between the sensor and the surface varies between $0°$ and $50°$. The blue line indicates the ground truth while the error bars indicate the RMSE between the ground truth and measured normals. Image taken from Ottenhaus et al. (2018b), © 2018 IEEE.

Validation using ARMAR-III and ARMAR-6

Two sets of experiments are performed using the humanoid robots ARMAR-III and ARMAR-6. First, the sensor is attached to the arm of ARMAR-6 Asfour et al. (2018). The robot arm has eight joints, also called degrees of freedom (DOF). A human operator guides the arm in zero torque mode to bring the tactile sensor into contact with the object's surface.

In the second experiment, the humanoid robot ARMAR-III performs an autonomous exploration of the object's surface, while the sensor is attached to the robot's fingertip.

In both cases, contact events are detected using the pressure sensing, while the contact locations are calculated from the forward kinematics of the robots. The contact normals are computed from the absolute orientation of the IMU.

After each experiment, the gathered data is used to reconstruct the surface of the object. The reconstruction results are given in Table 3.6. For each experiment, the surface is reconstructed twice. Once without the observed surface normals, using only the contact positions as the input for the standard formulation of GPIS. In the second reconstruction, the extended version of GPIS is used, where the derivative of the underlying Gaussian Process is defined by

Object	Robot	Reconstruction RMSE without normals	Reconstruction RMSE with normals
Box	Simulation	5.2 mm / 25°	2.7 mm / 17°
Sphere	Simulation	1.9 mm / 6°	0.5 mm / 1°
Cylinder	Simulation	6.0 mm / 27°	3.2 mm / 17°
Banana	Simulation	5.9 mm / 27°	1.8 mm / 10°
Ground Coffee	ARMAR-6	4.6 mm / 22°	3.5 mm / 20°
Cheez It	ARMAR-6	7.0 mm / 30°	3.5 mm / 24°
Flat surface	ARMAR-6	9.5 mm / 23°	2.8 mm / 1°
Bowl	ARMAR-III	7.6 mm / 39°	4.7 mm / 17°

Table 3.6: Reconstruction results

the observed surface normals. The reconstruction quality is measured by comparing the distance between the estimated surface and the ground truth object model. Furthermore, the surface normals of the estimate are compared against the ground truth. Exemplary reconstruction results are shown in Figure 3.40. As can be seen in Table 3.6 and Figure 3.40 the addition of surface normals improves the surface estimation accuracy in every experiment. Furthermore, the reconstruction error at sharp edges is improved, as can be seen in Figure 3.41.

3.7 Discussion

In the beginning of the chapter, four questions were asked:

1. How to collect contact information by efficiently selecting the next best touch?
2. How to generate object shape models efficiently, based on the acquired sparse tactile data?
3. How to gather as much information per contact as possible?
4. How to plan grasps based on the approximate object model?

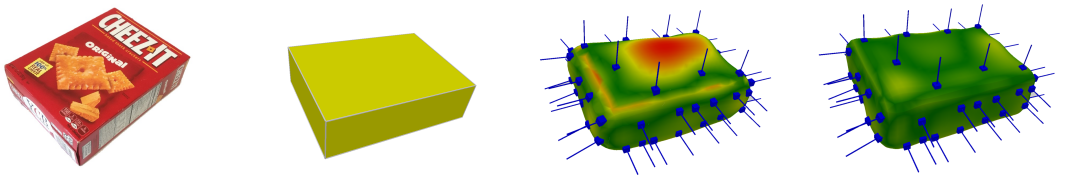
For each question, the previous sections have proposed an approach:

1. **Next-Best-Touch strategy** for efficient exploration.
2. **Extension of GPIS** to include **surface normal observations** for efficient surface estimation.
3. A **surface normal and contact sensor** to provide the necessary sensing modalities.

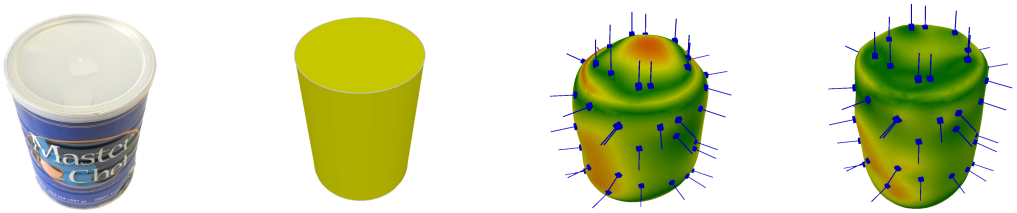
4. **Grasp planning** by using existing grasp planners in the form of the skeleton-based grasp planner.

By combining all proposed approaches, it is possible to generate grasp candidates for unknown objects. First, the robot explored the object, following the Next-Best-Touch exploration strategy using the proposed tactile sensor. Thereafter, the obtained contacts and contact normals are used to estimate the object's surface with the extended formulation of GPIS. Finally, the estimated surface is fed to the grasp planner, that is then able to generate grasp candidates for the unknown object.

This shows that tactile exploration alone is a feasible approach to grasp unknown objects. While the proposed methods aim to explore efficiently and to gather as much information per contact as possible this exploration process still takes time. Therefore, one question arises: *How many exploration actions are necessary for grasping?* The next chapter will introduce an approach to generate grasp candidates by combining visual and tactile information, which will greatly reduce the number of necessary exploration actions.



(a) *"Cheez-It"* (b) Ground truth model (c) GPIS reconstruction normals excluded (d) GPIS reconstruction normals included



(e) *"Ground Coffee"* (f) Ground truth model (g) GPIS reconstruction normals excluded (h) GPIS reconstruction normals included

Figure 3.40: Comparison of the GPIS reconstruction results with included and excluded normal information. The reconstruction error is displayed using a color-coding where green indicates a small error and red corresponds to a large error. For both objects, the GPIS reconstruction with included normals follows the ground truth more accurately. Images taken from Ottenhaus et al. (2018b), © 2018 IEEE.

(a) GPIS reconstruction without normals, RMSE: 20.7 mm (b) GPIS reconstruction with normals, RMSE: 9.5 mm

Figure 3.41: Reconstruction quality near edges of the objects. The closest points on the reconstruction and the actual object edge are shown as colored lines. Green indicates small errors whereas red denotes larger errors. Images taken from Ottenhaus et al. (2018b), © 2018 IEEE.

4 Visuo-Haptic Grasping

Using tactile exploration alone to estimate a surface model of an unknown object can take a long time, since the object has to be touched several times from multiple sides to gather enough contact points. Therefore, in the second part of the thesis the tactile exploration is combined with visual perception to get an initial view of the object. The approach is split into the three classical parts: sense, plan and act, as shown in Figure 4.1. The robot captures a point cloud of the object and explores the unseen back of the object to gather contact information. The visually acquired points and the tactile contact information is then fused in one joint model of the object. Thereafter, grasp candidates are generated and one candidate is chosen for grasp execution. Finally, the robot executes the chosen grasp candidate by moving the hand to the target 6D pose and closes the fingers to lift the object.

In order to find a suitable approach to implement the different parts several questions arise:

1. How should the exploration targets be chosen?
2. How many tactile exploration actions are necessary for grasping?
3. How can visual and tactile information be fused?
4. How can grasp candidates be generated, based on the fused data?
5. Which of the available grasp candidates has the highest success rate?

The following sections cover these questions in detail. First, the experiment setup is introduced in section 4.1. Thereafter, section 4.2 gives an overview over the developed grasping pipeline, that combines all necessary steps, including perception and model estimation, grasp candidate generation, selection and grasp execution. In section 4.3 a data-driven grasp metric is introduced that enables the robot to rate available grasp candidates according to grasp success probability. The complete pipeline is evaluated in section 4.4 and transferred to the robot ARMAR-6 in section 4.5, including validation experiment.

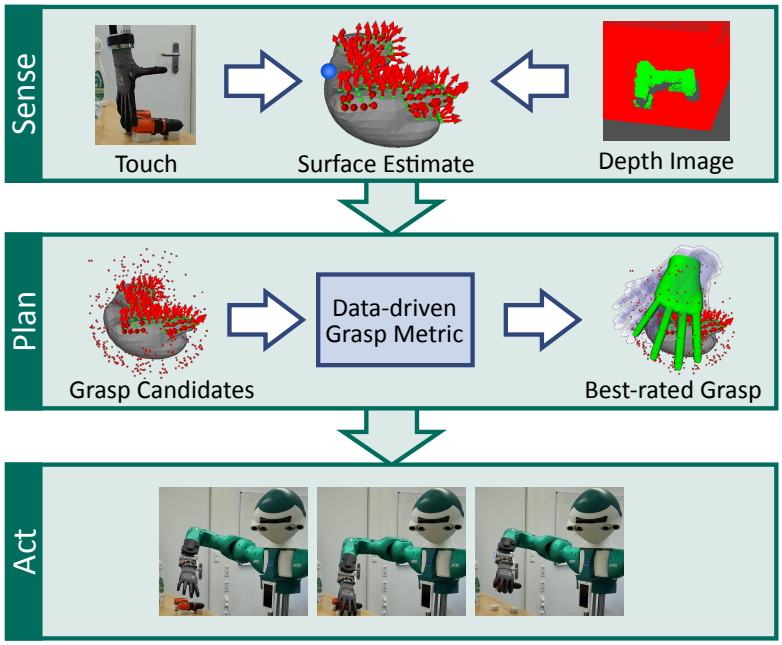


Figure 4.1: Visuo-haptic grasping pipeline: Visual and tactile information are fused in one surface model. Grasp planners synthesize grasp candidates, which are rated by a data-driven grasp metric, implemented as a neural network. The best-rated grasp candidate is selected for execution. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

4.1 Experiment Setup

During the grasping experiments, the humanoid robot ARMAR-6 is located in front of a table. On top of the table, one unknown object is placed. The approximate location of the object is known beforehand, i.e. segmentation of the object is not a central part of this thesis. The robot uses its depth camera, located in its head to capture a point cloud of the scene. It then separates the points belonging to the object from the background and the table. Once the object is segmented, the robot chooses a next-best-touch location and moves its hand to the exploration target, until contact with the object is measured. The exploration is repeated up to five times, while the optimal amount of exploration actions is determined during the evaluation in section 4.4. Thereafter, the visual and tactile information is fused and grasp candidates are generated. One candidate is selected for execution, according to the rating provided by the data-driven grasp metric. Finally, the robot executes the chosen grasp candidate by approaching the chosen target pose and closes the hand around the object.

This experiment setup is constructed in simulation as well as using the hu-

manoid robot ARMAR-6. The setup in simulation and in reality is similar; however, some modifications are necessary: In simulation, the visual perception is simulated in form of a virtual depth camera, provided by the ArmarX[1] simulation environment. This virtual depth camera renders a depth map of the scene. This depth map is then converted to a point cloud. The tactile exploration is also simulated using the ArmarX simulator. The models of the hand and the object are continuously checked for collisions with each other. If such a collision is detected a tactile contact is generated at the location of the collision. During the simulation of the grasp execution, only the robot's hand is considered. The hand is placed directly at the target grasp pose and the fingers are closed. If the hand configuration leads to a force closure around the object, the grasp is considered successful.

During the experiments on the real robotic system, a Primesense Carmine camera mounted in the robot's head performs the visual perception. To this end, the head of the robot is pointed at the object lying on the table. The ArmarX framework provides a driver for the depth camera, so that the point cloud is directly available. At the time of the experiments, the hands of the robot were not equipped with any tactile sensors. However, the robot has a precise force torque sensor (FT sensor) mounted in the wrist. Using the measurements from the FT sensor tactile contacts can be inferred. This process will be described in detail in the section transfer to ARMAR-6 in section 4.5. The fusion of visual and tactile data as well as the grasp candidate generation and selection do not have to be adapted for the real world experiments. The execution of the grasps however had to be adopted for the real robot, as the used hand is underactuated, with only two actuated degrees of freedom. One motor is used to control the thumb, while the other motor controls all fingers. The publication by Asfour et al. (2018) offers a detailed description of the hand.

4.2 Grasping Pipeline

In this section, one grasping pipeline will be introduced that combines all necessary steps to grasp unknown objects based on visual perception and tactile exploration. The developed grasping pipeline consists of six stages: Visual perception, haptic exploration, surface estimation, grasp candidate generation, candidate scoring and selection and grasp execution. The pipeline is depicted in Figure 4.2 and explained in the following.

[1] Available online: `https://armarx.humanoids.kit.edu/`

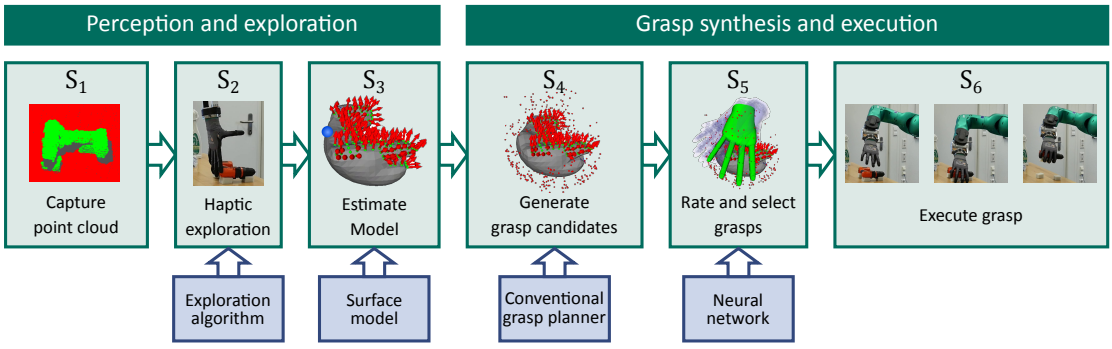


Figure 4.2: Full visuo-haptic grasping pipeline integrating visual perception, tactile exploration, model estimation, grasp candidate generation and scoring and grasp execution. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

4.2.1 Visual Perception

In the first pipeline step (S_1*: Visual Perception*), the robot uses the depth camera in its head to capture a point cloud of the unknown object. The object lies on a flat supporting plane and can therefore be easily separated from the background. The RANSAC (Fischler and Bolles, 1981) algorithm is used to find the supporting plane in the point cloud. All points belonging to the plane are subtracted, and the remaining points are filtered for outliers, so that only points belonging to the object remain. Thereafter, the point cloud is down sampled and normals are computed from the points. The result of the first pipeline step is a set of points with normals on the front and top surface of the object.

4.2.2 Tactile Exploration

During *Tactile Exploration* (S_2) the exploration algorithm presented in section 3.1 is initialized using the point cloud from S_1. Then the next-best-touch is chosen, according to the Information Gain Estimation Function, see subsection 3.3.2. The robots explores several points on the surface of the object, primarily in unseen regions at the back of the object.

4.2.3 Surface Estimation

The visually acquired point cloud and the tactile contact points are combined in one GPIS model in the third pipeline stage (S_3). The computed normals from the point cloud are used to define the gradient of the *Implicit Surface Potential*

(ISP) of the GPIS, as introduced in subsection 3.2.2. During the surface estimation process, GPIS requires the inversion of the covariance matrix. To speed up this matrix inversion the visual point cloud is further reduced to about 100 points, using methods from the point cloud library (PCL, Rusu and Cousins, 2011).

4.2.4 Grasp candidate generation

During the *Grasp candidate Generation* (S_4) stage the *Implicit Surface Potential* is converted to a triangle mesh using the marching cubes algorithm. Thereafter, two different grasp planners are used to generate grasp candidates. To this end, the grasp planners optimize a given grasp metric with respect to the surface mesh and the model of the robot hand. The used grasp planners are both part of the Simox robotic toolbox, developed by Vahrenkamp et al. (2013):

- The first grasp planner is the standard grasp planner provided by Simox. It selects a random triangle from the triangle mesh and places the robot's hand with respect to the normal of the triangle. Thereafter, the hand is moved towards the object and the hand is closed until all fingers collide with the surface mesh. Then the grasp metric is calculated.
- The second grasp planner is the skeleton-based grasp planner introduced by Vahrenkamp et al. (2018). The grasp planner first extracts a skeleton from the estimated surface mesh. The straight sections of the skeleton are identified and checked for possible hand placements. Additional possible hand placements are generated at the end points of the skeleton. A detailed description of the skeleton-based grasp planner can be found in Vahrenkamp et al. (2018).

