Grasp Pose Detection in Open Worlds

by

Andreas ten Pas

A dissertation submitted in partial satisfaction of the requirements

for the degree of Doctor of Philosophy in Computer Science

in the Khoury College of Computer Sciences of Northeastern University

in Boston, Massachusetts, United States of America

Committee in charge:

Associate Professor Robert Platt, Chair
Assistant Professor Christopher Amato
Professor Hanumant Singh
Professor Odest Chadwicke Jenkins

Spring 2021

Northeastern University
Khoury College of
Computer Sciences

PhD Thesis Approval

Thesis Title: Grasp Pose Detection in Open Worlds

Author: Andreas ten Pas

PhD Thesis Approval to complete all degree requirements for the PhD in Computer Science.

Thesis Advisor	8/4/2021 *Date*
Thesis Reader	8/2/2021 *Date*
Thesis Reader	6/3/2021 *Date*
Thesis Reader	8/3/2021 *Date*
Thesis Reader	*Date*

KHOURY COLLEGE APPROVAL:

Associate Dean for Graduate Programs	9/22/2021 *Date*

COPY RECEIVED BY GRADUATE STUDENT SERVICES:

Recipient's Signature	1 October 2021 *Date*

Distribution: *Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI Website.*

Abstract

Grasping is an essential prerequisite for autonomous robotic manipulation. The traditional approach to robotic grasping is to first estimate the pose of the object to be grasped and then to calculate a grasp configuration. However, there are two deficiencies that make it challenging to apply this approach to real world settings: (i) a model is required for each object, (ii) object pose estimation given noisy and incomplete sensor data can be extremely difficult. More recent approaches localize grasp configurations directly without prior object pose estimation. In this dissertation, we present grasp detection methods that fall within this direction. Our methods localize enveloping and two-finger grasp affordances in noisy and unstructured point clouds obtained from one or multiple depth sensors. We introduce an efficient method for sampling grasp candidates with six degrees of freedom by constraining the robot hand to be aligned with the local minor curvature axis of the object surface to be grasped. We represent grasps by images which describe the geometry of the object surface in the vicinity of the grasp. Having defined these representations, we present how to collect ground truth data in order to train machine learning models to classify grasp candidates as actual grasps. In simulation and robot experiments, we demonstrate that our methods enable highly reliable robotic grasping of novel objects with a wide variety of shapes, sizes, and materials in both isolated tabletop scenarios and dense clutter. We deploy our methods on two custom robotic systems for assisting people with motor disabilities in activities of daily living. These systems can autonomously pick up an object indicated by the user with a laser pointer.

Contents

List of Figures

List of Tables

Acknowledgments

During my graduate research work at Northeastern University, I have been fortunate to have Robert Platt as an advisor. His countless hours of feedback and discussions were highly influential on this work. Christopher Amato, Hanumant Singh, and Chadwicke Jenkins provided insightful and helpful feedback on the dissertation.

My colleagues in the Helping Hands group deserve my thanks. First and foremost, I very much enjoyed working with Marcus Gualtieri and Ulrich Viereck on topics which are slighly out of the scope of this dissertation, but still very related. I had many fruitful discussions and received highly valuable feedback from Marcus. I would also like to thank Dian Wang, Yuchen Xiao, and Colin Keil for our joined works. Colin Kohler provided valuable advice on deep learning. Within the robotics and artificial intelligence community of Northeastern, Andrea Baisero and Sammie Katt provided helpful software engineering insights and comments on papers.

A special mention outside of Northeastern University goes to Abraham Shultz who made significant contributions to our first assistive robotic system and who, each time, made the effort to take Marcus and me from the train station to the University of Massachusetts campus in Lowell and back.

Over my journey, I had two memorable internship experiences, with Renaud Detry at NASA's Jet Propulsion Laboratory, and Naveen Kuppuswamy, Alex Alspach, Kunimatsu Hashimoto, and Russ Tedrake at Toyota Research Institute, among many others. I also thank Luca Colasanto for his work with me when consulting at Realtime Robotics.

I had many interesting interactions with other researchers in the field, including Li Yang Ku, Jeffrey Mahler, Kevin Eckenhoff, Juxi Leitner, Benjamin Burchfiel, Dimitrios Kanoulas, Barrett Ames, Jiaji Zhou, John Oberlin, and Lucas Manuelli, and Yifan Hou.

Lastly, I want to thank my friends and family for their support and encouragement.

Part I

Introduction and Background

Chapter 1

Introduction

1.1 Robotic Grasping

Whereas robots have been deployed successfully to repetitive tasks, such as those encountered in factory automation, for many years, other areas such as personal health care, space exploration, hazardous environments, or kitchen assistancy, require a much larger degree of autonomy. An important step toward more autonomous robots is the capability to manipulate objects and tools in ways which enable successful execution of the desired tasks. Autonomous robotic grasping is an essential prerequisite toward this capability.

There are many issues that an autonomous robotic grasping system has to deal with. Objects to be grasped may vary in shape, size, texture, mass, deformability. Friction and contact forces further complicate the problem. Perception is limited by sensor noise, occlusions, and partial observability. Control is limited by actuation noise and inaccurate robot kinematics. Robot hands may be restricted to a small number of degrees of freedom, difficult to control, or easy to break.

While there has been tremendous progress on visual sensors which allow to perceive both color and depth information, the inference of grasps from such data is a challenging problem. Even for a simple two-finger hand, such as a parallel jaw gripper, the space of grasp configurations is highly dimensional and continuous. Another challenge is that real world environments typically lack structure, e.g., picking up items out of grocery bag, which makes the grasp perception problem even harder. This dissertation aims to address this problem by predicting grasp poses in the complete six-dimensional space of positions and orientations from visual sensor data obtained for unknown objects presented in unstructured environments.

1.2 Thesis Contributions

The general contribution of this work is to enable robotic grasping of novel and partially observable objects in unstructured environments. We present a number of methods that

predict grasp poses with six degrees of freedom (DOFs) from visual sensor data that provides only partial information of the robot's environment. Our specific contributions can be listed as follows.

1. A model-free approach to detect grasp configurations where the fingers enclose around an object part. The approach is completely geometric and does not require machine learning. A key idea here is to orient the robot hand such that its major axis is parallel to the minor curvature axis of the object surface that is enclosed by the fingers of the robot hand when the grasp is executed. Another key idea is to cluster grasps which are geometrically aligned in order to reduce the number of false predictions.

2. A model-free approach to detect 6-DOF grasp poses in point clouds. This method is a two-stage detector, where the first stage generates a set of grasp candidates and the second stage classifies each candidate as an actual grasp or not.

3. A representation of grasps by a visual feature descriptor of a projection of the points contained within the closing region of the robot hand onto the plane parallel to the hand's approach axis. This representation enables the use of a support vector machine in order to classify the grasps.

4. A representation of grasps as multi-channel images. This representation incorporates surface normals and multiple views. It is used as input to train convolutional neural networks in order to classify the grasps.

5. A grasp representation that enables the use of convolutional neural networks to evaluate grasp candidates in a fast and accurate manner. This representation enables us to replace geometric candidate generators that are slower and whose output restricts the action space of the robot after grasping.

6. Two mobile robotic systems that assist with picking up objects and are aimed at people with motor disabilities. A laser pointer on the system is used to select an object of interest and one of our grasp perception methods is used to detect grasps on that object. Both systems provide autonomous grasping for open world environments which are more challenging than fixed tabletop scenarios, e.g., the inside of a shelf. The second system also provides basic pick-and-place capabilities.

7. Experimental evaluations of all grasp perception methods on physical robots which demonstrate that our methods can be used to grasp a large variety of novel objects presented in isolation or in clutter with low failure rates.

8. Open source implementations for most of the algorithms presented in this dissertation. These implementations are available as ROS packages or C++ libraries.

1.3 Outline

This dissertation is organized in four parts. Part I gives the background on the robotic grasp pose detection problem.

- Chapter 2 gives a mathematical definition of the grasp detection problem. Subsequently, we describe approaches to the problem, beginning with the traditional approaches and ending with contemporary ones. The background given by this chapter forms the basis for the algorithms presented in the following chapters (3-6).

Part II presents a number of methods for grasp pose detection that enable two-finger grasps given a 3D point cloud of the objects to be grasped. On a high level, all chapters in this part follow a similar structure. First, we describe how to represent grasps. Second, we describe how to infer grasps using the proposed reprsentation. Third, we evaluate how well the approach can detect grasps in simulation. Fourth, we characterize how well the approach can grasp actual objects on a physical robot in scenarios where the objects are presented in isolation or in clutter.

- Chapter 3 deals with the problem of finding grasps for objects with handles. We address this problem by searching a point cloud for enveloping grasp affordances where the fingers of the robot hand encircle some object part. The research for this chapter was previously published [84].

- Chapter 4 presents a two-stage method for the detection of 6-DOF grasp poses. We introduce an algorithm for sampling the space of 6-DOF grasp poses and a method to train a binary classification model to make predictions about whether the sampled poses are actual grasps or not. The research for this chapter was previously published [85].

- Chapter 5 extends the method presented in Chapter 4 to convolutional neural networks (CNNs) for grasp classification. Grasps are represented as images based on the visible points and their surface normals, and imagined points that lie behind the object surface. The research for this chapter was previously published [38, 86].

- Chapter 6 extends the method presented in Chapter 5 to CNNs for grasp candidate classification. While the models learned in Chapters 4 and 5 learns a one-to-one function from images to grasp poses, the model presented here learns a one-to-many function from images to grasp poses. The research for this section has not yet been published.

Part III describes two robotic systems aimed at the assistance of people with motor disabilities in activities of daily living.

- Chapter 7 presents a robotic system for picking up objects that consists of a mobility scooter, a Baxter robot arm, and a laser pointer. The laser pointer is used to indicate the object to be grasped. Our system then finds a grasp using the algorithm presented

in Chapter 5, and picks up the object. The research for this chapter has been previously published [37].

- Chapter 8 presents a second robotic system that improves up-on the performance of the system presented in Chapter 7 and enables simple pick-and-place tasks. The Baxter arm is replaced with a UR5 robot arm, fixed cameras are used instead of a visual SLAM method to obtain a point cloud, among other changes which are made to improve the grasp success rate. The research for this chapter has been previously published [112].

Part IV concludes the dissertation, provides a discussion of limitations of the proposed methods, and outlines opportunities for further research on robotic grasp perception in relation to the work done for this dissertation.

Chapter 2

Grasp Pose Detection

In this chapter, we define the problem addressed in this dissertation: grasp detection. Subsequently, we describe the challenges that make grasp detection a difficult and interesting problem to address. Finally, we elaborate on approaches that have been introduced in the literature to solve grasp detection.

2.1 Problem Statement

We define a robot, $\mathcal{R}$, as a single robot hand that can move arbitrarily in $SE(3)$, where $SE(3) = SO(3) \times \mathbb{R}^3$ is the special Euclidean group in three dimensions, $SO(3)$ is the special orthogonal group in three dimensions, i.e., the space of 3-dimensional rotations, and $\mathbb{R}^3$ is the three-dimensional Euclidean space, i.e., the space of translations in three dimensions. The **robot state** is defined by the pose of the hand, $h \in SE(3)$, and the configuration of the fingers. For the most part, we assume the hand is a parallel jaw gripper whose jaws are actuated by a single degree of freedom. The **world state** $\mathcal{W} \in \mathbb{W}$ is a description of the state of the environment and objects around the robot, where $\mathbb{W}$ is the set of all possible world states. A **point cloud**, $\mathcal{C} \in \mathbb{C}$, is a finite set of points in the robot's environment which are obtained by one or more depth sensors, where $\mathbb{C} \subset 2^{\mathbb{R}^3}$ denotes the space of possible point clouds that can be generated by the sensor arrangement of the robot. The world state is a latent variable that is observed only via the point cloud. In particular, we model the depth sensor(s) as a function $\Lambda : \mathbb{W} \to \mathbb{C}$ that encodes aspects of world state as a point cloud.

We define a grasp as follows:

Definition 2.1.1 (Grasp) *The predicate* $grasp(\mathcal{W}, h) \in \{0, 1\}$ *denotes whether a grasp exists at pose* $h \in SE(3)$ *in world* $\mathcal{W} \in \mathbb{W}$*. If such a grasp exists, then an object in world* $\mathcal{W}$ *can be grasped by moving the hand to* h *and closing the fingers.*

This function classifies hand poses as grasps or not. There are many ways to determine if a grasp will be successful. For example, a human could annotate the grasp or the grasp could be virtually executed in a simulator such as OpenRAVE [26] or Gazebo [59]. Another

possibility, used in our prior work [85, 38, 86], is to check for mechanical stability, e.g., using force closure conditions, and to check if the grasp is not in collision. Having the robot lift the object without dropping it is another way to determine grasp success.

The *grasp detection* problem is to detect a hand pose from which a grasp can be formed by closing the fingers:

Problem 2.1.1 (Grasp detection) *Given a robot $\mathcal{R}$ and a point cloud $\mathcal{C} = \Lambda(\mathcal{W})$ for some hidden world state $\mathcal{W}$, the problem of* ***grasp detection*** *is to find a hand pose, $h \in SE(3)$, such that $grasp(\mathcal{W}, h) = 1$.*

To solve this problem, we follow a two-stage approach. In the first stage, we generate a set of grasp candidates $H \subset SE(3)$ given a point cloud $\mathcal{C}$ where each grasp candidate is a hand pose $h \in H$. This step can be expressed as a function $f : \mathbb{C} \to \mathbb{H}$, where $\mathbb{H} = 2^{SE(3)}$ is the space of possible grasp candidate sets. The idea of candidate generation is to reduce the search space from $\mathbb{H}$ to a more manageable discrete set H that can be passed onto the second step for classification. In the second step, we predict for each $h \in H$ whether $grasp(\mathcal{W}, h) = 1$ given a point cloud $\mathcal{C} = \Lambda(\mathcal{W})$. Specifically, we learn a function $g : \mathbb{C} \times SE(3) \to \{0, 1\}$ where $g(\mathcal{C}, h)$ predicts whether $grasp(\mathcal{W}, h) = 1$.

2.2 Challenges

Grasp detection is a challenging problem for a number of reasons. A primary problem is that the objects to be grasped are typically not known in advance. In addition, objects can have many properties that make it difficult to perceive or grasp them, such as shape, texture, weight, deformability, and material properties which can make it hard for a sensor to perceive them such as reflectivity, transparency, and so on. Sensors are often noisy, work only indoors or outdoors, and may be sensitive to light and extreme temperatures. The quality of the obtained data, in particular the depth information, can differ greatly between different sensors. Precise calibration is necessary to let the robot make use of the sensor data. Another issue is partial observability: the data obtained from a sensor does not represent the whole scene. While it can help to add more sensors, objects may still be occluded by their surroundings. Robotic hardware provides another layer of potential issues. The kinematics of robot arms and hand can be inaccurate or suffer from non-perfect calibration. Furthermore, the grasp configurations produced by a grasp detection method may be kinematically infeasible or unreachable by the robot. A further challenge are computational resources, particularly on mobile platforms.

Many traditional approaches make a **closed world** assumption, where the objects to be grasped are known in advance. Some approaches also store grasp poses in a database, making this assumption even more restrictive. For example, Kehoe et al. [55] use a database that stores a CAD model and a set of grasp candidates for a set of objects. Collet et al. build models using local descriptors from several images [19] and match them to new images using RANSAC [30].

In this dissertation, we consider **open world** problems, where the objects to be grasped are not known a priori and where we can only obtain partial observations of the robot's environment. While closed world problems are well structured, open world problems are unstructured. A common, open world problem that we consider in this work is clutter, i.e., multiple objects which are placed very close to each other. This is a particularly challeging perception problem for two main reasons. The first reason is that, because certain objects are behind or below other objects, the data that can be obtained from a sensor contains occlusions. The second reason is that the objects are so close to each other which makes it difficult to distinguish one object from another adjacent object. This is a signifcant challenge for many computer vision techniques applied to clutter, such as object detection and segmentation. Similarly, it makes it difficult to find collision-free grasp configurations. Another issue with clutter is that motion planning is hard because the objects surrounding the target object might become obstacles when the robot attempts to move to the intended grasp pose. The literature makes a distinction between structured clutter [2] and dense clutter [85], with the main difference being that the objects in structured clutter are not as close to each other as in dense clutter. The free space between the objects makes structured clutter simpler than dense clutter because object boundaries are easier to perceive. In this dissertation, we are primarily concerned with dense clutter scenarios.

2.3 Approaches

Approaches to the grasp detection problem can be divided into three different categories: (i) analytical, (ii) model-based, and (iii) model-free approaches. While analytical approaches deal mostly with finding optimal grasps under certain given criteria and typically assume that the geometry of the object is known, model-based and model-free approaches take data from actual sensors and infer grasp configurations from this data. The sensor data given to such approaches is typically visually, but can also come from other sources such as tactile. Nevertheless, in this dissertation, we mostly consider visual data.

