Technische Universität Berlin

School IV - Electrical Engineering and Computer Science

Department of Computer Engineering and Microelectronics

Robotics and Biology Laboratory

Master Thesis

Extracting shape primitives from point clouds for grasping

---

presented by

Stefan Schrandt

Matr.-Nr.: 324492

stefan.schrandt@me.com

Technische Informatik

Date of submission: **07. June 2012**

Examiners: **Prof. Dr. Oliver Brock, Prof. Dr. Olaf Hellwich**

Advisors: **Prof. Dr. Oliver Brock, Dipl. Inf. Clemens Eppner**

Eidesstattliche Erklärung

Die selbständige und eigenhändige Ausfertigung versichert an Eides statt

Berlin, den

Unterschrift

Abstract

Grasping of unknown objects in robotics is an ongoing topic in robotic research and is not uniquely understood or solved. Robot grasping is an essential feature that enables robots to solve a lot of tasks. If a robot cannot grasp, it is very limited in solving tasks. Most tasks that autonomous robots need to do, requires grasping.

To find a solution for the grasp problem the neuroscience research gives suggestions. The compliance of the human hand and the observation that humans use this compliance to grasp during the closing procedure of the hand shows that the grasping problem is reduced from finding the complete hand configuration to finding a pre-shape of the hand. In addition to the pre-shape of the hand, robots need the position and orientation of that object to grasp it.

This thesis depicts an approach that perceives the environment with a 3D depth sensor and extracts shape primitives out of point clouds. The point cloud contains the whole scene including the objects to grasp. In addition to the shape primitive the pose of that object is estimated. The best estimated shape primitive found by this approach is utilized by a simple heuristic to find the best pre-grasp for that object. The shape primitives are initialized with a RANSAC algorithm and tracked and evaluated over time with a particle filter. The real world experiments show that a robot that uses this approach is able to grasp a range of various objects.

The results of this work shows that the grasping problem can be reduced from searching the complete hand configuration to search a pre-grasp and the corresponding pose of an object.

Key words: robotics, grasping, hand compliance, shape primitives, pre-grasp.

Kurzfassung

Das Greifen von unbekannten Objekten ist ein andauerndes Problem in der Robotikforschung. Das Greifproblem wurde bisher weder eindeutig verstanden, noch gelöst. Das Greifen des Roboters ist eine wesentliche Eigenschaft, welche dem Roboter die Erledigung einiger Aufgaben ermöglicht. Wenn ein Roboter nicht in der Lage ist zu greifen, so ist seine Einsetzbarkeit stark eingeschränkt. Viele Aufgaben von autonomen Robotern benötigen das Greifen um sie erfolgreich zu erledigen.

Neurowissenschaften geben Impulse für das Finden einer Lösung dieses Greifproblems. Die Nachgiebigkeit der menschlichen Hand und die Beobachtung, dass Menschen die Nachgiebigkeit beim Schließen der Hand benutzen, zeigt, dass das Greifproblem reduziert werden kann. Es muss nicht mehr der komplette Konfigurationsraum der Hand durchsucht werden, sondern es muss nach einem Vorgriff gesucht werden. Zusätzlich zu dem Vorgriff benötigt der Roboter die Position und die Orientierung des zu greifenden Objekts.

Diese Thesis beschreibt einen Ansatz, der das Umfeld mit einem 3D Sensor wahrnimmt und aus den Eingangssignalen geometrische Primitiven extrahiert. Das Eingangssignal bei diesem Ansatz ist eine Punktwolke von der Umgebung. Diese Punktwolke enthält neben den Umfeldpunkten auch die Punkte für das zu greifende Objekt. Neben den geometrischen Primitiven werden die Position und die Orientierung des Objektes geschätzt. Die Primitive mit der höchsten Wahrscheinlichkeit wird von einer einfachen Heuristik genutzt um den daraus resultierenden Vorgriff zu generieren. Initial werden die geometrischen Primitiven durch ein RANSAC Algorithmus ermittelt. Ein Partikelfilter wird genutzt um die Primitiven über die Zeit zu verfolgen und um die Primitiven zu validieren. Die Experimente in der echten Welt zeigen, dass Roboter mit diesem Ansatz in der Lage sind eine Reihe von unterschiedlichen Objekten zu greifen.

Das Ergebnis dieser Arbeit zeigt, dass das Greifproblem vom Durchsuchen des gesamten Konfigurationsraum auf die Suchen eines Vorgriffs reduziert werden kann.

Titel: Extrahieren von Form Primitiven aus Punktwolken für das Greifen

Table of Contents

List of Figures

List of Tables

Notation Remarks

Some abbreviations used in this work are introduced:

- Vectors are lower case bold: $\mathbf{v}$
- Matrices are upper case bold: $\mathbf{M}$
- Scalars are mixed case italic: S, r
- Transformations are T with source s and target t frame: ${}^{s}_{t}T$

1 Introduction

This chapter gives a short overview about the use of robots and the main difference between industrial and autonomous robots. It shows some examples of autonomous robots and indicates why grasping is an essential feature to enable robots to interact with humans or manipulate the physical environment. Additionally it includes the aim and the outline of this work.

1.1 Motivation

Robots can support humans in a lot of situations like working in a hazardous environment, doing physically hard work for humans, or do work that humans do not want to do like cleaning up or similar things. Today most robots perform their tasks in industries. These industrial robots are build for performing only a few or maybe just one - mostly static - step of the complete production cycle. Therefore a lot of robots are used in industries to produce a product. These robots do not work with humans. They do the hard work for the humans.

There are a lot more situations in our daily life where robots can assist or even interact with humans in everyday tasks. The so called service robots shown in Figure 1.1 perform tasks in this fields. In the left image the personal robot PR1 cleans the living room. The Care-o-bot 3 robot in the middle picture serves an elderly and disabled woman something to drink. The right image shows the robot Justin preparing a drink.

Figure 1.1: Some examples for robots that operate autonomously in unstructured environments. From left to right: Personal robot PR1 cleaning up toys [3], Service robot Care-o-bot 3 supporting disabled or elderly people [1], Services robot Justin doing domestic tasks [2].

All examples in Figure 1.1 show autonomous service robots that operate in unstructured and dynamic environments. The ability to operate in this unstructured

and dynamic environment is the key difference between robots in industries and autonomous robots. Dynamic environment, in this case, means the environment can change through outside influences or by the robot itself. Outside influences could be a human that moves through the environment. That implies the unpredictability and uncertainty in real world environments. The service robots themselves can move through the physical world. Additionally, the robots often need to manipulate the physical environment to accomplish their tasks. Some industrial robots can also change their environments but the environment is engineered for the industrial robots. In general the tasks the autonomous robots have to do is, besides collision free motion, the ability to interact with and directly manipulate the physical environment.

All robots in Figure 1.1 on the preceding page need to solve its task manipulation of the physical world. In these cases manipulation includes grasping objects. The PR1 in the left image needs to grasp the toys that are lying on the ground and put them in a box. The care-o-bot 3 in the second image has to grasp the cup in order to deliver it to the elderly woman. In the right image Justin needs to grasp the can to tip the powder in the glass.

The above mentioned examples show that grasping is an essential feature to enable robots to interact with humans or manipulate the physical environment in order to solve a lot of tasks. Most current grasping approaches need perfect geometrical models of the object, in order to plan the grasp. However, finding a perfect model of a simple object (e.g. a massage ball) is hard due to input noise. These approaches are successful only when a perfect model is available. Nevertheless, those grasping techniques often fail during real world experiments where the model is not known precisely.

For humans however, grasping seems to be a simple task. Whenever one grasps an object, the hand is oriented and pre-shaped regarding the volume of the object. Additionally, the human hand uses its compliance to achieve a better contact surface on that object and therefore a more stable grasp. The compliance of the hand is a result of the hands mechanical structure and the hands surface structure.

Many robot hands have mechanical structures similar to human hands. They sometimes even have a soft surface structure. The compliance of the hand allows us to achieve a stable grasp without knowing the exact shape. We rather need a coarse model that describes the rough pose (position and orientation) of the hand and choses a pre-grasp. A pre-grasp is pre-configuration and pre-forming of the robot

hand. Having the pose and the pre-grasp there is no need for searching the high dimensional configuration space of the hand for the best grasp.

1.2 Aim of Work

The aim of this thesis is to find shape primitives that graspable objects can be decomposed into. This decomposition allows a stable grasp of those objects. Geometrical shapes like cone, cylinder, sphere and plane are considered as shape primitives in this work. These geometrical shape primitives are used to create a coarse model of perceived objects. The coarse model consists only of one or multiple of the above named shape primitives that describe the objects to grasp. The results of this approach are the shape primitives with their position, orientation and size that describes the coarse model of each object. This results can be used as an input of a grasping planner like graspIt! [10]. More interesting is the fact that this approach enables grasping of an object, described by the coarse model, with simple heuristics. That means this approach needs no object recognition in order to get a model and grasp objects based on the recognition. The heuristic estimates the position and orientation of the robots hand and additionally choses the pre-grasp to perform the grasp.

1.3 Thesis Outline

The remainder of the thesis is organized is as follows. Chapter 2 deals with background information and describes related work in robot grasping and shape approximation. In Chapter 3, I describe the problem that is solved in this work. The description gives a more detailed view of the problem started in this introduction. The methods I used to solve this problem are written in Chapter 4. To present the results of my work I illustrate the experiments of this work in Chapter 5. In the last chapter, Chapter 6, I finally conclude my work.

2 Background and Related Work

This chapter starts with a short background section of the field of grasping and continues with the related work section. The related work section contains works of shape approximation in the field of robot grasping.

2.1 Background

This section contains background information used in the related work chapter and in the rest of this thesis. It starts with the mechanical background, describes form and force closure and depicts afterwards the different contact models in robot grasping.

2.1.1 Mechanics of Grasping

In the field of robot grasping, grasps are mostly defined by hand posture and hand position. The orientation and the position of all links of the hand determine the hand posture. The orientation and position of the hand itself defines the hand position.

The internal degrees of freedom (internal DoFs) define the hand posture. The number of links of robot hands do not have to be the same as the number of the internal DoFs. If multiple links share one DoF, they share one actuator. This is called shared links. The Barrett hand shown in figure 2.1 on page 6, as an example, has two links in every finger that share one actuator. Such hand uses the advantages of the passive mechanical adaption. That means if the first link has contact with an object and has reached its final position the second link can still move until they touch the object too. The first link in this case means the link that is connected with the palm. The third link is the fingertip and the second connects the first and third link.

The external DoFs assign the hand position. The position and the orientation define the hand position. Therefore, six variables are describing the hand position in its entirety. Three variables are need for translation and three for rotation. That means six variables for the hand position plus d variables for the hand posture are needed

to describe a complete grasp, where d is the number of the internal DoFs of the robot hand.

2.1.2 Form and Force Closure

The first step in robot grasping is holding an object rather than manipulating it. In robot grasp planning, there are two concepts that describe the stable holding of an object. A stable grasp in the field of grasping means that the robot grasps an object and moves it, without losing the contact between the object and the hand. The first concept that describes if a grasp is stable is form closure. Form closure means that the fingers of the robot hand encloses the object. The enclosed object is not movable in the hand through outside disturbances. This fact is given, even if the hand has frictionless contact points. Figure 2.1 on the following page shows an example of form closure where the Barrett hand grasps an sphere which is not completely round. Would the sphere be completely round the sphere could rotate in the hand which would mean it is not form closure.

The counterpart of form closure is force closure. If a grasp is force closure the fingers of a hand do not need to enclose the object. The object is held by the hand's contact points. These contact points need friction in order to grasp successfully. With a force closure grasp the hand can perform a stable grasp even when external forces apply to that object. How a contact point with friction is calculated is considered in the next section. Figure 2.2 on the next page shows an example of a force closure grasp.

2.1.3 Contact Models

Many approaches in robot grasping use simulation to search for the best grasp. As mentioned in the introduction these approaches need perfect models to simulate grasps. They also need to calculate the contact models to determine the best grasps. This is the mathematical way to identify if a grasp is form closure or force closure. The number of points where the hand has contacts to the objects defines a grasp. These contacts can have three different contact model properties in robot grasping: frictionless contact point model, contact point model with friction or soft finger contact model. If the fingers have frictionless contact point models they apply force only along the normal of the contact point. The left image in Figure 2.3 on the

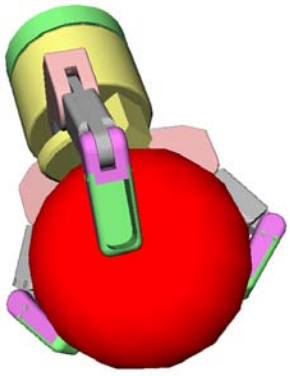

Figure 2.1: The Barrett hand holds a small sphere with a form closure grasp. Through the enclosing of the object the grasp is stable.