4.2.5 Candidate Rating and Selection

The *Rating and selection* (S_5) stage first estimates the success probability for each grasp candidate using the data-driven grasp metric. Each candidate is encoded as a 6D grasp pose and a voxel grid containing a local view of the gathered visual and tactile points around the grasp pose. Based on the grasp pose and the voxel grid the grasp metric predicts the grasp success probability between 0 and 1 for each grasp candidate. Then the grasp candidate with the highest success probability is chosen for execution.

4.2.6 Grasp Execution

An ArmarX state chart implements the *Grasp Execution* (S_6) stage. Within this state chart, the position of the hand pose is controlled using a Cartesian velocity controller. The state chart first moves the hand just above the grasp pose and the starts lowering the hand. The force torque sensor measures the interaction force between the hand and the object during the approach phase. When a given force threshold is exceeded the fingers of the hand are closed and the object is lifted.

4.3 Data-Driven Grasp Metric

When the pipeline execution reaches the stage of the grasp metric, many possible grasp candidates have been generated by the previous pipeline stage. Traditional grasp planners, using the estimated surface of the object, have generated the grasp candidates. While these grasp planners calculate grasp metrics on their own, these ratings often do not correspond well with real grasp success rates. The main reason is that the grasps were planned using the estimated surface. This surface deviates from the actual object, in particular in unseen regions of the object. Therefore, a different metric has to be developed that is able to predict the grasp success when only partial information is available. To this end a new data-driven grasp metric is developed. This metric has two inputs:

- The 6D grasp pose, relative to the object center.
- A 3D voxel grid containing features related to the captured point cloud and the explored points on the object.

This grasp metric is implemented as a deep neural network, as shown in Figure 4.3. The network structure is inspired by Voxnet, introduced by Maturana and Scherer (2015). In the original Voxnet publication, the neural network is used to classify objects based on a 3d occupancy grid calculated from a point cloud. The structure of the original Voxnet architecture is extended to allow the prediction of the grasp success probability. In the Voxnet architecture, the input is encoded as an occupancy grid. For the grasping application, the input is expanded to contain two features per voxel. The first feature measures the distance of the voxel center to the nearest point in the fused visual and tactile point cloud. The second feature is derived from the Implicit Surface Potential that was calculated during the GPIS surface estimation. As a second input, the 6D grasp pose relative to the object center is added. The pose is encoded as

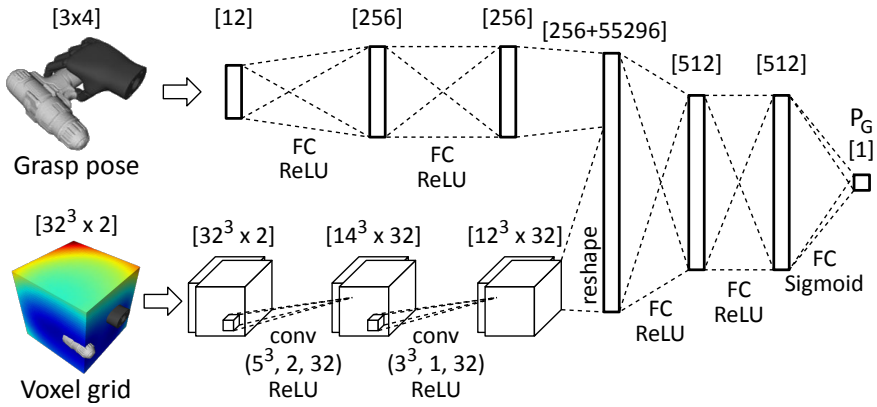


Figure 4.3: Data-driven grasp metric implemented as a deep neural network. The first input is derived from the homogeneous grasp pose matrix, taking the top three rows of the matrix (3×4). The grasp pose is preprocessed by two fully connected layers. The second input encodes the captured visual and tactile points as a 3D voxel grid. The voxel grid is processed by two 3D convolutional layers $conv(d, s, f)$, where d is the kernel size, s is the stride and f denotes the number of filters. At each voxel, center two features are observed: The *Implicit Surface Potential (ISP)* and the distance to the closest point within the fused visual and tactile point cloud. The information is fused in two fully connected layers, resulting in the success probability of grasp execution. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

the top three rows of the homogeneous pose matrix (3×4). The bottom row of the pose matrix is omitted, since it does not contain any information. The output of the network is a single value that predicts the grasp success probability between 0 and 1. The structure of one training sample is given in Table 4.1.

Type	Dimension	Description
Input	4×3	6D Pose
Input	32^3	3D Voxel grid
Output	1	Grasp success probability

Table 4.1: Structure of one training sample

4.3.1 Training Data Generation

The training of deep neural networks requires large amounts of labeled training samples. In the present application case, one training sample consists of the grasp pose, the voxel grid and the corresponding grasp success probability. As

introduced in the related work (subsection 2.2.2) four sources of training data can be considered:

1. *Hand labeling* of training samples.
2. A self-supervised approach, executed directly *on the target system*.
3. *Learning from demonstration*: A human teacher provides training examples.
4. Dataset creation by *training data generation in simulation*.

As discussed in the related work, the approaches *hand labeling*, *learning on the target system* and *learning from demonstration* have high costs, either with respect to necessary human work or due to the need of thousands of robotic experiments. Therefore, in this thesis the necessary dataset was created in simulation.

The robot is placed in a simulated environment in front of a table with an unknown object on top. The objects used for training are taken from the KIT object database (Kasper et al., 2012) and the YCB object and model set (Calli et al., 2015). The simulation chooses one object from one object set (KIT or YCB) at random and loads the corresponding object mesh. The object mesh is rotated and scaled randomly and placed on the table. Then the pipeline stages $S_1 \dots S_4$ are executed using simulated cameras and simulated tactile exploration. Thereafter, the grasp pose and corresponding voxel grid are calculated for each grasp candidate. The grasp success probability is determined via a stochastic process, as will be described in the next section.

4.3.2 Sim2real Transfer Considerations

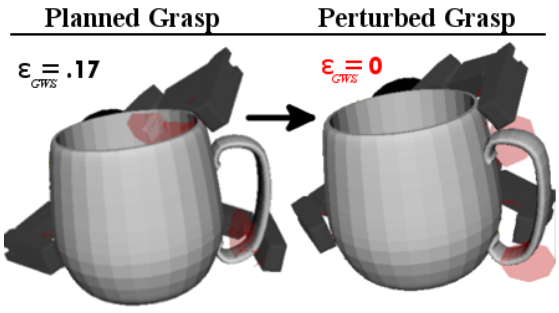


Figure 4.4: The ϵ-metric is not robust under pose uncertainty, as shown by Weisz and Allen (2012), © 2012 IEEE.

In the field of humanoid robotics, a perfect simulator is hard to find, especially when it comes to grasping. In a study regarding grasp quality metrics Weisz

and Allen (2012) have determined that grasp quality metrics, such as the ϵ-metric, are not robust regarding pose uncertainty. In their work, they state: „*[...] we have shown that the planned ϵ of a planned grasp is not predictive of the probability of achieving a force closed grasp in the presence of uncertainty, neither in simulation nor physical experiments.*“ The effect of this uncertainty is shown in Figure 4.4. Kappler et al. (2015) made similar observations in their work: „*[...] the metric based on physics simulation is a more consistent predictor for grasp success than the standard ϵ-metric. [...] labels based on the physics-metric are less noisy than those from the ϵ-metric and therefore lead to a better classification performance.*“

Therefore, the grasp success probability is predicted using the approach proposed by Weisz and Allen (2012). A given grasp candidate pose is perturbed slightly in position and orientation. The simulator determines if this perturbed grasp will still result in a force closure configuration. This process is repeated multiple times and the grasp success probability is determined by averaging over the force closure results.

4.3.3 Preprocessing and Network Training

Training of a neural network requires the input and output data to be encoded in a suitable format. This includes the dimensionality of the data as well as the representation. The input data is preprocessed and normalized to speed up the network training process.

- The *grasp pose* is encoded as a 4×4 matrix. The bottom row of this pose matrix does not contain any information, as it is always 0001. Therefore, only the tow three rows are input to the neural network as a flat vector.
- The base coordinate system of the *voxel grid* is aligned with the grasp pose. The center of the grid is aligned with the position of the grasp candidate pose. The grid axis are aligned with the axis of the grasp pose. The grid therefore represents a local view, relative to the grasp candidate. In particular, the relative location of the hand is fixed within the voxel grid. The grid has a side length of $30\,\mathrm{cm}$.
- The *grasp success probability* is encoded as a singular value between 0 and 1.

An aligned voxel grid has multiple advantages: The grasp success probability is only dependent on the local object geometry, which is encoded in the grid. The network is not required to learn the transformation from a global orientation to

the relevant local orientation, so the learning can focus on the relevant features. A local view allows the transfer of grasps between objects with similar local geometry.

The training set consists of $1.6\,\mathrm{million}$ samples that were generated in simulation. During the training $50\,\%$ dropout is applied to the CNN part of the network. The chosen leaning rate was 10^{-4}. The network weights converged after 4 epochs.

Figure 4.5: Object test set for evaluation (1-4 YCB, 5-12 KIT): 1 Power Drill, 2 Apple, 3 Racquetball, 4 Jello, 5 Bottle, 6 Shampoo, 7 Spray Bottle, 8 Vitalis Cereal, 9 Tomato Soup, 10 Schaumküsse, 11 Koala Candy, 12 Fruit Drink. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

4.4 Evaluation

The evaluation focuses on the effectiveness of the data-driven grasp metric (pipeline stage S_5) and the tactile exploration (S_2). Two main questions are addressed:

- What is the benefit of the data-driven grasp metric, when compared to conventional grasp planning?
- How much tactile exploration is necessary for successful grasping?

The evaluation uses a test set comprised of 12 unseen test objects, depicted in Figure 4.5. The objects are taken from the KIT and YCB object sets. The test set is separated into two categories: Unknown objects and objects that are unknown, but have familiar shapes. The unknown objects (*Power Drill* and *Spray Bottle*)

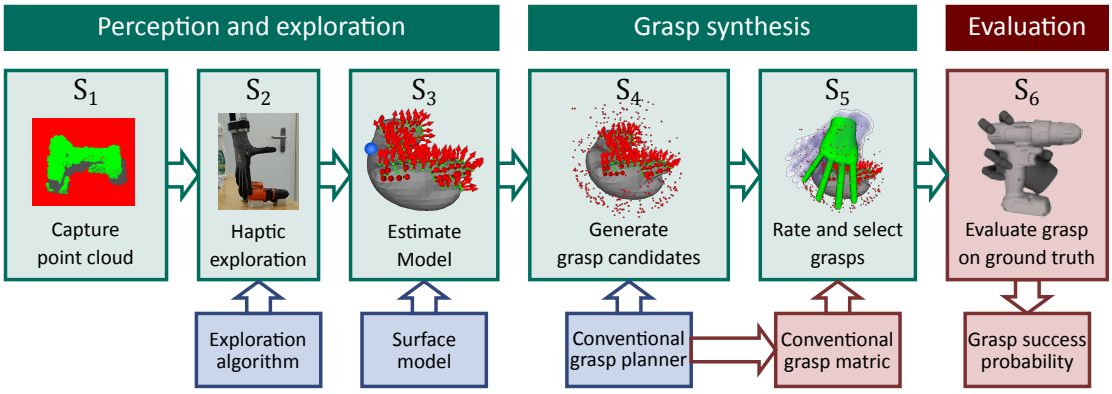


Figure 4.6: Evaluation pipeline: The grasping pipeline is adapted (red parts) to provide baseline results and evaluation in simulation. The data-driven grasp metric is replaced with the grasp metrics computed by the grasp planners. The execution is replaced by the evaluation of the grasp candidate against the ground truth mesh of the object. Image adapted from Ottenhaus et al. (2019), © 2019 IEEE.

differ significantly from the raining set. The familiar objects from the test set have similar shapes to some objects within the training set.

The evaluation section is structured as follows: First, a baseline is established using conventional grasp planners. The data-driven grasp metric is then compared against this baseline. Finally, the grasp success probability is determined depending on the number of exploration actions.

4.4.1 Baseline and Evaluation Pipeline

In their work, Bjorkman et al. (2013) explore unknown objects and fuse tactile and visual data using GPIS. The focus of the publication is on the exploration and modeling, however the authors suggest that the final GPIS model can be used for grasp synthesis by means of a conventional grasp planner. This idea can be easily applied to the proposed grasp pipeline by replacing the data-driven grasp metric with the grasp metrics computed by the conventional grasp planners, as shown in Figure 4.6. During evaluation, the execution stage of the pipeline is also replaced. The simulator moves the simulated hand of the robot to the pose of the selected grasp candidate. Then the closure probability is calculated using the ground truth mesh and the hand model, as described in subsection 4.3.2. A grasp candidate is considered successful, if the computed grasp success probability is above $80\,\%$.

4.4.2 Benefit of the Data-Driven Grasp Metric

The evaluation of the data-driven grasp metric compares the predicted grasp success with the actual grasp success. Each the predicted grasp success and the actual grasp success are considered as binary values. The predicted grasp success is derived from the predicted grasp success probability (P_G) by the data-driven grasp metric. If P_G is larger than a given threshold $P_{\delta,G}$ the prediction is considered positive. The same threshold mechanism is applied to the actual grasp success, derived from the ground truth mesh. For the given grasp candidate the force closure probability ($P(FC)$), according to Weisz and Allen (2012), is calculated. If $P(FC)$ is larger than a given threshold $P_\delta(FC)$ the ground truth is considered a successful grasp. During the evaluation, the values $P_{G,\delta} = 0.95$ and $P_\delta(FC) = 0.8$ were chosen. The possible combinations of predicted grasp success and actual grasp success are listed in Table 4.2.

Prediction	**Ground truth**	**Type**
Failure	Failure	True negative (TN)
Failure	Success	False negative (FN)
Success	Failure	False positive (FP)
Success	Success	True positive (TP)

Table 4.2: Possible combinations of predicted grasp success and actual grasp success.

After the possible four cases have been enumerated in the table, the question is if all of the cases are relevant during the evaluation. To this end, the final grasp execution by the robot is considered. When the robot is tasked with grasping an unknown object, the pipeline stages S_1 through S_4 are executed. Within S_4 arbitrarily many grasp candidates can be generated. The grasp metric is evaluated for each of the grasp candidates. However, the robot has to pick exactly one grasp candidate for execution. This candidate will be picked from the set that the grasp metric considers successful. This corresponds to the two bottom rows of the table.

The data-driven grasp metric is evaluated by evaluation the *precision* of the predictions. The precision is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \,. \tag{4.1}$$

In this case, the precision of the predictions corresponds to the question: How

good is the data-driven grasp metric in predicting grasp candidates, that are actually successful?

To answer this question the evaluation loads each of the 12 objects from the test set into the simulator. Each object is rotated randomly in five different orientations. For each of the orientations the evaluation pipeline is run, while 800 grasp candidates are generated. For each of the grasp candidates the predicted grasp success and the actual grasp success are computed. Thereafter, only the grasp candidates are considered for which the grasp metric predicted grasp success.

The same procedure is repeated to generate the ground truth data. In this case, the data-driven grasp metric is replaced by the grasp metric computed by the conventional grasp planner, as described in subsection 4.4.1.