2.3.1 Analytical Grasp Planning

Analytical approaches to plan grasps can be divided into two classes of approaches based on (i) contact geometry and (ii) contact wrenches (forces and torques). Approaches in the former class evaluate the quality of a grasp by checking if the geometry of the contact region matches a set of prespecified criteria. Davidson and Blake proposed a geometric caging method to find immobilizing grasps [21]. Vahedi and van der Stappen presented a similar approach [107]. Calli finds graspable regions on the contour of an object based on the contour curvature [14]. The object contour is also the basis for Blake's work who looked at its rotational and reflectional symmetry [8].

Approaches based on contact wrenches analyze the wrenches which act on the object at contact. A review of this area is given by Bicchi and Kumar [7]. Typically, such approaches

look for grasps which are in force closure [80]. A simple and widely used method for the construction of force closure grasps was introduced by Nguyen [81]. A popular grasp quality measure was developed by Ferrari and Canny, the Q measure, which evaluates how well a grasp can resist external disturbances. Later works proposed variations of this measure which improve it [91, 69, 116].

While we do not explicitly study analytical grasp planning in this dissertation, we do consider it to be an important component for detecting grasps based on visual inputs. The methods that are described in Chapters 4-6 calculate the ground truth for a grasp by checking geometric conditions which are derived from Nguyen's construction method for force closure grasps [81]. Generally, the chosen grasp quality measure could be exchanged with one of the choices described above if this is desired by the application.

2.3.2 Traditional Grasp Perception

The traditional approach to grasp perception consists of two steps: perception and planning. Based on perceptual data, the first step estimates the pose of the object to be grasped. The second step then uses the object pose to calculate a grasp configuration and a trajectory that enables the robot to grasp the desired object. The ROS grasping pipeline, introduced by Chitta et al. [17], is a typical example of this approach. Here, the perception component obtains a point cloud from an RGB-D sensor and registers the point cloud to a CAD model. The planning component then calculates a feasible arm and hand trajectory that results in a grasp of the localized object.

Ikeuchi et al. detect grasps in an image by segmenting the image into regions and estimating the attitude and position of the object [45]. They assume that all objects in the image are of the same type and that the object's material and shape are known. Other approaches in this area commonly first calculate feature descriptors to recognize an object and estimate its pose in a 2D image. Early research following this type of approach was surveyed by Chin and Dyer [16]. Local point features have been popular in the more recent literature. Among many others, the SIFT descriptor [71], SURF descriptor [5], and Harris corner detector [39] have often been used to calculate features. Based on the features, the object of interest can be matched against a database of object models. The 6D object pose can then be estimated from 2D-3D point correspondences. To recognize and localize an object in a 2D image, Vahrenkamp et al. iteratively estimate an affine transform based on feature point correspondences and estimate the 3D pose of the object with a stereo triangulation method [108]. Klank et al. perform shape matching by simulating the object appearance in the 2D image space [57]. Bohg et al. provide a comprehensive overview of approaches for grasping known objects [11].

Under ideal conditions, the traditional approach can work quite well. However, there are two deficiencies that make it challenging to apply this approach to real world settings. First, this approach inherently makes a closed world assumption that a model exists for every object to be grasped. Second, localization of the object pose given a noisy and incomplete point cloud can be very difficult [34].

2.3.3 Contemporary Grasp Perception

A more recent direction of research finds grasp configurations directly without prior object pose estimation. A database that stores object models and is used to estimate object pose does not need to be used.

A survey by Bohg et al. [11] on data-driven grasp perception divides approaches into three types based on the information available about the objects to be grasped: (i) known objects (as outlined in the previous section), (ii) familiar objects, and (iii) unknown objects. This ordering of types also corresponds to going from a closed world assumption (known objects) over relaxing that assuption to some degree to completely eliminating that assumption and arriving at open world scenarios.

Our previous work falls within the last two types described by Bohg et al. in their survey. We mostly consider an open world scenario where object models are not available. Another distinction from earlier research is that we address the problem of detecting grasp poses in the full 6-DOF space of poses. This sets our work apart from many of the earlier approaches (and also more recent ones) in grasp perception. In summary, we have developed methods which predict 6-DOF poses that correspond to enveloping and two-finger grasp affordances in noisy and unstructured point clouds obtained from one or multiple depth sensors [84, 85, 38, 86]. These methods will be described in the chapters of Part II of this dissertation. We end each of these chapters by discussing the relevant related work at the time of publication.

Part II

Grasp Pose Detection

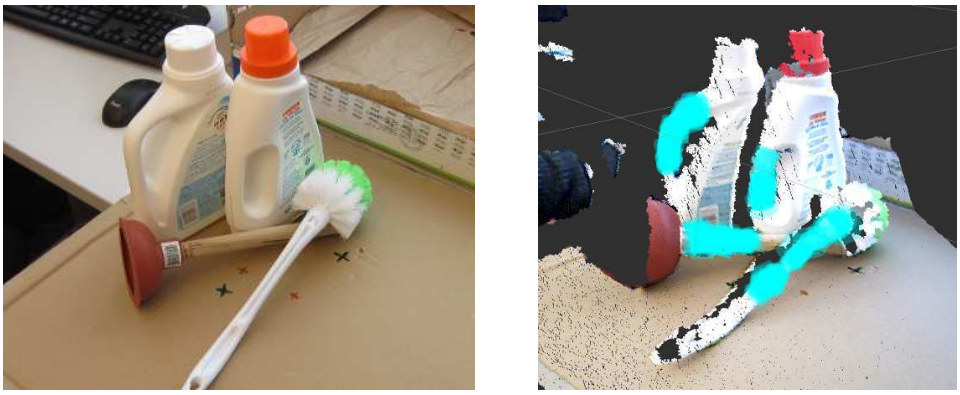
Chapter 3

Geometric Grasp Pose Detection

In this chapter, we present a purely geometric approach to grasp pose detection. Our method detects handle-like grasps where the robot's fingers can enclose some part of the object. A complete handle configuration is determined by seven parameters: a 6-DOF pose $X \in SE(3)$ and a radius r. A brute force search of the point cloud for such configurations is infeasible in real time. Our main idea is to search for pairs of co-linear cylinders that satisfy a set of geometric conditions with respect to local neighborhoods of the point cloud. First, we constrain the robot hand to grasp in a plane orthogonal to the minor principal curvature axis of the local object surface at the point where the grasp occurs. This constraint reduces the search space to three spatial dimensions. Second, we require a cylindrical gap to be present around an object surface to accommodate the robot fingers. This constraint enables us to eliminate many grasp candidates quickly. In robot experiments, we show that our method has high grasp success rates for both isolated objects and objects presented in clutter. Runtime measurements indicate that our method is very fast. We have made our method available to the robotics community as ROS package: `http://wiki.ros.org/handle_detector`.

3.1 Problem Statement

We want to solve Problem 2.1.1, i.e., to detect hand poses corresponding to a robotic grasp. In this chapter, we use the concept of *affordances*, defined by Gibson [31] as "a resource or support that the environment offers an animal; the animal in turn must possess the capabilities to perceive it and to use it" [31]. In the context of robotic perception, affordances are usually defined as those action possibilities that a robot can execute on an object and that are ready to be perceived [82]. In particular, we are looking for **handle-like grasp affordances**, where the object part to be grasped is roughly a handle. Such a grasp affordance affords a grasp of the object where the robot hand encloses the handle-like part of the object to be grasped. For example, all the objects shown in Figure 3.1a have handles and therefore they also have handle-like grasp affordances as depicted in Figure 3.1b. Given a point cloud of a set of objects with graspable handles, we want to solve the problem of finding a grasp



(a) (b)

Figure 3.1: (a) An RGB image of a typical scene. (b) Handle-like grasp affordances highlighted in cyan.

pose for each handle in the scene.

3.2 Approach

An **enveloping grasp affordance** is a handle-like object geometry that can be grasped by encircling it with the thumb and fingers of the robot hand. To locate these geometries in a point cloud, we search for cylindrical shells which satisfy a number of conditions with respect to local point neighborhoods of the point cloud. We define a **cylindrical shell** as a pair of co-linear cylinders with two different radii. For a cylindrical shell to be classified as an enveloping grasp affordance, we require the following conditions on the local point neighborhood to be satisfied.

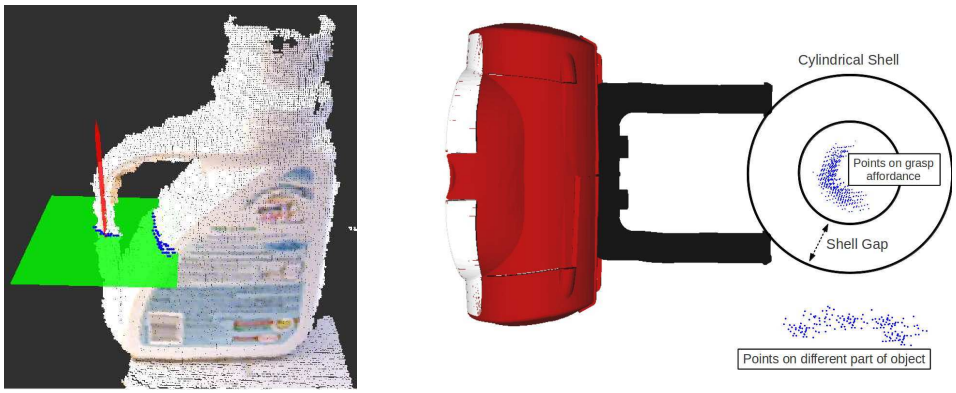
1. Points near the center of the neighborhood must lie on a curved object surface (with respect to a parameterized threshold on curvature).

2. The cylindrical shell's major axis must be parallel to the secondary axis of curvature of the local object surface.

3. The gap between the inner and outer cylinders must contain zero points and be wide enough to contain the robot fingers.

4. The radius of the innermost cylinder must be no larger than the maximum hand aperture.

An enveloping grasp affordance exists in the corresponding configuration if the above conditions are satisfied. If we assume that these conditions are satisfied, if we assume that points



(a) (b)

Figure 3.2: Illustration of an enveloping grasp affordance for a point neighborhood. (a) Points in the neighborhood (blue) are projected onto a plane (green) orthogonal to the minor curvature axis (red) of the object surface. (b) A cylindrical shell is found that satisfy the conditions for an enveloping grasp: points are contained within the inner shell and there's a large enough gap between the inner and outer circle to enable the robot fingers to envelop the object part.

lie densely on all object surfaces in the neighborhood, and if we assume that the neighborhood can be reached by the robot hand, then these conditions are sufficient to enable the robot to grasp the object by performing an enveloping grasp. An example of such a grasp configuration is given in Figure 3.2.

Algorithm 1 describes our algorithm. First, we sample a point uniformly from the point cloud (Step 3) and find the spherical neighborhood of a fixed radius with that point as the centroid (Step 4). Second, we fit an implicit quadratic function in three variables to each point neighborhood using a least squares algebraic fit with Taubin normalization [105] (Step 5). From the fitting, we can accuractely measure the axes of principal surface curvature and their magnitudes in the point neighborhood (Step 6). We then prune all neighborhoods for which the surface curvature is below a fixed threshold (Step 7) and project the points in the neighborhood onto the plane orthogonal to the axis of minor principal curvature (Step 8). Next, we fit a circle to the projected points (Step 9). We then set the center of the shell to the center of the fitted circle and perform a 1-D search for cylindrical shells which satisfy the conditions for an enveloping grasp affordance given above (Step 10). Last, we search for sets of enveloping grasp affordances which are roughly aligned and that exceed a minimum length (Step 16).

The details of the above algorithm are given in the next sections. We first describe how to estimate the curvature of the object surface (Step 5-6). Then, we elaborate on the cylindrical shell search (Step 8-10). Next, we outline the handle search (Step 16). Finally, in order to reduce the runtime of the algorithm, we propose two alternative strategies for sampling point

Algorithm 1 Handle Localization

$\mathcal{A} = \emptyset$
for $i = 0$ to I **do**
  Sample x uniformly from point cloud.
  Calculate neighborhood about x.
  Fit a quadratic surface S to point neighborhood.
  Estimate the median curvature $\hat{k}$ of S.
  if $\hat{k} > K$ **then**
    Project point neighborhood onto orthogonal plane.
    Fit a circle to points in plane and calculate the circle's center c.
    Search for cylindrical shell, a, centered at c.
    if a is found **then**
      $A = A \cup a$
    end if
  end if
end for
$\mathcal{H} \leftarrow$ findHandles($\mathcal{A}$).

neighborhoods which replace the uniform sampling of points from the point cloud (Step 1).

3.2.1 Estimation of Object Surface Curvature

To find regions of the point cloud with high curvature and to estimate the curvature axes accurately, we fit an implicit quadratic surface in three variables to the local point neighborhood. A quadratic surface can be described by $f(c, x) = 0$, where

$$f(c, x) = c_1x_1^2 + c_2x_2^2 + c_3x_3^2 + c_4x_1x_2 + c_5x_2x_3 + c_6x_1x_3 + c_7x_1 + c_8x_2 + c_9x_3 + c_{10}, \quad (3.1)$$

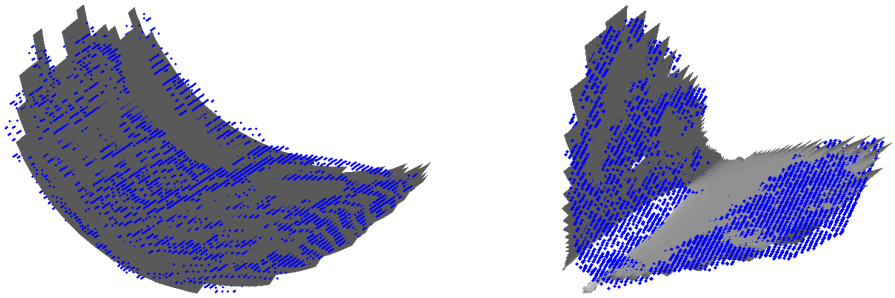
and $\mathbf{c} = \{c_1,\ldots,c_{10}\}$ denotes the parameters of the quadratic and $\mathbf{x} = \{x_1, x_2, x_3\}$ denotes the Cartesian coordinates of a point on the surface.

While there is no known fast (convex or closed form or etc.) method for calculating the implicit quadratic surface that minimizes least squares geometric distances to a set of points (called the *geometric fit*), there are fast methods for solving for an *algebraic fit*. The algebraic fit is a surface that solves the following optimization problem:

$$\min_c \sum_{i=1}^{n} f(\mathbf{c}, \mathbf{x}^i)^2 = \mathbf{c}^T M \mathbf{c}, \quad (3.2)$$

where $M = \sum_{i=1}^{n} l(\mathbf{x}^i)$, $l(\mathbf{x}^i)^T, \mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^3$ are the points to which the surface is fitted, and $l(\mathbf{x}) = (x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3, x_1, x_2, x_3)$.

To avoid the trivial solution $c = 0$, it is necessary to impose constraints on this problem. Different constraints give different results. To arrive at a fit that is intuitively close to the



(a) (b)

Figure 3.3: Two examples of an implicit quadratic surface fit using Taubin normalization.

geometric fit, we can use Taubin's method [105] that sets the constraint $||\nabla_x f(\mathbf{c}, \mathbf{x}^i)||^2 = 1$. Equation 3.2 can be reformulated as a Generalized eigendecomposition, $(M - \lambda N)\mathbf{c} = 0$, where

$$N = \sum_{i=0}^{n} l_x(\mathbf{x}^i) l_x(\mathbf{x}^i)^T + l_y(\mathbf{x}^i) l_y(\mathbf{x}^i)^T + l_z(\mathbf{x}^i) l_z(\mathbf{x}^i)^T. \tag{3.3}$$

Here, $l_x(\mathbf{x}^i)$ denotes the derivative of $l(\mathbf{x})$ taken with respect to x_1 and the other derivatives are taken similarly. The best-fit parameter vector is given by the eigenvector corresponding to the smallest eigenvector. Two examples for this fitting method are given in Figure 3.3.

To align the major axis of the cylindrical shell with the axis of minor principal curvature of the surface, we need to estimate the direction and magnitude of the curvature of the surface. At a particular point on the surface, the curvature can be calculated by evaluating the shape operator on the plane that is tangent to the point. The eigenvectors of the shape operator describe the principal directions of the surface and the corresponding eigenvalues give the curvature in these directions. For a point $\mathbf{x}$ on the surface, this can be calculated by solving the eigendecomposition of:

$$(I - N(\mathbf{x})N(\mathbf{x})^T)\nabla N(\mathbf{x}), \tag{3.4}$$

where $N(\mathbf{x})$ denotes the surface normals of the surface. It can be calculated by differentiating and normalizing the implicit surface:

$$N(\mathbf{x}) = \frac{\nabla f(\mathbf{c}, \mathbf{x})}{||\nabla f(\mathbf{c}, \mathbf{x})||}, \tag{3.5}$$

where

$$\nabla f(\mathbf{c}, \mathbf{x}) = \begin{pmatrix} 2c_1x_1 + c_4x_2 + c_6x_3 + c_7 \\ 2c_2x_2 + c_4x_1 + c_5x_3 + c_8 \\ 2c_3x_3 + c_5x_2 + c_6x_1 + c_9 \end{pmatrix} \tag{3.6}$$

After a quadratic surface is fit to a point neighborhood, we estimate the median curvature of the surface in the neighborhood by sampling several points at random from the

local surface and calculating the maximum curvature, i.e., the maximum of the two principal curvatures, at each of those points. We then take the median of these maximum values. We make the assumption that all enveloping grasp affordances are found in point neighborhoods with high curvature and thus eliminate from consideration all neighborhoods with an associated median curvature that is less than a fixed threshold.