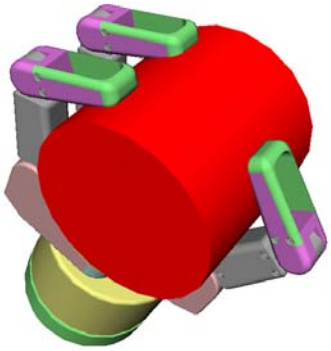

Figure 2.2: The Barrett hand holds a cylinder with a force closure grasp: Through the contact points with friction the grasp is stable.

following page shows this property. Additionally to the frictionless contact points, the contact point models with friction, called also hard contact model, can apply force tangential to the contact points normal. The middle image in Figure 2.3 on the next page depicts this property. The right image in Figure 2.3 on the following page shows the soft contact models which also perform a moment around the contact point normal.

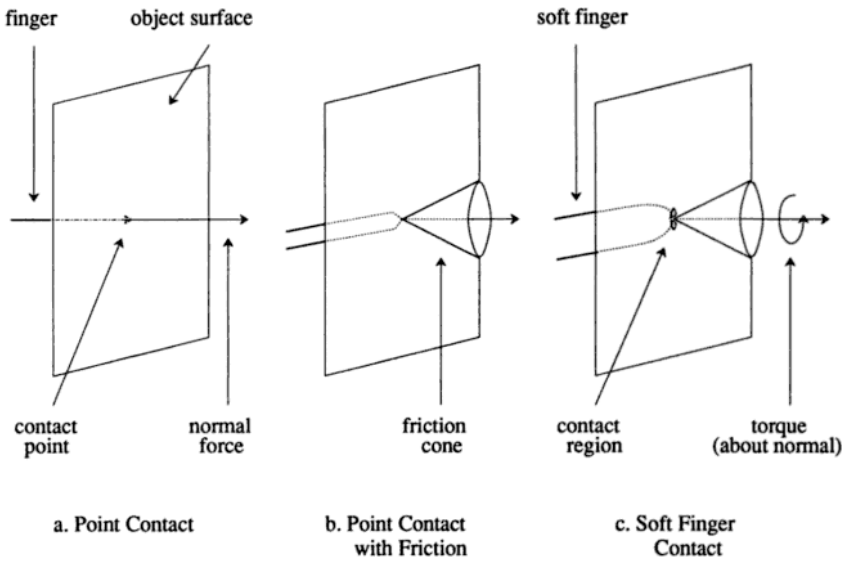

Figure 2.3: The three different contact point models. The first is the point contact model without friction, then there is a depiction of a point contact with friction and the last model is the soft finger contact point. Image taken from [11]

2.2 Related Work

This section deals with the related work in the field of shape approximation for robot grasping. It starts with a grasp planner that uses shape primitives to plan the grasp for an object. Afterwards, the section treats different approaches on shape approximation in the field of robot grasping.

In related work on robot grasping many approaches rely on a grasp planner in order to find a stable grasp. To utilize this grasp planners, these approaches need a detailed model of the object in order to grasp that object. These approaches mostly search for the best grasp using sampling techniques. Other approaches like Miller et al.[12] do not need a detailed model of the object, but rather a rough description of the object. Geometric primitives mostly describe the rough model of objects in this category.

Miller et al.[12] show an approach that generates a set of grasping points and pre-grasps from modeled object. A set of shape primitives, such as boxes, cones, cylinders and spheres model that object. Figure 2.4 on the next page shows a mug modeled with shape primitives. To create this set of grasping points and pre-grasps they use a set of heuristic rules. An example of the heuristics is that spheres should

Figure 2.4: Shape primitives model of a mug. The left image shows an CAD model of a mug. The right image shows the shape primitive model of the mug. The modeled mug uses two shape primitives: a cylinder and a box. Image taken from [12]

be grasped with the spherical pre-grasp shape and the approach vector of the palm should point trough the sphere center. From these rules they sample a set of starting positions. Figure 2.5 shows hypotheses for the four above named shape primitives.

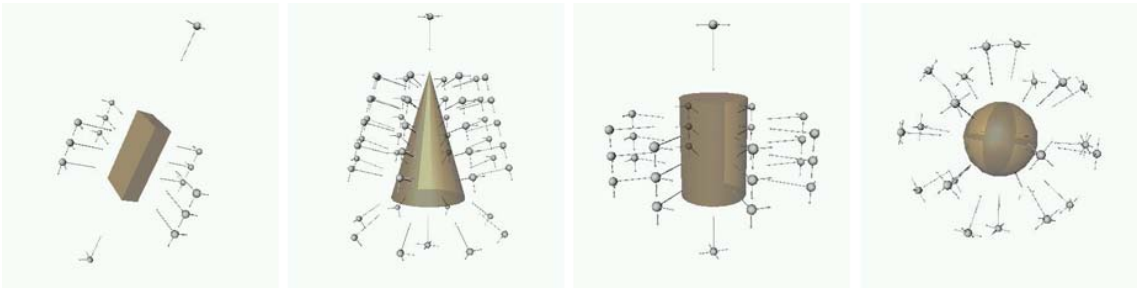

Figure 2.5: Grasp hypotheses of the different shape primitives. The first image shows the box primitive with the gasp hypothesis. The small spheres symbolize the start positions of the grasp hypothesis and the lines symbolize the approach vector of the palm. The second image shows the cone and its hypothesis. The last two images show the cylinder and the sphere with their hypotheses. Image taken from [12]

My approach extracts shape approximations from objects and I show how the work of Miller et al. deals with the input coming from my approach. This is shown in the experiments in Chapter 5.

Their planning framework GraspIt![10] evaluates every starting position and its corresponding pre-grasp shape. The robot hand moves in simulation along the approach vector until the hand contacts the object. In that position the simulation closes the fingers until they contact the object, too. The planning framework searches the best force closure grasp for the given criteria. This approach needs between 1-5 minutes to plan that grasp. They also show only simulation experiments. They assume that

the shape approximation of the object to grasp is known a-priori.

Huebner et al.[13] show in their work how to select robot pre-grasps using box-based shape approximation. They wrap 3D data points of that object to grasp in a box shape primitive. They do this with a fit and split algorithm that is based on minimum volume bounding boxes. Figure 2.5 on the previous page shows box primitives that model the scan of a stapler.

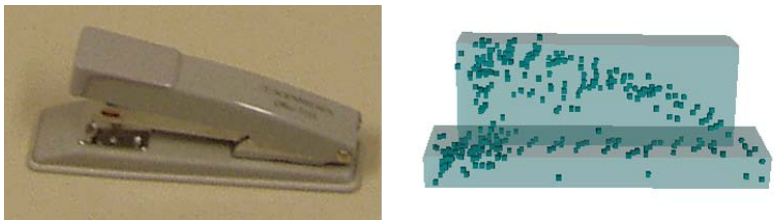

Figure 2.6: Box shape primitives model a staple scan. The left image shows the stapler and the right image shows the scan of the stapler wrapped in box shape primitives. Image taken from [14]

They define that each face of a found box primitive has maximal 4 grasp hypothesis. Thereby, the normal of the faces is the approach vector of the grasp hypothesis and the orientation of the hand bases on one of the four edges of the box face. To find the best grasp hypothesis they use some heuristics like using the largest box and using the face pointing to the viewpoint. These heuristics are used to reduce the grasp hypotheses. To choose the most promising hypothesis they trained a grasp predictor with a supervised learning method. In an example of that work they use a toy duck that starts with 72 hypotheses. The number of hypotheses reduces through the heuristics to eight. From these eight, the grasp predictor selects the one with the best predicted quality. In comparison to the work of Huebner et al. my work uses multiple shape primitives that allows using different pre-grasp shapes like cylindrical or spherical pre-grasps.

Goldfeder et al.[15] present a grasp planning approach with superquadric decomposition trees. To prune the intractably large grasp space into a subspace, they use decomposition trees. They claim that this subspace contains many good grasps to grasp that object. The left image in Figure 2.7 on the following page shows a decomposition of a toy bear to 8 superquadrics. These decompositions result in the decomposition tree shown in the right image of Figure 2.7 on the next page.

Within this subspace of grasps they sample for each superquadric separately. This is done with a heuristic that places approach points on the superquadric surface at

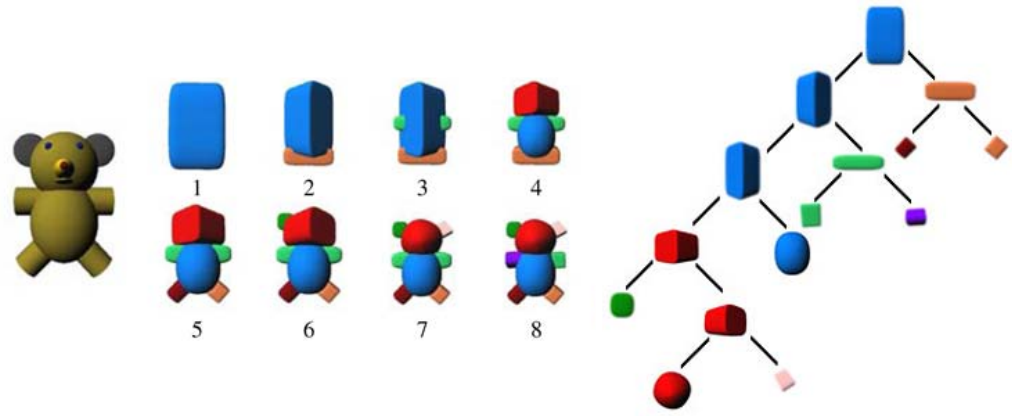

Figure 2.7: Superquadric decomposition of a toy bear and the resulting decomposition tree. The left image shows a toy bear decomposed in 8 superquadrics. The right image depicts the resulting decomposition tree out of the superquadrics. Image taken from [15]

approximately 1 inch intervals. For each point they start position the hand with a small distance between hand and surface. The approach vector of the hand is along the surface normal. They simulate their approach with GraspIt![10]. In the same way as Miller et al they move and close the hand until it touches the object. This approach has only experiments in simulation, whereas my approach is evaluated in real world experiments.

Przybylski et al.[16] present a grasp planning approach. Their approach uses the medial axis approximation of a-priori known objects. The medial axis used in their work describes the set of center points of maximally inscribing balls. These inscribing balls have the maximum diameter and have at least two contacts with the object shape boundary. The authors claim that their representation with medial axes is much more precise than the box or superquadrics decompositions described above by Huebner et al. and Goldfeder et al.. Przybylski et al. say box or superquadrics decompositions do not resemble the geometry carefully enough. To compute the grasp hypothesis they first sample the object surface and generate from that object the medial axis representation. Then they slice the medial axis parallel to the support plane and construct the minimum spanning tree of each slice. In the last step they classify the minimum spanning trees into four categories: circles, stars with ring, trees and symmetry axis elements. Out of these categories they generate grasp candidates by using the OpenRAVE grasp planner[17] that searches force closure grasps in the same way as done by the GraspIt! simulator. The hand moves along the approach vector until the hand touches the object. In that position the

simulation closes the fingers until they touch the object, too.

Another approach is the work of Bone et al.[18]. Their approach uses silhouettes and structured light to obtain a 3D model of an a-priori unknown object. The unknown objects used in their work can be grasped with a parallel-jaw gripper. The camera in their work is mounted at the hand. The hand with the camera moves in a circular trajectory around the object. During this movement, their approach captures silhouettes out of the 2D image from different viewpoints. The sub-optimal trajectory results into a bad modeling of the objects top surface. Therefore, the authors apply a structured light based method to estimate the top surface. The combination of the silhouettes and the structured light results create the model in their work. In this model, a region-growing like algorithm determines a set of flat surfaces. They extract flat surface pairs that fall within a parallelism tolerance. Their approach choses the flat surface pair with the largest area. The robot uses this flat surface pair to grasp that object with its parallel-jaw gripper. This work is limited for a parallel-jaw gripper whereas my approach is in generally usable with all kinds of hands. To use different hands the pre-grasps of the hands need to determine and the heuristics has to change for the available pre-grasps.

In the work of Dune et al.[19] they estimate the shape of an unknown object to grasp. They approximate the object shape by a quadric surface. The parameters of this quadric surface are estimated using multiple views of the object. The motion from one view to the next is continuously optimized to reduce the uncertainty on the estimated parameters. The solutions of their approach is based on active contours which can be computed in real time.

3 Problem Analysis

This chapter gives an overview of the grasping problem and depicts advantages and disadvantages of different robotic hands. It contains a short review of the neuroscience research on the research field of robot grasping. Additionally it shows how grasping can use knowledge from the neuroscience community.

A big problem in robot grasping is the high dimensionality of the configuration space of the hand. Beside the six external DoFs of the position and the orientation, the hand itself has additional internal DoFs. Nowadays, robot hands have between one and 20 internal DoFs or even more. The amount of DoFs depends on the design and the mechanics of the hand. The human hand, as an inspiring example for many robotic hand designs, has 20DoF. The fact that the human hand works so well is a big motivation to develop and use also high complex robot hands.