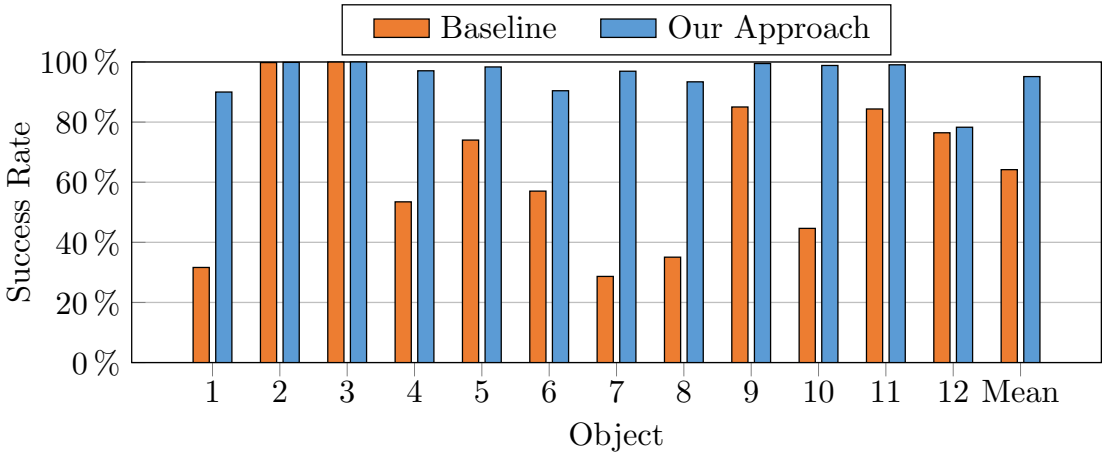


Figure 4.7: Comparison of the data-driven grasp metric (blue) against the baseline (orange). The data-driven grasp metric outperforms the conventional grasp planning baseline for complex shaped objects. The difference is most significant for object 1, 7, and 8. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

Finally, the success rate in terms of the prediction precision of the baseline and the data-driven grasp metric are compared or all 12 objects. The results are given in Figure 4.7. Each of the bars in the figure corresponds to the average of the prediction success. As the figure shows the data-driven approach outperforms the baseline by a significant margin, i.e. the data-driven approach yields higher or equal average prediction success for all objects. In the following the results are analyzed for the different groups of objects: ball-shaped objects (2 and 3), cylindrical (5, 6, 9 and 11), boxes (4, 8, 10 and 12) and complex objects (1 and 7).

For the ball-shaped objects both the baseline and the data-driven grasp metric yield almost perfect results. This can be explained by the surface estimation produced by GPIS. The captured depth image from the front of the object describes a half sphere. In this case, the GPIS surface completion generates the rest of the sphere. Therefore, the estimated surface is quite close to the actual surface of the object and the planned grasp candidates have a high success rate already. If all of the predicted grasp candidates work as actual grasps, the prediction of the chosen grasp metric does not matter, since the expected grasp success is almost $100\,\%$, independent of the chosen grasp candidate.

This consideration can be extended to the cylindrical objects from the test set. The GPIS estimate follows the ground truth surface of the object closely. The largest deviations are to be expected at the ends of the cylindrical parts, i.e. the flat bottom and top side. However, the baseline can still achieve high grasp success rates for objects 5, 9 and 11. The baseline drops below to $60\,\%$ success rate for object 6, namely the shampoo. The reason is that the shampoo has an oval shape. The GPIS estimate completes the object following a round shape at the unseen back. This can lead to misplaces grasp candidates. Here the data-driven grasp metric can identify these incorrect grasp candidates and can improve the grasp success rate significantly.

The same consideration can be made for the boxes within the test set. The GPIS estimate competes the unseen back of the object with a round shape. Therefore, the estimated surface of the object is too large, i.e. the enclosed volume of the GPIS model is larger than the actual object. Since the boxes have sharp edges, the depth camera can only observe up to three sides of the boxes at once. This means that at least three sides of the objects are estimated incorrectly. Therefore the predicted grasp candidates deviate from the actually successful grasp candidates in many cases.

The largest improvement can be observed for the complex shaped objects 1 and 7: the spray bottle and the power drill. For these objects, the GPIS estimate deviates from the actual object surface in many cases. An example can be seen in Figure 4.8 on the right side. Each arrow in the figure denotes one grasp candidate. The arrows are colored according to the grasp success of the respective grasp candidate. The baseline produces many false positives, while the data-driven grasp metric can filter out many of these false positives, leading to a higher grasp success rate.

Overall, the experiments show that the data-driven grasp metric outperforms the conventional baseline for all tested objects. Regarding the grasp failures,

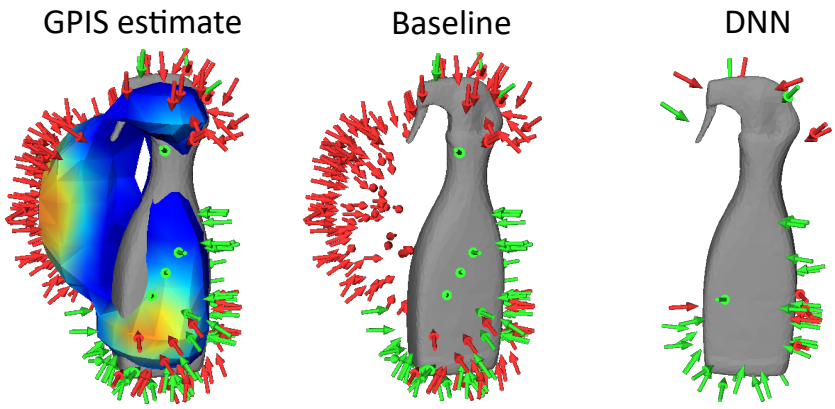


Figure 4.8: Comparison of the baseline and the data-driven grasp metric: True positives are shown as green arrows and false positives are shown as red arrows. The GPIS estimate (colored surface) deviates from the actual object. The data-driven grasp metric is able to filter out most of the false positive grasp candidates. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

the baseline fails in $35\,\%$ of the cases while the data-driven grasp metric fails in only $5\,\%$ of the tested cases, resulting in an average of 7-times less grasp failures. Exemplary successful grasp candidates are displayed in Figure 4.9.

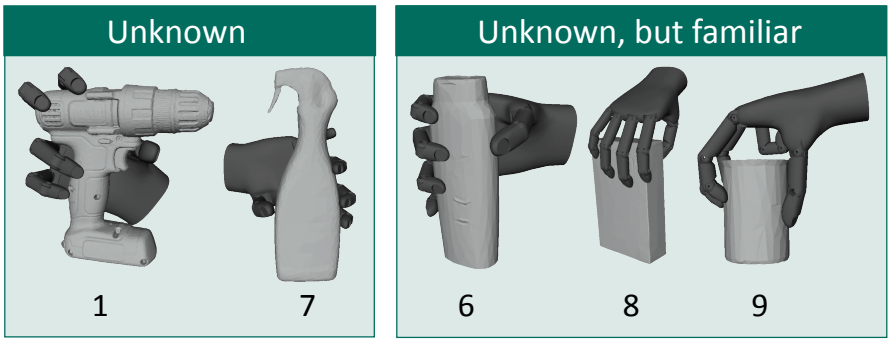


Figure 4.9: Examples grasps generated by the data-driven grasp metric. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

4.5 Transfer to ARMAR-6 and Validation

The evaluation showed, that the data-driven grasp metric is able to reliably predict the success probability of a given grasp candidate in simulation. When transferring to the humanoid robot ARMAR-6 several questions have to be answered:

- How does the real robot differ from the simulation?

- How many tactile exploration actions are necessary?
- How should the grasp execution be implemented?

The following sections address these questions and describe the transfer of the approach from simulation to the real robot.

4.5.1 Validation Setup

The developed grasping pipeline is validated using the humanoid robot ARMAR-6. The validation setup is depicted in Figure 4.10.

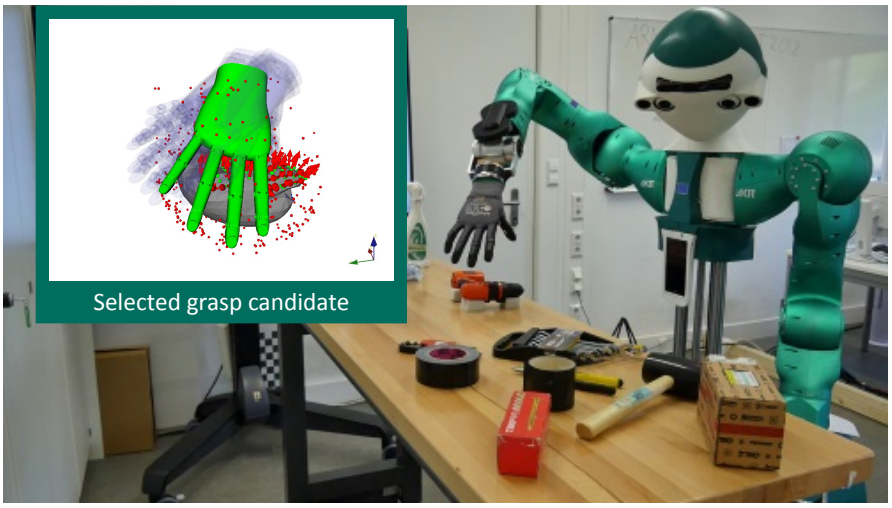


Figure 4.10: Validation setup: The humanoid robot ARMAR-6 is tasked with grasping an unknown object from the workbench. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

The robot is placed in front of a table with an unknown object on top. The object should be grasped and lifted from above with the right hand.

The main differences of the real robot system compared to the simulation are:

1. The robot must actively control the arm so that the hand is moved to the desired destination.
2. The hand has no tactile sensors; however, the 6D force / torque sensor in the wrist can be used to infer contacts.
3. The hand is underactuated and has two motors. The first motor controls the thumb while the second motor controls all the fingers.

4.5.2 Control of the Robot's Arm

During the experiments, the robot moves its arm using velocity bound Cartesian control. The control of the arm is split into three different layers, namely:

- A **Cartesian position controller** that takes in the target pose of the hand and the actual pose of the hand and calculates a Cartesian velocity.
- A **Cartesian velocity controller** that converts a Cartesian velocity to a joint velocity vector by means of the pseudo-inverse Jacobian.
- A **Null-space controller** that calculates the null-space of the current joint configuration from the Jacobian. The controller moves the joints in this null-space to avoid the joint limits.

As Cartesian position and velocity control are standard, only the null-space control will be explained here. If the Jacobian of the kinematic chain of the arm is given by J then

$$J_\theta \dot{\theta} = \dot{x} \tag{4.2}$$

denotes the relation between the joint angle velocity $\dot{\theta}$ and the end-effector velocity $\dot{x}$. In case of ARMAR-6 the dimension of θ is 8, i.e. the robot has two more joints than needed to sweep a 6D workspace. However, these joints are very useful to increase the volume of the workspace. In particular, the clavicle joint in the shoulder allows the robot to reach far in front of it. The kinematic structure of the arm has 8 degrees of freedom (DOF), therefore the dimension of the null-space is 2. The null-space of the arm configuration coincides with the null-space N_θ (also known as kernel) of the Jacobian:

$$N_\theta = \left\{ \dot{\theta} | J_\theta \dot{\theta} = 0 \right\} . \tag{4.3}$$

In matrix form this can be written as

$$J_\theta N_\theta = 0 \tag{4.4}$$

The goal of the null-space controller is to calculate a joint velocity of the arm, that moves the arm in this null-space, i.e. changes the configuration of the arm without moving the end-effector. The null-space controller uses the available motion in the null-space to move the joints away from the joint limits. First, a desired *joint limit avoidance velocity vector* $\dot{\theta}^a$ is calculated that moves all joints away from their limits.

$$\dot{\theta}_i^a = \cos\left(\pi \frac{\theta_i - \theta_i^-}{\theta_i^+ - \theta_i^-} \right) \tag{4.5}$$

where θ_i^- and θ_i^+ denote the lower and upper joint limit of joint i. Then this desired velocity vector $\dot{\theta}^a$ is mapped onto the available null-space of the Jacobian.

$$\dot{\theta}^N = \sum_i \frac{N_{\theta,j} N_{\theta,j} \cdot \dot{\theta}^a}{\|N_{\theta,j}\|^2} \tag{4.6}$$

Here $N_{\theta,j}$ denotes the j-th column of the null-space matrix N_θ. Finally, the resulting null-space joint limit avoidance velocity is scaled with a proportional factor p_N and added to the joint velocity $\dot{\theta}$

$$\dot{\theta}' = \dot{\theta} + p_N \dot{\theta}^N \tag{4.7}$$

When $\dot{\theta}$ is commanded to the joints of the robot's arm the desired Cartesian velocity is reached and the available null-space is used to avoid the joint limits. Values of $p_N \in [1..2]$ have proven suitable for most operations during tests on the robot. In practice, this additional null-space control has been beneficial to the overall Cartesian control of the robot's arm, as joint limits are avoided as long as possible. If a joint limit is reached during Cartesian control without the null-space controller active, the end-effector usually deviates from the desired velocity vector. Therefore, the null-space controller developed in this thesis was added to the standard implementation of the Cartesian controllers used for ARMAR-6 and ARMAR-III. The implementation is part of ArmarX and available online[2]

4.5.3 Emulation of Tactile Sensing

At the time of the experiments, the robot's hand did not include any tactile sensors. However, a 6D force/torque sensor is available in the wrist of the hand, shown in Figure 4.11. This FT-sensor can be used to infer tactile contacts: Just before a contact between hand and object is expected, the current force values of the FT-sensor are stored as F_0. Then the difference between the current measured force and the initial force value is compared and a tactile contact is assumed if this difference exceeds a predefined threshold ΔF.

$$\text{Tactile contact if } \|F - F_0\| > \Delta F \tag{4.8}$$

The position of the tactile contact can be inferred from the model of the hand.

[2] `https://gitlab.com/ArmarX/RobotAPI/blob/master/source/RobotAPI/libraries/core/CartesianVelocityController.cpp`

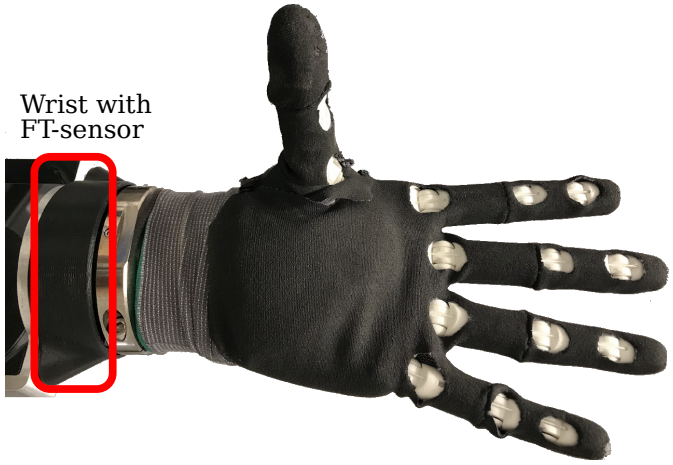


Figure 4.11: The underactuated hand of ARMAR-6. A 6D force/torque sensor is mounted in the wrist of the hand. Image adapted from Asfour et al. (2018), © 2018 IEEE.

4.5.4 Control of the Underactuated Hand

During initial grasping experiments, fine hand-control was mostly neglected and the predominant grasping strategy can be summarized as: *Move the hand's TCP to the grasping point on the object and close the fingers*. This approach worked for large and bulky objects, when grasped from the side. Grasping objects from the top with the underactuated hand is more challenging and requires the coordination of hand and finger motions.

Underactuated Mechanism of the ARMAR-6 Hand

The ARMAR-6 hand features a mechanism in the palm that realizes its underactuation. This mechanism follows the design of the TUAT/Karlsruhe mechanism first described by Fukaya et al. (2000). Starke et al. (2018) described the implementation of the mechanism in detail for the KIT prosthetic hand, while the mechanism of the ARMAR-6 hand is similar to the one in the KIT prosthetic hand.