Instead of fitting a quadratic surface to calculate local curvature axes and magnitudes for a point neighborhood, an alternative approach is to estimate curvature based on the surface normals of each point in the neighborhood. This approach works by calculating the eigendecomposition of the covariance matrix $\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T$, where $\mathbf{n_i}$ is the surface normal associated with point $\mathbf{x_i}$. The eigenvector that corresponds to the minimum eigenvalue gives the major principal axis of curvature. Ratios between the eigenvectors approximate the curvature magnitudes. This approach is common in point cloud processing [99]. In comparison, the method of fitting a quadratic surface described above seems to be more accurate, less noisy, and is faster to compute for large sets of points.

3.2.2 Cylindrical Shell Search

Once low curvature regions are eliminated and the axes of principal curvature and their magnitudes are estimated, we search for cylindrical shells in three steps (Step 8-10 in Algorithm 1). The first step is to project the points in the neighborhood onto a plane that is orthogonal to the minor principal curvature axis (see Figure 3.2a). The second step is to calculate the shell's center by fitting a circle to the points near the neighborhood's center. The circle fit is accomplished by minimizing algebraic distance as follows [22, 53]. Let x_i and y_i denote the coordinates of the i-th point in the plane. Let h_x, h_y denote the coordinates of the center of the circle and let r denote its radius. Then we calculate $\mathbf{w} = [a, b, c]$ as:

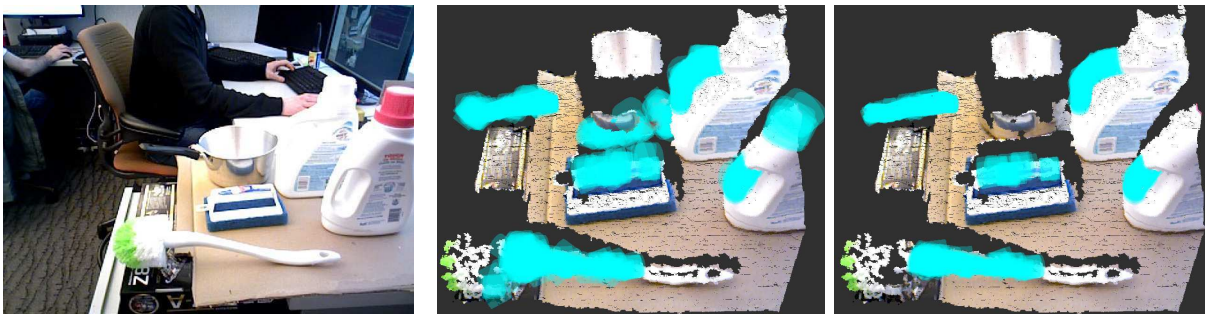
$$\mathbf{w} = -(\sum_{i=1}^{n} \mathbf{l}_i \mathbf{l}_i^T)^{-1} \sum_{i=1}^{n} \lambda_i \mathbf{l}_i, \tag{3.7}$$

where $\lambda_i = x_i^2 + y_i^2$ and $\mathbf{l}_i = [-x_i, -y_i, 1]^T$. The circle's center can then be calculated as $h_x = -0.5a, h_y = -0.5b$ and its radius as $r = \pm\sqrt{h_x^2 + h_y^2 - c}$.

The third step is to set the shell's center to the circle's center and then to search over different radii for a shell such that the gap between the inner and outer cylinders of the shell contains no points and the radius of the inner cylinder is less than the diameter of the robot hand (Condition 3 and 4 for the existence of an enveloping grasp affordance; see Figure 3.2b). This is a brute-force search.

3.2.3 Handle Search

As long as all object surfaces in the local area are densely covered with points, the presence of an enveloping grasp affordance guarantees that a grasp is possible in that configuration. Unfortunately, this is not always the case. In particular, objects with certain material



(a) (b) (c)

Figure 3.4: Illustration of handle search. (a) An RGB image of the scene. (b) All enveloping grasp affordances found by our affordance search in the point cloud. False positives are found on the surface of the brush and the pot, caused by measurement errors of the depth sensor. (c) The handles found which satisfy the alignment and minimum length constraints. The false positives discovered in the affordance search are eliminated by the handle search.

properties can be hard to perceive for a depth sensor, e.g., the highly reflective surface of the body of the pot shown in Figure 3.6b.

To mitigate this problem, we search for sets of enveloping grasp affordances which are roughly aligned and that have at least a certain minimum length. We call these sets *handles*. This process helps to reduce the number of false positive grasp configurations because true enveloping grasp affordances are typically found aligned along object handles. Typically, false positives are found in arbitrary configurations. An example where our handle search method eliminates all false positives is show in Figure 3.4.

To find handles, we perform a brute-force search over all pairs of enveloping grasp affordances. For each pair of affordances, i and j, with centroids h_i and h_j, major principal axes v_i and v_j, and radii r_i and r_j, we calculate the following three distance functions:

$$d_o = ||(I - v_i v_i^T) v_j||, \tag{3.8}$$

$$d_c = ||(I - v_i v_i^T)(h_i - h_j)||, \tag{3.9}$$

$$d_r = |r_i - r_j|. \tag{3.10}$$

The function d_o evaluates the distance to ..., d_c evaluates the distance between the centroids along ..., and d_r measures the distance between the radii of the two affordances i and j. If these three functions are below fixed thresholds, then affordance i is considered to be aligned with affordance j. If an enveloping grasp affordance i is aligned with at least a minimum number of other grasp affordances, then it is considered to define a handle affordance. The output of our algorithm is the set of all handles found using this method (Step 16 in Algorithm 1).

3.2.4 Sampling Strategy

An important step in our algorithm is sampling. Uniform random sampling, as show in Step 3 of Algorithm 1 is the basic approach. A point is sampled uniformly from the point cloud and our algorithm operates on the point neighborhood that is centered at the sampled point. However, uniform random sampling might require a large number of samples to be drawn in order to find all the handles in a scene. This is computationally expensive. For a scene such as the one depicted in Figure 3.1a, our experience indicates that 20,000 samples are sufficient. It takes about 1.7s to execute Algorithm 1 for 20,000 samples (see Section 3.3.3). To reduce runtime, we can use a more effective sampling strategy. Here, we explore a a sequential importance sampling method that can be seen as an implementation of the Cross Entropy Method [10]. Our method samples a fixed number of point neighborhoods in a series of rounds. In the first rounds, neighborhoods sre chosen uniformly at random from the point cloud. In subsequent rounds, samples are drawn from a proposal distribution that is parameterized by the positions of the enveloping grasp affordances which were found in all prior rounds.

A key choice that affects the performance of sampling is the form of the proposal distribution. Here, we explore two variants on the Gaussian kernel density proposal distribution: (i) a distribution expressed as a sum of Gaussians and (ii) a distribution expressed as a maximum over Gaussians. Let $x_i \in \mathbb{R}^3, i \in [1, n]$ denote the centroids of n enveloping grasp affordances found in all prior rounds. The sum of Gaussians distribution is:

$$g_{\text{sum}}(x) = \frac{1}{n} \sum_{i=1} n\mathcal{N}(x|x_i, \Sigma), \tag{3.11}$$

where Σ is a constant parameter. The maximum of Gaussians distribution is:

$$g_{\text{max}}(x) = \eta \max_{i \in [1,n]} \mathcal{N}(x|x_i, \Sigma), \tag{3.12}$$

where η is the normalization constant. Sampling from the sum of Gaussians distribution is straight forward. To draw k samples from g_sum, initialize $\mathcal{X} = \emptyset$ and repeat for k times: (i) select an enveloping grasp affordance index $j \in [1, n]$ uniformly at random, and (ii) draw one sample from $\mathcal{N}(x|x_i, \Sigma)$ and add it to $\mathcal{X}$. Sampling from the max of Gaussians distributions is more complicated. To draw samples from g_max, we use a method based on rejection sampling, presented in Algorithm 2.

The two proposal distributions g_{sum} and g_{max} differ in the way that they allocate samples to particular regions of space, i.e., to regions about potential handle locations. Samples are allocated by g_{sum} to a region in direct proportion to the number of grasp affordances that have been found in that region. If there are multiple handles present in a scene, but one handle is more likely to be populated by enveloping grasp affordances than the others (e.g., it could be longer or more densely covered with points in the point cloud), this can be a problem. The handle where grasp affordances are more likely to be found is sampled even more densely on the next round. This strategy tends to oversample some handles and ignore

Algorithm 2 Max of Gaussians Distribution Sampling

$\mathcal{X} = \emptyset$.
$j = 1$.
while $j < k$ **do**
  Select i uniformly from $[1, n]$.
  Sample $\hat{x} \sim \mathcal{N}(x|x_i, \Sigma)$.
  $m \leftarrow \max\{\mathcal{N}(\hat{x}|x_1, \Sigma), \mathcal{N}(\hat{x}|x_2, \Sigma), \ldots, \mathcal{N}(\hat{x}|x_n, \Sigma)\}$.
  if $\mathcal{N}(x|x_i, \Sigma) \geq m$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \hat{x}$.
  end if
end while

other handles in the scene. This effect can be corrected to some degree by sampling from g_{max} because it samples from all handle regions with a similar probability.

Figure 3.5 illustrates the difference between sampling from the two distributions. Suppose that our algorithm has found all of the enveloping grasp affordances highlighted in cyan. For $k = 100$, the set of samples drawn from g_{sum} is shown in Figure 3.5a, and the set of samples drawn from g_{max} is shown in Figure 3.5b. The samples in Figure 3.5a are more densely distributed on the object on the right than the one on the left. The reason for this is that the object on the right was more densely covered with enveloping grasp affordances in prior rounds. The samples drawn from g_{max} cover both objects more evenly, as depicted in Figure 3.5b.

(a) (b)

Figure 3.5: Illustration of the difference in sampling strategy. (a) Samples drawn from g_{sum}. (b) Samples drawn from g_{max}. Notice that the distribution in (b) covers the two handles more evenly than in (a).

3.3 Robot Experiments

We evaluated our method on a Rethink Robotics Baxter research robot. We used the right arm of the robot which has seven DOFs. For the robot hand, we used Baxter's standard parallel jaw gripper and configured the fingers such that they can open from 0 up to 4cm. To acquire point clouds, an Asus Xtion Pro RGB-D camera was mounted near the bottom of the robot's chest. Objects to be grasped where placed in front of the robot on a table top. This setup is depicted in Figure 3.6a.

To perform a grasp, we first acquired a point cloud from the Asus camera. Second, we then ran our algorithm on this point cloud that outputs handle affordances. Third, we moved the robot to the handle that is closest to the base link position of the right arm. In particular, the robot was translated such that a point in between the fingers matched the handle's centroid, and it was oriented such that the gripper was perpendicular to the handle axis and an axis pointing outward from the gripper was co-linear with a line between the right arm's base and the handle. Once the robot had reached the target pose, the gripper was closed and the object was transported above a box on the right side of the table where it was then dropped. If a grasp failed on the first attempt, the robot continued to attempt to grasp by repeating this process. During each motion, the arm followed a straight line through configuration space.

Our object set for the experiments consists of the 12 common household objects shown in Figure 3.6b. All of these objects have an object part that is a handle and that can be grasped with the width setting of the Baxter gripper (as described above). The objects were emptied to make it easier for the arm to pick them up.

(a) (b)

Figure 3.6: (a) Experimental setup. (b) The 12 objects in our test object set. All objects have a handle.

3.3.1 Isolated Object Grasping

In this experiment, we performed a series of grasp trials for each of the objects. On each trial, we placed the object in one of 12 different configurations: four orientations at three positions. Objects were placed such that a significant number of points on the object's handle were visible to the Asus camera and such that the handle was within the workspace of the robot's right arm. For each object configuration, we allowed the robot at most three attempts to grasp the object.

Object	Grasped on 1st try	Grasped on 2nd try	Grasped on 3rd try
Blue Bottle	10/12	10/12	12/12
White Purex Bottle	11/12	12/12	12/12
White All Bottle	9/12	12/12	12/12
Carrying Case	11/12	11/12	11/12
Brush 1	10/12	11/12	12/12
Pot	11/12	12/12	12/12
Plunger	11/12	12/12	12/12
Sprayer	11/12	12/12	12/12
Dust Pan	11/12	12/12	12/12
Brush 2	8/12	12/12	12/12
Sponge	8/12	12/12	12/12
Lint Roller	11/12	12/12	12/12

Table 3.1: Results for the isolated object grasping experiment.

The results of this experiment are shown in Table 3.1. Our method had a first-try grasp success rate of about 85%, a two-tries grasp success rate of 97.2%, and an almost perfect grasp success rate of 99.3% by the third try. The Carrying Case was the only exception: during a failed grasp attempt, it was pushed out of the robot's workspace (due to a collision between the gripper and the object).

3.3.2 Grasping in Clutter

In this experiment, we evaluated the capability of our method to grasp a series of objects in the presence of clutter. We performed 10 runs of grasping a set of five objects which were chosen randomly from the object set, and placed close to each other on the table top in configurations similar to the ones in the isolated object grasping experiment. A video that demonstrates a run of this experiment can be found at: https://youtu.be/qjE5X7pTjKE.

The results of the clutter grasping experiment are shown in Table 3.2. On average, we obtained a grasp success rate of 79.3%. Sometimes our method failed to grasp one out of the five objects. Furthermore, it could take up to eight attempts to grasp all of the objects in a single trial.

Trial #	1	2	3	4	5	6	7	8	9	10
# grasped / # total objects	5/5	4/5	5/5	4/5	5/5	4/5	5/5	5/5	4/5	5/5
Total grasp attempts	5	5	5	7	7	6	5	5	5	8

Table 3.2: Results for the clutter grasping experiment.

3.3.3 Algorithm Runtime

In this experiment, we measure the runtime of Algorithm 1. The algorithm was implemented in C++ on an Intel i7 3.5GHz system with four physical CPU cores and 8GB of system memory. The runtime measurements were averaged over ten runs for 20,000 samples. We measure the runtime of the major steps of our algorithm as well as the total time.

Figure 3.7 shows the results. The total runtime is a little more than 0.5Hz, with the majority of the runtime taken by by the 1-D brute-force shell search. We suspect that a closed-form approximation to the brute-force search exists that would reduce the runtime. Nevertheless, we expect that our implementation is fast enough for most application scenarios.

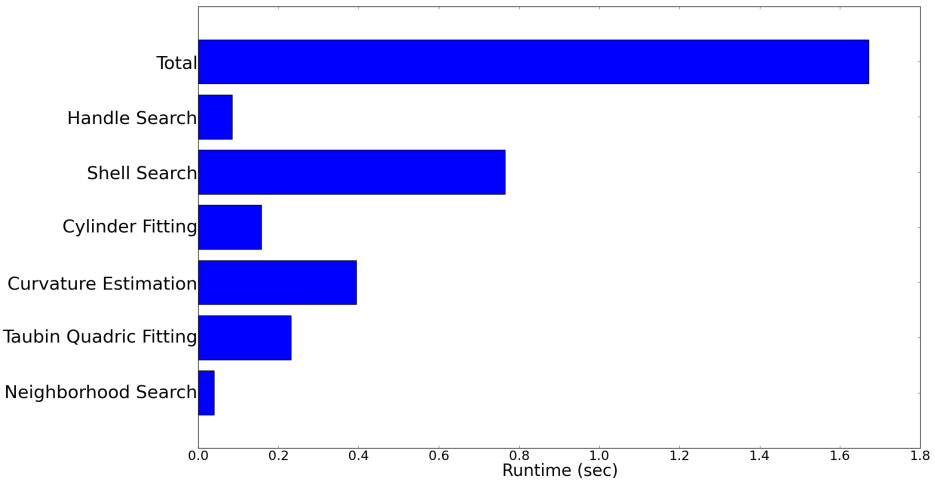


Figure 3.7: Runtime of the handle localization algorithm for 20,000 samples averaged over 10 runs.

3.3.4 Comparison of Sampling Strategies

In this experiment, we evaluate the number of handles in a scene which are missed by the algorithm as a function of the number of point neighborhoods and as a function of the sampling strategy used. Our evaluation set consists of point clouds generated for seven different scenes of which the first five contain five handles and the last two contain nine

and ten handles, respectively. We compared three different sampling strategies: (i) uniform random Monte Carlo (MC), (ii) sequential importance sampling with g_{sum}, and (iii) sequential importance sampling with g_{max}. For each strategy, we evaluate our algorithm for a total of 2000 and 5000 neighborhoods. For the Monte Carlo strategy, we just sampled all 2000 or 5000 neighborhoods in one batch. For sequential importance sampling with 2000 samples, we sampled 1000 neighborhoods in the first round and 100 neighborhoods in each of ten subsequent rounds. For sequential importance sampling with 5000 samples, we sampled 2000 neighborhoods in the first round and 300 neighborhoods in each of ten subsequent rounds.