Another problem in robot grasping is perception. The perception coming from a camera, 3D depth sensor or laser scanner is noisy and has sensor errors. As mentioned in the introduction in Section 1 a lot of grasp planners work with perfect models. Therefore, such approaches need perfect models of the object to grasp successfully. The perfect model can only be perceived if the sensor moves around that object as done by [18] or the object itself moves for example on a turntable as done by [20]. In the last approach the models need to be stored in a database and an object recognition algorithm is used to find that model. This last approach is not able to grasp unknown objects which is a significant disadvantage.

3.1 Robot Hands

In order to achieve the manipulation ability of the human hand, anthropomorphic robot hands have been designed in the past years. The right-most image in Figure 3.1 on the following page shows an example of a highly complex anthropomorphic robot hand from the DLR (German Aerospace Center) [21]. The disadvantage of these dexterous hands is the complex control mechanism that the robot needs to control the hand. To benefit from these human hand-like designs, effective algorithms are

Figure 3.1: Different robot hands starting from a simple gripper to a human like robot hand. From left to right: The gripper of the PR2 with 2 joints and 1 actuator [7], SDM robot hand with 8 joints and 1 actuator. [8], Barrett robot hand with 7 joints and 4 actuators [6], Meka H2 robot hand with 12 joints and 5 actuators [9] and the DLR robot Hand II with 16 joints and 13 actuators [5].

needed to handle the high dimensional configuration space. After solving the problem of the control algorithm, these complex hands could than manipulate and use objects rather than just grasp them. Other disadvantages of these type of hands are the expensive production cost and the fragility of these hands.

In contrast to the anthropomorphic robot hands some really simple end-effectors like parallel jaw gripper are developed and used for robot grasping. An example is shown in the first image in Figure 3.1: the PR2 gripper. The advantages of these grippers compared to the dexterous hands are the easy use of these grippers and the simpler control algorithms that the robot needs for this grippers to grasp. Additionally, the production cost of a gripper is much lower than the cost for the dexterous counterparts. However, the applicability of these grippers is limited, due to the low versatility which is needed for complex tasks.

The third and fourth image in Figure 3.1, the Barrett hand and the Meka hand are two examples of robot hands which designs and mechanics are arranged between the two above named hand designs. The Barrett hand is often used in the robotics research community. As a last example the second image in Figure 3.1, the Shape Deposition Manufacturing (SDM) hand shows a hand in which only one actuator controls 8 joints. These so called underactuated robot hands take advantage of the passive mechanical adaption.

3.2 Neuroscience View on Grasping

Results of the neuroscience research indicate that humans do not control all joints of their hands in a decoupled way. Santello et al.[22] showed that humans select the preshape of their hand in a much lower dimensional space compared to the configuration

space of the hand. He shows in his results that the first two components of a principle component analysis over the human hand pre-shapes could account for 80% of the variance. After humans found the pre-shape, they move to the right position and simply close their hand. The object shape and the compliance of the hand moves the fingers in the final position.

A second result of the neuroscience was introduced by Cutkosky [23] and continued by Feix et al. [24]. They showed in their work a grasp taxonomies of human hands. A grasp taxonomy classifies different type of a grasp. In the work of Feix et al. they presented a grasp taxonomy which includes 33 different grasps types of the human hand. These grasp types are grouped in the grasp taxonomy into 17 groups.

3.3 Benefits from Neuroscience for Robot Grasping

The above named results from neuroscience show that most stable grasps can be found in a small number of discrete states. These states are considered pre-grasps in this thesis.

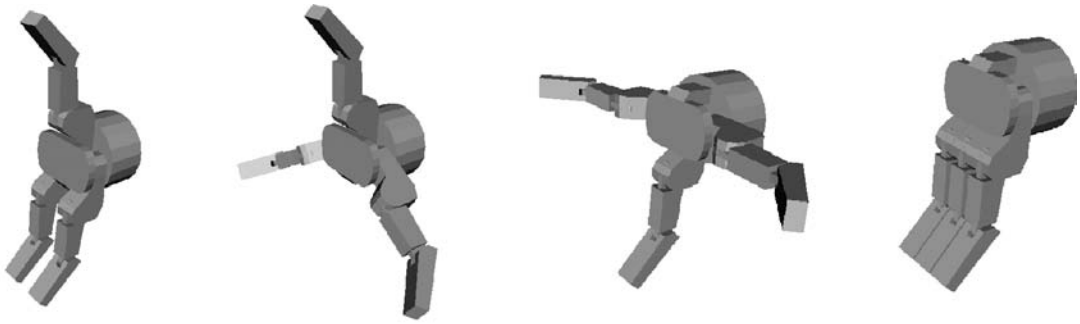

Figure 3.2: Possible pre-grasps of the Barrett hand [4]. From left to right: cylindrical grasp, spherical grasp, pinch grasp and the hook grasp.

The Barrett hand, used in this work, has compared to the human hand only four different grasp types, in this work called pre-grasps. Figure 3.2 shows the possible pre-grasps of the Barrett hand. The first image shows the cylindrical pre-grasp. In this pre-grasp the thumb, the single finger pointing up, is opposed to the two other fingers. The second image depicts the spherical grasp. In this grasp the two fingers are spread out to approximately 120 degrees to the other fingers. In the pinch grasp, shown in the third image, the two fingers are on opposite sides of each other. In the fourth and last image all three fingers are parallel to form the hook grasp. In

this work I focus on the first two pre-grasps: cylindrical pre-grasp and spherical pre-grasp.

My approach extracts shape primitives from the point clouds of the object to grasp. These shape primitives are used to find the best possible pre-grasp for the object to grasp it. In addition to the pre-grasps the robot needs the position and the orientation of the shape primitive in order to grasp that object. Cylinders, cones, spheres and planes are considered shape primitives in this work. Derived from these shape primitives the approach estimates the best known pre-grasp.

In this work I show that successful grasps can be found if the pre-grasps with corresponding position and orientation are known. This approach can deal with sensor noise because it computes only a rough model out of shape primitives. My approach uses this rough model to find the pre-grasp. After finding the pre-grasp the robot only closes the hand. During the closing procedure the compliance of the robot hand moves the finger in the final position. This is done in the same way as described in the introduction in Section 1, by the human hand. Due to this fact the complexity of the hand control is reduced and the sensor noise can be handled. The main part in my work is extracting shape primitives in order to derive the pre-grasp from it.

4 Methods

This chapter contains the methods used in my thesis. It starts with a brief overview of the cooperation of the single modules. The modules in this work are the object detection module, the shape approximation module and the shape primitive improvement module. The modules are described in detail. In addition some results of the different modules are presented.

My approach is basically structured in three modules. The first module detects the dominant horizontal plane on which the objects be located. After the plane is detected my approach considers only the input points that are located above the plane, where the objects are assumed. The input points located above the plane are clustered into objects. The clustering method segments the objects based on the euclidean distance. The second module approximates shapes with the Random Sample Consensus (RANSAC) algorithm. That algorithm fits geometric shapes in the object point clouds and extract the points correspond to the best found shape primitive. After all shape primitives found the third module validates and improves the shape primitives over the time. This is realized with a particle filter. Figure 4.1 shows the cooperation between the modules.

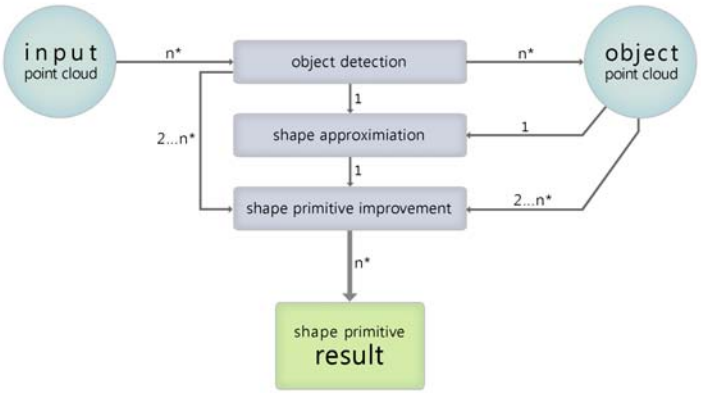

Figure 4.1: Cooperation between the modules.

4.1 Object Detection

The goal of the object detection is to find out which points correspond to an object. Therefore, the first step is to find the dominant horizontal plane. This plane could be in the real world a table a sideboard or equivalent horizontal flat surfaces where the objects to grasp are placed on. After the plane is found the perception focuses only on the area above the plane. In the following I name this the area of interest. To focus on the area of interest all points whose position does no be in this area is cut out of the point cloud. In addition the detected plane is cut out of the point cloud, too. All remaining points in the resultant point cloud correspond to objects placed on in the area of interest. The last step of the object detection is to cluster the points in point clouds that depict separate objects in the area of interest. The points are clustered based on euclidean distance. Therefore, the result of this object detection is a point cloud for every object placed in the area of interest. The euclidean distance clustering has one drawback, it clusters two objects which are placed side by side as one object cluster. This is done because of the missing space between those objects. The resulting point clouds of the clustered objects are the inputs for the shape approximation with the Random Sample Consensus (RANSAC) algorithm.

4.1.1 Dominant Horizontal Plane Detection

As mentioned above the first step to detect the objects out of a scene is searching the dominant horizontal plane. To detect the dominant horizontal plane one needs first to calculate the normals of each point of the point cloud. After the normal calculation the Random Sample Consensus (RANSAC) algorithm is used to extracts a plane that is horizontal. This is achieved if the normal of the plane does not deviate more than α degrees to the z-axis of the base coordinate frame. The z-axis of the base frame is pointing upwards. The right image in Figure 4.2 on the following page shows the result of the table detection out of the input point cloud. The left image shows the whole input point cloud.

Figure 4.2: Table extraction with RANSAC.

Normal estimation of point clouds

Surface normals are important properties of a geometric surface which are used to detect a plane. The normals are also used to find the other shape primitives like cylinder, cone and sphere. The surface normal is a vector that points perpendicular to the surface. The dataset of the input we acquire represent a set of point samples on the real surface. To compute the normal of a point one estimates the normal of a a plane tangent to the surface. That is done by a least-square plane fitting algorithm. To estimate the surface normal one analysis the eigenvectors and eigenvalues of a covariance matrix. This covariance matrix is created from k-nearest neighbors of the point for which the normal is searched.

The covariance matrix C of each point is calculated as follows:

$$C = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \overline{\mathbf{p}}) \cdot (\mathbf{p}_i - \overline{\mathbf{p}})^T \tag{4.1}$$

Where k is the number of the the k-nearest-neighbors of $\mathbf{p}_i$ and $\overline{\mathbf{p}}$ is the 3D centroid of the k-nearest-neighbors.

The eigenvectors and eigenvalues of a covariance matrix are obtained as follows:

$$C \cdot \vec{\mathbf{v}_j} = \lambda_j \cdot \vec{\mathbf{v}_j}, j \in \{0, 1, 2\} \tag{4.2}$$

Where λ_j is the j-th eigenvalue of the covariance matrix and $\vec{\mathbf{v}_j}$ the j-th eigenvector.

A problem of these method is the sign of the normals. The orientation of normals computed with Principal Component Analysis (PCA) is ambiguous. To solve this ambiguity one uses the known viewpoint. To orient all normals n_i of the point cloud in the opposite direction as the viewpoint the following equation needs to be true:

$$\vec{\mathbf{n}_i} \cdot (\mathbf{v}_p - \mathbf{p}_i) > 0 \tag{4.3}$$

In general, to compute the normals of all points at first one select the k-nearest neighbors of point p_i and computes the surface normal n_i of point p_i. The last step is checking if n_i is oriented towards the viewpoint. If that is not given n_i is flipped towards the viewpoint.

Plane Fitting with RANSAC

The RANSAC algorithm described bellow in Section 4.2 searches the plane by randomly selecting three point and normal pairs out of the input point cloud. RANSAC calculates the coefficients of the plane and returns the best result. RANSAC calculates and evaluates models a previous defined number of times. The best result is the plane model that is approximated by the most points of the input data. These coefficients are used to extract the plane out of the input point cloud.

4.1.2 Estimation of Object Points

After the dominant horizontal plane is detected the convex hull of the plane is calculated. To consider only the points of the input point cloud that are located above the table the convex hull is calculated. The set of point indices that represent the convex hull of the dominant horizontal plane and a predefines maximum height above the plane are used to generate a 3D polygonal prism located above the detected plane [25]. The polygonal prism starts close above the plane. That means in this polygonal prism are all points located that correspond to objects positioned on the detected plane. The plane itself is not included in the point cloud located in the polygonal prism. This point cloud is the result of the object point detection and is used to cluster the objects. Figure 4.3 on the next page shows the result of the

object points estimation out of the input point cloud. The left image shows the whole input point cloud.