The mechanism in the ARMAR-6 hand can be summarized as follows: The hand has two motors: one for the fingers and one for the thumb. The thumb motor is connected via a tendon to the thumb. The tendon is routed through the thumb and enables the closing of both thumb joints. The finger motor is connected via a tendon pulley mechanism to the fingers. Each finger has three joints and the hand has four fingers, plus the thumb. The finger motor has to pull 12 joints and has to overcome the losses within the mechanism, while the

thumb motor only has to pull two joints.

If both motors are commanded to close their respective fingers, the thumb will close much faster as the rest of the fingers. In practice, this resulted in failed grasping attempts, as the thumb moved the object out of the hand, while fingers were still closing.

A second challenge lies in the coordination of the hand posture and the finger closing strategy. When a small object is approached from above, the fingers and thumb usually touch the table before the hand touches the object. In this configuration, neither fingers nor thumbs can be closed because the table blocks them. Therefore, the hand position and closing of the fingers must be coordinated. The goal of the coordination is to realize a „collecting movement“: The fingertips and thumb should remain as close to the table surface as possible while the hand closes.

Development of a Coordinated Finger and Thumb Grasping Strategy

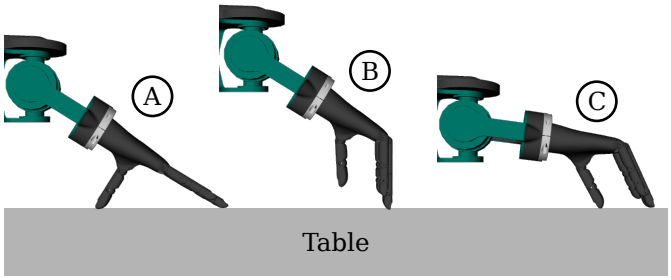


Figure 4.12: Design of the grasping strategy: Coordination of hand position and rotation with the closing of the fingers and the thumb.

In the initial state, both fingers and thumb are in contact with the table surface; see state (A) in Figure 4.12. To allow the fingers to move, the hand must be raised slightly. The hand is lifted until the fingers are orthogonal to the table surface (state B). The hand must then be lowered again until the fingers are completely closed. The same applies to the thumb. However, the thumb is shorter than the fingers. This results in two contradictory requirements for the position of the hand: Closing the fingers requires a higher raising of the hand than closing the thumb. Therefore, by simply controlling the position of the hand it is not possible to ensure that both fingers and thumbs are at a small distance from the table surface while the hand is being closed. This can be compensated by an additional rotation of the hand during closing. The rotation causes the thumb to move closer to the table as the fingers move away from the table (state C).

Parameterization of the Grasping Strategy

To enable the robot to perform a coordinated grasping movement, this must first be described as a trajectory. The relevant parameters of this trajectory include (1) the position of the hand, (2) the orientation of the hand, (3) the posture of the thumb and (4) the posture of the fingers.

In detail, these four parameters are divided into:

1. 3D position of the Tool Center Point of the hand.
2. 3D orientation matrix of the hand.
3. Tendon length change of the thumb.
4. Tendon length change of the fingers.

The change in tendon length for thumb and finger is measured in percent, where $0\,\%$ corresponds to the hand opening and $100\,\%$ means completely closed.

Model of the Fingers

The first idea is to develop a suitable grasp trajectory using the model of the robot. This can be applied to the 3D position and the 3D orientation of the hand straight forward, since the kinematic model of the arm is sufficiently calibrated. However, the 3D position of the fingertips and thumb cannot be calculated directly due to the underactuated mechanism. An approach to calculate the position of the fingertips is to analyze the structure of the fingers and derive the necessary equations. The finger in the open state is depicted in Figure 4.13. The tendon coming from the TUAT/Karlsruhe mechanism is in the default state, i.e. no force is applied. When the tendon is pulled, the finger closes. The fully closed state is shown in Figure 4.14.

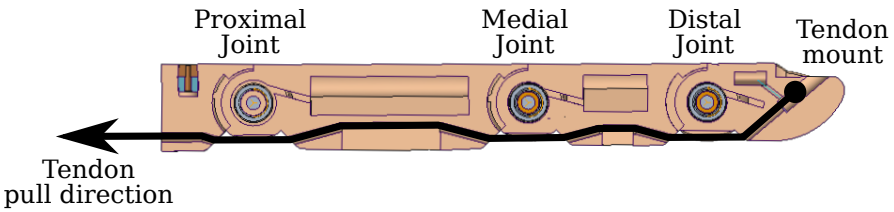


Figure 4.13: One finger of the ARMAR-6 hand in the open state.

In each joint of the finger, one torsion spring is present. This spring helps to open the finger again, when the tendon is no longer pulled. To understand how this closing works one has to look at the length of the tendon that runs through

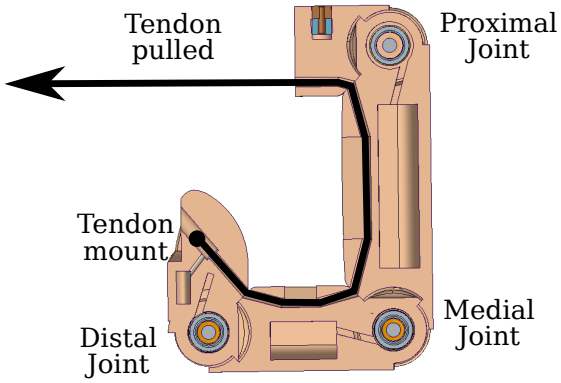


Figure 4.14: One finger of the ARMAR-6 hand in the closed state.

the finger. The overall length of the tendon within one finger is determined by the parts where the length cannot change, i.e. the static routing part. The actual length change occurs under the joints, where the tendon is not routed through the finger and is not constrained. If the tendon is pulled by a length of ΔL, this change of length is distributed to all three joints, namely the proximal joint, the medial joint and the distal joint.

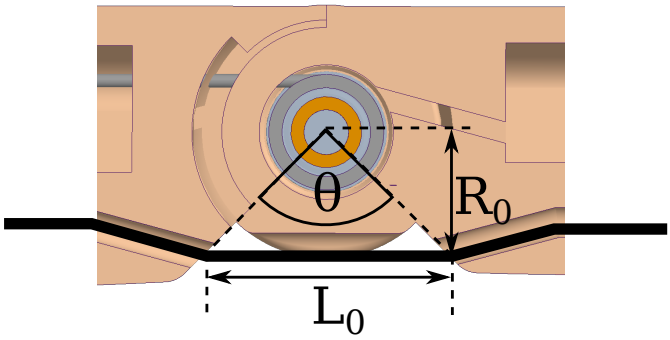


Figure 4.15: Detail view of one finger joint. The tendon acts as a lever, transferring the force within the tendon to a torque in the joint.

In order to determine how much the three joints contribute to the change in length, the change in length per joint must be expressed as a function of the torque in the joint. All used symbols are listed in Table 4.3. The joint angle θ_i is 0 if the joint is open and $1/2\pi$ if the joint is closed. The change in tendon length is defined to be 0 if the joint is open and $L_{0,i}$ if the joint is closed.

The torque M_i is determined by the force pulling on the tendon at the joint and the effective lever length.

$$M_i = R_i(\theta_i)F_i \tag{4.9}$$

The force within the tendon is assumed to be the same throughout the tendon,

Symbol	Description
$\theta_i \in [0, 1/2\pi]$	Joint angle of joint i
$\Delta L_i \in [0, L_{0,i}]$	Change of the tendon length in joint i
$R_{0,i}$	Effective lever length in the open state
$R_i(\theta)$	Effective lever length as a function of the joint angle
$\theta_{0,i}$	Pretension angle of the torsion spring
K_i	Spring constant of the torsion spring

Table 4.3: Symbols used for the calculation of the joint angles.

while friction is neglected.

$$F_1 = F_2 = F_3 = F \tag{4.10}$$

By solving Equation 4.9 for F_i and inserting it into Equation 4.10, a dependency between all three joints can be derived.

$$\frac{M_1}{R_1(\theta_1)} = \frac{M_2}{R_2(\theta_2)} = \frac{M_3}{R_3(\theta_3)} \tag{4.11}$$

The effective lever length is equal to $R_{0,i}$ if $\theta_i = 0$. This effective lever changes to $\sqrt{2}R_{0,i}$ when the joint is closed at $\theta_i = 1/2\pi$

$$R_i(\theta) = \sqrt{2}R_{0,i} \sin\left(\frac{\pi}{2} + \frac{\theta_i}{2}\right) \tag{4.12}$$

The dependency between ΔL_i and θ_i is simplified and assumed linear.

$$\theta_i \approx \frac{\pi}{2}\frac{\Delta L_i}{L_{0,i}} \tag{4.13}$$

The torque in each joint is determined by the torsion spring and can be expressed by the torsion spring constant K_i and the effective angle of the torsion spring, given by the sum of the joint angle and the pretension angle.

$$M_i = K_i(\theta_i + \theta_{0,i}) \tag{4.14}$$

The overall change in tendon length is given by the sum of the tendon length

change per joint.

$$\Delta L = \sum_{1}^{3} \Delta L_i \tag{4.15}$$

If ΔL is the input to the system, it can be assumed to be known. Using all equations above an equation system can then be derived with three equations and three unknown variables, namely θ_1, θ_2, θ_3.

$$\Delta L = \theta_1 \frac{2}{\pi} L_{0,1} + \theta_2 \frac{2}{\pi} L_{0,2} + \theta_3 \frac{2}{\pi} L_{0,3} \tag{4.16}$$

$$\frac{K_1(\theta_1 + \theta_{0,1})}{\sqrt{2} R_{0,1} \sin\left(\frac{\pi}{2} + \frac{\theta_1}{2}\right)} = \frac{K_2(\theta_2 + \theta_{0,2})}{\sqrt{2} R_{0,2} \sin\left(\frac{\pi}{2} + \frac{\theta_2}{2}\right)} \tag{4.17}$$

$$\frac{K_2(\theta_2 + \theta_{0,2})}{\sqrt{2} R_{0,2} \sin\left(\frac{\pi}{2} + \frac{\theta_2}{2}\right)} = \frac{K_3(\theta_3 + \theta_{0,3})}{\sqrt{2} R_{0,3} \sin\left(\frac{\pi}{2} + \frac{\theta_3}{2}\right)} \tag{4.18}$$

This system of equations has some nonlinear terms, therefore a closed form solution is difficult to find. However, approximate solutions can be found by means of non-linear optimization. To this end, the popular NLopt library is used (Johnson, 2014). NLopt is presented with the equations 4.16 through 4.18, reformulated as an optimization problem.

$$E(\theta_1, \theta_2, \theta_3) = (\Delta L - \Delta L_{target})^2 + (F_1 - F_2)^2 + (F_1 - F_3)^2 + (F_1 - F_3)^2 \tag{4.19}$$

Here ΔL_{target} denotes the target value of ΔL. This target value is enumerated in the range of $[0, L_{0,1} + L_{0,2} + L_{0,3}]$. Using the actual parameters of the hand (Table 4.4), NLopt can find approximate solutions for the three joint angles, for each target of ΔL. The resulting joint angles are graphed in Figure 4.16. Using the segment lengths S_i from the hand model these joint angles can be translated to joint configurations. Several examples of joint configurations are shown in Figure 4.17.

While a numerical solution could be found for the equation system, an experiment on the robot could not confirm the analytical solution. The closing behavior of the fingers and the thumb is depicted in Figure 4.18. While the analytical model predicted that all three joints of the fingers close at the same rate, with a small offset, the behavior on the real robot differs. The proximal joint begins to close immediately, while the other two joints move little up to about $40\,\%$. The reasons for this could be other, not modeled effects:

- The glove over the fingers adds an additional spring, which is non-linear.

Description	Symbol	Proximal	Medial	Distal
Index	i	1	2	3
Spring part no.	-	0T032-180-218	0T035-180-187	0T035-180-187
Spring constant	K_i	$0.67\,\mathrm{N\,mm}$	$0.82\,\mathrm{N\,mm}$	$0.82\,\mathrm{N\,mm}$
Spring pretension	$\theta_{i,1}$	$30°$	$30°$	$35°$
Initial lever length	$R_{0,i}$	$7.5\,\mathrm{mm}$	$7.5\,\mathrm{mm}$	$7.5\,\mathrm{mm}$
Maximum lever length	$R_i(90°)$	$10.6\,\mathrm{mm}$	$10.6\,\mathrm{mm}$	$10.6\,\mathrm{mm}$
Initial tendon length	$L_{0,i}$	$15\,\mathrm{mm}$	$15\,\mathrm{mm}$	$15\,\mathrm{mm}$
Segment length	S_i	$60\,\mathrm{mm}$	$39\,\mathrm{mm}$	$30\,\mathrm{mm}$

Table 4.4: Physical parameters of the three joints in the fingers, taken from the model.

- Gravity adds additional torque to the joints.
- Friction was not modeled.
- Contact with the environment change the force equilibrium.

Due to these difficult to model effects, in practice a different approach was necessary to design a coordinated grasping motion.

Design of a Coordinated Grasp Trajectory

In the previous section, the model of the fingers was examined analytically. Experiments with the real hardware showed that there are factors that are difficult to model, such as friction and environmental contacts. However, these environmental contacts can be very useful to close the fingers around the target object.

The forward kinematics of the fingers are difficult to model, but the hand has a repetitive accuracy, i.e. the same control inputs result in the same finger configuration. Therefore, a tool was developed which enables the design of grasp trajectories using the real robot hand in the loop. This tool is implemented as an ArmarX state chart that actively controls the hand of the robot using a Cartesian position controller. The user is presented with a graphical interface that allows changing the target of the controller online. In addition, the target values for the fingers and the thumb can be adjusted. The user can change the target position and the target orientation of the hand in small increments. The controller within the state charts follows this changed target and the hand moves accordingly. The tool also has an option to get the currently available grasp candidates, exacted from vision.

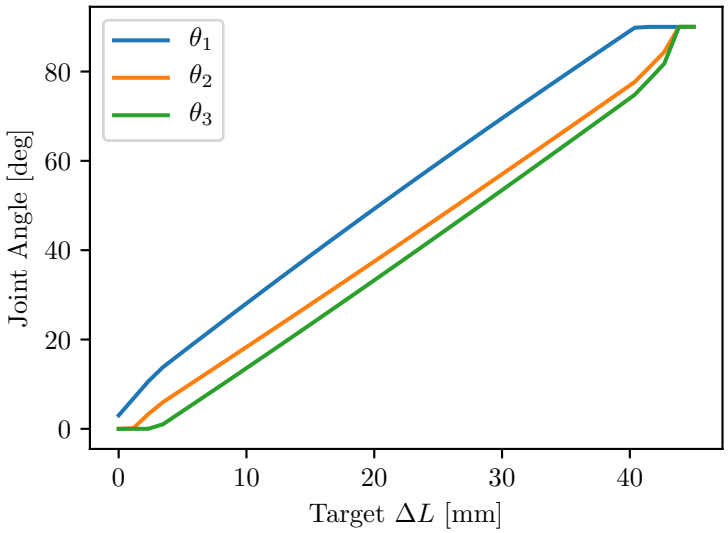


Figure 4.16: Finger joint angles derived from the shortening of the tendon.

To design a new grasp trajectory, first all available grasp candidates are queried from the ArmarX framework. Then the user selects one grasp candidate and commands the target of the hand to be above this selected grasp candidate. When the hand has reached this target, the user can visually verify that the hand is in the correct location. The user can then command the robot to move the hand downwards, while keeping the orientation, until the force torque sensor in the robot's wrist detects a force. When the hand touches the table, the grasp trajectory is initialized. The current position and location of the hand is stored, together with the current joint values. This tuple, consisting of hand position, hand orientation, finger target and thumb target is stored as one key point in the grasp trajectory. After the initial key point has been set, the user can move the hand in small increments, rotate the hand and close the fingers in small steps. Using this tool, a grasp trajectory was designed by setting several key points. During grasp execution, the robot uses these key points to control the hand leveraging feed forward control.