Figure 3.8 presents the results for this experiment. Each bar gives the mean and standard deviation of 20 runs for the corresponding test scene. The ground truth bar (yellow) shows the actual number of handles present in each scene. These results indicate the following. First, our method can be expected to find two or three handles in any scene with as few as 2000 samples using any sampling method. This is sufficient for some tasks, e.g., clearing a table, where it is only necessary to grasp one object at a time. However, even 5000 sampled neighborhoods might not be enough to find all handles in a complex scene, especially if uniform random Monte Carlo sampling is used. We found that it was necessary to use as many as 20000 sampled neighborhoods in order to localize all handles using this sampling strategy. Second, the results indicate that it is generally better to use a sequential sampling strategy. Moreover, the results show that sequential importance sampling using the g_{max} proposal distribution has the best performance because it is able to find nearly all handles with 5000 sampled neighborhoods.

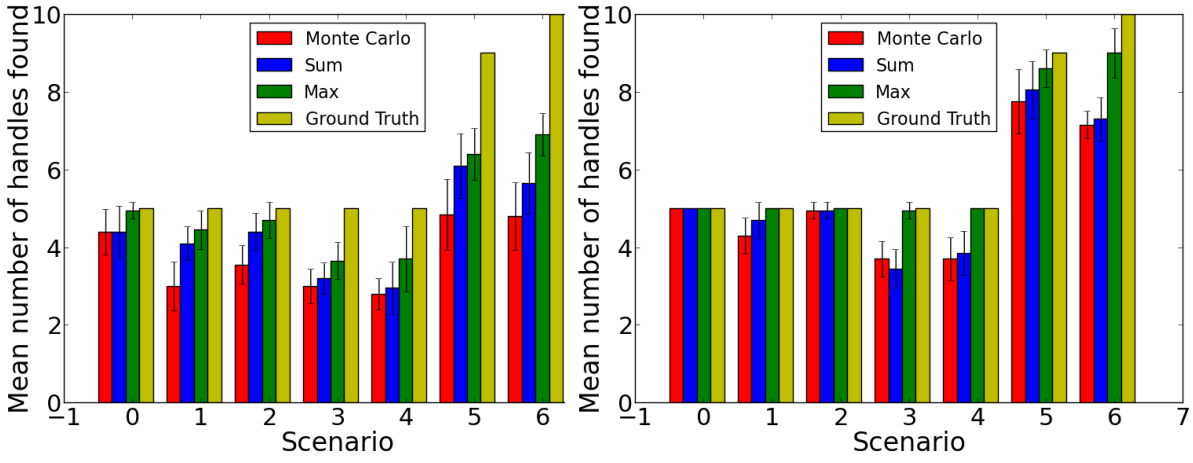


Figure 3.8: Comparison of sampling strategies. (a) 2000 sampled neighborhoods. (b) 5000 sampled neighborhoods.

3.4 Discussion

The strength of this method relative to related approaches is that it is very practical: it has a high grasp success rate for the types of affordances under consideration, it runs in real-time, and it is easy to adapt to different robots and operating scenarios. Enveloping grasp affordances can be executed with many different types of robot hands: parallel jaw grippers can be moved into the gap between the two cylindrical shells in opposing locations (see Figure 3.2b and multi-fingered hands might enclose the corresponding object part with their fingers. Furthermore, it does not require to train a machine learning model because the approach is purely geometric.

In robot experiments, we observed a few grasp failure modes. For isolated object grasping, nearly all failures were caused by trying to grasp false positives. These false positives were found because of depth measurement errors or because of insufficient point density on object surfaces in the neighborhood of the false positive. For example, the grasp failures of Brush 2 were caused by false positives detected on the brush part of the object because of large measurement errors in that region. Grasping objects in clutter suffered from similar failures, however, these were amplified because of the clutter. Sometimes, a failed attempt to grasp some object pushed other objects out of the robot's workspace so that it was impossible to clear all objects from the table.

A weakness of this method is that the grasp configurations can only be detected for object parts that resemble a handle. Another weakness is that such grasp configurations are ideally executed by a robot hand with fingers that are able to enclose around a handle-like object part. We expect that such grasps are necessary to perform some tasks effectively, e.g., to use a hammer to put nails into a wall. On the other hand, such grasps might be used for pick and place tasks but are not necessary. A simple planar two-finger grasp with a parallel jaw gripper would be sufficient for many such tasks.

3.5 Related Work

Current research for detecting grasp poses is mostly focused on deep learning based approaches. Localizing graspable geometries has been a research subject even before those more recent approaches. Klingbeil et al. search for geometries in a range image that can be grasped by a parallel-jaw gripper [58]. A three-dimensional search for a planar position and an angle about the gripper's approach axis is performed over the range image. The gripper is constrained to approach the object from a single direction. Jiang et al. search an RGBD image for regions that score high on a grasp score function where feature weights were learned offline [49]. Similary, Fischinger and Vincze perform a 3-DOF search of a height map that is calculated from a point cloud [28]. Herzog et al. learn graspable height map templates from user demonstrations of grasps [42]. A method that depends on physical interaction with the objects to be grasped was developed by Katz et al. [54], where pushing of the target object is used to segment it accurately.

Another line of research explores searching for known modeled objects in a scene for which feature matching is commonly used. Brook, Ciocarlie, and Hsiao develop a database-driven method that segments the point cloud into clusters and compare these clusters against 3D models in a database [13]. They use a Bayesian framework that incorporates uncertainty in object pose, object shape, and robot motion.

Our approach distinguishes itself by the introduction of a new feature type used to develop a graspability score function for object parts. This feature type describes the curvature of the part of interest. Furthermore, we directly search the point cloud for grasp affordances instead of a depth image or height map. This enables us to structure the search in different ways and allows more freedom in choosing the orientation of the robot hand for the grasp.

Chapter 4

Supervised Learning for Grasp Pose Detection

In this chapter, we present a supervised learning approach to grasp pose detection. Our approach looks for antipodal grasp configurations by first sampling hand pose candidates and then classifying each candidate as a mechanically stable (i.e., antipodal) grasp or not. To sample candidates from the space of robot hand poses, we introduce a method that enables us to sample from a low dimensional subspace of $SE(3)$. This space is constrained by the point cloud geometry and contains poses which are more likely to result in a stable grasp. To classify the grasp candidates generated this way, we propose a grasp representation that encodes the geometry of the object part in the hand closing region as an image. This enables us to train a support vector machine (SVM) on a feature descriptor derived from these images. In simulation experiments, we show that the learned SVM has high classification accuracy, even in the case of novel objects and under partial observability. In robot experiments, we demonstrate that we can grasp objects presented in isolation with a high success rate and that we can grasp objects presented in dense clutter. We have made our method available to the robotics community as a ROS package: `http://wiki.ros.org/agile_grasp`.

4.1 Problem Statement

To solve Problem 2.1.1, i.e., to detect hand poses corresponding to a robotic grasp, our goal in this chapter is to detect **antipodal hand poses**, a type of grasp configuration based on the concept of an antipodal grasp. Nguyen defines an antipodal grasp as follows [81].

Definition 4.1.1 (Antipodal Grasp) *A pair of point contacts with friction is antipodal if and only if the line connecting the contact points lies inside both friction cones.*

If an antipodal grasp exists for an object, then the robot can hold that object by applying sufficiently large forces along the line connecting the two contact points. An antipodal grasp

is in *force closure* [80]: any external wrench applied to the grasped object can be balanced by the contact forces applied by the robot hand.

In this work, we only consider parallel jaw grippers. A parallel jaw gripper has parallel fingers and a single DOF of position for moving the fingers along a single direction of motion. Because of this motion, we require the two contact points in Definition 4.1.1 to be located on a line parallel to the finger motion direction. Instead of direct localization of antipodal contact configurations, our goal is to localize hand configurations where we expect an antipodal grasp to be achieved when the robot hand is closed.

Definition 4.1.2 (Antipodal Hand) *An* ***antipodal hand*** *is a pose of the hand, hinH, such that the hand is not in collision with any objects or obstacles, $B(h) \cap \prime = \emptyset$, and at least one pair of antipodal contacts will be formed when the fingers close such that the line connecting the two contacts is parallel to the direction of finger motion.*

Here, $\mathcal{O} \subseteq \mathcal{W}$ denotes the space occupied by objects or obstacles in world $\mathcal{W}$, and $B(h) \subseteq \mathcal{W}$ denotes the volume occupied by the hand in configuration $h \in H$, where the fingers are completely open.

4.2 Approach

At a high level, Algorithm 3 outlines our algorithm for detecting grasp poses that satisfy Definition 4.1.2. As input, our algorithm takes a point cloud $\mathcal{C} \subseteq \mathbb{R}^3$ and a geometric model of the robot hand θ. As output, it produces a set of hand poses $\mathcal{H} \subseteq H$ which are predicted to be antipodal. There are two main steps. The first step is to sample a set of grasp candidates. The second step is to classify each candidate as an antipodal hand or not. In the following two sections, we describe these two steps in detail.

Algorithm 3 Grasp Pose Detection

1: $\mathcal{H}_{\text{candidates}} = \text{SampleHands}(\mathcal{C})$.
2: $\mathcal{H} = \text{ClassifyHands}(\mathcal{H}_{\text{candidates}})$.

4.3 Hand Sampling

A key part of our algorithm is the approach to sampling from the space of robot hand poses. A naive approach would be to sample directly from $H \subset \text{SE}(3)$. However, this would not be efficient because the SE(3) space has six dimensions and many hand poses sampled this way would have a large distance to any points in the cloud. Instead, we define a lower dimensional sample space that is constrained by the geometry of the point cloud. In the next three sections, we first give a geometric model for a robot hand and the object surface

to be grasped, we then define the sample space by introducing constraints on the hand poses to be generated, and finally we propose an algorithm for sampling from this space.

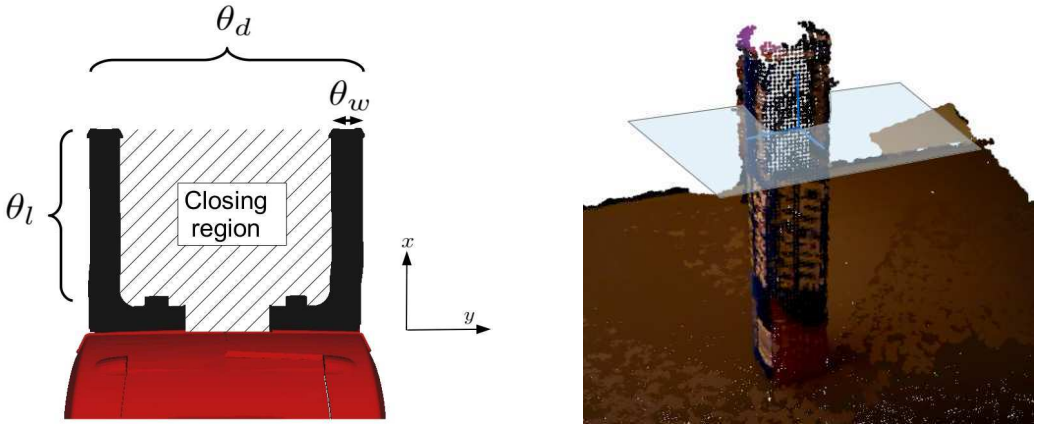
4.3.1 Robot Hand Model

We assume that the robot hand $\mathcal{R}$ is a parallel jaw gripper, like the Baxter gripper shown in Figure 4.1a. We model the two parallel fingers of such a hand as rectangular prisms which move parallel to a common plane. For some hand pose $h \in SE(3)$, let $\hat{a}(h)$ denote the normal vector of this plane. A full specification of the hand is given by the parameter vector $\theta = (\theta_l, \theta_w, \theta_d, \theta_t)$ where θ_l and θ_w denote the length and width of the fingers, θ_d denotes the distance between the fingers when the hand is completely open, and θ_t denotes the thickness (or height) of the fingers (orthogonal to the page in Figure 4.1a).

We define the **closing region**, $R(h) \subseteq \mathcal{W}$, to be the volumetric region that is swept out by the fingers when they close. Let $r \in R(h)$ denote an arbitrary reference point in the closing region. We define the **closing plane**, $C(h)$, to be the subset of the plane that intersects r, is orthogonal to $\hat{a}(h)$, and is contained within $R(h)$, i.e.:

$$C(h) = \{p \in R(h) | (p - r)^T \hat{a}(h) = 0\}. \tag{4.1}$$

We characterize the object surface to be grasped using differential geometry. Each point on a differentiable surface is associated with a surface normal and two principal curvature axes. These three vectors form orthogonal basis known as a Darboux frame. Technically, any frame that is aligned with the surface normal is a Darboux frame but we here consider only the special case where the frame is also aligned with the principal axes. We denote the Darboux frame at a point $x \in \mathcal{C}$ as $F(x) = [\hat{n}(x)\ (\hat{a}(x)\hat{n}(x))\ \hat{a}(x)]$, where $\hat{n}(x)$ denotes the unit surface normal and $\hat{a}(x)$ denotes the axis of minor principal curvature at the point x.



(a) (b)

Figure 4.1: (a) Robot hand geometry. (b) Cutting plane geometry.

We define the **cutting plane** to be the plane that is orthogonal to $\hat{a}(x)$ and that passes through x, as illustrated in Figure 4.1b.

We cannot measure the Darboux frame exactly at each point because we do not have access to the ground truth geometry of the observed object surfaces. We only have access to a point cloud that can be both incomplete and noisy. Therefore, we estimate the surface normal and principle directions over a small point neighborhood. To calculate the Darboux frame, we fit a quadratic surface to the neighborhood, as described in Section **??**.

4.3.2 Constraints for Hand Sampling

To arrive at a lower dimensional space from which we can sample hand poses, we define the set $\mathcal{H} \subset \mathrm{SE}(3)$ where $h \in \mathrm{SE}(3)$ is an element of $\mathcal{H}$ if it satisfies the following three constraints. For every $h \in \mathcal{H}$, the first constraint is:

Constraint 1 *The body of the hand is not in collision with the point cloud:* $B(\mathcal{R}) \cup \mathcal{C} = \emptyset$.

For the second and third constraint to be satisfied, there must exist a point in the point cloud, $p \in \mathcal{C}$, such that:

Constraint 2 *The hand closing plane* C *contains* x*:* $p \in C$.

Constraint 3 *The hand closing plane* C *is parallel to the cutting plane at* x*:* $\hat{a}(p) = \hat{a}(h)$.

Now, we can define $\mathcal{H}$ as $\mathcal{H} = \bigcup_{x \in \mathcal{C}} H(x)$, where

$$H(x) = \{h \in \mathbb{H} | x \in C, \hat{a}(x) = \hat{a}(h), B(\mathcal{R}) \cup \mathcal{C} = \emptyset\}. \tag{4.2}$$

Essentially, Constraint 3 is a heuristic that reduces the size of the set of hand poses that we need to consider by constraining two DOFs of orientation and one DOF of position. Therefore, the sample space $\mathcal{H}$ is significantly smaller than the space of all hand poses $\mathbb{H}$ and it can hence be covered with many fewer samples. Another motivation for this constraint is that humans prefer grasps for which the wrist is oriented perpendicularly to one of the principal axes of the object [4]. Furthermore, drawing samples from $\mathcal{H}$ is an easy two-step procedure: (i) sample a point $p \in \mathcal{C}$ from the point cloud, and (ii) sample one or more hand poses from $\mathcal{H}(p)$.

4.3.3 Sampling Algorithm

Algorithm 4 gives an algorithm for sampling hand poses that satisfy the constraints specified in the previous section. First, we preprocess the point cloud, $\mathcal{C}$, by voxelizing and applying workspace limits (Step 2). Second, we iteratively sample a set of n points from the point cloud (Step 4). For each point, $p \in \mathcal{C}$, we calculate a neighborhood, $N(p)$, in the θ_d-ball around p (Step 5). We then estimate the Darboux frame at p by fitting a quadratic surface,

as described in Section **??**, and calculate the surface normal and principal curvature axes (Step 6). The next step is to sample a set of hand poses over a coarse 2-DOF grid in a neighborhood about p (Step 7). Finally, we add the hand poses which are produced by the grid search to the output set (Step 8).

Algorithm 4 Hand Sampling

1: $\mathcal{H} = \emptyset$.
2: Preprocess the point cloud $\mathcal{C}$.
3: **for** $i = 1$ to n **do**
4: Sample $p \in \mathcal{C}$ uniformly at random.
5: Calculate θ_d-ball about p: $N(p) = \{q \in \mathcal{C} : ||p - q|| \leq \theta_d\}$.
6: Estimate local Darboux frame at p: $F(p) = \text{EstimateDarboux}(N(p))$.
7: $H = \text{GridSearch}(F(p), N(p))$.
8: $\mathcal{H} = \mathcal{H} \cup H$.
9: **end for**

The subroutine for this step is outlined in Algorithm 5. With respect to the Darboux frame $F(p)$, let $h_{x,y,\phi}$ denote the hand pose at position $(x, y, 0)$ and orientation ϕ. Let Φ denote a discrete set of hand orientations ($|\Phi| = 8$ in our implementation), and let X denote a discrete set of hand positions ($|X| = 10$ in our implementation). For each hand configuration $(\phi, x) \in \Phi \times X$, we calculate the hand configuration furthest along the y axis of the Darboux frame that remains collision free, $y^* = \max_{y \in Y}$ such that $B(h_{\phi,x,y}) \cap N \neq \emptyset$, where $Y = [-\theta_d, \theta_d]$ (Step 3). Then, we evaluate whether the closing plane for this hand configuration contains any points from the point cloud (Step 4). If it does, we add the hand pose to the output set (Step 5).