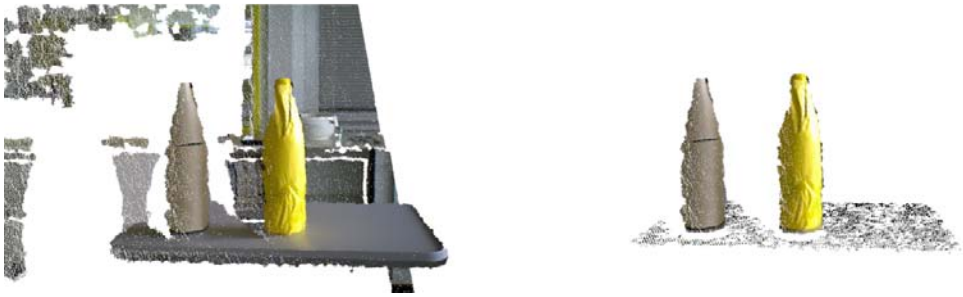

Figure 4.3: Object points extraction out of the whole input cloud.

4.1.3 Object Clustering

The clustering of the objects is done by the euclidean distance. Therefore, the nearest neighbors algorithm is used and the clustering is done with a clustering technique that is similar to a flood fill algorithm [25]. For the point cloud P coming from the object point estimation the Kd-tree representation is created. For every point $p_i \in P$ the cluster algorithm adds the point $p_i \in P$ in a queue Q. The cluster algorithm then searches for the neighbor point set P_k^i of p_i in a given radius r. For every neighbor $p_k^i \in P_k^i$ check if the point is in the queue Q and if not add the point to the queue Q. After all points in the queue Q have been processed, add the queue Q to the list of clusters C and reset Q. The cluster algorithm terminates if all points $p_i \in P$ are processed which means all points are part of the list of clusters C. Figure 4.4 on the following page shows the result of the different clusters resulting out of the object point cloud shown in Figure 4.3. The left image shows the point cloud for theleft bottle and the right image shows the point cloud for the right bottle.

Figure 4.4: Object points extraction out of the whole input cloud.

4.2 Shape Approximation with RANSAC

The Random Sample Consensus (RANSAC) algorithm, described in the next section, initializes the shape approximation in our approach. Therefore, a given input point cloud $P = \{p_1, \ldots, p_N\}$ with corresponding normals $N = \{n_1, \ldots, n_N\}$ is used to estimate a set of primitive shapes $S = \{s_1, \ldots, s_n\}$. The input data comes from the object point detection module described in Section 4.1. In addition to the primitive shapes my approach extracts the corresponding set of points $P_s = \{P_{s_1} \subset P, \ldots, P_{s_n} \subset P\}$ to ensure that points correspond only to one primitive shape. The remaining points $R = P \backslash \{P_{s_1}, \ldots, P_{s_n}\}$ are used for the next iteration of the shape approximation. My approach searches primitive shapes in the remaining point cloud for all given models. The best candidate is taken and stored if the number of points $size|b|$ of the found primitive shape is larger than a given threshold th_2. This loop is done until the remaining point cloud is smaller then another given threshold th_1. The threshold th_1 is given as a factor of the whole input point cloud i.e. 0.3. Algorithm 4.1 shows the above described steps.

The resulting primitive shapes of the RANSAC algorithm have no limitations in their size. Therefore, my approach estimates the sizes of the different models. The size estimation of the different models varies and is described in Section 4.2.6.

4.2.1 Random Sample Consensus

The Random Sample Consensus (RANSAC) algorithm extracts primitive shapes out of point clouds. Therefore, RANSAC randomly picks a minimal set of samples from the point data and constructs the corresponding shape primitive models with this samples. In my work the shape primitives cone, cylinder, plane and sphere are

Algorithm 4.1 Shape approximation with RANSAC

$S \leftarrow \emptyset$
$C \leftarrow \emptyset$
$R \leftarrow P$
while $R < th_1 \cdot P$ **do**
  $C \leftarrow C \cup newCandidates()$
  $b \leftarrow bestCandidate(C)$
  if $count(P_b) > th_2$ **then**
    $R \leftarrow R \backslash P_b$
    $S \leftarrow S \cup b$
  end if
end while
return S

considered. A minimal set RANSAC uses to construct shape primitive models is the smallest number of points that one needs to define shape primitives in unique way. The size of the minimal sets differs by the geometric shape primitives. After computing the model coefficients for the resulting candidate shape RANSAC verifies the model over all points of the input point cloud. Therefore, the distance between each point and the model needs to be smaller then a given threshold ϵ. In addition to the distance the angular distance between the normals of the point and the normals of the models needs to be smaller then a given threshold α. The distance estimation of the different models varies and is described in Section 4.3.2.

If the absolute distance is smaller than the given threshold t the point is counted. The resulting number determines how many points are well approximated by the geometric shape primitive described by the current model. This number is called the score of the shape. The RANSAC algorithm extracts, after a given number of iterations, the shape primitive with the highest score. Figure 4.5 on the following page shows a result of the RANSAC algorithm.

RANSAC is a conceptually simple algorithm, that easily allows extensions of further models. In addition it is usable in a wide range of settings and can deal with data that contains more than 50% of outliers[26].

4.2.2 Cone-Shaped Sample Consensus Model

The cone model is fully defined by two points and the corresponding normals. Because it is computational simpler the cone model is generated with three points

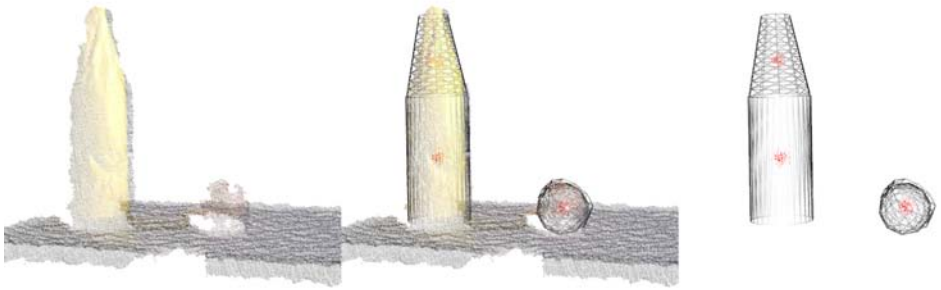

Figure 4.5: Resulting shape primitives of a bottle and a massage ball. The bottle is represented by a cylinder for the bottom and a cone for the top. The sphere represents the massage ball. The left image is the raw point cloud. The middle image shows the shape primitives found for the object and the right image depict the shape primitives separately. The red dots in the middle of the shape primitives depict the center points of the shape primitives.

$(\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2})$ with corresponding normals $(\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_2})$. figure 4.6 shows how the coefficients of the cone model (light blue in the image) are computed with the three point and normal pairs (dark blue in the image).

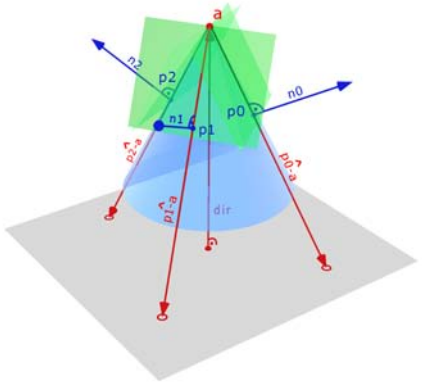

Figure 4.6: Cone-shape RANSAC model.

The cone coefficients contain the vector of the apex **a**, the vector for the direction of the cone's axis **dir** and the opening angle ω.

To calculate the apex a my approach intersects the three planes which are defined by the three point and normal pairs (green in the image). A plane is fully defined by a point with corresponding normal. The intersection of these three planes results

in one point if non of the planes are parallel to one of the other planes

$$\mathbf{n_1} \times \mathbf{n_2} \neq 0 \text{ and } \mathbf{n_2} \times \mathbf{n_3} \neq 0 \text{ and } \mathbf{n_3} x \mathbf{n_1} \neq 0 \tag{4.4}$$

and if the normals of the planes are not coplanar.

$$\mathbf{n1} \cdot (\mathbf{n2} \times \mathbf{n3}) \neq 0 \tag{4.5}$$

If two planes are parallel or the normals are coplanar the one can not compute the model coefficients for the cone with the three given point normal pairs. The intersection is done as follows:

$$\frac{d_1 * (\mathbf{n_2} \times \mathbf{n_3}) + d_2 * (\mathbf{n_3} \times \mathbf{n_1}) + d_3 * (\mathbf{n_1} \times \mathbf{n_2})}{\mathbf{n_1} \cdot (\mathbf{n_2} \times \mathbf{n_3})} \tag{4.6}$$

Where d describes the planes with the point and normal pairs.

$$d_i = \mathbf{p_i} \cdot \mathbf{n_i}, i \in \{1, 2, 3\} \tag{4.7}$$

To calculate the direction of the axis **dir** the normalized vectors between the apex **a** and the three points $(\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2})$ are define a plane by:

$$\{\mathbf{a} + \frac{\mathbf{p_1} - \mathbf{a}}{|\mathbf{p_1} - \mathbf{a}|}, \mathbf{a} + \frac{\mathbf{p_2} - \mathbf{a}}{|\mathbf{p_2} - \mathbf{a}|}, \mathbf{a} + \frac{\mathbf{p_3} - \mathbf{a}}{|\mathbf{p_3} - \mathbf{a}|}\} \tag{4.8}$$

In Figure 4.6 on the preceding page the red vectors ($\mathbf{p_i} \hat{-} \mathbf{a}$) depict the normalize vectors. The three end points of the vectors, marked in the image with a small red circle, define a plane which is gray in the image. The normal of this plane is the direction of the axis **dir** of the cone model.

The last missing coefficient of the cone is the opening angle ω. My approach calculates ω as follows:

$$\omega = \frac{\sum_i arccos((\mathbf{p_i} - \mathbf{a}) \cdot \mathbf{dir})}{3} \tag{4.9}$$

4.2.3 Cylinder-Shaped Sample Consensus Model

A cylinder is fully defined by two points $(\mathbf{p_0}, \mathbf{p_1})$ and the corresponding normals $(\mathbf{n_0}, \mathbf{n_1})$. Figure 4.7 shows how the coefficients of the cylinder model, that is light blue in the image, are computed with the two point and normal pairs. The point and normal pairs are dark blue in the image.

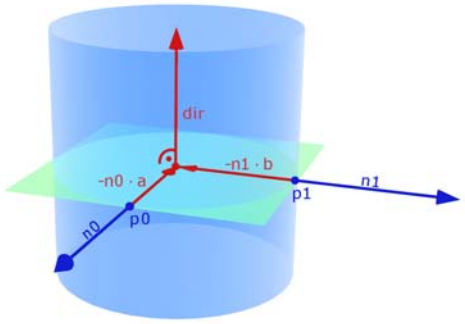

Figure 4.7: Cylinder-shaped RANSAC model.

The cylinder coefficients contain the vector for the point that lies on the axis $\mathbf{p_a}$, the vector for the direction of the axis **dir** and the radius r

To generate a cylinder from two points with normals one calculates the direction of the axis **dir** by:

$$\mathbf{dir} = \mathbf{n_0} \times \mathbf{n_1} \tag{4.10}$$

In Figure 4.7 the direction of the axis **dir** is the marked with a red vector. Then the approach projects the two lines $p0 + t * n_0$ and $p_1 + t * n_1$ along the axis onto

the $\mathbf{a} \cdot \mathbf{x} = 0$ plane. This plane is green marked in the image. The intersection of the two projected lines define the point on the axis of the cylinder $\mathbf{p_a}$.

$$\mathbf{p_a} = \mathbf{p_0} + \mathbf{n_0} + t * \mathbf{n_0} \tag{4.11}$$

where t is

$$t = \frac{(\mathbf{n_0} \cdot \mathbf{n_1}) * (\mathbf{n_1} \cdot (\mathbf{n_0} + \mathbf{p_0} - \mathbf{p_1})) - (\mathbf{n_1} \cdot \mathbf{n_1}) * (\mathbf{n_0} \cdot (\mathbf{n_0} + \mathbf{p_0} - \mathbf{p_1}))}{(\mathbf{n_0} \cdot \mathbf{n_0}) * (\mathbf{n_1} \cdot \mathbf{n_1}) - (\mathbf{n_0} \cdot \mathbf{n_1}) * (\mathbf{n_0} \cdot \mathbf{n_1})} \tag{4.12}$$

The last coefficient of the cylinder, the radius r, is calculated as follows:

$$r = \frac{|\mathbf{dir} \times (\mathbf{p_a} - \mathbf{p_0})|}{|\mathbf{dir}|} \tag{4.13}$$

4.2.4 Plane-Shaped Sample Consensus Model

The plane model is fully defined by three points $(\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2})$. The normals of these points only used to confirm the plane. Figure 4.8 on the next page shows how to calculate the coefficients of the plane model marked in light blue with the three points $(\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2})$ marked in dark blue.

The plane coefficients contain the normal of the plane **dir** and the distance d that lies between the origin to the plane.