Coordinated Grasp Trajectory

The developed coordinated grasping motion is depicted in Figure 4.19 where the robot grasps a flash light. First, the robot orients the hand so that the line between the fingertips and the thumb is parallel to the table surface (1). The hand is moved down, until the contact force is measured by the FT-sensor (2). The finger closing starts (4) while the wrist is continuously rotated to match the

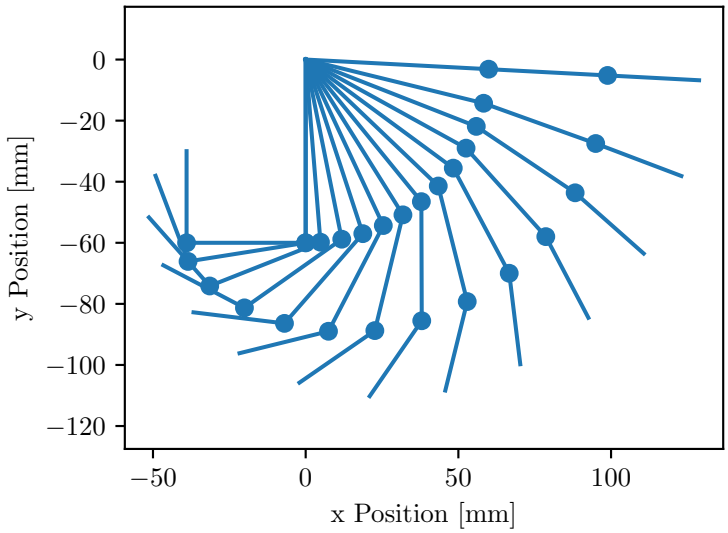


Figure 4.17: Finger configuration for different solutions of the equation system.

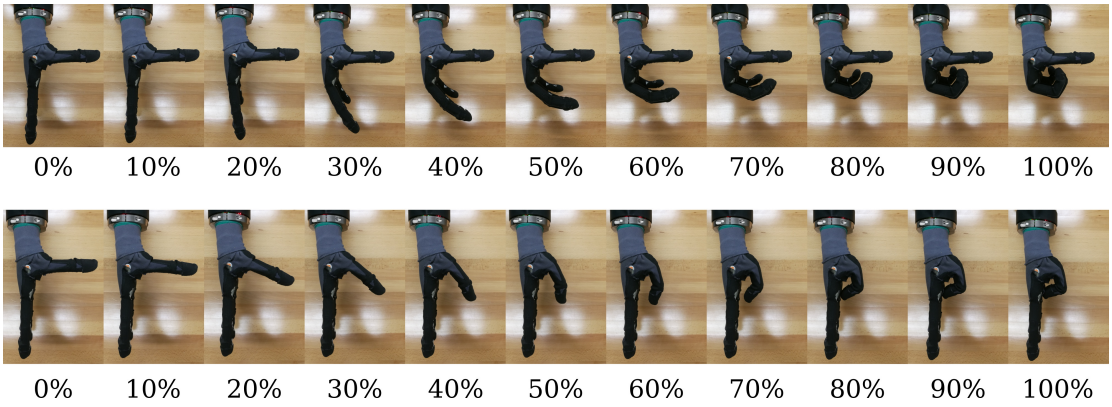


Figure 4.18: Closing behavior of the fingers and the thumb of the real robot hand.

finger closing speed (5). The fingertips and the thumb are kept at a constant height, just above the surface of the table. When the fingers are almost fully closed, the hand is moved down slightly to exploit the interaction between the underactuated fingers and the table (6). This shapes the fingers around the object. During the final closing sequence, the weight of the object is transferred from the table to the robot's hand (7). When the object is grasped, it can be lifted (8).

The grasp trajectory is comprised of 10 key points, excluding the initial key point. The key points are spaced by $0.5\,\mathrm{s}$ each, resulting in an overall $5\,\mathrm{s}$ duration. The linearly interpolated trajectory is depicted in two plots in Figure 4.20. The right plot shows the linearly interpolated trajectory of the fingers and the

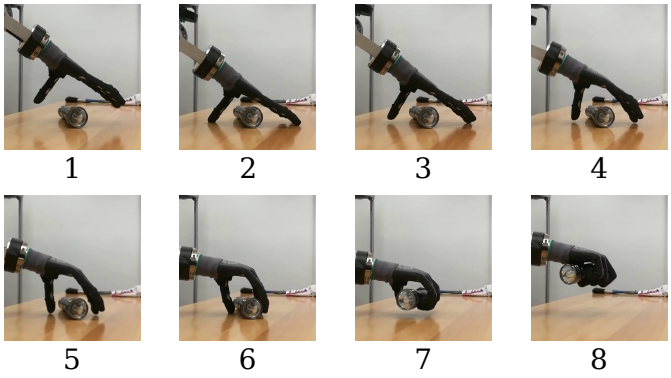


Figure 4.19: ARMAR-6 grasps a flashlight with its underactuated hand using a precise motion coordinating finger closing and hand orientation.

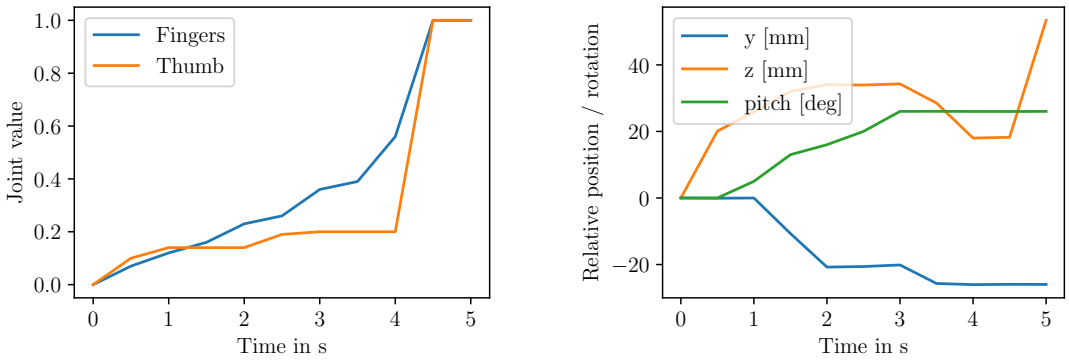


Figure 4.20: Coordinated grasp trajectory for the underactuated hand of ARMAR-6. The joint values for finger and thumb are given relatively, where 0 means open and 1 means completely closed (left side). The hand is controlled in Z-direction (height) and Y-direction (distance from the robot) and in rotation to match the finger trajectory (right side).

thumb. The finger joint value increases steadily to about $50\,\%$ at $4\,\mathrm{s}$. The thumb is first closed to $20\,\%$ and then remains mostly unchanged until 4 seconds have passed. After $4\,\mathrm{s}$ the fingers and the thumb are completely closed. The control of the position and the orientation takes place simultaneously with the control of the fingers. The right plot shows the relative position change and the relative orientation change of the hand. The plot summarizes the height of the hand (z), the displacement of the hand in the direction of the robot (y) and the rotation around the pitch axis. The initial raising of the hand is clearly visible in the Z-component. In addition, the lowering of the hand, between second 3 and 4, is visible. While the Z-position is adjusted, the hand is also moved closer to the robot. This is necessary because the center of rotation is close to the palm of the hand and not between the fingertips. During the grasping process, the hand is

rotated by approx. $25°$ so that the fingertips and the tip of the thumb always remain at a constant height. After the fingers are completely closed at $4.5\,\mathrm{s}$, the hand is raised 30mm to complete the grasp.

4.5.5 Minimal Tactile Exploration

During execution on the robot, acquiring tactile contact with the objects is a time consuming endeavor, which might move the objects resulting in loss of precision. However, adding more tactile contact points improves the GPIS estimate gradually (Bjorkman et al., 2013). Therefore, a trade-off between execution time and model completeness arises. Prior to transfer to the real robot, the minimum number of tactile exploration actions is determined. The grasping pipeline is executed in simulation using the 12 test objects. The number of tactile exploration actions is varied between 0 and 5. Here 0 exploration actions is equivalent to using visual information alone. The results of this evaluation are listed in Table 4.5.

	Number of exploration actions					
Object	**0**	**1**	**2**	**3**	**4**	**5**
1	70 %	73 %	82 %	68 %	94 %	90 %
2	100 %	100 %	100 %	100 %	100 %	100 %
3	100 %	100 %	100 %	100 %	100 %	100 %
4	88 %	96 %	98 %	98 %	96 %	97 %
5	89 %	99 %	98 %	99 %	99 %	98 %
6	89 %	95 %	92 %	96 %	93 %	90 %
7	77 %	91 %	70 %	74 %	79 %	97 %
8	92 %	88 %	92 %	92 %	94 %	93 %
9	90 %	100 %	99 %	99 %	100 %	99 %
10	96 %	94 %	99 %	93 %	94 %	99 %
11	97 %	99 %	99 %	100 %	99 %	99 %
12	83 %	97 %	97 %	99 %	96 %	78 %
Mean	**89 %**	**94 %**	**94 %**	**93 %**	**95 %**	**95 %**

Table 4.5: Grasp success probability depending on the number of tactile exploration actions.

The table shows that the information gain from the first exploration action is the greatest. The grasp success rate is increased from $89\,\%$ to $94\,\%$, which is equivalent to halving the grasp error rate. Further exploration actions yield only little improvement of the grasp success rate. Therefore, one exploration action was chosen for the execution on the real robot.

4.5.6 Validation Results

Figure 4.21: Object set used for validation. Objects from top to bottom and left to right: Aluminum Profile, Hammer, Multimeter, Screw Box, Power Drill, Cutter, Pliers, Spray Bottle. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

During the validation, different objects are used, including the „Power Drill“ and the „Spray Bottle“. The validation objects are shown in Figure 4.21. During the validation, the grasping pipeline is executed. The camera in the robot's head captures a point cloud. The robot performs exactly one exploration action, as determined in subsection 4.5.5. The visual and tactile information is fused in one GPIS model. The two grasp planners provided by Simox synthesize 500 grasp candidates using the GPIS estimate. The grasp candidates are filtered to leave only top grasps. The data-driven grasp metric rates the remaining candidates and selects one candidate for grasp execution.

The robot tries to grasp each of the objects from the validation set. Example grasping results can be seen in Figure 4.22. Each of the objects could be lifted successfully, while in some cases multiple attempts were necessary, see Table 4.6. Further examples for successful and failed grasps are shown in the accompanying video[3].

[3] http://ottenhaus.de/simon/vhgrasping/

Figure 4.22: Examples of grasping results. Image taken from Ottenhaus et al. (2019), © 2019 IEEE.

Object	Mass	Lift success?	Attempts
Alum. Profile	1.2 kg	Yes	2
Hammer	0.8 kg	Yes	2
Multimeter	0.4 kg	Yes	1
Screw box	0.4 kg	Yes	2
Power drill	0.9 kg	Yes	2
Cutter	0.3 kg	Yes	2
Pliers	0.3 kg	Yes	1
Spray bottle	0.2 kg	Yes	1

Table 4.6: Results of the validation on the real robot, taken from Ottenhaus et al. (2019), © 2019 IEEE.

4.6 Discussion

This chapter introduced a visuo-haptic grasping pipeline that enables the humanoid robot ARMAR-6 to grasp unknown objects. The developed pipeline combines visual perception with tactile exploration and fuses both modalities in one joint surface estimate. Two grasp planners generate grasp candidates using this estimated surface. A newly developed data-driven grasp metric scores all available grasp candidates, according to the predicted grasp success probability. A simulator executes the whole grasp pipeline many times in simulation using different objects. The grasp metric is trained by simulating the grasp success probability using the ground truth surface mesh. The approach is transfered to the humanoid robot ARMAR-6 and validated by grasping various objects using the underactuated hand of the robot. Several requirements were identified at the beginning of the chapter and are discussed below.

Choice of Exploration Targets After the robot has captured an initial visual view of the object the unseen sides of the object should be explored via tactile exploration. However, tactile exploration actions are time consuming, as the robot needs to physically move its end-effector. Therefore, the tactile exploration is performed using the developed next-best-touch approach introduced

in section 3.1. This enables the robot to maximize the information gain per cost in order to improve the surface estimate as quickly as possible.

Fusion of Visual and Tactile Data The point cloud coming from the depth camera and the positions of the contact point acquired by tactile exploration are fused using *Gaussian Process Implicit Surfaces (GPIS)*. GPIS was introduced in subsection 3.2.1 and was extended to include surface normals in subsection 3.2.2. The *Implicit Surface Potential* provided by the GPIS method is then triangulated; resulting is an estimated surface mesh of the object.

Grasp Candidate Generation Two grasp planners, namely the skeleton grasp planner and the surface based grasp planner, developed by Vahrenkamp et al. (2009, 2018) use the estimated surface of the object to synthesize grasp candidates.

Minimum Required Exploration The question at the end of the last chapter (chapter 3) was, *How many exploration actions are necessary for grasping?* Regarding the combination of visual and tactile information this was answered during the transfer of the grasping pipeline to the real robot. The surprising answer was: *The first exploration action yields the most new information that is necessary for grasping.* After this first exploration action, the estimated model of the object still differs from the actual surface of the object. However, the data-driven grasp metric is able to extract the necessary information from this one touch at the back of the object to reliably produce grasp candidates with high success rate.

Rating and Selection of Grasp Candidates After the grasp planners have generated many grasp candidates the central question is: *Which of the available grasp candidates has the highest success rate?* One simple answer could be that one can simply use the computed grasp scores by the grasp planners to select the grasp candidate with the highest grasp score, i.e. the ϵ-metric. However, this is not possible since the grasps were planned on an estimated surface. The computed metrics are based on this estimated surface and do not reflect the actual grasp success probability. To overcome this, a new data-driven grasp metric was developed that predicts the grasp success of a given grasp candidate. This grasp metric uses the captured visual and tactile points as well as the surface estimate as an input. The output value of this metric is trained in simulation using an improved grasp score, namely the force closure rate, introduced by Weisz and Allen (2012).

Transfer to the Humanoid Robot ARMAR-6 The developed grasping pipeline was transferred to the humanoid robot ARMAR-6. The robot's hand does not

have any tactile sensors; therefore, the tactile exploration was adapted. Instead of sensors at the fingertips, tactile contacts are inferred from the force/torque sensor reading from the wrist. The hand model is then used to calculate the position of the contact. The grasp execution was optimized to fit to the underactuated hand of the robot, see subsection 4.5.4. A grasping procedure was developed that coordinates the closing of the fingers with the rotation and translation of the wrist. The grasping procedure exploits the interaction between the underactuated fingers and the supporting surface to shape the fingers around the object. In validation experiments, the robot was able to grasp and lift several objects using the developed grasping pipeline.

5 Conclusion

The goal of this work was to develop new approaches to address the following question:

How can a humanoid robot grasp unknown objects?

The present work addressed this question as follows: First, the robot gathers information about the object by performing a tactile scan of the object. Then the robot uses a data-driven grasp metric in combination with an initial visual view to select a suitable grasp. The grasping problem was thus divided into two sub problems, (1) the efficient tactile scanning of the object with the aim of exploring the object surface for subsequent grasp planning, and (2) the generation, evaluation and selection of grasp candidates based on visual and tactile perception with a data-driven grasping metric.

Both parts of the work are closely related, since the exploration strategy developed in the first part is used in the second part. A self-imposed constraint in the entire work was that the developed methods were to be used for the humanoid robots ARMAR-III and ARMAR-6. Thus, only sensors directly available in the respective robot system were used. In addition, the developed approaches should be data-efficient. The exploration was required to move the robot arm as little as possible. For the visual perception, only one view per object and grasping attempt was used, which originated from the camera in the head of the robot.

5.1 Scientific Contributions

The scientific contributions of this thesis can be summarized as follows.