Algorithm 5 Grid Search

1: $H = \emptyset$.
2: **for all** $(\phi, x) \in \Phi$ **do**
3: Push hand until collision: $y^* = \max_{y \in Y}$ such that $B(h_{\phi,x,y}) \cap N \neq \emptyset$.
4: **if** $C(h_{\phi,x,y^*}) \cap N \neq \emptyset$ **then**
5: $H = H \cup h_{x,y,\phi}$.
6: **end if**
7: **end for**

4.4 Grasp Classification

After grasp candidates have been generated, the second step in our grasp pose detection method is to classify each of those candidates as antipodal or not. If we had access to the full geometry of the object, we could check which candidates satisfy Definition 4.1.1.

However, point clouds only provide partial and noisy information about the world and a lot of candidates could not be checked for Definition 4.1.1 because not all relevant object surfaces are visible. To tackle this problem, we infer which grasp candidates are likely to be antipodal using supervised learning, i.e., classification. We pose the inference of antipodal grasps as a binary classification problem.

4.4.1 Generation of Training Labels

Our method automatically labels a set of training images by checking a relaxed version of the conditions of Definition 4.1.1. In order to check whether a hand pose, $h \in H$, is antipodal, we need to determine whether an antipodal pair of contacts will be formed when the hand closes. Let $\hat{f}(h)$ denote the closing direction of one finger of the robot hand. When the fingers close, they will first make contact with an extremal pair of points, $s_1, s_2 \in \mathbb{R}$, for which the following condition holds:

$$\forall s \in R(h), s_1^T \hat{f}(h) \geq s^T \hat{f}(h) \wedge s_2^T \hat{f}(h) \leq s^T \hat{f}(h). \tag{4.3}$$

For an antipodal hand, we require two such extremal points to be antipodal. Additionally, the line that connect these two points needs to be parallel to the finger closing direction. We relax this condition slightly in two ways. First, rather than finding extremal points, we consider points with a surface normal parallel to the closing direction. Essentially, this is a first order condition for an extremal point that is more robust to point cloud outliers. Second, instead of requiring the line that connects the two contacts to be parallel to the finger closing direction, we require that at least k points are found with an appropriate surface normal, i.e., a surface normal that is almost parallel to the closing direction. This latter condition improves the robustness of detecting antipodal grasps. If there are at least k points close to each finger with have surface normals which are parallel to the closing direction, then it is likely that there exists at least one pair for which the connecting line is nearly parallel to the closing direction. In summary, we look for hand poses which satisfy the following definition.

Definition 4.4.1 (near antipodal) *A hand, $h \in H$, is **near antipodal** for thresholds $k \in \mathbb{N}$ and $\theta \in [0, \pi/2]$ if there exist k points $p_1, \ldots, p_k \in R(h) \cap \mathcal{C}$ such that $\hat{n}(p_i)^T \hat{f}(h) \geq \cos(\theta)$ and k points $q_1, \ldots, q_k \in R(h) \cap \mathcal{C}$ such that $\hat{n}(p_i)^T \hat{f}(h) \leq -\cos(\theta)$.*

When Definition 4.4.1 is satisfied, we label the corresponding hand pose as a positive instance. Otherwise, we label it as a negative instance.

In order to check for the conditions in Definition 4.4.1, it is necessary to register at least two point clouds produced by range sensors which have observed the scene from different perspectives (see Figure 4.4a). The reason for this is that two nearly opposite object surfaces need to be visible in the point cloud. Even with two sensors, many hand poses cannot be identified as near antipodal or not because only one side of the object is visible. We call such hand poses *indeterminate* and omit them from the training set (see Section 4.4.3).

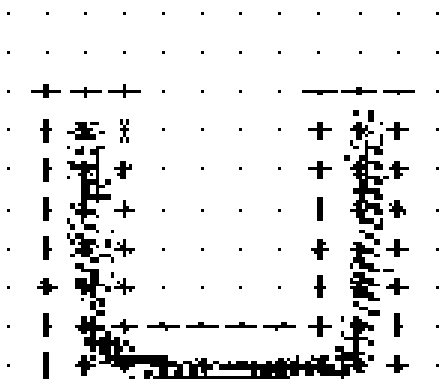


Figure 4.2: HOG feature descriptor of a grasp candidate for the box depicted in Figure 4.1b.

4.4.2 Grasp Representation

To classify grasp candidates, we need to represent the candidates so that they can be used as input for a classifier. For a given hand pose $h \in H$, we encode the geometry of the points which are contained within the hand closing region, $\mathcal{C} \cap R(h)$. Here, we encode this geometry with a simple feature descriptor that is based on Histogram of Oriented Gradients (HOG) features [20]. We create a two-dimensional of the hand closing region by projecting the points in $\mathcal{C} \cap R(h)$ onto the closing plane, $I(\mathcal{C}, h) = S_{12}F(h)^T(N \cap C(h))$, where S_{12} is a projection matrix that selects the first two rows of $F(h)^T$. We call $I(\mathcal{C}, h)$ the **grasp image**, and encode it using the HOG descriptor, $\text{HOG}(I(\mathcal{C}, h))$. In our implementation, we choose a HOG cell size such that the grasp image was covered by 10×12 cells with a standard 2×2 block size. Figure 4.2 gives an example of this encoding.

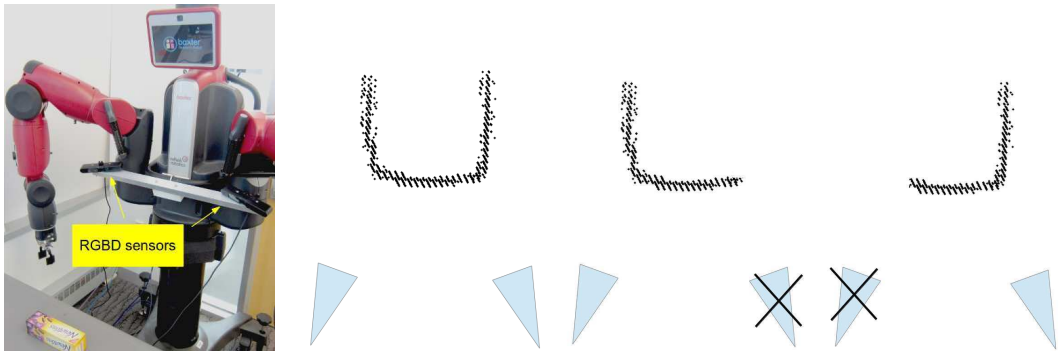
4.4.3 Training Data Generation

Our training set consists of the 18 objects shown in Figure 4.3a. For each object, we collected data for two two different configurations: one upright configuration and one configuration where the object is placed on its side (a total of 36 configurations). To obtain point clouds, we mounted two Asus Xtion Pro RGBD cameras to the torso of a Baxter research robot, as illustrated in Figure 4.4a. For each configuration, we obtained a point cloud from each of the two sensors. Let $\mathcal{C}_1$, $\mathcal{C}_2$ denote the individual point clouds, and let $C_{12} = \mathcal{C}_{12} = \mathcal{C}_1 \cup \mathcal{C}_2$ denote the two-view point cloud. We generate training data in three steps. First, using the method from Section **??**, we create grasp candidates from the two-view point cloud C_{12}. Second, for each candidate $h \in H$, we check if it satisfies the conditions in Definition 4.4.1 and determine if it is a positive, negative or indeterminate instance. Third, for each positive or negative instance, we extract three feature descriptors, one for each of the three sensor configurations depicted in Figure 4.4(b-d): $\text{HOG}(I(\mathcal{C}_1, h))$, $\text{HOG}(I(\mathcal{C}_2, h))$, and $\text{HOG}(I(\mathcal{C}_{12}, h))$. Each descriptor is paired with the same label and added to the training

(a) (b)

Figure 4.3: (a) The 18 objects in our training set. (b) The 30 objects in our test set.

set. Including all three descriptors makes our method more robust to partial point cloud information. This is because HOG($I(\mathcal{C}_1, h)$) and HOG($I(\mathcal{C}_2, h)$) emulate what the point cloud would look like for a single view. For the 18 objects in our training set, this procedure generated about 6500 positive and negative instances.



(a) (b) (c) (d)

Figure 4.4: (a) Our robot with two depth sensors. (b-d) The three grasp images added to the training set for each grasp candidate: image created with two sensors (b), image created with only one sensor (c,d).

4.5 Simulation Experiments

In this section, we evaluate the performance of a machine learning model for the grasp classification task. First, we measure classification accuracy on two-view point clouds of known objects from the training set with ten-fold cross validation. Second, we characterize performance on single-view point clouds of novel objects from a test set.

4.5.1 Cross Validation

We use the instances generated by the procedure described in the previous section to train a support vector machine (SVM) [12] for a single round of training. To find the SVM kernel that performs best, we performed cross validation on a dataset that is derived from the 18 training objects shown in Figure 4.3a. For each object, we obtained a registered point cloud for two configurations (a total of 36 configurations). Following the procedure described in Section 4.4.3, we obtained 6500 labeled features with 3405 positives and 3095 negatives. We performed ten-fold cross validation of the SVM on this dataset for the various polynomial and Gaussian kernels which are available in Matlab [76]. With a polynomial kernel of degree three, we achieved 97.8% classification accuracy over then ten folds. We use this kernel for the remainder of our experiments in this chapter.

4.5.2 Novel Objects and Partial Views

To evaluate how our algorithm deals with novel objects and less visual information, as in the case of single-view point clouds, we performed an experiment as follows. First, we trained the SVM using the degree-three polynomial kernel on the 6500 labeled examples as described above. Second, we obtained single-view point clouds for each of the 30 novel test objects shown in Figure 4.3b, where each object was presented in isolation. In total, we obtained 122 single-view points clouds. Using the ground truth labeling procedure outlined in Section 6.3.1, we gathered 7250 single-view grasp poses on novel objects, out of which 1130 were positives and 6120 were negatives. Our algorithm obtained a classification accuracy of 94.3% on this dataset. Despite not mining hard negatives to train the SVM and using a relatively simple feature descriptor, the high accuracy achieved in this experiment indicates that the way our hand sampling method and the way in which we encode a grasp candidate as an image simplify the grasp classification task compared to approaches that do not follow this kind of structure [28, 29, 49, 64, 101].

4.6 Robot Experiments

In this section, we investigate the performance of our method on a physical robot. We explore two experimental scenarios: (i) a single object presented to the robot in isolation, (ii) multiple objects presented in dense clutter. In this section, we first describe how we select a grasp to be executed from the outputs of our algorithm. Next, we describe the experimental setup. Finally, we provide results for the two experimental scenarios.

4.6.1 Grasp Selection

Our algorithm typically outputs a large number of potential antipodal robot hand poses. Therefore, it is necessary to choose one out of them to execute. One way would be to choose a grasp on an object of interest. However, we ignore object identity in this work and we

want to execute any feasible grasp. On the robot, we choose a grasp to be attempted by first reducing the number of grasp choices. We cluster antipodal hands based on distance and orientation. Grasp poses that are nearby each other and that are approximately aligned in orientation are assigned to the same cluster. We require each cluster to contain a minimum number of constituent grasps. If a cluster is found, then we create a new grasp pose whose position is the mean and whose orientation is the "average" orientation of the grasps which belong to the cluster. Second, we select a grasp by evaluating how easy it is for the robot to reach. We solve the inverse kinematics (IK) for each of the potential grasp poses and eliminate those which have no solutions. Then, we rank the remaining grasps according to the following three criteria. The first criterion evaluates the distance from the robot's joint limits and is expressed as a a piece-wise function that is zero far from the arm joint limits and quadratic close to the limits. The second criterion evaluates the distance from the hand joint limits and is zero far from the limits and quadratic near the limits. The third criterion evaluates the workspace distance that is traveled by the hand starting from a fixed pre-grasp arm configuration. We order grasps using these three criteria by first selecting those which minimize the first criterion, and then out of those, we select the ones which minimize the second criterion. Finally, we select the one to be executed by the robot that minimizes the third criterion out of the latter subset.

4.6.2 Setup

All our robot experiments were performed on a Baxter research robot made by Rethink Robotics. We use the right arm of the Baxter robot. The arm has seven DOFs and is equipped with the stock two-finger Baxter gripper. A key constraint of the Baxter gripper is the limited finger stroke: each finger has only 2 cm stroke. In these experiments, we adjust the finger positions such that they have a distance of 3cm when closed and of 7cm when open. This means that we cannot grasp objects smaller than 3cm or larger than 7cm. The objects in our training and test sets were chosen according to these constraints. Two-view point clouds were acquired using Asus Xtion Pro sensors (see Figure 4.4a) and transformed to the robot's base frame.

We implemented our algorithm in C++ on an Intel i7 3.5GHz system with four physical CPU cores and 16GB of system memory. While learning the SVM parameters for the 18 training objects takes about five minutes on average, online grasp detection takes about 2.7s for 4000 hand samples. Motion plans and inverse kinematics solutions were generated with ROS and the Baxter SDK.

4.6.3 Isolated Object Grasping

In this experiment, we characterize how well our method is able to grasp objects that are placed in isolation on a table in front of the robot. The object set consists of the objects shown in Figure 4.3b, except for the objects shown in Figure 4.5 which we excluded because they were hard to see with the Asus Xtion depth sensor. This is due to the objects' specularity,

transparency, or color. We look at the performance for these particular objects in a separate experiment at the end of this section. We compare three variations on our algorithm.

1. **No classification (NC)**: We assume that all grasp candidates generated by the hand sampling algorithm (Algorithm 4) are antipodal and do not use classification.
2. **Antipodal (A)**: We classify candidates by evaluating the conditions in Definition 4.1.2 directly for each hand pose (instead of using an SVM).
3. **SVM**: The complete algorithm proposed in the previous sections. Hand poses are classified with a SVM that was trained on the 18 objects shown in Figure 4.3a.

We consider a grasp attempt to be successful if the robot localized, grasped, lifted and transported an object to a box on the side of the table. We evaluate the three variations above on single-view and two-view point clouds for a total of 214 grasp attempts over the 30 test objects. Each object was placed in between six and eight different orientations.

The results of this experiment are shown in Table 4.1. Columns *NC, 1V* and *NC, 2V* show the results for the *No Classification* variation for one- and two-view point clouds, respectively. As shown in column *NC, 1V*, with a point cloud created from a single depth sensor, sampling with no additional classification results in an average grasp success rate of only 58%. However, as illustrated by column *NC, 2V* it is possible to raise this success rate to 73% by the addition of a second depth sensor and using the resulting two-view registered point cloud. This is a surprisingly large success rate because the sampling strategy uses rather simple geometric constraints. It suggests that even simple geometric constraints can improve grasp detection significantly.

The results for the *Antipodal* variation are given by column *A, 2V*. We only evaluated this variation for two-view and not for one-view point clouds because two views of an object are required for Definition 4.4.1 to find near antipodal hand poses. This variation only finds a small number of positives because it needs to be able to see two opposing sides of a potential grasp surface to verify if a grasp is present. Therefore, we only evaluated this method for three poses for each object. For four of the objects, this method failed to find any grasps. While the success rate is very high (94.7%), it is not a practically useful method because it only works for a small number of object orientations.

The results for the complete detection method are shown in columns *SVM, 1V* and *SVM, 2V* for one- and two-view point clouds, respectively. Compared to *NC*, adding a classifier improves the success rate to 85% for a one-view point cloud. Notice that the addition of a second sensor does not give a large improvement here as this increases the success rate by just about 2% to 87.8%. There are three major failure modes for this method: (1) collisions between the gripper and the object to be grasped due to calibration errors of the robot kinematics or collisions with unobserved parts of the environment (5.6% in both scenarios), (2) dropping of objects after an initially successful grasp (3.5% in both scenarios), (3) perceptual failures of our detection algorithm (3.7% in the one-view and 2.3% in the two-view scenario). Since not all of these failure are due to perception, we would need to make improvements in multiple areas in order to improve the success rate beyond 87.8%.