The normal of the plane **dir** is calculated with the two vectors $\mathbf{p_1} - \mathbf{p_0}$ and $\mathbf{p_2} - \mathbf{p_0}$.

$$\mathbf{dir} = (\mathbf{p_1} - \mathbf{p_0}) \times (\mathbf{p_2} - \mathbf{p_0}) \tag{4.14}$$

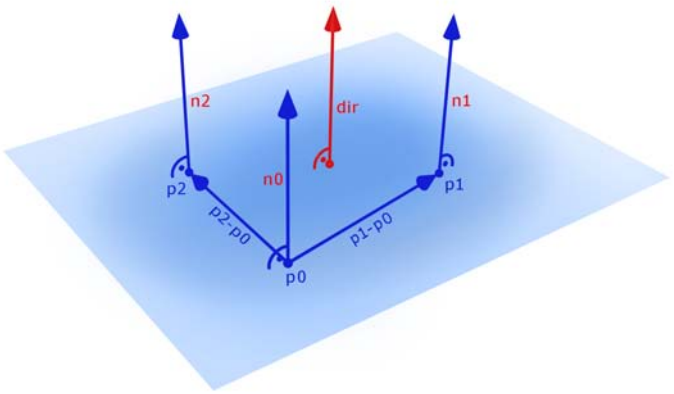

Figure 4.8: Plane-shaped RANSAC model.

To calculate the distance d between the origin and the normal the following is calculated:

$$p = -1 * (\hat{\mathbf{n}} \cdot \mathbf{p_0}) \tag{4.15}$$

The deviation of the normal of the plane and $\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_2}$ is determine. The model is only valid if all deviations are less than a known angel α.

4.2.5 Sphere-Shaped Sample Consensus Model

The sphere model is fully defined by two points $(\mathbf{p_0}, \mathbf{p_1})$ and the corresponding normals $(\mathbf{n_0}, \mathbf{n_1})$. Figure 4.9 on the following page shows how the coefficients of the sphere model marked in light blue in the image, are computed with the two point and normal pairs (marked in dark blue).

The sphere coefficients contain a vector for the center point $\mathbf{c}$ and the radius r.

The center point of the sphere is calculated by determine the midpoint of the shortest distance between the two lines defined by the point and normal pairs. The shortest

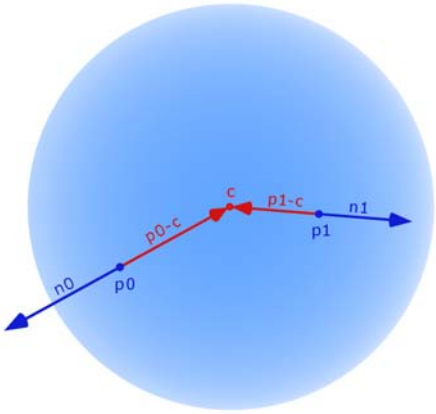

Figure 4.9: Sphere-shape RANSAC model.

distance **d** is calculated as follows:

$$d = \left| \frac{(\mathbf{p_0} - \mathbf{p_1}) \cdot (\mathbf{n_0} \times \mathbf{n_1})}{|\mathbf{n_0} \times \mathbf{n_1}|} \right| \tag{4.16}$$

The sphere radius r is calculated with the two vectors between the points $\mathbf{p_0}, \mathbf{p_1}$ and the center point **c** as follows:

$$r = \frac{|\mathbf{p_0} - \mathbf{c}| + |\mathbf{p_1} - \mathbf{c}|}{2} \tag{4.17}$$

4.2.6 Estimation of Shape Primitive Sizes

As mentioned above the RANSAC algorithm extracts models without size limitations. That means the cone and the cylinder have no height and the plane has no height and no width. These sizes are calculated after RANSAC extracts the primitive shapes.

For the cone the point cloud corresponding to the found cone model is transformed to estimate the height and the offset of a cone. Thereby, is the offset the distance between the apex **a** and the highest point of the truncated cone. To calculate the height and the offset the axis of the cone **dir** is transformed to the z-axis and the apex

a is transformed in the origin (${}^{dir}_{z-axis}T$). After transforming the points, the minimal and maximal values of the z-axis are used to estimate the sizes as follows:

$$height = z_{max} - z_{min} \tag{4.18}$$
$$offset = z_{min} \tag{4.19}$$

For the cylinder the point cloud corresponding to the found cylinder model is transformed in the same way as the cone model. The axis **dir** is transformed on the z axis and the point on the axis $\mathbf{p_a}$ is transformed in the origin (${}^{dir}_{z-axis}T$). After this transformation of the point cloud corresponding to the cylinder, I use the minimum and maximum values to estimate the height of the cylinder with:

$$height = z_{max} - z_{min} \tag{4.20}$$

The sphere size is completely defined by the radius. Therefor no size estimation is needed for the sphere.

To estimate the plane height and the plane width the points corresponding to the plane are transformed that the normal of the plane **dir** is transformed to the z-axis and the point defined by $d \cdot (-dir)$ is transformed in the origin (${}^{dir}_{z-axis}T$). After transformation of the points corresponding to the cylinder model, the height and the width of the cylinder is estimated using the minimum and maximum values of the x- and y-axis as follows:

$$height = y_{min} - y_{max} \tag{4.21}$$
$$width = x_{min} - x_{max} \tag{4.22}$$

The coefficients of the plane are extended by the center point of the plane. The estimation of the center point **c** is done by calculate the center **pos** of the transformed

point cloud as follows:

$$pos_x = x_{max} \pm width \tag{4.23}$$
$$pos_y = y_{max} \pm height \tag{4.24}$$
$$pos_z = 0 \tag{4.25}$$

The sign of *width* and *height* varies depending of the position of the plane. If both maximum values are positive *pos* is estimated by:

$$pos_x = x_{max} - width \tag{4.26}$$
$$pos_y = y_{max} - height \tag{4.27}$$
$$pos_z = 0 \tag{4.28}$$

If both maximum values are negative *pos* is calculated by:

$$pos_x = x_{max} + width \tag{4.29}$$
$$pos_y = y_{max} + height \tag{4.30}$$
$$pos_z = 0 \tag{4.31}$$

If only the maximum of the x-axis is negative *pos* is calculated by:

$$pos_x = x_{max} + width \tag{4.32}$$
$$pos_y = y_{max} - height \tag{4.33}$$
$$pos_z = 0 \tag{4.34}$$

The last case is if only the maximum of the y-axis is negative *pos* is calculated by:

$$pos_x = x_{max} - width \tag{4.35}$$
$$pos_y = y_{max} + height \tag{4.36}$$
$$pos_z = 0 \tag{4.37}$$

To get the center point the vector **pos** is transformed back with the inverse transformation ${}^{dir}_{z-axis}T^{-1}$.

4.3 Shape Primitive Improvement with Particle Filter

To validate the hypothesis coming from the RANSAC algorithm and to circumvent occlusion we use a particle filter. The particle filter in general is divided in four steps. The first step is the motion model, also called prediction step. In the motion model, the particle filter predicts the new particles with the given motion. In a simple example a robot moves 10 cm along the x-axis with no rotation. The result is that all previous seen objects are now predicted -10cm in x direction. Because of the inaccuracy motion the predicted noise is added to the new particles. After motion model is done the sensor model verifies the new particles. The sensor model is also called correction step. In the sensor model the particle filter gets input data from a sensor and proves all particles against the sensor data. All particles get a weight form the sensor model corresponding to the sensor data. The particle filter then normalizes the weights of all particles. With the normalized particles the particle filter resamples new particles from the current particle set. The new particles are sampled randomly with respect to their weights. The new sampled particles used for the next iteration of the particle filter.

In the first iteration the particle filter is initialized with the shape primitives resulting from the shape approximation module. From these shape primitives the particle filter generates a predefined number of particles. The particles are multiple copies of the found shape primitives with added noise. The noise is used to generate particles, whose coefficients vary minimally in comparison to the original shape primitives.

4.3.1 Motion Model

The motion model calculates the motion between the end effector at time $t-1$ and time t. The time $t-1$ is the time of the previous iteration of the particle filter and t the current one. Figure 4.10 shows the coordinate frames of the different parts of the robot.

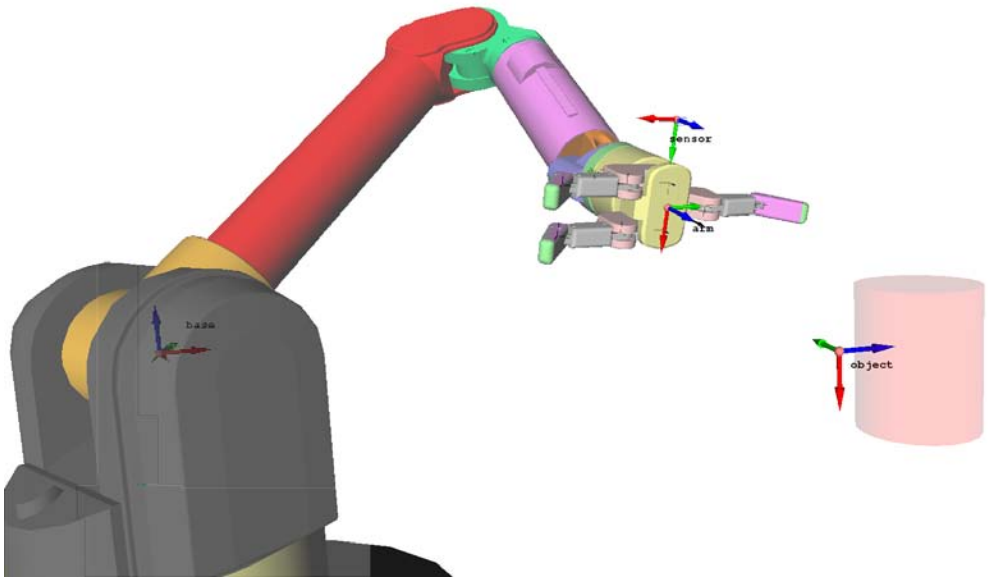

Figure 4.10: Transformation frames of the robot.

The transformation between base frame B and end effector frame E is published by the robot. The transformation between the end effector frame E and sensor frame S is static and the resultant transformation is given by:

$$ {}^{E}_{B}T *^{S}_{E} T =^{S}_{B} T \tag{4.38} $$

These transformation is known from the previous and from the current iteration. Out of these two transformations the sensor motion M is calculated as follows:

$$ \mathbf{M} =^{S}_{B} T^{-1}_{prev} *^{S}_{B} T_{curr} \tag{4.39} $$

The motion model predicts the new particles with this motion. Because the coefficients of the models are point vectors (center point) and direction vectors (axis direction) the motion model calculates the new coefficients by:

$$\mathbf{p}_{curr} = \mathbf{M}^{-1} * \mathbf{p}_{prev} \tag{4.40}$$
$$\mathbf{d}_{curr} = \mathbf{M}_{rot}^{-1} * \mathbf{d}_{prev} \tag{4.41}$$

Where $\mathbf{p}_{curr}$ is the current center point calculated by inverse motion $\mathbf{M}^{-1}$ times the previous center point $\mathbf{p}_{prev}$. For the current direction vector $\mathbf{d}_{curr}$ the rotation part of inverse motion $\mathbf{M}_{rot}^{-1}$ is multiplied by the previous direction vector $\mathbf{d}_{prev}$. Because of the inaccuracy of the transformation the motion models adds on each new calculated coefficient noise.

4.3.2 Sensor Model

The sensor model of the particle filter checks each particle against the current sensor data. Therefore, each particle type has a separate prove function. Particle types are the shape primitive models that can be detected by the shape approximation module. The sensor model proves the particles by the distance between the model defined by its coefficients and each point of the current sensor data. In addition to the distance the normals of the sensor data is considered. In general, only a point which distance to the model is smaller than the threshold ϵ and which normal does not deviate by more than α degree from the models normal.

Sensor Model for Cones

For the cone the sensor model first projects the point $\mathbf{p}$ on the cone axis $\mathbf{dir}$ and calculates the radius $\mathbf{r}$ at the position of the projected point $\mathbf{p}_{proj}$ as follows:

$$k = \frac{\mathbf{p} \cdot \mathbf{dir} - \mathbf{a} \cdot \mathbf{dir}}{\mathbf{dir} \cdot \mathbf{dir}} \tag{4.42}$$
$$\mathbf{p}_{proj} = \mathbf{a} + k * \mathbf{dir} \tag{4.43}$$

Where k is the factor for the distance between the apex $\mathbf{a}$ and the projected point along the axis direction $\mathbf{dir}$. The radius r is calculated with the distance between the apex $\mathbf{a}$ and the projected point $\mathbf{p}_{proj}$ and the opening angle ω of the cone with the following equation:

$$r = \tan(\omega) * |\mathbf{a} - \mathbf{p}_{proj}| \tag{4.44}$$

The sensor model calculates the distance d of the point p to the model defined by the apex a, the axis direct dir and the opening angle ω as follows:

$$d = \frac{|\mathbf{dir} \times (\mathbf{a} - \mathbf{p})|}{|\mathbf{dir}|} - r \tag{4.45}$$

The sensor model calculates the normal of the cone model $\mathbf{n}_{cone}$ as follows:

$$n_{cone} = \sin\omega * (\mathbf{a} - \hat{\mathbf{p}}_{proj}) \cos\omega * (\mathbf{p} - \hat{\mathbf{p}}_{proj}) \tag{4.46}$$

Where ω is the opening angle of the cone, $\mathbf{a}$ is the apex of the cone and $\mathbf{p}_{proj}$ is the projection of the point $\mathbf{p}$ on the axis direction of the cone.