Next-Best-Touch for Grasping of Unknown Objects

A novel next-best-touch algorithm was developed that efficiently plans exploration actions to maximize the information gain with respect to the expected

costs of the exploration actions. The result of each exploration action is a new contact point on the object's surface, including the local surface orientation encoded as a surface normal. *Gaussian Process Implicit Surfaces (GPIS)* were used as a data-efficient surface model. The standard formulation of GPIS was extended to include surface normals by adding derivative information to the underlying Gaussian process. A new tactile sensor, comprised of an inertial measurement unit *(IMU)* and a tactile pressure sensor, was developed to measure the local surface orientation. The approach was evaluated in simulation using over 120 objects from the KIT and YCB object datasets. The developed next-best-touch algorithm has outperformed state-of-the-art approaches with respect to the exploration cost. The evaluation of the developed surface normal sensor has shown that the added surface normals significantly reduced the surface estimation error. The exploration algorithm was published in Ottenhaus et al. (2018a), while the sensor concept was initially published in Kaul et al. (2016) and extended and evaluated in Ottenhaus et al. (2018b).

Data-Driven Grasping of Unknown Objects through added Visual Information

In the second part of this thesis, a method was developed that enables the humanoid robot ARMAR-6 to grasp unknown objects. To this end, a complete grasping pipeline was developed that combines visual and tactile information, estimates the object's surface, synthesizes grasp candidates, rates the candidates, and executes the grasp candidate with the highest success rate. First, a depth camera in the robot's head is used to capture an initial view of the object. Since this initial view captures only part of the object, i.e. the back of the object is missing, the previously developed next-best-touch algorithm is used to explore the unseen back of the object. The visually acquired point cloud and the tactile contact points are fused into a single surface model by means of *GPIS*. In the next step, multiple grasp candidates are generated by feeding the estimated surface of the object to two conventional grasp planners. To answer the question regarding which of the generated grasp candidates has the highest success rate, a novel data-driven grasp metric was developed. This grasp metric predicts the probability of success of a given grasp candidate based on visual and tactile sensor data, as well as the estimated surface model. The data-driven grasp metric was trained using synthetic grasping data obtained in a simulated environment. In simulation, the developed data-driven grasp metric has outperformed conventional grasp planning by a significant margin. During

the transfer to the real robot ARMAR-6, special attention had to be paid to the grasp execution. The robot's hand is underactuated, where only two motors control all five fingers by means of a specialized mechanism. However, this underactuation could not be simulated sufficiently accurately with the available simulators. Therefore, a specialized grasping procedure was developed that coordinates the closing of the fingers and the wrist rotation. Furthermore, interactions with the environment were exploited to shape the under-actuated fingers around the object. The grasping pipeline including the data-driven grasp metric was published in Ottenhaus et al. (2019).

5.2 Discussion and Future Work

This work led to novel approaches in two investigated aspects of robotic grasping: next-best-touch for grasping and data-driven grasping of unknown objects.

Next-Best-Touch for Grasping of Unknown Objects

The extensive evaluation of the developed next-best-touch algorithm presented in chapter 3 has shown that tactile exploration is a promising approach for a humanoid robot to gather contact information about an unknown object. It was shown that this contact information can be used to generate a surface estimate of the object. To this end, the data-efficient surface model GPIS was used.

The evaluation of the newly developed next-best-touch algorithm has also shown that balancing expected path costs with the predicted information gain per exploration action can lead to a significant reduction in overall exploration costs. Furthermore, the extension of GPIS to include surface normal information reduced the surface estimation error by $50\,\%$.

A newly developed tactile sensor was used to collect surface normal data as well as contact data at the same time by combining an IMU with a pressure sensor. The evaluation of the sensor has shown that the orientation accuracy of the IMU was accurate enough to gather the surface normals of unknown objects. Using these surface normals together with the contact positions as input for the GPIS model, the surface of unknown objects could be reconstructed. The resulting surface reconstruction was precise enough to allow grasp planning using a conventional grasp planner. This is an interesting result, as conventional grasp planners tend to require precise object models.

Data-Driven Grasping of Unknown Objects through added Visual Information

The evaluation of the developed data-driven grasping pipeline and the subsequent transfer to the humanoid robot ARMAR-6 have produced some interesting findings. First and foremost, it became clear that successfully grasping unknown objects with humanoid robots depends on a variety of factors. These factors include (1) the hardware in the form of the hand, the arm and the camera systems, (2) the grasp planning and (3) the execution of the grasp. In this thesis, the hardware was specified by the humanoid robot system ARMAR-6. Therefore, the developed data-driven grasp planning and the grasp execution could be optimized especially for the given underactuated hand of the robot. From this it can also be deduced that precise grasp planning and execution cannot be robot agnostic, especially with regard to underactuated hands. This clearly distinguishes grasping with humanoid robots from bin-picking tasks performed by industrial yaw grippers, where most gripper share similar physical properties.

In this thesis, the data-driven grasp planning was performed by first generating a large set of possible grasp candidates, which were then rated by a data-driven grasp metric. This discriminative approach circumvents the need for searching good grasp candidates in the full 6D space that is theoretically available for hand placements. However, a discriminative approach always hinges on a suitable candidate generator. In the present work, this candidate generator was implemented by reusing the surface estimation method of the next-best-touch method presented in chapter 3. A conventional grasp planner uses this estimated surface to synthesize possible grasp candidates. As the evaluation showed, some of these candidates have a high success rate, while a large portion of the candidates will result in failed grasps. Here, the data-driven discriminative grasp metric, implemented as a deep neural network, was able to distinguish between promising grasp candidates and unsuitable grasp candidates. On the one hand, it was shown that the use of deep learning in the area of humanoid robot grasping can lead to significant improvements compared to conventional methods. On the other hand, it was found that when learning grasping data in simulation, special attention must be paid to the transfer to the real robot system.

After the data-driven grasp metric was evaluated in simulation, the approach was transferred to the robot. In the first iteration of validation on the robot, a relatively simple grasping strategy was used. Some objects had to be lifted

slightly from the tabletop by placing small foam blocks underneath, to ensure that the fingers could be closed around the object. In later experiments, it became clear that a controlled interaction between the fingers and the tabletop can be exploited to close the fingers of the hand around small objects. A large part of the grasp success is thus not only dependent on the hardware and grasp planning, but depend also on a precise execution of the grasp. Both the fingers and the position and the orientation of the wrist must be continuously controlled.

Another interesting result arose during the evaluation of the minimally necessary tactile exploration. If an initial point cloud of the object had already been captured with a depth camera in the head of the robot, the first exploration action yielded the greatest information gain in terms of the grasp success rate. Further exploration actions could only marginally improve the grasping success. It can be argued that the main reason for this is that the first contact on the back of the object provides information about the overall size of the object. Therefore, the deep learning model was able to implicitly predict the center of the object and thus was able to filter out grasp candidates that placed the fingers or the palm in unsuitable locations.

Extensions in Future Work

The next-best-touch algorithm and the data-driven grasping of unknown objects presented in this thesis lay the foundation for a variety of possible extensions in future work. As the grasping of unknown objects is a central and challenging problem in humanoid robotics, many aspects can be investigated in more detail in the future. The developed next-best-touch algorithm also offers several starting points for future work.

Extension of the Next-Best-Touch Algorithm to Whole Hands

The presented next-best-touch algorithm predicts the information gain for one exploration action at a time, using one robotic finger. The used Information Gain Estimation Function (IGEF) evaluates possible exploration targets, where each target is defined as one point on the estimated surface. Although this function is limited to one finger at the moment, it can be extended to include multiple exploration targets at the same time. Thereby multiple fingertips of a robotic hand could be evaluated simultaneously and an optimal next-best-touch for the whole hand could be planned.

Extension of the Information Gain Estimation Function Exploration Horizon

The current IGEF approach estimates the information gain with an exploration horizon of one exploration action. In future research this could be extended by planning not one, but multiple exploration actions at the same time, e.g. by means of optimizing an exploration policy aiming to minimize the overall expected exploration cost of all exploration actions. In the field of deep reinforcement learning, promising techniques have been developed for such policy optimization.

Joining the Surface Estimation with the Exploration Planning

Using deep learning techniques, it is also possible to combine the dual problem of exploration action planning and surface estimation in one joint approach. Thereby, the sampling of possible next-best-exploration actions would not be limited to exploring targets on the estimated surface of the object. This could significantly speed up the exploration process, since the estimated surface of the object usually deviates from the actual object, especially in regions where the object has not been explored yet. Thereby the exploration would need to spend less time to restore contact with the object, but could focus more on acquiring relevant contact information in interesting regions of the object.

Grasping with Different or Improved Hands

The presented data-driven grasping approach was evaluated and validated solely using the under-actuated hand of the humanoid robot ARMAR-6, due to the fact that this hand was the only available hand for the robot at the time of evaluation. As the hand development continues, an upgraded version of the hand could be used for grasping, especially since the robot offers the ability to easily mount different hands. Furthermore, the impact of the fingers' friction properties can be investigated in detail. For some objects, the friction of the fingers might not be of relevance, since the object is fully enclosed by the fingers. However, other objects require highly adherent fingertips. As an example, the screw box is too large and cannot be enclosed by the fingers of the robot. Therefore, the fingers can only be placed on two opposite, parallel faces on the box. In future work, the suitability of different materials for fingertip coating can be investigated to improve the overall grasp success rate of the hand.

Extension of the Grasp Metric to Include of Finger Configurations

The presented data-driven grasp metric rates possible grasps according to the 6D pose of the hand's TCP. In future work this can be extended to include the configuration of the fingers. During the grasping experiments, it became apparent that some small objects might require precise pre-shapes of the hand. The

prediction of such a suitable pre-shape could be included in the data-driven grasp metric to extend the set of objects that can be grasped.

Data-Driven Grasp Execution

The coordinated grasp execution presented in subsection 4.5.4 had a significant positive effect on grasp success. However, it was mainly developed using visual inspection and trial and error methods. The synthesis of object-specific grasp executions can therefore be of interest in the future. Objects requiring precise finger placement at specific locations could particularly benefit from such methods. This could include part-specific grasping of objects, e.g. grasping an object at a handle or grasping a cup or bowl-shaped object at the rim. These grasping procedures either could be generated from a set of predefined grasping primitives or could be transferred from human grasping experiments by learning from demonstration methods. Another approach could be the application of reinforcement learning techniques to enable the robot to find new grasp execution strategies autonomously.

Iterative Refinement and Learning from Failures

Iterative learning is one of topics in machine learning, that currently remain challenging. However, robotics and in particular grasping could benefit from such methods. Especially learning from mistakes could improve the grasp success rate. The presented implementation of the grasp execution is feed forward, i.e. the whole trajectory is precomputed and then precisely executed. If a new iteration of the hand provides additional sensor feedback, e.g. in the form of tactile sensors, these signals could be used to continuously monitor the grasp execution. The robot could then learn to distinguish between tactile signals that correlate with stable grasps and signals that indicate an imminent grasp failure. In case of a predicted failure, corrective motions could be executed.

Appendix

A Evaluation Table of the Next-Best-Touch Strategy

The following table lists the exploration results for each tested object from the KIT and YCB object sets. Each object was explored 100 times using each exploration strategy. The table lists the average result for each strategy and object. The strategies include:

- *Random*: Random sampling of the next-best-touch from all available targets.
- *GP-V*: Selection of the next-best-touch with the highest Gaussian process variance, according to state-of-the art approaches.
- *IGEF*: Selection of the next-best-touch according to the newly developed *Information Gain Estimation Function*.

Nr	Object Name	Random	GP-V	IGEF
1	KIT_Amicelli	534.9 cm	480.7 cm	261.8 cm
2	KIT_BakingSoda	263.3 cm	413.1 cm	128.5 cm
3	KIT_BakingVanilla	270.9 cm	395.7 cm	143.6 cm
4	KIT_BlueSaltCube	471.4 cm	467.8 cm	223.9 cm
5	KIT_BroccoliSoup	607.8 cm	568.9 cm	265.5 cm
6	KIT_CatLying	492.5 cm	493.0 cm	233.0 cm
7	KIT_CatSitting	521.8 cm	508.7 cm	239.4 cm
8	KIT_CeylonTea	780.1 cm	598.3 cm	356.8 cm
9	KIT_ChickenSoup	467.6 cm	534.5 cm	225.0 cm
10	KIT_ChocSticks	483.7 cm	578.8 cm	226.2 cm
11	KIT_ChocolateBars	567.9 cm	557.1 cm	272.7 cm
12	KIT_ChoppedTomatoes	447.2 cm	454.2 cm	224.6 cm
13	KIT_Clown	459.3 cm	531.1 cm	227.1 cm
14	KIT_CoffeeBox	1060.0 cm	811.9 cm	528.6 cm
15	KIT_CoffeeCookies	360.2 cm	424.7 cm	189.4 cm
16	KIT_CondensedMilk	332.2 cm	426.8 cm	158.6 cm
17	KIT_CoughDropsBerries	399.4 cm	472.6 cm	196.3 cm
18	KIT_CoughDropsHoney	413.7 cm	474.6 cm	198.9 cm
19	KIT_CoughDropsLemon	410.3 cm	515.3 cm	194.5 cm
20	KIT_Curry	244.1 cm	523.8 cm	119.2 cm

Nr	Object Name	Random	GP-V	IGEF
21	KIT_DanishHam	545.5 cm	593.4 cm	269.2 cm
22	KIT_Deodorant	228.2 cm	444.0 cm	111.8 cm
23	KIT_DropsCherry	217.8 cm	497.7 cm	104.0 cm
24	KIT_DropsOrange	203.2 cm	574.7 cm	100.7 cm
25	KIT_Fish	446.9 cm	447.6 cm	222.7 cm
26	KIT_FizzyTablets	271.7 cm	358.1 cm	127.8 cm
27	KIT_FizzyTabletsCalcium	279.1 cm	422.3 cm	124.3 cm
28	KIT_FlowerCup	462.1 cm	477.7 cm	230.6 cm
29	KIT_FruitTea	448.5 cm	446.8 cm	209.7 cm
30	KIT_GreenSaltCylinder	233.5 cm	565.9 cm	107.6 cm
31	KIT_HamburgerSauce	513.2 cm	493.7 cm	254.4 cm
32	KIT_HeringTin	418.9 cm	581.4 cm	195.9 cm
33	KIT_HotPot	488.3 cm	608.3 cm	222.0 cm
34	KIT_HotPot2	533.2 cm	574.9 cm	243.3 cm
35	KIT_HygieneSpray	376.0 cm	471.3 cm	196.5 cm
36	KIT_InstantDumplings	827.6 cm	681.3 cm	389.3 cm
37	KIT_InstantIceCoffee	597.8 cm	521.1 cm	285.9 cm
38	KIT_InstantMousse	555.3 cm	606.6 cm	257.5 cm
39	KIT_InstantSauce	570.5 cm	564.4 cm	268.0 cm
40	KIT_InstantSoup	591.8 cm	548.0 cm	281.5 cm
41	KIT_InstantTomatoSoup	469.8 cm	540.6 cm	225.9 cm
42	KIT_JamSugar	466.0 cm	462.0 cm	233.4 cm
43	KIT_Knaeckebrot	972.7 cm	767.8 cm	462.8 cm
44	KIT_KnaeckebrotRye	794.8 cm	656.8 cm	390.7 cm
45	KIT_LivioClassicOil	538.5 cm	510.3 cm	275.7 cm
46	KIT_LivioSunflowerOil	572.0 cm	514.4 cm	276.2 cm
47	KIT_Margarine	439.7 cm	487.3 cm	227.8 cm
48	KIT_Marjoram	257.3 cm	536.5 cm	119.5 cm
49	KIT_MilkDrinkVanilla	555.2 cm	478.7 cm	264.1 cm
50	KIT_MilkRice	656.5 cm	597.8 cm	315.8 cm
51	KIT_MuesliBars	740.0 cm	700.0 cm	354.6 cm
52	KIT_NutCandy	892.3 cm	762.7 cm	402.1 cm
53	KIT_NutellaGo	297.1 cm	346.9 cm	145.3 cm
54	KIT_OrangeMarmelade	426.7 cm	427.8 cm	199.0 cm
55	KIT_OrgHerbTea	589.7 cm	537.7 cm	289.9 cm
56	KIT_Paprika	250.4 cm	512.8 cm	121.7 cm
57	KIT_Patches	322.7 cm	457.8 cm	154.2 cm
58	KIT_PatchesSensitive	454.2 cm	506.3 cm	210.5 cm
59	KIT_Peanuts	490.1 cm	458.6 cm	228.6 cm
60	KIT_Peanuts2	485.5 cm	471.8 cm	226.5 cm
61	KIT_Peas	441.1 cm	439.4 cm	214.1 cm
62	KIT_PineappleSlices	547.1 cm	481.1 cm	257.9 cm
63	KIT_Pony	474.1 cm	576.7 cm	232.3 cm
64	KIT_PotatoeDumplings	587.6 cm	631.7 cm	266.9 cm
65	KIT_PotatoeStarch	827.6 cm	673.1 cm	385.8 cm
66	KIT_PotatoeSticks	645.3 cm	536.6 cm	310.0 cm
67	KIT_PowderedSugar	500.0 cm	537.5 cm	250.5 cm