Object	**# poses**	**Succ. Rate** A, 2V		**# poses**	**Grasp Success Rate**			
					NC, 1V	NC, 2V	SVM, 1V	SVM, 2V
Plush drill	3	100.00%		6	50.00%	66.67%	100.00%	66.67%
Black pepper	3	100.00%		8	62.50%	62.50%	75.00%	100.00%
Dremel engraver	3	100.00%		6	33.33%	50.00%	66.67%	100.00%
Sand castle	3	100.00%		6	50.00%	33.33%	83.33%	83.33%
Purple ball	0	n.a.		6	66.67%	100.00%	83.33%	100.00%
White yarn roll	3	100.00%		8	87.50%	87.50%	87.50%	75.00%
Odor protection	0	n.a.		8	50.00%	87.50%	87.50%	75.00%
Neutrogena box	3	66.67%		8	25.00%	87.50%	87.50%	75.00%
Plush screwdriver	3	100.00%		6	83.33%	87.50%	83.33%	100.00%
Toy banana box	3	100.00%		8	100.00%	83.33%	87.50%	75.00%
Rocket	3	100.00%		8	50.00%	87.50%	100.00%	87.50%
Toy screw	3	100.00%		6	100.00%	100.00%	83.33%	100.00%
Lamp	3	100.00%		8	62.50%	83.33%	87.50%	87.50%
Toothpaste box	3	100.00%		8	87.50%	100.00%	87.50%	87.50%
White spray bottle	3	100.00%		8	25.00%	12.50%	75.00%	87.50%
White rope	3	100.00%		6	66.67%	83.33%	83.33%	100.00%
Whiteboard cleaner	3	100.00%		8	62.50%	75.00%	100.00%	100.00%
Toy train	0	n.a.		8	87.50%	100.00%	87.50%	100.00%
Vacuum part	3	100.00%		6	33.33%	66.67%	100.00%	83.33%
Computer mouse	0	n.a.		6	33.33%	33.33%	66.67%	83.33%
Vacuum brush	1	100.00%		6	50.00%	83.33%	66.67%	50.00%
Lintroller	3	100.00%		8	75.00%	75.00%	87.50%	100.00%
Ranch seasoning	3	100.00%		8	50.00%	75.00%	100.00%	100.00%
Red pepper	3	100.00%		8	75.00%	75.00%	100.00%	100.00%
Crystal light	3	100.00%		8	25.00%	37.50%	75.00%	75.00%
Red thread	3	100.00%		8	75.00%	100.00%	100.00%	100.00%
Kleenex	3	100.00%		6	33.33%	33.33%	83.33%	83.33%
Lobster	3	66.67%		6	16.67%	83.33%	66.67%	83.33%
Boat	3	100.00%		6	83.33%	100.00%	83.33%	100.00%
Blue spray bottle	2	100.00%		8	25.00%	50.00%	75.00%	62.50%
Average		**94.67%**			**57.50%**	**72.92%**	**85.00%**	**87.78%**

Table 4.1: Experimental results for isolated object grasping. Algorithm variations are denoted as: *A* for antipodal, *NC* for no classification, and *SVM* for the full algorithm. The number of views used to create point clouds is denoted as: *1V* for one view and *2V* for two views.

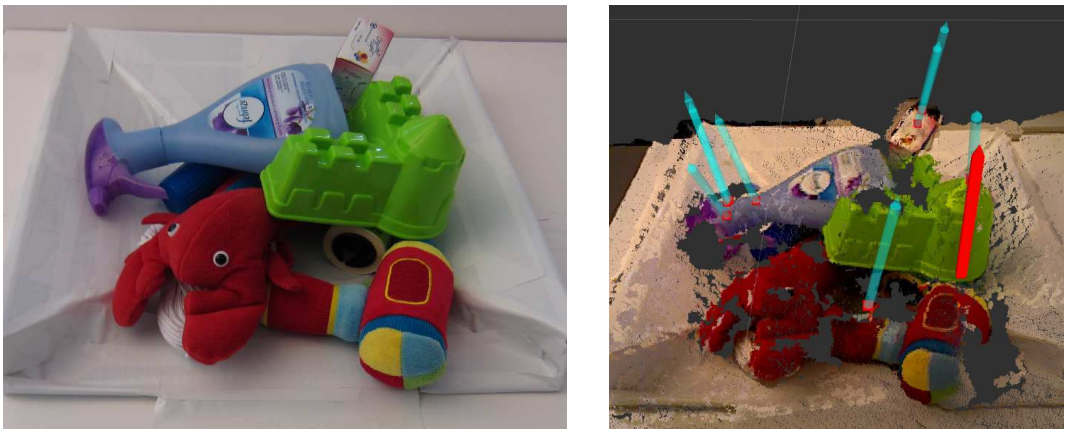
Figure 4.5: The set of objects which are hard to see for the depth sensor.

In a separate experiment, we investigated grasp performance for the seven "hard-to-see" objects depicted in Figure 4.5. For each object, we attempted grasps for eight different poses. Over the total 56 grasp attempts, we achieved a grasp success rate of 66.7% for SVM with a single-view point cloud and a 83.3% success rate for a two-view point cloud. These results indicate that our success rate drops by about 5% for objects which are difficult to see for the depth sensor, and that performance can be significantly improved by adding more sensors in non-perfect viewing conditions.

4.6.4 Dense Clutter Grasping

In this experiment, we evaluate our method in a dense clutter setting where multiple objects are located very close to each other. Here, we pile ten objects together in a shallow box. An example of such a scene is given in Figure 4.6a. A two-view point cloud was used for all scenes. The object set consists of 27 objects out of the 30 test objects depicted in Figure 4.3b. We excluded the engraver and the computer mouse because their cables can get stuck in the clutter. We also excluded the vacuum brush because the brush part cannot be grasped by the Baxter gripper in some configurations due to the aperture limits of the hand (3-7cm). At the beginning of each run, we randomly selected 10 out of the 27 objects and placed them in a small rectangular container. Then, we shook the container to mix up its contents and emptied it into the shallow box on top of the table. If the sandcastle was included in a run and it landed upside down, we repeated this process because the Baxter gripper cannot grasp the sandcastle in that configuration. If three detection failures happened in a row, we terminated the run. In total, we performed ten runs of this experiment.

The robot performed a total of 113 grasps over the ten runs of this experiment. On average, it succeeded in clearing 85% of the objects from the box. The remaining objects were not grasped because the system failed to detect grasp poses three times in a row. Over all grasp attempts, 73% succeeded. The four major failure modes were: (1) kinematic errors (3%), perceptual failures caused by our algorithm (9%), (3) collisions with the environment



(a) (b)

Figure 4.6: Example of a scene for dense clutter grasping. (a) RGB image of the scene. (b) The approach vectors of the grasps which were output by our algorithm.

(4%), and (4) objects dropped after an initially successful grasp (4%). Compared to the isolated object grasping experiment, the perceptual failure rate is higher. We suspect that a reason for this are the extensive occlusions caused by the clutter. As for the isolated object grasping experiment, to improve the grasp success rate beyond 73%, we would need to make improvements in other areas than perception.

4.7 Discussion

In this chapter, we proposed a new approach for the detection of 6-DOF grasp poses that can deal with both novel objects and objects presented in clutter. We contributed a method that uses the geometry of the robot hand and the object surface to be grasped to generate a set of grasp candidates. This method focuses grasp pose detection on areas of the search space where grasps are more likely to occur. We also contributed a method to automatically label a training set which enabled us to train a machine learning model to classify grasps in order to improve detection. Compared to previous methods from the literature, our method is able to detect 6-DOF robot hand poses from which a grasp is expected to be successful rather than detecting grasp points, which are typically only 3-DOF, in a height map or depth image, as has been proposed in [28, 64, 65]. Furthermore, automatically labeling grasp poses based on objective mechanical conditions for a stable grasp can reduce label noise compared to manual labeling by humans. Notice that our grasp candidate sampling method can be used as a proposal mechanism for any classifier. The high success rate that this sampling method obtained when used without any classification for grasping of isolated objects with a robot suggests that it should boost the performance of such classifiers. As a final remark, during robot experiments, we observed a decrease in grasp success rate from 87% for objects

presented in isolation to 73% for objects presented in clutter. This significant drop indicates that clutter is a challenging scenario that warrants further investigation. We address this issue in the upcoming chapter.

4.8 Related Work

Saxena, Driemeyer, and Ng were among the first to search an image for grasp targets without taking object identity into account [101]. They used a sliding window classifier to localize grasps based on local visual features. Later work from the same group extended this idea to range data [49] and subsequently proposed a deep learning approach [64, 65]. Fischinger and Vincze presented a similar method that uses height maps instead of range images and employs a Haar-like feature representation [28, 29]. Another type of approach searches a point cloud or range image for manually specified geometries which are expected to be associated with a successful grasp. Klingbeil et al. search a range image for a gripper-shaped pattern [58]. A different type of approach is template-based where grasps which have been demonstrated on a set of training objects are generalized to novel objects. Herzog et al. learn to choose a grasp from a library of templates based on features of the novel object [42]. Detry et al. model the geometry of local object shapes and fit these shapes to novel objects in order to grasp them [23]. Kroemer et al learn object affordances by matching shape templates against a number of actions which are afforded by those templates [62]. Another type of approach is based on the interaction with a stack of objects. Katz et al. interactively push objects to improve object segmentation in order to grasp novel objects [54]. Chang, Smith, and Fox physically manipulate objects to segment them [15].

In comparison to the literature described above, the approach presented in this chapter distinguishes itself because of the way in which we make use of geometric information. Our method to sample grasp candidates based on the robot hand geometry and the object surface in the vicinity of the grasp is novel. Additionally, generating large amounts of labeled training data in the way that we presented is important to further improve detection accuracy. With respect to experiments, we demonstrate a reasonably high success rate of 73% for objects presented in dense clutter which is a worst-case scenario for grasping. Out of the above works, only Fischinger and Vinze explored grasping in clutter. While they report a higher success rate of 86.7%, the scenes they evaluate are much less cluttered and have more structure than the ones in our own evaluation. With respect to isolated object grasping, our success rate of 87% on novel objects is similar to related methods. Lenz et al. report an 84% grasp success rate on Baxter and a 92% success rate on a PR2 [64], and Fischinger and Vincze report a 92% success rate on the PR2 [29]. Furthermore, our results are based on a much larger number of test objects than in related work.

Chapter 5

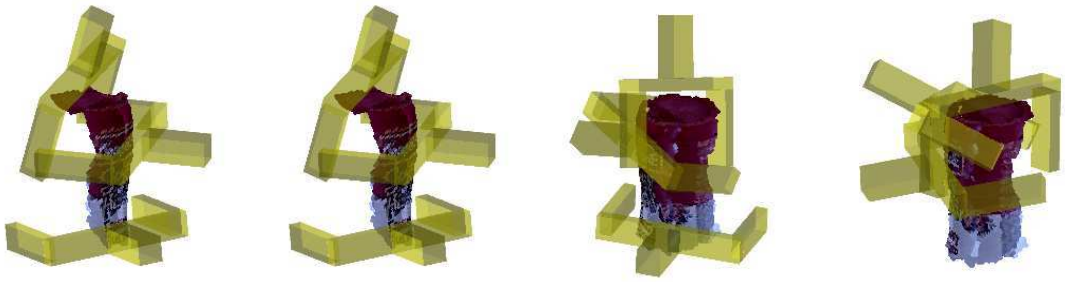
Deep Learning for Grasp Pose Detection

In this chapter, we present a supervised learning approach to grasp pose detection that uses convolutional neural networks (CNNs). Our method detects 6-DOF grasp poses which can be executed with two fingers that move along the same line, such as with a parallel jaw gripper. The presented method extends the one presented in the previous chapter by replacing the support vector machine with a CNN and introducing advanced grasp representations. We evaluate the grasp representations with respect to runtime and accuracy. Our robot experiments demonstrate high grasp success rates for objects presented in dense clutter. Our method is available as an open source C++ library, https://github.com/atenpas/gpd, accompanied by a ROS wrapper package: https://github.com/atenpas/gpd_ros.

5.1 Problem Statement

Our algorithm takes three inputs. The first input is a triple $\text{CV} = (C, \mathcal{V}, V)$, where C is a point cloud obtained from one or multiple view points, and $V : \mathcal{V} \rightarrow V$ is a mapping from viewpoints to points in C such that each point is paired with at least one viewpoint. The second input is a set of geometric parameters of the robot hand, θ. The third input is a subset of points $C_G \subset C$ that identifies which parts of the point cloud contain the objects to be grasped.

The output of the algorithm is a set of robot hand poses, $H \in \text{SE}(3)$, such that if the robot hand is moved to a hand pose in H and the robot fingers are closed, then a force closure grasp is expected to be achieved for some object in the scene. Here, we only consider the case where the robot hand is a 1-DOF two-finger parallel jaw gripper.



(a) (b) (c) (d)

Figure 5.1: Examples of grasp candidates found using the first step of our algorithm. Three examples of a gripper placed at randomly sampled grasp candidate configurations are depicted in each image.

5.2 Approach

Our grasp pose detection algorithm consists of two steps: (i) sampling grasp candidates, (ii) classifying grasp candidates as actual grasps. The first step is to sample a number of grasp candidates. Each candidate is a 6-DOF hand pose, $h \in \mathrm{SE}(3)$. First, we sample points uniformly at random from the point cloud. For each sampled point, we find a point neighborhood of a fixed radius and centered at the sampled point. We then calculate the surface normal of the point and an axis of major principal curvature of the object surface in the point neighborhood. Potential hand candidates are generated at regular orientations which are orthogonal to the major curvature axis. For each orientation, we evaluate hand candidates at a fixed set of positions along the closing direction of the robot hand. Out of the feasible position candidates, we choose the most centered one, and push the hand forward as much as possible without collisions with the point cloud. Once we have a configuration that is in contact with the point cloud, we check if there are any points in between the fingers. If there are none, the grasp candidate is discarded. Figure 5.1 visualizes grasp candidates found using this method.

The second step in our algorithm is to classify each grasp candidate as a grasp or not using a convolutional neural network (CNNs). CNNs are considered state-of-the-art for many classification tasks. To classify grasp candidates with a CNN, we need to decide what structure to use for the CNN and how to represent the geometry and appearance of the object region to be grasped. As the CNN structure, we use LeNet [63]: two convolutional layers, each followed by a max pooling layer, one inner product layer with a rectified linear unit at the output and one inner product layer with a softmax at the output. As the representation, we encode the object geometry in the vicinity of the grasp position as a multi-channel image.

5.3 Grasp Classification

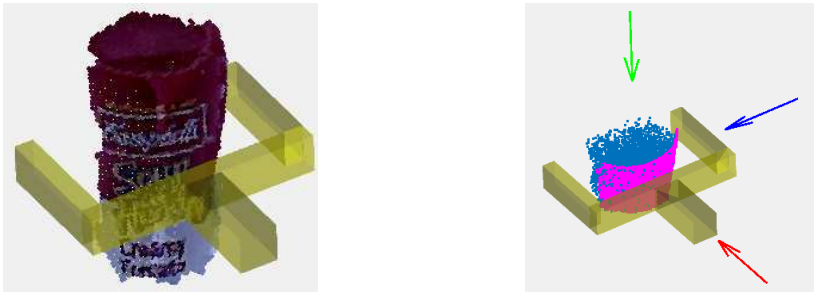
In this section, we first introduce a representation for the grasp such that it can be fed as input to the CNN that performs the classification. We then present a procedure to automatically generate ground truth labels for our grasp candidates. Pairing the representation with the labels, we arrive at a dataset that can be used to train the CNN.

5.3.1 Grasp Representation

To represent a grasp candidate to the classifier, we use the geometry of the observed surfaces and unobserved volumes which are contained in a region, $R \subset \mathbb{R}^3$, and swept out by the fingers are they close. We model the fingers as rectangles that move away or toward each other, the region of interest is a rectangular cuboid. Figure 5.2 illustrates our grasp representation. A grasp candidate generated based on a partial point cloud is depicted in Figure 5.2(a). The sets of points associated with this candidate are shown in Figure 5.2(b). One set of points are points in the point cloud which are contained in the region R, colored in magenta. The other set of points are sampled from the portion of R that is unobserved, i.e., that is occluded from view by every sensor, colored in blue.

For a cuboid region R, we scale the points to the unit cube and voxelize them to a grid of a fixed size of $60 \times 60 \times 60$ voxels. For every triple, $(x, y, z) \in [1, 60] \times [1, 60] \times [1, 60]$, $V(x, y, z) \in \{0, 1\}$ denotes if the corresponding voxel is occupied and $U(x, y, z) \in \{0, 1\}$ denotes if the corresponding voxel has been observed. We calculate, for each occupied voxel in R, a unit, outward-pointing surface normal vector, $\hat{n}(x, y, z) \in S^2$, that denotes the orientation of the object surface at that point. This information can be calculated from the point cloud.

An ideal representation for the classifier's input would be the 3D geometry of the object surface that is contained between the fingers of the robot hand. However, this would require to pass a large number of voxels to the CNN. Instead, we project the voxels onto planes



(a) (b)

Figure 5.2: (a) A grasp candidate sampled for a partial point cloud. (b) Local hand frame: red is the approach axis, blue is the closing axis, and green is the curvature axis.

(a) (b) (c)

Figure 5.3: Examples of our grasp representation as an image. (a) Averaged height map of the occupied points. (b) Averaged height map of points sampled from the unobserved region. (c) Averaged surface normals for the occupied points.

orthogonal to the standard basis axes of the voxel grid and pass these to the CNN as input. These planes are parallel to the arrows shown in Figure 5.2(b). The x coordinate in the voxel grid V goes along the hand approach vector, shown as a red arrow, the y coordinate goes along the hand closing direction, shown as a green arrow, and the z coordinate goes along the axis of major principal curvature of the object surface, shown as a blue arrow.