The angular distance d_{angle}between the normal of the cone $\mathbf{n}_{cone}$ and the normal of the point $\mathbf{n}_p$ is calculated by:

$$d_{angle} = \frac{\mathbf{n}_{cone} \cdot \mathbf{n}_p}{\sqrt{|\mathbf{n}_{cone}|^2 \cdot |\mathbf{n}_p|^2}} \tag{4.47}$$

The sensor model counts each point for which the following equation is true. The number of found points is defines the weight of the particle.

$$d < \epsilon \text{ and } d_{angle} < \alpha. \tag{4.48}$$

Sensor Model for Cylinders

For the cylinder the sensor model first projects the point on the axis of the cylinder in the same way as described by the cone in Equation 4.43. After the projected point p_{proj} is determined the sensor model calculates the distance d between the point $\mathbf{p}$ and the cylinder model defined by the center point $\mathbf{c}$, the direction of the axis $\mathbf{dir}$ and the radius r of the cylinder by:

$$d = \frac{|\mathbf{dir} \times (\mathbf{a} - \mathbf{p})|}{|\mathbf{dir}|} - r \tag{4.49}$$

The normal of the cylinder $\mathbf{n}_{cyl}$ is defined by

$$\mathbf{n}_{cyl} = p - p_p roj \tag{4.50}$$

The angular distance d_{angle}between the normal of the cylinder $\mathbf{n}_{cyl}$ and the normal of the point $\mathbf{n}_p$ is calculated as follows:

$$d_{angle} = \frac{\mathbf{n}_{cyl} \cdot \mathbf{n}_p}{\sqrt{|\mathbf{n}_{cyl}|^2 \cdot |\mathbf{n}_p|^2}} \tag{4.51}$$

The weights of the particle is defined by the number of points that approximates well the particle type. All points that fulfill Equation 4.48 approximate the particle well.

Sensor Model for Planes

The sensor model calculates for each point the distance between the point p and the plane that is defined by the normal n_plane and the distance d_{plane} between the origin and the plane along the normal direction n_plane. This is done by:

$$d = n_{plane} \cdot p + d_plane \tag{4.52}$$

The angular distance between the plane's normal $\mathbf{n}_{plane}$ and the normal of the point p $\mathbf{n}_p$ is calculated as described in Equation 4.51. And again all points that satisfy Equation 4.48 are counted and define the weight of the particle.

Sensor Model for Spheres

For the particles of type sphere the sensor model computes first the sphere's normal. That does the sensor model by subtract the center point $\mathbf{c}$ of the point p.

$$\mathbf{n}_{sphere} = \mathbf{p} - \mathbf{c} \tag{4.53}$$

With the normal of the sphere $\mathbf{n}_{sphere}$ and the radius of the sphere r the sensor model calculates the distance between the point and the sphere model by:

$$d = |\mathbf{p} - \mathbf{c}| - r \tag{4.54}$$

The angular distance between the normal of the sphere $\mathbf{n}_{sphere}$ and the normal of the point p $\mathbf{n}_p$ is calculated as described in Equation 4.51. The weight of the particle of the plane type is defined by the number of points that satisfy Equation 4.48. Those points approximate the plane model of the particle well.

4.3.3 Normalize Particle Weights

Before the particle filter resamples new particles, the weights of the particles need to be normalized. Therefore, shape primitive groups are defined in the initialization step of the particle filter. In the initialization multiple copies of a shape primitive, found by the shape approximation module, are generated with small differences in the coefficients. Out of these shape primitive group the highest particle weight is divided by all other weights of the shape primitive group. This is done to avoid that shape primitive groups with a large number of points has a better weight than a smaller shape primitive group in the same object. As an example one can consider the bottle shown in Figure 4.5 on page 23. The cone that approximates the upper part of the bottle has less points than the cylinder that approximates the lower part

of the bottle. If I would not divide the weight by the maximum weight of the shape primitive group the particle filter would sample more and more particles out of the cylinder group because their weight would be higher than the weight of the cone.

If my approach finds more than one object the normalization step of the particle filter normalizes the weights for each object separately. This is done because the objects should not have influences against each other. After the normalization step the sum of all particles containing to one object is 1.

4.3.4 Resampling

After all particle weights are normalized the particle filter resamples new particles with replacement. Therefore the resample step samples N times with replacement from the set of all particles. Where N is the predefined particle number. The resample step samples the new particles accordingly to the weight of the particles. Therefore, all particles with a high weight have a high probability to be sampled where particles with low weights have only a low probability to be sampled.

4.4 Heuristic for Pre-Grasp Estimation

For testing purposes the shape primitive result coming from the particle filter is used with a heuristic to do real world experiments. The heuristic searches the possible pre-grasp, the hand position and the orientation of the hand. For the pre-grasp selection the heuristic choses that a cone, a cylinder and the plane shape primitives are grasped with the cylindrical pre-grasp. The heuristic selects the spherical pre-grasp if the best shape primitive coming from the particle filter is a sphere. The cylindrical pre-grasp in generally is grasped from the viewpoint direction of the camera. The sphere is grasped from the top. To avoid grasping in the table the orientation of the best shape primitives are considered. If the axis of the cone and the axis of the cylinder is oriented nearly perpendicular to the table normal, the objects lies on the table. In that case the heuristic calculates the grasp from the top. In addition to the grasping from the top the size of the shape primitives are considered. Therefore the position of the grasp for the lying objects is higher if the radius of the cone or the cylinder is not big enough. The heuristic calculates this

higher position to avoid grasping in the table. Section 5.4 shows the results of the real world experiments with this heuristic.

5 Experiments and Results

This chapter starts with the presentation of the hardware configuration used in the experiments. The first part of the experiments shows the results of the shape approximation. How the resulting shape primitives can be used in the grasp planner GraspIt!. The results of the real world experiments are shown in the last part of this chapter.

5.1 Hardware Configuration

For the experiments the Barrett Whole Arm Manipulator (WAM) is used. The end-effector of the arm is a Barrett hand. To perceive the scene a 3D depth sensor is mounted on the hand as shown in Figure 5.1. The 3D Sensor used for the experiments are a Asus Xtion Pro Live sensor.

Figure 5.1: Hardware configuration of the experiments settings. A Barrett WAM is mounted on a XR4000 base. The 3D depth sensor is connected with the Barrett hand which is mounted at the arm.

In addition to the robot the experiments setting shown in Figure 5.1 depicts the table used during the experiments. On this table the objects to grasp are placed on.

5.2 Estimation of shape primitives

The first part of the experiments consider the estimation of the shape primitives. Therefore, different objects are placed on the table shown in Figure 5.1 on the previous page. The shape primitive estimation is done for single and multiple objects. Figure 5.2 shows the objects used in the experiments. The objects consist of two categories rounded and elongated objects. There are two bigger rounded objects: the orange soccer ball and the blue cuddly toy, and two smaller rounded objects: the green sponge and the beige massage ball. The objects contain three bigger elongated object: the white box, the wooden pentagon and the white/red bottle. The smaller elongated objects are: the red spectacle case and the green/red creme tube.

Figure 5.2: Objects used in the experiments. There are two large round objects: soccer ball and cuddly toy, two smaller round objects: sponge and massage ball, three bigger oblongness objects: box, pentagon and bottle, and two smaller oblongness objects: spectacle case and creme tube

All object shown in Figure 5.2 are tested separately to estimate the shape primitives. After all objects are tested separately, my approach estimates shape primitives for a group of objects. Figure 5.3 on the following page depicts the found shape primitives of a bottle. A cylinder for the bottom part and a sphere for the upper part was found. The red dots in the center of the shape primitives represent the center point of the particles. The particles are created from the found shape primitives.

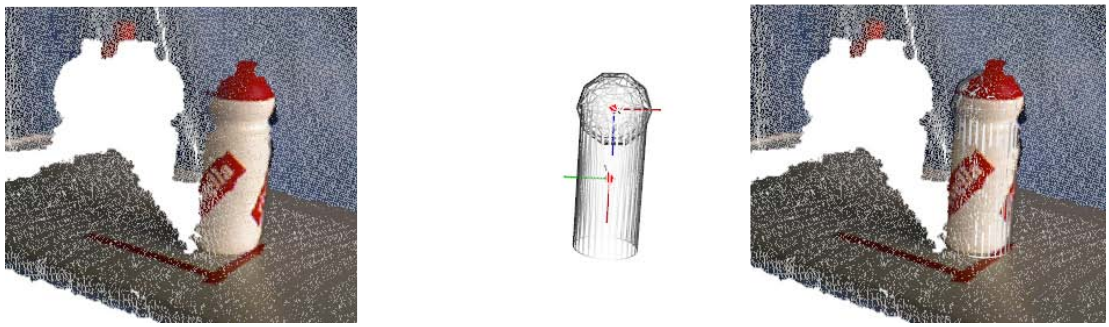

Figure 5.3: Initial estimation of shape primitives from the first view. The left image shows the raw input cloud. The picture in the middle shows the shape primitives found in the raw input cloud. The right image depicts a projection of the found shape primitives in the input cloud. The cylinder and the sphere represent the object: a bottle. The red points display the particles of found shape primitives. In this case the center points represent the particles

The results of the shape primitive estimation is listed in Tabular 5.1 on the next page. This tabular lists in the second column the found shape primitives. If multiple shape primitives are detected, like the two primitives of the bottle in the example above, all found shape primitives are listed. The third column of the tabular depicts the quantity of the points that approximates the shape primitive well. In the case that multiple shape primitives are detected the sum of the quantities are listed. The forth column shows the quantity of the whole object. The ratio of the object quantities and the primitive quantities are displayed in the fifth column. The last column shows the quantity of the points that correspond to the best primitive.

This results shows that most objects can be described with a small number of shape primitives. All but one of the shape primitives have a better ratio than 0.5. That means more than 50% of the object's points are approximate the shape primitive well. The cuddly toy, the only object with a ration less than 50%, has a lot of small shape primitives which approximation is well for only a small number of points. My approach considers only shape primitives which at least approximated by 10% of object points. The visualization of the other objects can be found in the Appendix A.

Table 5.1: Results of the shape primitive extraction.

Object	found primitives	point quantity of primitives	points quantity of object	object primitive ration	point quantity of best primitive
ball	sphere	614	639	0.961	614
bottle	cylinder, sphere	151	252	0.599	123
box	plane	217	316	0.687	217
cuddly toy	sphere, cone	174	488	0.357	144
massage ball	sphere	14	25	0.56	14
pentagon	plane	134	183	0.732	134
spectacle case	cylinder, cone	36	60	0.6	25
sponge	sphere	42	68	0.618	42
creme tube	plane	46	71	0.648	46

5.3 Shape Primitives for GraspIt!

In the related work in Section 2.2, I mentioned the work of Miller et al. [12]. They showed in their work how their grasp planner GraspIt! estimates grasps of shape primitives. They generated the shape primitives manually and presented their work with these manually generated shape primitives. My approach estimates the shape primitives out of the point cloud that the robot perceives with a 3D depth camera. Therefore no manually generation of the shape primitives is used. In the experiment I show how my approach generates shape primitives of the bottle. The object is estimated by two shape primitives. A sphere is estimated on top of a cylinder. Figure 5.4 on the following page shows the shape estimation.

These two shape primitives are used as input for the GraspIt! grasp planner. GraspIt! generates grasps based on the shape primitives. Figure 5.5 on the next page shows the result for the bottle which is presented by a cylinder and a sphere. The right image shows the GraspIt! results. It depicts dots which represent the position of the hand. The lines depict in the image are the approach vectors for that

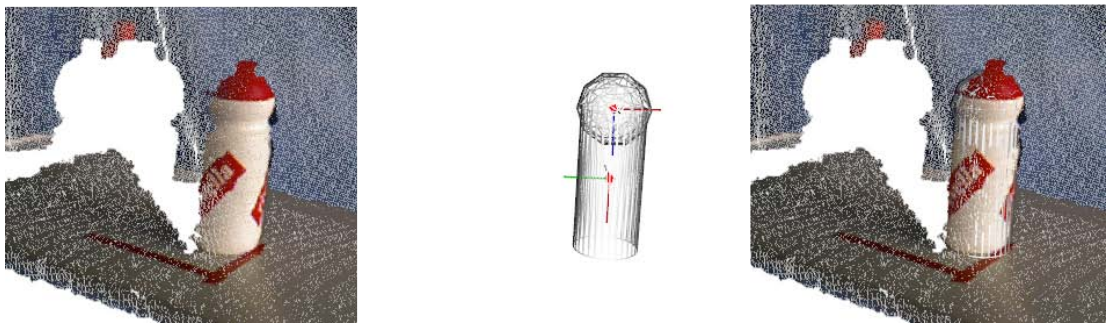

Figure 5.4: Estimation of shape primitives of a bottle object. The resulting shape primitives used as input for the grasp planner. The left image shows the raw input cloud. The middle image shows the estimated shape primitives including the pose. The coordinate systems in the object depicts the pose of the object. The right image shows the estimated shape primitives projected in the input cloud.

grasp. The diameter of the points symbolize the grasp quality. The bigger the dots the better is the quality of that grasp.