Nr	Object Name	Random	GP-V	IGEF
68	KIT_PowderedSugarMill	586.7 cm	494.6 cm	285.8 cm
69	KIT_RavioliLarge	666.4 cm	551.3 cm	320.7 cm
70	KIT_SardinesCan	288.4 cm	396.9 cm	139.2 cm
71	KIT_SauceThickener	375.8 cm	407.0 cm	186.2 cm
72	KIT_Sauerkraut	654.2 cm	547.3 cm	321.2 cm
73	KIT_SauerkrautSmall	382.2 cm	477.1 cm	183.3 cm
74	KIT_Seal	446.6 cm	496.0 cm	213.6 cm
75	KIT_Shampoo	429.8 cm	470.3 cm	223.4 cm
76	KIT_ShowerGel	448.5 cm	496.8 cm	238.1 cm
77	KIT_SmallGlass	340.1 cm	421.2 cm	162.2 cm
78	KIT_SoftCheese	400.6 cm	562.7 cm	177.8 cm
79	KIT_StrawberryPorridge	749.0 cm	689.2 cm	363.6 cm
80	KIT_Sweetener	248.0 cm	374.6 cm	131.3 cm
81	KIT_TomatoHerbSauce	457.9 cm	476.0 cm	232.1 cm
82	KIT_TomatoSauce	430.2 cm	444.6 cm	213.4 cm
83	KIT_TomatoSoup	363.3 cm	414.4 cm	179.1 cm
84	KIT_Tortoise	421.8 cm	471.3 cm	211.1 cm
85	KIT_Wafflerolls	622.9 cm	520.2 cm	292.6 cm
86	KIT_Waterglass	517.1 cm	494.6 cm	256.4 cm
87	KIT_WhiteCup	451.9 cm	380.3 cm	236.3 cm
88	KIT_YellowSaltCube	463.0 cm	446.9 cm	226.6 cm
89	KIT_YellowSaltCube2	475.1 cm	469.8 cm	227.2 cm
90	KIT_YellowSaltCylinder	456.3 cm	427.5 cm	225.6 cm
91	KIT_YellowSaltCylinderSmall	226.7 cm	558.0 cm	108.0 cm
92	YCB_black_and_decker_lithium_drill_driver_unboxed	726.4 cm	694.8 cm	363.4 cm
93	YCB_block_of_wood_6in	529.5 cm	513.4 cm	261.9 cm
94	YCB_brine_mini_soccer_ball	878.8 cm	549.7 cm	342.7 cm
95	YCB_campbells_condensed_tomato_soup	398.2 cm	411.9 cm	175.4 cm
96	YCB_champion_sports_official_softball	580.3 cm	330.8 cm	190.9 cm
97	YCB_clorox_disinfecting_wipes_35	816.7 cm	657.0 cm	397.6 cm
98	YCB_comet_lemon_fresh_bleach	663.3 cm	554.6 cm	320.9 cm
99	YCB_dark_red_foam_block_with_three_holes	309.5 cm	460.7 cm	147.8 cm
100	YCB_domino_sugar_1lb	630.6 cm	575.3 cm	304.1 cm
101	YCB_frenchs_classic_yellow_mustard_14oz	515.8 cm	473.7 cm	247.5 cm
102	YCB_jell-o_chocolate_flavor_pudding	450.5 cm	485.8 cm	205.1 cm
103	YCB_jell-o_strawberry_gelatin_dessert	320.9 cm	430.8 cm	150.1 cm
104	YCB_large_black_spring_clamp	522.1 cm	546.8 cm	264.6 cm
105	YCB_master_chef_ground_coffee_297g	733.6 cm	574.3 cm	349.2 cm
106	YCB_medium_black_spring_clamp	233.4 cm	480.2 cm	117.8 cm
107	YCB_melissa_doug_farm_fresh_fruit_apple	412.9 cm	443.4 cm	143.4 cm
108	YCB_melissa_doug_farm_fresh_fruit_banana	347.1 cm	443.0 cm	179.1 cm
109	YCB_melissa_doug_farm_fresh_fruit_orange	357.7 cm	420.1 cm	133.3 cm
110	YCB_melissa_doug_farm_fresh_fruit_pear	306.9 cm	444.3 cm	129.7 cm
111	YCB_morton_salt_shaker	218.8 cm	575.4 cm	104.8 cm
112	YCB_play_go_rainbow_stakin_cups_10_blue	532.9 cm	554.9 cm	295.9 cm
113	YCB_play_go_rainbow_stakin_cups_5_green	327.3 cm	397.7 cm	135.6 cm
114	YCB_play_go_rainbow_stakin_cups_6_purple	358.1 cm	342.0 cm	206.8 cm

Nr	Object Name	Random	GP-V	IGEF
115	YCB_play_go_rainbow_stakin_cups_7_yellow	394.8 cm	371.1 cm	161.5 cm
116	YCB_play_go_rainbow_stakin_cups_8_orange	450.7 cm	411.2 cm	262.3 cm
117	YCB_red_metal_bowl_white_speckles	824.1 cm	666.5 cm	404.2 cm
118	YCB_red_metal_cup_white_speckles	590.8 cm	446.1 cm	297.7 cm
119	YCB_soft_scrub_2lb_4oz	748.2 cm	647.0 cm	385.8 cm
120	YCB_spam_12oz	466.4 cm	464.8 cm	205.8 cm
121	YCB_sponge_with_textured_cover	271.5 cm	564.1 cm	126.4 cm
122	YCB_starkist_chunk_light_tuna	332.6 cm	445.6 cm	149.9 cm
123	YCB_thick_wood_block_6in	826.5 cm	645.3 cm	399.6 cm
124	YCB_wescott_orange_grey_scissors	268.0 cm	490.7 cm	151.9 cm
	average	**243.3 cm**	**255.3 cm**	**116.6 cm**

Acronyms

AI Artificial intelligence
ARMAR Antropromatic multi-arm robot

CNN Convolutional Neural Network
CPU Central processing unit

DOF Degrees of freedom

GP Gaussian process
GP-V Gaussian process variance
GPIS Gaussian Process Implicit Surface

IGEF Information Gain Estimation Function
IMU Inertial measurement unit
ISP Implicit surface potential

LfD Learning from demonstration

RANSAC Random sample consensus
RMSE Root-mean-square error
RRT Rapidly-exploring random tree

SVM Support vector machine

TCP Tool center point

List of Figures

List of Tables

List of Algorithms

Bibliography

Alagi, H., Navarro, S. E., Mende, M., and Hein, B. (2016). A versatile and modular capacitive tactile proximity sensor. In *IEEE Haptics Symposium (HAPTICS)*, pages 290–296. IEEE. *Cited on page 17.*

Allen, P. K. and Roberts, K. S. (1989). Haptic object recognition using a multi-fingered dextrous hand. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 342–347. IEEE. *Cited on page 9.*

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., Awwal, A. A. S., and Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*. *Cited on page 30.*

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2018). Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*. *Cited on page 28.*

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483. *Cited on page 28.*

Asfour, T., Kaul, L., Wächter, M., Ottenhaus, S., Weiner, P., Rader, S., Grimm, R., Zhou, Y., Grotz, M., Paus, F., Shingarey, D., and Haubert, H. (2018). ARMAR-6: A collaborative humanoid robot for industrial environments. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 447–454. *Cited on pages 96, 103, and 119.*

Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., and Dillmann, R. (2006). ARMAR-III: An integrated humanoid platform for sensory-motor control. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 169–175. IEEE. *Cited on page 14.*

Asif, U., Tang, J., and Harrer, S. (2018). Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices. In *IJCAI*, pages 4875–4882. *Cited on page 26.*

Atkeson, C. G., An, C. H., and Hollerbach, J. M. (1985). Rigid body load identification for manipulators. In *IEEE Conference on Decision and Control*, pages 996–1002. IEEE. *Cited on page 13.*

Barr, A. H. (1981). Superquadrics and angle-preserving transformations. *IEEE Computer graphics and Applications*, 1(1):11–23. *Cited on page 9.*

Bartolozzi, C., Natale, L., Nori, F., Metta, G., et al. (2016). Robots with a sense of touch. *Nature materials*, 15(9):921–925. *Cited on page 14.*

Biegelbauer, G. and Vincze, M. (2007). Efficient 3d object detection by fitting superquadrics to range image data for robot's object manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1086–1091. IEEE. *Cited on page 9.*

Bierbaum, A., Rambow, M., Asfour, T., and Dillmann, R. (2008). A potential field approach to dexterous tactile exploration of unknown objects. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 360–366. IEEE. *Cited on pages 8 and 18.*

Bierbaum, A., Rambow, M., Asfour, T., and Dillmann, R. (2009a). Grasp Affordances from Multi-Fingered Tactile Exploration using Dynamic Potential Fields. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 168 – 174, Paris, France. *Cited on pages 9 and 11.*

Bierbaum, A., Schill, J., Asfour, T., and Dillmann, R. (2009b). Force position control for a pneumatic anthropomorphic hand. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 21–27. IEEE. *Cited on page 14.*

Bierbaum, A., Welke, K., Burger, D., Asfour, T., and Dillmann, R. (2007). Haptic exploration for 3d shape reconstruction using five-finger hands. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 616–621. IEEE. *Cited on page 9.*

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Robot programming by demonstration. *Springer handbook of robotics*, pages 1371–1394. *Cited on page 28.*

Bjorkman, M., Bekiroglu, Y., Hogman, V., and Kragic, D. (2013). Enhancing visual perception of shape through tactile glances. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3180–3186. IEEE. *Cited on pages 10, 11, 12, 68, 69, 85, 111, and 129.*

Bohg, J., Johnson-Roberson, M., León, B., Felip, J., Gratal, X., Bergstrom, N., Kragic, D., and Morales, A. (2011). Mind the gap - robotic grasping under incomplete observation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 686–693. *Cited on pages 9 and 33.*

Bohg, J., Morales, A., Asfour, T., and Kragic, D. (2014). Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309. *Cited on pages 19, 20, 21, and 25.*

Bütepage, J., Cruciani, S., Kokic, M., Welle, M., and Kragic, D. (2018). From visual understanding to complex object manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*. *Cited on pages 21 and 24.*

Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). The YCB object and model set: Towards common benchmarks for manipulation research. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 510–517. IEEE. *Cited on pages 83 and 108.*

Cannata, G., Maggiali, M., Metta, G., and Sandini, G. (2008). An embedded artificial skin for humanoid robots. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 434–438. IEEE. *Cited on page 16.*

Chen, I.-M. and Burdick, J. W. (1993). Finding antipodal point grasps on irregularly shaped objects. *IEEE transactions on Robotics and Automation*, 9(4):507–512. *Cited on page 38.*

Chu, F.-J., Xu, R., and Vela, P. A. (2018). Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, 3(4):3355–3362. *Cited on page 26.*

Dahiya, R. S., Metta, G., Valle, M., and Sandini, G. (2010). Tactile sensing—from humans to humanoids. *IEEE Transactions on Robotics*, 26(1):1–20. *Cited on page 18.*

Do, M., Schill, J., Ernesti, J., and Asfour, T. (2014). Learn to wipe: A case study of structural bootstrapping from sensorimotor experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1858–1864. *Cited on pages 7 and 14.*

Dragiev, S., Toussaint, M., and Gienger, M. (2011). Gaussian process implicit surfaces for shape estimation and grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2845–2850. IEEE. *Cited on pages 10, 54, 59, 68, and 85.*

Duncan, K., Sarkar, S., Alqasemi, R., and Dubey, R. (2013). Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4238–4243. IEEE. *Cited on pages 9 and 10.*

Esrom, J. (1989). Tactile sensors for robotics and medicine, edited by webster-john g.j. wiley and sons, new york, 1988, 359 pp. *Robotica*, 7(3):265–265. *Cited on page 14.*

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. *Cited on page 104.*

Fishel, J. A. and Loeb, G. E. (2012). Bayesian exploration for intelligent identification of textures. *Frontiers in neurorobotics*, 6:4. *Cited on page 14.*

Fishel, J. A., Santos, V. J., and Loeb, G. E. (2008). A robust micro-vibration sensor for biomimetic fingertips. In *IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 659–663. IEEE. *Cited on pages 14 and 17.*

Fukaya, N., Toyama, S., Asfour, T., and Dillmann, R. (2000). Design of the tuat/karlsruhe humanoid hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1754–1759. IEEE. *Cited on page 119.*

Gariépy, A., Ruel, J.-C., Chaib-draa, B., and Giguère, P. (2019). Gq-stn: Optimizing one-shot grasp detection based on robustness classifier. *arXiv preprint arXiv:1903.02489*. *Cited on page 26.*

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. *Cited on page 30.*

Gorges, N., Fritz, P., and Wörn, H. (2010). Haptic object exploration using attention cubes. In *Annual Conference on Artificial Intelligence*, pages 349–357. Springer. *Cited on page 9.*

Guo, D., Sun, F., Liu, H., Kong, T., Fang, B., and Xi, N. (2017). A hybrid deep architecture for robotic grasp detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1609–1614. IEEE. *Cited on page 26.*

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48. *Cited on pages 29 and 30.*

Harmon, L. D. (1980). Touch-sensing technology- a review. *Society of Manufacturing Engineers, 1980. 58*. *Cited on page 14.*

Harmon, L. D. (1982). Automated tactile sensing. *The International Journal of Robotics Research*, 1(2):3–32. *Cited on page 14.*

Huebner, K. and Kragic, D. (2008). Selection of robot pre-grasps using box-based shape approximation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1765–1770. IEEE. *Cited on page 9.*

Huebner, K., Ruthotto, S., and Kragic, D. (2008). Minimum volume bounding box decomposition for shape approximation in robot grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1628–1633. IEEE. *Cited on pages 9 and 10.*

Huebner, K., Welke, K., Przybylski, M., Vahrenkamp, N., Asfour, T., Kragic, D., and Dillmann, R. (2009). Grasping known objects with humanoid robots: A box-based approach. In *IEEE International Conference on Advanced Robotics (ICAR)*, pages 1–6. IEEE. *Cited on page 9.*

Jamali, N., Maggiali, M., Giovannini, F., Metta, G., and Natale, L. (2015). A new design of a fingertip for the icub hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2705–2710. IEEE. *Cited on page 16.*

Jiang, Y., Moseson, S., and Saxena, A. (2011). Efficient grasping from rgbd images: Learning using a new rectangle representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3304–3311. IEEE. *Cited on pages 20, 25, 26, and 30.*

Johansson, R. S. and Flanagan, J. R. (2009). Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359. *Cited on pages 1 and 43.*

Johnson, S. G. (2014). The nlopt nonlinear-optimization package. *Cited on page 124.*

Kaboli, M. and Cheng, G. (2018). Robust tactile descriptors for discriminating objects from textural properties via artificial robotic skin. *IEEE Transactions on Robotics*, 34(4):985–1003. *Cited on page 17.*

Kaboli, M., Feng, D., Yao, K., Lanillos, P., and Cheng, G. (2017). A tactile-based framework for active object learning and discrimination using multi-modal robotic skin. *IEEE Robotics and Automation Letters*. *Cited on pages 14 and 17.*