For each of these three projections, we calculate three images: an averaged height map of the occupied points, I_o, an averaged height map of the unobserved region, I_u, and averaged surface normals, I_n. For example, to project onto the plane constituted by the hand and curvature axes, i.e., the (x,y) plane, these maps are calculated as follows:

$$I_o(x,y) = \frac{\sum_{z\in[1,60]} zV(x,y,z)}{\sum_{z\in[1,60]} V(x,y,z)} \tag{5.1}$$

$$I_u(x,y) = \frac{\sum_{z\in[1,60]} zU(x,y,z)}{\sum_{z\in[1,60]} U(x,y,z)} \tag{5.2}$$

$$I_n(x,y) = \frac{|\sum_{z\in[1,60]} \hat{n}(x,y,z)V(x,y,z)|}{\sum_{z\in[1,60]} V(x,y,z)} \tag{5.3}$$

The first two images, I_o and I_u, have a width and height of 60 and one channel. The last image, $I_n(x,y)$, has a height and width of 60 and three channels. The three dimensions of the normal vector make up these three channels. In total, we have 15 channels of geometric information: five channels for each of the three projections. An example for these three images is given in Figure 5.3.

5.3.2 Generation of Training Labels

To train the classifier for the second step in our algorithm, we need to generate a large amount of training data. In particular, we need a set of instances where each instance is a pair that

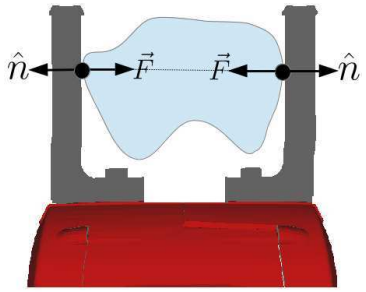


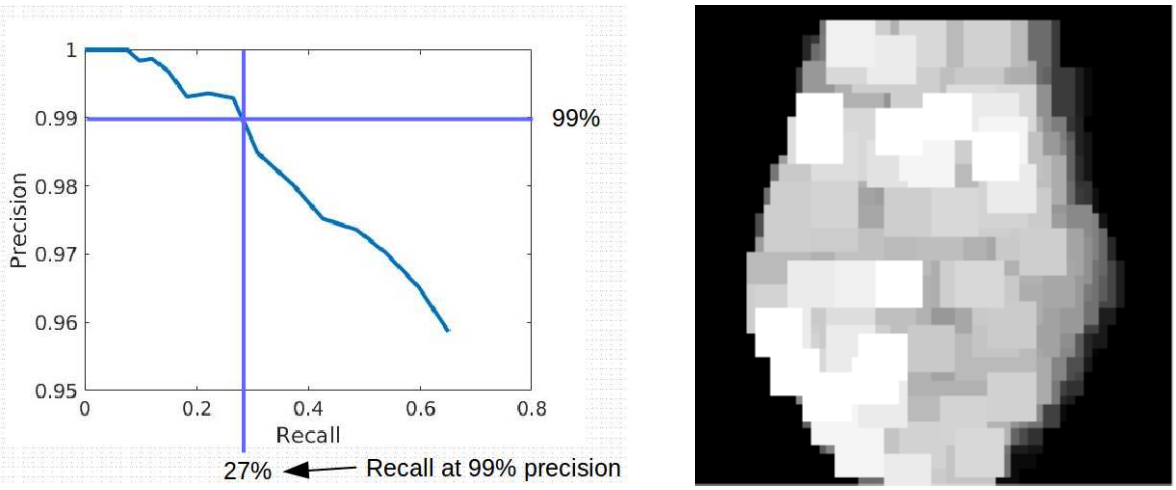
Figure 5.4: The conditions for a frictionless antipodal grasp: contact surface normals are anti-parallel to each other and co-linear with the line that connects the contacts.

consists of a representation of the appearance or geometry of the object surface relative to a hand pose and a label that indicates if a grasp exists at that hand pose. To generate training data, we first use the grasp candidate sampling method described in Section 5.2 to generate a large set of grasp candidates. For each candidate, we then evaluate if a frictionless, antipodal grasp would be formed by closing the fingers on the object mesh from the hand pose associated with the candidate. In the case of a two-finger parallel jaw gripper, a frictionless antipodal grasp means that each of the object surface normals at contact is anti-parallel with the direction in which the contacting finger closes and co-linear with the line that connects the contacts [81]. These conditions are illustrated for the Baxter robot gripper in Figure 5.4. For any positive coefficient of friction, a frictionless antipodal grasp is in force closure [80], i.e., it can resist external disturbances.

To generate the training data, we need a set of object meshes and point clouds where each cloud is registered to a mesh. This data can be obtained from synthetic object sets such as 3DNET [113] where point clouds are generated by simulating a depth sensor or from point clouds acquired by a physical depth sensor which are used to reconstruct a mesh of the object, such as BigBird [104]. However, object meshes are often noisy which makes it difficult to determine whether an antipodal grasp exists. We address this problem by slightly relaxing the antipodal condition. In particular, we make the assumption that each vertex of the object mesh is subject to a small amount of position error and evaluate whether an antipodal grasp exists under any perturbation of the vertices. We identify small regions in each finger where contact may be established and check if the frictionless antipodal condition described above holds for any pair of contacts in these regions.

5.4 Simulation Experiments

In this section, we evaluate our algorithm in simulation. First, we compare different grasp representations in terms of grasp classification accuracy. Second, we measure the runtime of our algorithm. These experiments let us decide which variation to use for robot experiments.



(a) (b)

Figure 5.5: Example of recall-at-high-precision (RAHP). (a) Precision-recall plot. (b) Grasps recalled at 99% precision for a shampoo bottle instance.

5.4.1 Evaluation Metrics

Evaluations of many grasp detection methods in the literature have focused on grasp classification accuracy [28, 29, 23, 42, 62, 52, 85], i.e., the ratio of the number of correct predictions to the total number of predictions. Under this evaluation criterion, methods from the literature achieve accuracies in the range of 75% to 95%. However, accuracy is not a good indicator of the actual grasp success rate. A better way of evaluating a grasp detector is to look at precision and recall. Precision is the ratio of correct positive predictions (true positives) to the total number of positive predictions (the sum of true positives and false positives). A false positive prediction has a high cost with respect to grasping because it can cause a grasp attempt to fail. Recall is the ratio of true positives to the total number of actual positives.

We are interested in characterizing *recall-at-high-precision* (RAHP) because precision is much more important for a grasp detector than recall. We want to avoid false positive predictions because they come at a high cost as described above. Furthermore, since grasp detectors can predict hundreds of grasps for a single object, we do not need to recall all of the grasps to have a diverse set of grasps from which we can choose one to be executed by a robot. Figure 5.5 illustrates this metric for a particular instance of a shampoo bottle, where our method is able to recall 27% of all the grasps at 99% precision (see Figure 5.5a). As illustrated by Figure 5.5b, this low recall value still gives us plenty of grasp choices.

5.4.2 Comparison between Grasp Representations

The accuracy of the grasp classifier depends on how the grasp candidates are represented to the CNN that learns to classify them. Here, we compare the following four representations.

- **Representation #1**: The 15-channel representation described in this chapter. For each of the three projections, there are three channels of averaged surface normals, one channel for the average height map of the observed points, and one channel for the averaged height map of the unobserved region.
- **Representation #2**: A 12-channel ablation of the 15-channel representation, where we remove the three channels for the unobserved region, i.e., remove one I_u channel for each of the three views.
- **Representation #3**: The 3-channel representation used in the previous chapter. This representation consists only of the I_n channels projected onto the plane orthogonal to the negative z axis of $F(h)$. Whereas we encode this 3-channel image using HOG features in the previous chapter, we here give the image directly to a CNN.
- **Representation #4**: The 3-channel representation used in both Herzog et al. [42] and Kappler et al. [52]. This representation consists of three channels of information projected along the hand approach axis. The first channel is I_o, the second channel is I_u, and the third channel is a description of the unoccupied voxels in the space $I_f = I_u \cup I_o$.

To compare the above representations, we created a dataset of grasp candidates that is equally balanced between positives and negatives. For each of the 55 BigBird objects, we randomly sampled about 4,000 grasp candidates. In total, we obtained 216,000 grasp candidates and divided them in a training and test set with 185,000 and 31,000 instances, respectively. The split into the two sets was done by views, i.e., the test set contains only instances from views which were not used for the training set. For each of the above representations, we train the CNN from scratch, with random initial weights. We then measure classification accuracy on the test set.

Figure 5.6 shows the results of this comparison. The green curve shows the accuracy of the 15-channel representation and the blue curve shows the accuracy of 12-channels representation. The occlusion information gives about a 2% increase in accuracy. However, it is computationally expensive to compute as it takes approximately twice the time to encode the grasp candidates for classification. The red curve shows the accuracy of the 3-channel representation proposed in the previous chapter. Notice that it performs about as well as the 12-channel representation even though it contains a much smaller amount of channels of information. This representation is much faster to compute than the 15-channel representation, actually by a factor of four (see Table 5.1). The accuracy obtained by the 3-channel representation used in both Herzog et al. [42] and Kappler et al. [52] is shown in cyan. While

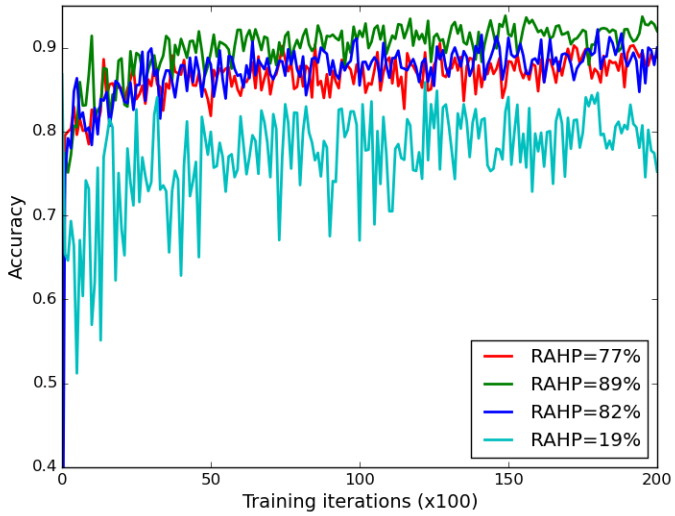


Figure 5.6: Comparison of classification accuracy between four different grasp candidate representations (colored curves). Green: 15-channel representation. Blue: same as green but without the occlusion channels. Red: the representation used in our prior work [85]. Cyan: the representation used in both Herzog et al. [42] and Kappler et al. [52]. The legend shows the recall-at-high-precision (RAHP) metric for each of these representations at 99% precision.

we use Herzog's representation, we evaluate on grasp candidates which were generated generated using our own method. Notice that, on average, Herzog's representation obtains at least 10% lower accuracy than the other representations. The reason for this loss in performance could be that this representation projects onto the plane orthogonal to the surface normal rather than the projection from Representation #3. Another potential reason is that this representation does not encode surface normals.

The 15-channel representation outperforms all other representations. If the main objective is to maximize the accuracy of grasp detection, then this representation should be used. However, it is also the most computationally expensive representation. Therefore, the 3-channel representation is a better choice if reducing runtime is the main objective because it has almost the same performance as the 12-channel representation but is much faster to compute. Compared to the 3-channel representation from Herzog et al. [42] and Kappler et al. [52], our method should be preferred because it achieves much better grasp classification accuracy.

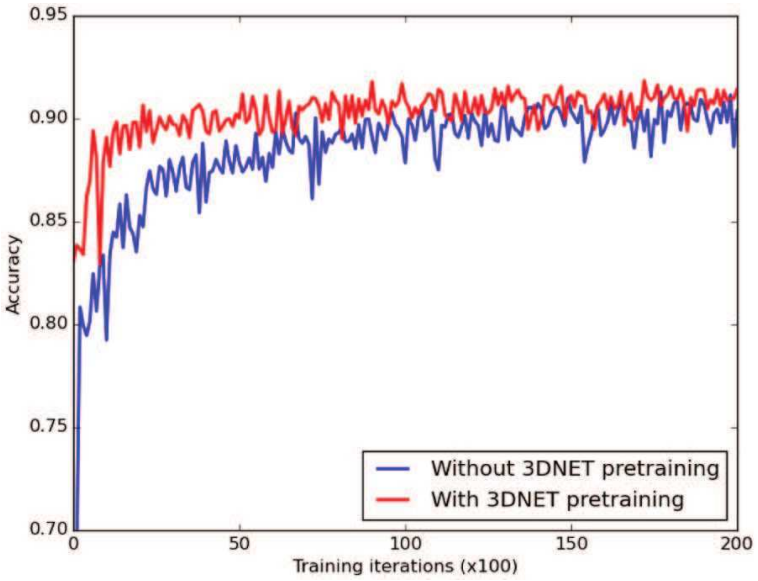


Figure 5.7: Classification accuracy with (red) and without (blue) 3DNET pretraining, evaluated on BigBird objects.

5.4.3 Pretraining on Simulated Data

To improve classification accuracy, we can pretrain our network on a synthetic dataset. A synthetic dataset consits of point clouds created by simulating a sensor that observes a CAD model. Such datasets are much simpler to create than real world datasets. There is a very large number of CAD models available online. Here, we consider the 3DNET dataset that contains models from 200 object categories [113]. First, we trained the CNN on 400 CAD models taken from 16 categories in 3DNET. We then measured the accuracy of this network on our test set for the BigBird objects. Second, we finetuned the CNN on the BigBird training set. Finally, we evaluted the performance of the finetuned CNN on the BigBird test set.

The learning curves for the pretrained and the non-pretrained network using the 15-channel grasp representation are presented in Figure 5.7. Initially, the pretrained network has a strong influence on the accuracy. At 4,000 iterations, it obtains the same accuracy as the non-pretrained network at 20,000 iterations. Nevertheless, over time, both networks converge to the same performance.

5.4.4 Using Prior Object Knowledge

To improve classification accuracy, we can incorporate prior knowledge about the object to grasped. One strategy to accomplish this is to change the contents of the training set. A large and diverse set of objects for the training set might be a good choice if we have no

prior knowledge of the object to be grasped because we expect that the learned model is able to generalize to unknown objects. If we have prior knowledge of the category of the target object, e.g., we know that the object is similar to a cylinder, we might use training data from cylindrical objects to train the grasp detector. If we have prior knowledge of the exact target object, we might use training data only from that particular object to train the detector. Generally, we expect that the more of such prior knowledge is incorporated int the network by training in this way, the better the classification accuracy will be. Object information could be given by standard object detection methods, as described later in Section 5.6.

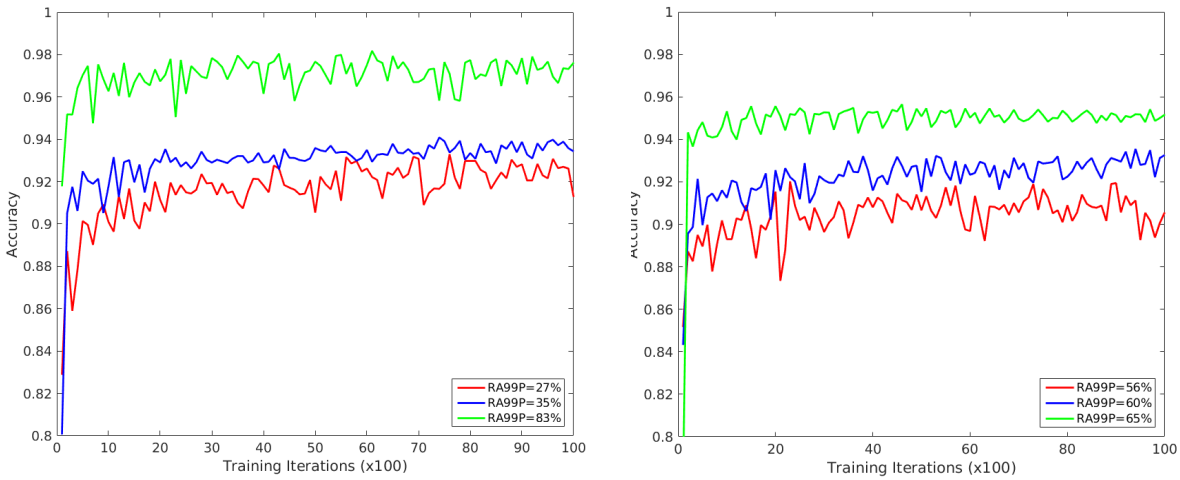
To characterize the impact of prior object knowledge as described above, we performed an experiment with two object categories which are part of the BigBird object set: (i) 29 objects with a shape similar to a box, and (ii) 16 objects with a shape similar to a cylinder. For each of these two categories, we compared classification accuracy in the three different prior knowledge scenarios: (a) no prior knowledge, (b) known object category, and (c) known object instance. In all cases, we first pretrained a network on 3DNET. For the no prior knowledge scenario, we trained the classifier on all other objects from BigBird. For the known object category scenario, we trained the classifier using a leave-one-out strategy. For the object instance scenario, we trained the classifier using data that is derived from only the single object to be grasped. We then evaluated the classification accuracy for each scenario on a test set of unseen views.

Figure 5.8 shows the results of this experiment. The ordering of the curves indicates that our expectation of more prior knowledge resulting in better performance is correct. With no prior knowledge, the performance is worst, it gets better with knowledge of the object category, and it is best with knowledge of the object identity.