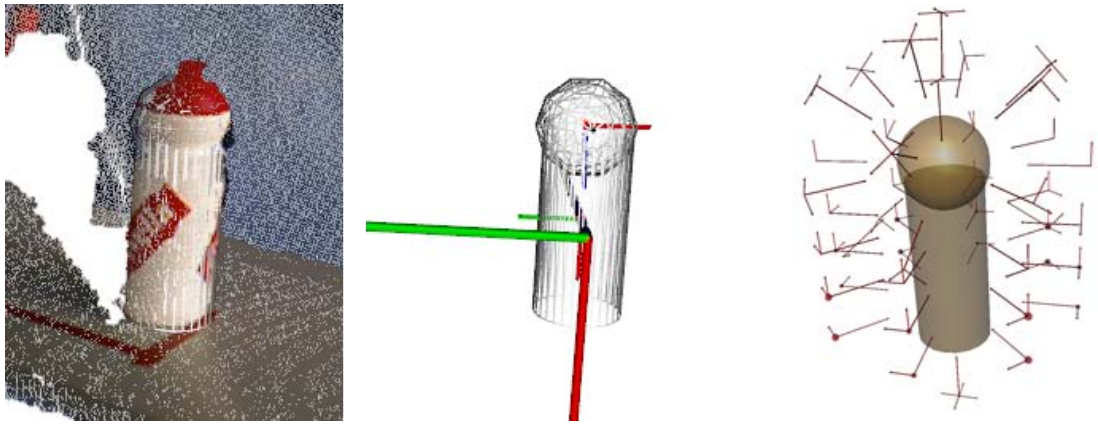

Figure 5.5: GraspIt! result of bottle represented by a cylinder and a sphere shape primitive. The dots declare the hand position and the lines indicate the approach vector of the hand. The size of the dots represent the probability of the grasp.

Out of these grasp hypothesis GraspIt! performs the grasp with the highest quality. Figure 5.6 on the following page shows the two best found grasps for the bottle. These grasp have equal quality. The Barrett hand grasps that object at the cylinder. The red lines in the cylinder symbolizes the friction cone that GraspIt! uses to calculate the quality of the grasp

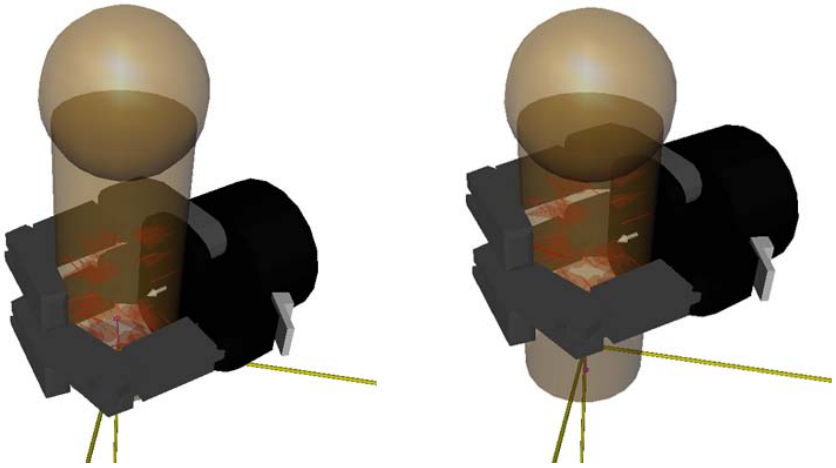

Figure 5.6: GraspIt! grasp planner results of the two best quality grasps. The red cones depict the friction cone that GraspIt! uses to calculate the grasp quality.

5.4 Grasping based on estimated shape primitives

In the real world experiments we use the setup shown in Figure 5.1 on page 39. The objects are placed on the table and my approach perceived that scene with the hand mounted 3D depth sensor. My approach extracts the shape primitives and uses the particle filter to validate the primitives. After some time the best shape primitive is taken and the heuristic generates the pre-grasp. The heuristic provides the the pre-grasp type, the hand position and orientation.

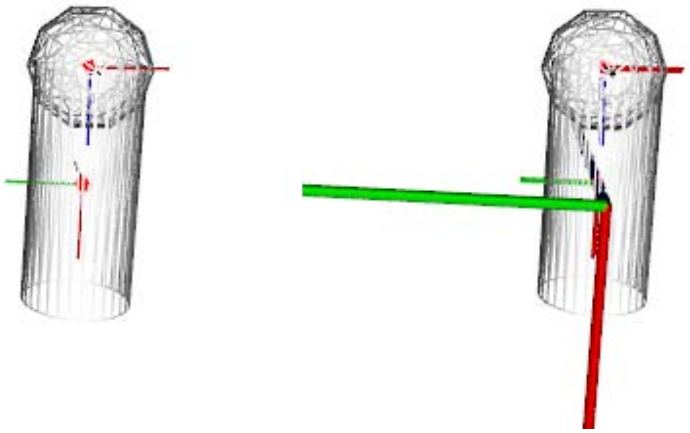

Figure 5.7: Resulting grasp pose of the bottle represented by the larger coordinate system. The origin of the coordinate system depicts the position of the hand and the orientation of the coordinate system presents the orientation of the hand.

Figure 5.7 on the preceding page shows the grasping point of the bottle. The grasping point is the origin of the large coordinate system. The approach vector for the pre-grasp is pointing through the cylinder. The approach vector is the blue z-axis of the large coordinate system. The other two axes of this coordinate system defines the wrist orientation. Figure 5.8 shows the end-effector frame of the robot. During the grasping, the robot moves its hand to the object until the end-effector frame and the coordinate system that defines the pre-grasp pose are superimposed upon each other.

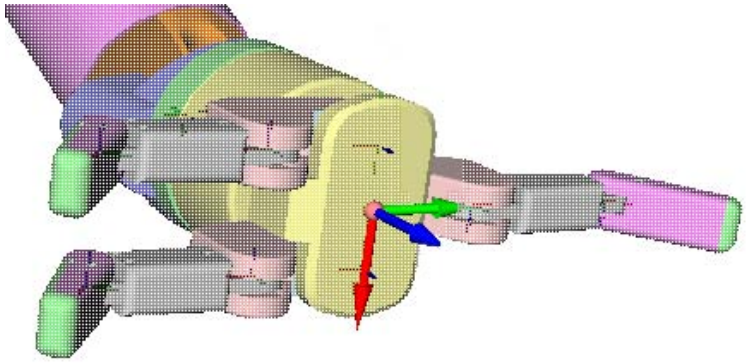

Figure 5.8: The coordinate system located at the hand inside is the reference frame of the Barrett hand in this experiments. During the grasping procedure the hand is moved until this coordinate system and the coordinate system that defines the pre-grasp pose are superimposed upon each other.

In this part of the experiments all objects shown in Figure 5.2 on page 40 are tested. Figure 5.9 on the next page shows the steps my approach runs through. It starts after the initialization with RANSAC to move and validate the shape primitives. The top left image shows the robot in this motion phase. In the top right image the robot found a pre-grasp and moves to the final position. The final position for that bottle is shown in the bottom left object. After closing the hand the robot raises its arm and lift up the bottle.

This experiment is done for all objects. Tabular 5.2 on page 47 shows the results for each object. The results show that most objects are grasped with this approach. In the three cases where the grasp where unsuccessful the approach found good shape primitive. Additionally the heuristic found a good pre-grasp. The problem with the ball was that the heuristic calculates the pre-grasp position with a small distance between the object and the hand. Because of the large diameter the ball has, it is only graspable if the hand pushes on the ball and then closes the hand.

The massage ball is only a small ball which rolls easily. My approach found a sphere

Figure 5.9: A real world experiment. The top left image shows the robot during the validation phase. In the top right image the robot found the pre-grasp and moves to the final position shown in the bottom left image. After close the hand the robot raise its arm.

as the best shape primitive and calculates the pre-grasp. The massage ball should be grasped from top and due to the small diameter with a cylindrical pre-grasp. Therefore the pre-grasp was not wrong but the movement from the observation position to the final position was not accurate enough. One side of the hand touched the massage ball earlier as the opposite site, and thus the ball was rolling and the massage ball was not grasped.

The spectacle case was the last object that was not grasped successfully. The spectacle case was lying on the table and my approach found a lying cylinder in it. Because of the lying position the object should be grasped from top. The small diameter causes that the space between the object and the hand needs to be bigger to not grasp in the table. Again, in the final position the hand was not posed good enough. It was posed to low and therefore the hand grasped in the table and could not grasp the object. The problem was the detection of the cylinder. Because of the ellipsoid shape of the spectacle case the cylinder was found to deep and reaches in the table. Therefore the pre-grasp was to close to the table

The other grasps were successfully grasped by the robot using my approach.

Table 5.2: Results of the shape primitive extraction.

Object	found primitives	grasp trails	successful grasps	unsuccessful grasps	used pre-grasp
ball	sphere	3	0	3	spherical
bottle	cylinder, sphere	3	3	0	cylindrical
box	plane	3	3	0	cylindrical
cuddly toy	sphere, cone	3	3	0	spherical
massage ball	sphere	3	1	2	cylindrical
pentagon	plane	3	3	0	cylindrical
spectacle case	cylinder, cone	3	0	3	cylindrical
sponge	sphere	3	3	0	cylindrical
creme tube	plane	3	3	0	cylindrical

6 Conclusion

This chapter conclude this work with a summary and makes suggestions for future work of this approach.

6.1 Summary

In this work, I describe why robot grasping of unknown objects is a problem and how other works try to solve this problem. The high dimensional grasp problem is reduced in this work to a lower subspace problem. I do not search the complete configuration of the hand to find a good grasp. I search for a good pre-grasp. If a good pre-grasps is found and the hand is posed regarding the object the hand closes. During the closing procedure of the robot hand, the compliance of the hand is used. The compliance of the hand causes that the fingers of the hand moves in the automatically in final position during the closing procedure. Therefore, the fingers of the robot hand fit the shape of the object well. This fact leads to a stable grasp. The general idea of this approach comes from the neuroscience research. In the research field of neuroscience the realize that humans do not control each joint separately. Humans control the fingers in a low dimensional subspace and use the compliance.

The result of this work is the extraction of the shape primitives in order to find the best pre-grasps for that object. In the input point cloud my approach searches shape primitives. The shape primitives are initialization with a RANSAC algorithm. The RANSAC algorithm extracts shape primitives like cones, cylinders, planes and spheres. A particle filter validates and improves these shape primitives over the time. The particle filter also tracks the shape primitives during the movement and can circumvent occlusion by using multiple views on that object. My approach generates the best found pre-grasp regarding the weight of the particles. Therefor, the particle with the highest weight is use to create the correct pre-grasps. The pre-grasp is determined with a heuristic regarding the orientation, size and type of the best shape primitive.

The experiments which where done for the shape primitive approximation separately

show how my approach detects the shape primitives. In the real world experiments my approach extracts the shape primitives and generates from it the best detected pre-grasps for the objects. These pre-grasps, determined by the heuristic is used with the estimated hand pose to grasp the objects located on the table. The experiments shows that the shape primitives can be extracted of various objects and that good grasping result can be achieved with a simple heuristic if the pre-grasp and hand pose is detected by this approach. Additionally this approach is not limited to a specific robot hand. It can be used with various hands by defining the possible pre-grasp and changing the heuristic.

6.2 Future Work

In future work on this approach one can expand the shape primitives. A shape primitive that could improve this approach could be a box or a combination of planes. A combination on planes could be a two parallel planed or perpendicular planes.

Another aspect that can improved in future work is using more pre-grasp. The Barrett hand, as an example, has two additional pre-grasps that are not considered in this work. The two additional pre-grasps are the precise grasp and the hook grasp. Consider this two additional pre-grasps one could chose new shape primitives regarding these pre-grasps.

To achieve a better result in the real world experiments one could improve the heuristic. One could also use tactile feedback of the robot hand to improve the heuristic. Also the problem that shape primitives can reach inside the table could be considered.

Images and Internet

[1] http://www.stuttgarter-zeitung.de/inhalt.stuttgart-roboter-im-pflegeheim.d0c9a7dd-9438-49b7-801a-bdfd13ece1ce.html.

[2] http://glia.ca/meanderings-wordpress/wp-content/uploads/robots-justin.jpg.

[3] http://personalrobotics.stanford.edu/.