Kaboli, M., Yao, K., Feng, D., and Cheng, G. (2018). Tactile-based active object discrimination and target object search in an unknown workspace. *Autonomous Robots*, pages 1–30. *Cited on pages 8 and 14.*

Kappassov, Z., Corrales, J.-A., and Perdereau, V. (2015). Tactile sensing in dexterous robot hands. *Robotics and Autonomous Systems*, 74:195–220. *Cited on page 18.*

Kappler, D., Bohg, J., and Schaal, S. (2015). Leveraging big data for grasp planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311. IEEE. *Cited on page 109.*

Kasper, A., Xue, Z., and Dillmann, R. (2012). The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934. *Cited on pages 83 and 108.*

Kaul, L., Ottenhaus, S., Weiner, P., and Asfour, T. (2016). The sense of surface orientation — a new sensor modality for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 820–825. IEEE. *Cited on page 136.*

Koiva, R., Zenker, M., Schürmann, C., Haschke, R., and Ritter, H. J. (2013). A highly sensitive 3d-shaped tactile sensor. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1084–1089. IEEE. *Cited on page 15.*

Kopicki, M., Detry, R., Adjigble, M., Stolkin, R., Leonardis, A., and Wyatt, J. L. (2016). One-shot learning and generation of dexterous grasps for novel objects. *The International Journal of Robotics Research*, 35(8):959–976. *Cited on pages 23 and 29.*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. *Cited on page 30.*

Kumra, S. and Kanan, C. (2017). Robotic grasp detection using deep convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 769–776. IEEE. *Cited on page 26.*

Lederman, S. J. and Klatzky, R. L. (1987). Hand movements: A window into haptic object recognition. *Cognitive psychology*, 19(3):342–368. *Cited on pages 6, 7, 14, and 149.*

Lederman, S. J., Loomis, J. M., and Williams, D. A. (1982). The role of vibration in the tactual perception of roughness. *Perception & Psychophysics*, 32(2):109–116. *Cited on page 14.*

Lenz, I., Lee, H., and Saxena, A. (2015). Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724. *Cited on page 26.*

Leonardis, A., Jaklic, A., and Solina, F. (1997). Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295. *Cited on page 9.*

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436. *Cited on pages 27, 28, 40, and 149.*

Lin, C. H., Erickson, T. W., Fishel, J. A., Wettels, N., and Loeb, G. E. (2009). Signal processing and fabrication of a biomimetic tactile sensor array with thermal, force and microvibration modalities. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 129–134. IEEE. *Cited on pages 14 and 17.*

Lin, Y. and Sun, Y. (2015). Robot grasp planning based on demonstrated grasp strategies. *The International Journal of Robotics Research*, 34(1):26–42. *Cited on pages 23 and 28.*

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88. *Cited on page 29.*

Liu, M., Pan, Z., Xu, K., Ganguly, K., and Manocha, D. (2019). Generating grasp poses for a high-dof gripper using neural networks. *arXiv preprint arXiv:1903.00425*. *Cited on pages 23, 32, and 149.*

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM. *Cited on pages 76 and 82.*

Lundell, J., Verdoja, F., and Kyrki, V. (2019). Robust grasp planning over uncertain shape completions. *arXiv preprint arXiv:1903.00645*. *Cited on pages 23, 35, and 149.*

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., and Goldberg, K. (2017). Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*. *Cited on pages 26, 38, and 39.*

Mahler, J., Patil, S., Kehoe, B., Van Den Berg, J., Ciocarlie, M., Abbeel, P., and Goldberg, K. (2015). Gp-gpis-opt: Grasp planning with shape uncertainty using gaussian process implicit surfaces and sequential convex programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4919–4926. IEEE. *Cited on page 10.*

Martens, W., Poffet, Y., Soria, P. R., Fitch, R., and Sukkarieh, S. (2017). Geometric priors for gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 2(2):373–380. *Cited on pages 10, 59, and 60.*

Marton, Z.-C., Goron, L., Rusu, R. B., and Beetz, M. (2011). Reconstruction and verification of 3d object models for grasping. In *Robotics Research*, pages 315–328. Springer. *Cited on pages 9 and 10.*

Matsubara, T. and Shibata, K. (2017). Active tactile exploration with uncertainty and travel cost for fast shape estimation of unknown objects. *Robotics and Autonomous Systems*, 91:314–326. *Cited on pages 10, 11, 12, 19, 68, 71, 72, 84, and 85.*

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE. *Cited on pages 30 and 106.*

Metta, G., Sandini, G., Vernon, D., Natale, L., and Nori, F. (2008). The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pages 50–56. ACM. *Cited on page 16.*

Mittendorfer, P. and Cheng, G. (2011). Humanoid multimodal tactile-sensing modules. *IEEE Transactions on robotics*, 27(3):401–410. *Cited on pages 14 and 17.*

Morales, A., Asfour, T., Azad, P., Knoop, S., and Dillmann, R. (2006). Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5663–5668. IEEE. *Cited on pages 20 and 26.*

Morrison, D., Corke, P., and Leitner, J. (2018). Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*. *Cited on page 26.*

Navarro, S. E., Marufo, M., Ding, Y., Puls, S., Göger, D., Hein, B., and Wörn, H. (2013). Methods for safe human-robot-interaction using capacitive tactile proximity sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1149–1154. IEEE. *Cited on page 17.*

Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2016). Detecting object affordances with convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770. IEEE. *Cited on pages 23, 35, 36, 37, and 149.*

Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2017). Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5908–5915. IEEE. *Cited on page 36.*

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528. *Cited on page 36.*

OpenAI (2018). Openai five. `https://blog.openai.com/openai-five/`. Accessed: 2019-04-16. *Cited on page 27.*

Ottenhaus, S., Kaul, L., Vahrenkamp, N., and Asfour, T. (2018a). Active tactile exploration based on cost-aware information gain maximization. *International Journal of Humanoid Robotics*, 15(01):1850015. *Cited on pages 3, 75, and 136.*

Ottenhaus, S., Renninghoff, D., Grimm, R., Ferreira, F., and Asfour, T. (2019). Visuo-haptic grasping of unknown objects based on gaussian process implicit surfaces and deep learning. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. *Cited on pages 4, 102, 104, 107, 110, 111, 113, 115, 116, 130, 131, and 137.*

Ottenhaus, S., Weiner, P., Kaul, L., Tulbure, A., and Asfour, T. (2018b). Exploration and reconstruction of unknown objects using a novel normal and contact sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1614–1620. IEEE. *Cited on pages 3, 66, 81, 95, 96, 99, and 136.*

Park, D. and Chun, S. Y. (2018). Classification based grasp detection using spatial transformer network. *arXiv preprint arXiv:1803.01356*. *Cited on page 26.*

Park, D., Seo, Y., and Chun, S. Y. (2018). Real-time, highly accurate robotic grasp detection using fully convolutional neural networks with high-resolution images. *arXiv preprint arXiv:1809.05828*. *Cited on page 26.*

Pelossof, R., Miller, A., Allen, P., and Jebara, T. (2004). An svm learning approach to robotic grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3512–3518. IEEE. *Cited on page 20.*

Pratt, J., Torres, A., Dilworth, P., and Pratt, G. (1996). Virtual actuator control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1219–1226. IEEE. *Cited on page 11.*

Przybylski, M., Asfour, T., and Dillmann, R. (2010). Unions of balls for shape approximation in robot grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1592–1599. IEEE. *Cited on page 9.*

Quispe, A. H., Milville, B., Gutiérrez, M. A., Erdogan, C., Stilman, M., Christensen, H., and Amor, H. B. (2015). Exploiting symmetries and extrusions for grasping household objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3702–3708. IEEE. *Cited on page 9.*

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*, volume 1 of *Adaptive Computation and Machine Learning*. MIT Press. *Cited on page 75.*

Redmon, J. and Angelova, A. (2015). Real-time grasp detection using convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE. *Cited on pages 26, 32, 33, and 39.*

Romero, J., Feix, T., Kjellström, H., and Kragic, D. (2010). Spatio-temporal modeling of grasping actions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2103–2108. IEEE. *Cited on page 28.*

Roncone, A., Hoffmann, M., Pattacini, U., Fadiga, L., and Metta, G. (2016). Peripersonal space and margin of safety around the body: Learning visuo-tactile associations in a humanoid robot with artificial skin. *PloS one*, 11(10):e0163713. *Cited on page 16.*

Roncone, A., Hoffmann, M., Pattacini, U., and Metta, G. (2014). Automatic kinematic chain calibration using artificial skin: self-touch in the icub humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2305–2312. IEEE. *Cited on page 16.*

Rosales, C., Spinelli, F., Gabiccini, M., Zito, C., and Wyatt, J. L. (2018). Gpatlasrrt: a local tactile exploration planner for recovering the shape of novel objects. *International Journal of Humanoid Robotics*, 15(01):1850014. *Cited on page 10.*

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China. *Cited on page 105.*

Santaera, G., Luberto, E., Serio, A., Gabiccini, M., and Bicchi, A. (2015). Low-cost, fast and accurate reconstruction of robotic and human postures via imu measurements. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2728–2735. IEEE. *Cited on page 18.*

Schiebener, D., Schmidt, A., Vahrenkamp, N., and Asfour, T. (2016). Heuristic 3d object shape completion based on symmetry and scene context. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 74–81. IEEE. *Cited on pages 10 and 33.*

Schmidt, P., Vahrenkamp, N., Wächter, M., and Asfour, T. (2018). Grasping of unknown objects using deep convolutional neural networks based on depth images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6831–6838. IEEE. *Cited on pages 23, 31, and 149.*

Schmitz, A., Maggiali, M., Natale, L., Bonino, B., and Metta, G. (2010). A tactile sensor for the fingertips of the humanoid robot icub. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2212–2217. IEEE. *Cited on page 16.*

Schmitz, A., Maiolino, P., Maggiali, M., Natale, L., Cannata, G., and Metta, G. (2011). Methods and technologies for the implementation of large-scale robot tactile sensors. *IEEE Transactions on Robotics*, 27(3):389–400. *Cited on page 16.*

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017a). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*. *Cited on page 27.*

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017b). Mastering the game of go without human knowledge. *Nature*, 550(7676):354. *Cited on page 27.*

Solina, F. and Bajcsy, R. (1987). Three dimensional object representation revisited. In *In Proceedings of the International Conference on Computer Vision*, pages 231–240. *Cited on page 9.*

Sommer, N., Li, M., and Billard, A. (2014). Bimanual compliant tactile exploration for grasping unknown objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6400–6407. IEEE. *Cited on pages 10 and 68.*

Starke, J., Weiner, P., Hundhausen, F., Beil, J., and Asfour, T. (2018). The kit prosthetic hand: Design and control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3328–3334. IEEE. *Cited on page 119.*

Tar, . and Cserey, G. (2011). Development of a low cost 3d optical compliant tactile force sensor. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 236–240. *Cited on pages 14 and 16.*

Tenzer, Y., Jentoft, L. P., and Howe, R. D. (2012). Inexpensive and easily customized tactile array sensors using mems barometers chips. *IEEE R&A Magazine*. *Cited on pages 15 and 80.*

Thrun, S. and Wegbreit, B. (2005). Shape from symmetry. In *IEEE International Conference on Computer Vision*, volume 2, pages 1824–1831. IEEE. *Cited on page 9.*

Tuthill, J. C. and Azim, E. (2018). Proprioception. *Current Biology*, 28(5):R194–R203. *Cited on page 6.*

Vahrenkamp, N., Barski, A., Asfour, T., and Dillmann, R. (2009). Planning and execution of grasping motions on a humanoid robot. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 639–645. IEEE. *Cited on page 132.*

Vahrenkamp, N., Koch, E., Wächter, M., and Asfour, T. (2018). Planning high-quality grasps using mean curvature object skeletons. *IEEE Robotics and Automation Letters*, 3(2):911–918. *Cited on pages 31, 83, 105, and 132.*

Vahrenkamp, N., Kröhnert, M., Ulbrich, S., Asfour, T., Metta, G., Dillmann, R., and Sandini, G. (2013). Simox: A robotics toolbox for simulation, motion and grasp planning. In *Intelligent Autonomous Systems*, pages 585–594. Springer. *Cited on pages 82 and 105.*

Varley, J., DeChant, C., Richardson, A., Ruales, J., and Allen, P. (2017). Shape completion enabled robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447. IEEE. *Cited on pages 23, 34, 35, and 149.*

Varley, J., Watkins-Valls, D., and Allen, P. (2018). Multi-modal geometric learning for grasping and manipulation. *arXiv preprint arXiv:1803.07671*. *Cited on pages 23 and 34.*

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`. Accessed: 2019-04-16. *Cited on page 27.*

Wang, S., Jiang, X., Zhao, J., Wang, X., Zhou, W., Liu, Y., et al. (2019). Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images. *arXiv preprint arXiv:1902.08950*. *Cited on page 26.*

Wang, Z., Li, Z., Wang, B., and Liu, H. (2016). Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering*, 8(9):1687814016668077. *Cited on page 26.*

Weiß, K. and Worn, H. (2005). The working principle of resistive tactile sensor cells. In *IEEE International Conference on Mechatronics and Automation*, volume 1, pages 471–476. IEEE. *Cited on page 15.*

Weisz, J. and Allen, P. K. (2012). Pose error robust grasping from contact wrench space metrics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 557–562. IEEE. *Cited on pages 29, 108, 109, 112, and 132.*

Wettels, N., Fishel, J. A., and Loeb, G. E. (2014). Multimodal tactile sensor. In *The Human Hand as an Inspiration for Robot Hand Development*, pages 405–429. Springer. *Cited on pages 14 and 17.*

Wettels, N., Santos, V. J., Johansson, R. S., and Loeb, G. E. (2008). Biomimetic tactile sensor array. *Advanced Robotics*, 22(8):829–849. *Cited on pages 14 and 17.*

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA. *Cited on page 60.*

Williams, O. and Fitzgibbon, A. (2007). Gaussian process implicit surfaces. *Gaussian Proc. in Practice*. *Cited on pages 10, 18, 54, 55, 58, 59, and 63.*

Yamaguchi, A. and Atkeson, C. G. (2016). Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1045–1051. IEEE. *Cited on page 16.*

Yang, S., Jeon, S., and Choi, J. (2016). Level-set based greedy algorithm with sequential gaussian process regression for implicit surface estimation. In *ASME Dynamic Systems and Control Conference*, pages V002T25A001–V002T25A001. American Society of Mechanical Engineers. *Cited on page 10.*

Yi, Z., Calandra, R., Veiga, F., van Hoof, H., Hermans, T., Zhang, Y., and Peters, J. (2016). Active tactile object exploration with gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930. IEEE. *Cited on pages 10, 68, 69, and 85.*

Yousef, H., Boukallel, M., and Althoefer, K. (2011). Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: physical*, 167(2):171–187. *Cited on page 18.*

Yuan, W., Dong, S., and Adelson, E. (2017). Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762. *Cited on page 16.*

Yuan, W., Li, R., Srinivasan, M. A., and Adelson, E. H. (2015). Measurement of shear and slip with a gelsight tactile sensor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 304–311. IEEE. *Cited on page 16.*

Zha, H., Hoshide, T., and Hasegawa, T. (1998). A recursive fitting-and-splitting algorithm for 3-D object modeling using superquadrics. In *International Conference on Pattern Recognition*, volume 1, pages 658–662. IEEE. *Cited on page 9.*

Zhang, Q., Qu, D., Xu, F., and Zou, F. (2017). Robust robot grasp detection in multimodal fusion. In *MATEC Web of Conferences*, volume 139, page 00060. EDP Sciences. *Cited on page 26.*

Zhou, X., Lan, X., Zhang, H., Tian, Z., Zhang, Y., and Zheng, N. (2018). Fully convolutional grasp detection network with oriented anchor box. *arXiv preprint arXiv:1803.02209*. *Cited on page 26.*