[4] http://www.emeraldinsight.com/content_images/fig/0490360408001.png.

[5] http://www.dlr.de/rm/Portaldata/52/Resources/images/bildgalerie/Hand-II-01.jpg.

[6] http://www.probot.fi/images/barrett_hand.jpg.

[7] http://www.flickr.com/photos/willowgarage/5363358698/sizes/s/in/photostream/.

[8] http://www.hizook.com/files/users/3/SDM_Robot_Hand_2.jpg.

[9] http://mekabot.com/wp-content/uploads/h2_hand_reach.jpg.

Bibliography

[10] A. Miller and P. Allen. *Graspit! a versatile simulator for robotic grasping.* Robotics & Automation Magazine, IEEE, volume 11 (4):110,Äì122, 2004.

[11]

[12] A. Miller, S. Knoop, H. Christensen and P. Allen. *Automatic grasp planning using shape primitives.* volume 2, 1824–1829 vol.2. 2003.

[13] K. Huebner and D. Kragic. *Selection of robot pre-grasps using box-based shape approximation.* In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 1765—1770. 2008.

[14] 2008. URL `http://dx.doi.org/10.1109/ROBOT.2008.4543434`.

[15] C. Goldfeder, P. K. Allen, C. Lackner and R. Pelossof. *Grasp Planning via Decomposition Trees.* In *2007 IEEE International Conference on Robotics and Automation*, 4679–4684. IEEE, 2007.

[16] M. Przybylski, T. Asfour and R. Dillmann. *Unions of balls for shape approximation in robot grasping.* In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1592–1599. IEEE, 2010.

[17] D. R and K. J. *Openrave: A planning architec- ture for autonomous robotics*, 2008. Robotics Institute, Pittsburgh, PA.

[18] G. M. Bone, A. Lambert and M. Edwards. *Automated modeling and robotic grasping of unknown three-dimensional objects.* In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*, 292–298. IEEE, 2008.

[19] C. Dune, E. Marchand, C. Collowet and C. Leroux. *Active rough shape estimation of unknown objects.* In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, 3622–3627. IEEE, 2008.

[20] W. Niem and R. Buschmann. *Automatic Modelling of 3D Natural Objects from Multiple Views.* Yakup Paker and Sylvia Wilbur: Image Processing for Broadcast and Video Production. Workshops in computing series, Springer, Hamburg 1994, ISBN 3-540-19947-0, volume 0 (0), 1994. URL `ftp://ftp.tnt.uni-hannover.de/pub/papers/1994/MONALISA94-WN.ps.gz`.

[21] J. Butterfass, M. Grebenstein, H. Liu and G. Hirzinger. *DLR-Hand II: next generation of a dextrous robot hand.* In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, 109–114 vol.1. 2001.

[22] M. Santello, M. Flanders and J. F. Soechting. *Postural Hand Synergies for Tool Use.* The Journal of Neuroscience, volume 18 (23):10105–10115, 1998. URL `http://www.jneurosci.org/content/18/23/10105.abstract`.

[23] M. R. Cutkosky. *On grasp choice, grasp models, and the design of hands for manufacturing tasks.* Robotics and Automation, IEEE Transactions on, volume 5 (3):269–279, 1989. URL `http://dx.doi.org/10.1109/70.34763`.

[24] T. Feix, R. Pawlik, H. Schmiedmayer, J. Romero and D. Kragic. *A Comprehensive Grasp Taxonomy.* In *Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation.* 2009. URL `http://grasp.xief.net`.

[25] R. B. Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* Ph.D. thesis, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[26] R. Schnabel, R. Wahl and R. Klein. *Efficient RANSAC for Point-Cloud Shape Detection.* Computer Graphics Forum, volume 26 (2):214–226, 2007.

A Shape primitive results

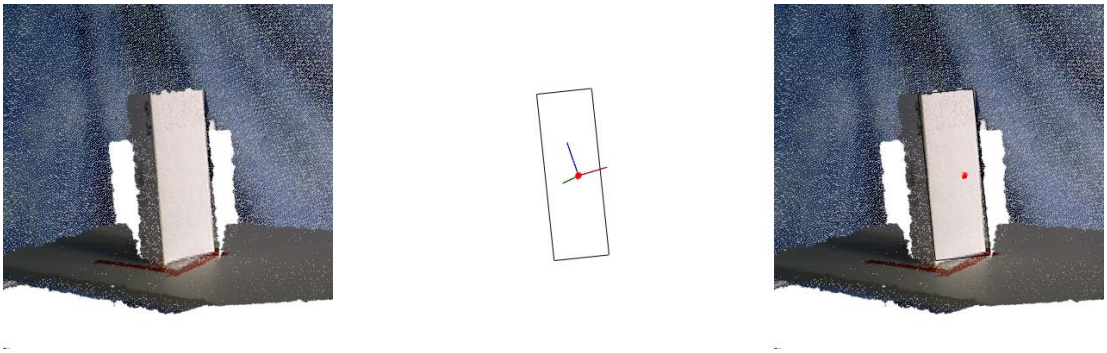

Figure A.1: Found shape primitives for the box

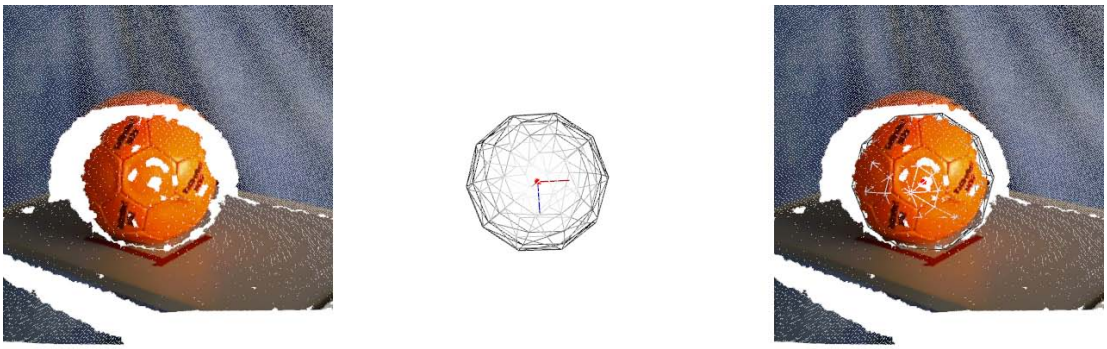

Figure A.2: Found shape primitives for the ball

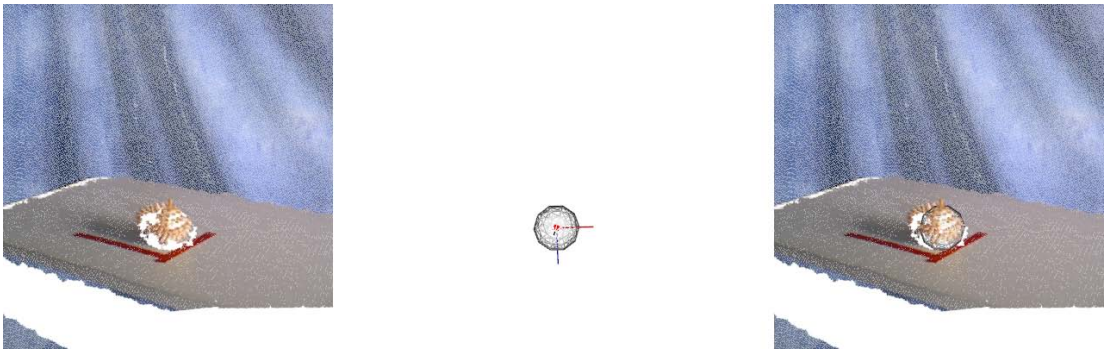

Figure A.3: Found shape primitives for the massage

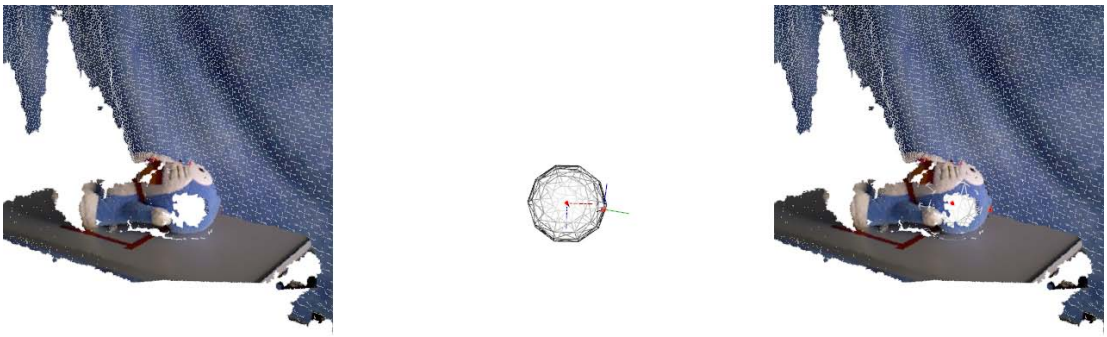

Figure A.4: Found shape primitives for the toy

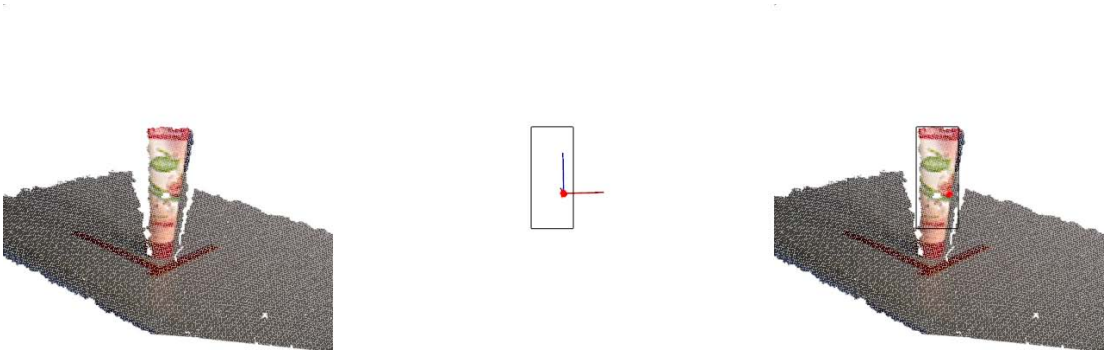

Figure A.5: Found shape primitives for the tube

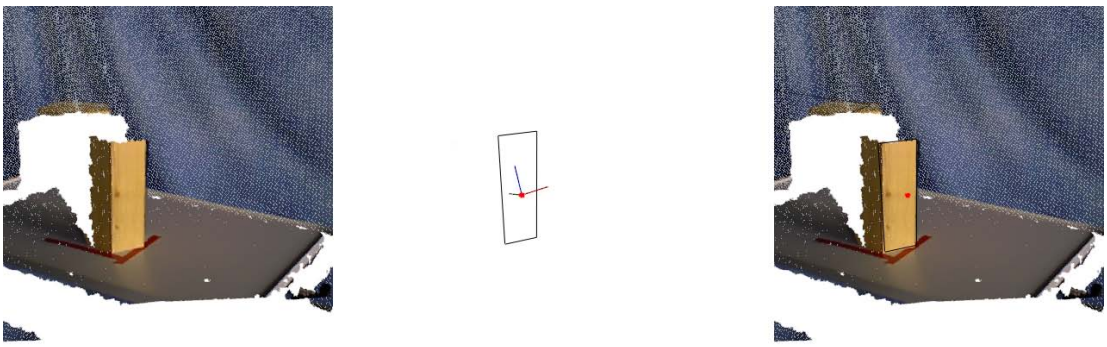

Figure A.6: Found shape primitives for the pentagon

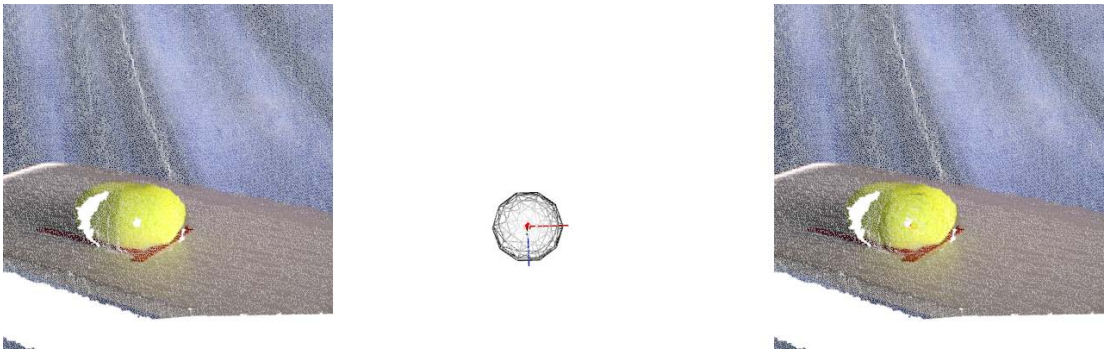

Figure A.7: Found shape primitives for the sponge