5.4.5 Algorithm Runtime

Since algorithm runtime is critical for real world applications, we have implemented our algorithm in C++ as a ROS package [1]. Both grasp candidate generation and classification are parallelized in our implementation. We implemented our neural network using Caffe [48]. We measured the runtime of the two main steps in our algorithm, i.e., grasp candidate sampling and grasp classification for both the 15-channel representation and the 3-channel representation. These runtime evaluations were run on an Intel Core i7-4770K Haswell Quad-Core 3.5GHz with 32GB of RAM and one NVIDIA GTX 970 GPU. We report the runtime to perform these steps for 1000 grasp candidates. Out of these 1000 candidates, we typically find that the classifier is able to identify a few high-confidence grasps. However, times scale approximately linearly with candidates, so it is possible to detect additional grasps by increasing the time spent. The point clouds for this experiment were derived from ten different cluttered scenes similar to the one shown in Figure 5.10c. We evaluate the runtime for both single- and two-view point clouds.

[1] github.com/atenpas/gpd



(a) (b)

Figure 5.8: Grasp classification accuracy when using prior object knowledge. (a) Results for cylindrical objects. (b) Results for box-like objects. The red line corresponds to no prior knowledge of the objects, the blue line to prior knowledge about the object category, and the green line to prior knowledge about the object instance.

Representation	**Number of points**	**Sample time**	**Classification time**
15-channel	39,000	0.8s	4.3s
15-channel	66,000	1.7s	6.2s
3-channel	39,000	0.8s	0.3s
3-channel	66,000	1.7s	0.4s

Table 5.1: Runtime averaged over ten different dense clutter scenes. Columns 1-3 contain averages.

Table 5.1 shows the runtime results averaged over the ten scenes. There are four different contingencies. The rows for which the second colum says 39,000 points are for one-view and the rows which say 66,000 points in the second column are for two-view point clouds. The first two rows measure runtime for the 15-channel and the last two rows measure runtime for the 3-channel representation. The results indicate that our algorithm can be applied to real world applications because it can be expected to require between one and eight seconds to run. Furthermore, the 3-channel version is much faster to compute than the 15-channel version. In particular, the former can process the 66,000 points point cloud in two seconds whereas the 15-channel version requires eight seconds. As shown in Figure 5.6, using the 3-channel representation instead of the 15-channel representation reduces grasp classification accuracy by roughly 2%. However, the 3-channel version is four times faster than the 15-

channel version. For this reason, we used the 3-channel representation in the robot grasping experiments which are reported in the next section. To counteract the lower accuracy, we raise the acceptance threshold for the classifier. While this decreases recall, it increases the number of high quality grasps which are produced by the algorithm.

5.5 Robot Experiments

To evaluate the performance of our GPD algorithm on a physical robot, we introduce a benchmark for object grasping in dense clutter. Given a set of objects located in a tray, the task that we want the robot to perform is to clear all objects from the tray by grasping them one-by-one and transporting them to another box where they are deposited.

5.5.1 Grasp Selection

Similar to the algorithm presented in the previous chapter, the algorithm presented here typically outputs hundreds of grasp poses out of which we need to choose one to execute on the robot. First, we eliminate grasps which are not feasible for the chosen setting of the fingers on the Baxter gripper, i.e., we eliminate grasps which are outside of the 3-7cm range. Second, we use IKFast to calculate IK solutions and OpenRAVE [26] to detect collisions. We eliminate grasps for which there are no collision-free IK solutions. After eliminating these infeasible grasps, we rank them by heuristics based on the dense clutter grasping scenario. First, there is a clear advantage to grasping objects at the top of the pile because it avoids collisions with the bottom of the pile. Similarly, grasps which approach the object from the top are less likely to collide with the clutter than grasps which approach the object from the side. Third, we prefer grasps whose position is closer to the robot because a smaller number of hand orientations can be reached by the robot if the grasp position is far away from the robot. We use these three criteria to constitute a cost function as follows. Let $z(h)$ denote the height of hand h along the gravity axis above the table. Let $\hat{a}(h)$ denote a unit vector that points in the approach direction of the hand. Let $\delta_q(h)$ denote the Euclidean distance in configuration space between the arm configuration at h and some nominal starting configuration. We use the following cost function:

$$J(h) = l\left(0.5(\hat{a}(h)^T\hat{g}+1)\right) l\left(\frac{\nabla q(h)}{\nabla q_m}\right) l\left(\frac{z_m - z(h)}{10 z_m}\right), \tag{5.4}$$

where $l(x) = \max(1-x, 0)$ denotes a hinge loss, z_m denotes the maximum height of a grasp above the table, $\hat{g}$ denotes a unit vector pointing in the direction of gravity, and ∇q_m the maximum distance that the robot hand can reach. We select the grasp that maximizes this cost function.

(a) (b) (c)

Figure 5.9: (a) Baxter in a typical dense clutter test scenario. (b) We mounted a depth sensor (Structure IO) to the robot hand to enable us to obtain more complete point clouds using SLAM software. (c) Dense clutter benchmark task scenario superimposed with an illustration of three configurations along the trajectory taken by the arm during active sensing.

5.5.2 Hardware Setup

We use the right 7-DOF arm of the Baxter Research Robot in the experiments (see Figure 5.9). Our robot hand is the off-the-shelf Baxter parallel-jaw gripper with the short fingers and square pads. The square pads were modified with a black rubber covering, and rubber-covered pieces of metal were added to the ends, as depicted in Figure 5.9b). These ends can bend slightly outward to initially widen the bite. This helped with minor, sub-centimeter kinematic or point cloud registration errors. This gripper is restricted to a 3 to 7cm width. Each object in the test set was selected given this restriction. We mounted two Asus Xtion Pro depth sensors to Baxter's waist, as shown in Figure 5.9a and an Occipital Structure IO sensor to the robot's wrist, as shown in Figure 5.9b.

We used two computer systems in the experiments. Each system consisted of a 3.5 GHz Intel Corei7-4770K CPU with four physical cores, 32 GB of system memory, and an Nvidia GeForce GTX 660 graphics card. One system was used to run our GPD algorithm, and we used InfiniTAM [50] on the other system to obtain a truncated signed distance function volume from the wrist-mounted sensor while moving the robot arm (see Figure 5.9c. The robot operating system (ROS) handled communication between the robot and the two computers.

5.5.3 Dense Clutter Benchmark

The dense clutter benchmark is a benchmark and protocol for evaluating robotic grasping systems. Figure 5.10 illustrates the task and Table 5.2 outlines the experimental protocol for this benchmark. The first step is to select ten objects uniformly at random from the set

1.	Randomly select 10 objects from the object set.
2.	Place objects into a box.
3.	Shake box until sufficiently mixed.
4.	Pour box contents into tray in front of robot.
5.	Run grasp algorithm.
6.	Terminate once any of these events occur: i) No objects remain in tray. ii) No grasp hypotheses were found after 3 attempts. iii) The same failure occurs on the same object 3 times.

Table 5.2: Experimental protocol for the dense clutter benchmark.

(a) (b) (c)

Figure 5.10: Dense clutter benchmark task. (a) Ten objects are selected at random from a set of 27 objects. (b) Box contents are poured into the tray. (c) Contents of the tray immediately after pouring.

of 27 objects show in Figure 5.10a. These 27 objects are common household items that are different from the 55 BigBird objects used for training our algorithm. The second step is to place the ten objects in a box. The third step is to shake the box to mix the objects. The fourth step is to pour the contents of the boxinto a tray placed in front of the robot on a table, as illustrated in Figure 5.10b. The fifth step is to let the robot grasp as many objects, one at a time, as it can. The robotic grasping continues until either no objects remain in the tray, the algorithm has run three times and no grasp poses were produced, or the same failure occurs on the same object three consecutive times.

The above protocol is repeated for a number of trials. For each grasp attempt, we record if the grasp was successful, and if not, we report the failure mode. Based on this data, we calculate the average grasp success rate and the average object removal rate.

5.5.4 Point Cloud Acquisition Strategies

To acquire a point cloud of the objects to be grasped, we explored two different strategies: a passive strategy and an active strategy. The passive strategy used the two Asus Xtion Pro depth sensors mounted to the body of the robot as shown in Figure 5.9a. We precisely calibrated the pose of each sensor in the robot's base frame and used this information to register the two point clouds obtained from the sensors together. The active strategy used a Structure IO depth sensor that we mounted near the end effector (see Figure 5.9b) to creates a point cloud with InfiniTAM [50], an off-the-shelf SLAM software. To acquire a point cloud with this strategy, we moved the arm through a collision-free trajectory on a hemisphere centered on the object pile while tracking using InfiniTAM. The trajectory is a 46cm geodesic between a fixed via point on one side of the pile and a fixed via point on the other. Based on the minimum range of the Structure IO sensor of approximately 35cm, we set the radius of the hemisphere to 40cm. The trajectory was generated using TrajOpt [103] is used to generate the trajectory where the sensor is constrained to always point directly towards a fixed point above the cluttered tray.

5.5.5 Results: Grasping in Dense Clutter

The results of this experiment are given by Table 5.3. We evaluated four contingencies: active point cloud, passive point cloud, no selection strategy, and no classification. All contingencies use a CNN that was trained on all 55 BigBird objects in our dataset. Grasp candidates were encoded using the 3-channel representation. Out of the 27 objects shown in Figure 5.10a, there were two objects, the sandcastle and the red lobster, which could not be grasped by the Baxter gripper when they came up in an upside down configuration. This is due to the minimum aperture of 3cm that we set the fingers of the Baxter gripper to. If such a case happened, we removed that object from the pile and placed it right side up back on the pile.

The active point cloud contingency obtains point clouds by running InfiniTAM while moving the wrist-mounted depth sensor above the tray. We used the grasp selection heuristics described above to select grasps. Over 30 rounds of the dense clutter benchmark task, with 10 objects per round, we observed a 93% grasp success rate (20 grasp failures out of 288 grasp attempts). Out of the 20 failures, 5 were due to point cloud registration errors or inaccuracies

	Active Point Cloud	Passive Point Cloud	No Selection Strategy	No Classification
Num objects	300	150	150	150
Num grasp attempts	288	138	155	142
Num grasp successes	268	116	117	75
% grasps successful	93%	84%	75%	53%
% objects removed	89%	77%	78%	50%

Table 5.3: Results of the dense clutter grasping experiment.

in the kinematic calibration of the robot, 9 were due to a failure of the algorithm to select what appeared to be a good grasp, 4 were due to a collision of the fingers with the object before the grasp, and 2 were due to the object dropping out after an initially successful grasp. In this scenario, 89% of the objects were cleared from the tray.

For the other three contingencies, we ran 15 rounds each of the benchmark task. The point cloud for the passive point cloud contingency was obtained using two Asus Xtion Pro sensors fixed to the body of the robot. We observed an 84% grasp success rate (22 grasp failures out of 138 grasp attempts). Of the 22 failures, 5 were due to point cloud registration errors or inaccuracies in the kinematic calibration of the robot, 13 were due to a failure of the algorithm to select what appeared to be a good grasp, 2 were due to a collision of the fingers with the object before the grasp, and 2 were due to the object dropping out after an initially successful grasp. In this scenario, 77% of the objects placed in front of the robot were cleared. Failures to clear objects were caused by objects which were moved out of the tray due to collisions, pushed too far forward to be visible by the sensors, or were located too close together to find a grasp. The reduction in performance for this contingency compared to the previous one can be attributed specifically to two factors: (a) the objects are covered better by the point cloud generated by the first contingency, and (b) some objects near the front of the tray are barely visible in the point cloud that is created with the second strategy.

The no selection strategy contingency omits the heuristics used to select a grasp. We still eliminated grasps which had no IK solutions, were in collision, or did not satisfy the constraints of the Baxter gripper aperture. However, this contingency selects grasps uniformly at random instead of ranking the remaining grasps. We observed a 75% grasp success rate (38 grasp failures out of 155 grasp attempts). Out of the 38 failures, 9 were due to a failure of the algorithm to select what appeared to be a good grasp, 8 were due to collision of the fingers with the object prior to forming a grasp, 10 were due to point cloud registration errors or inaccuracies in the kinematic calibration of the robot, and 11 were due to the object dropping out after an initially successful grasp. Compared to the first contingency, there were many grasps where the robot dropped the object after an initially successful grasp. These errors were likely happening because of inaccurate kinematic calibration or a collision prior to the grasp. This suggests that the grasp selection heuristics are useful for avoiding such errors.

The no classification contingency omits the grasp classification step. Once grasp candidates have been sampled, we used the grasp selection strategy to select one of the candidates. This contingency implicitly assumes that all sampled candidates are antipodal grasps. We observed a 53% grasp success rate (67 grasp failures out of 142 grasp attempts). Out of the 67 failures, 46 were due to a failure of the algorithm to select what appeared to be a good grasp, 6 were due to collision of the fingers with the object before the grasp, 14 were due to the object dropping out after an initially successful grasp, and 1 was caused by a point cloud registration error. These results indicate that it is important to identify which grasps are actually antipodal. If we do not eliminate grasp candidates which are not force closure, then the algorithm produces a larger number of grasp poses which result in failures because they correspond to poor grasp candidates. Furthermore, this contingency produces many

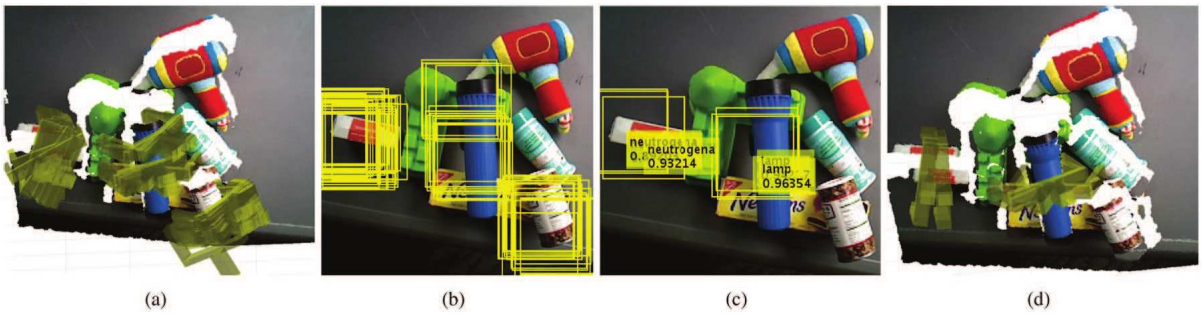


Figure 5.11: Illustration of combined grasp and object detection: (a) a set of grasps detected in a scene; (b) the corresponding object proposals; (c) high-confidence object detections; (d) grasps corresponding to detected objects. For either of the two detected objects, the robot has the option to execute one of two detected grasps.

poor grasp poses which cause a failure where the object is dropped during lifting.

5.6 Combined Object and Grasp Detection

While our approach ignores object identity, it is important to know the object of interest for many manipulation tasks. For example, in the traditional approach to grasp detection, where a CAD model is registered to a point cloud in a 6-DOF pose space, it is essential to determine the identity of the object to be grasped. Typically, we do not want the robot to grasp an abitrary object, but a particular one. Therefore, we should combine grasp detection with object detection.

Here, we propose a simple method to combine the two. Assume that we have a point cloud and an RGB image that is registered to the point cloud. Also assume that we have an object recgonition model that has already been trained on the objects. First, we detect a set of potential grasps. Second, we calculate a point that lies between the fingers and project this point into the RGB image. Third, we create a set of object proposals which consists of a fixed-size bounding box of 120x120 pixels whose center is the projected point.

This process is illustrated in Figure 5.11. In Figure 5.11(a), 68 grasps were detected. For each grasp, the corresponding object proposal in the RGB image is depicted in Figure 5.11(b). Our use of the grasp detector as an object proposal mechanism focuses attention on graspable objects in a way that an off-the-shelf solution such as selective search [106] would not. The top scoring object proposals which achieved a minimum threshold score are shown in Figure 5.11(c). Here, we obtained labels of high confidence for the white and red box on the left ("neutrogena") and the flashlight in the middle ("lamp"). The grasps corresponding to these object detections are depicted in Figure 5.11(d). The result of this process is a set of predictions of identity for each object that our GPD algorithm is able to grasp. We can therefore now select an object to grasp as desired.

Figure 5.12: The 11 objects used to evaluate combined object and grasp detection.

5.6.1 Network Training

In order to recognize objects, we trained the BVLC reference network [48] on the eleven objects shown in Figure 5.12. For each object, we first placed it by itself on a table in front of the robot. Then, we collected a sequence of depth images from different viewpoints by using a depth camera mounted to the end of our robotic arm. We repeated this process three times for each object, each time placing the object on the table in a different orientation. In total, we obtained about 2,000 images per object. We augmented this dataset by rotating and scaling. The result is a dataset of approximately 10,000 images per object, for a total of about 116,000 images. From this dataset, we used 75% for training and 25% for testing. We used a version of the reference network that was pretrained on ImageNet [61], and finetuned it on our eleven objects over 20,000 iterations of stochastic gradient descent with a learning rate of 0.000001, momentum of 0.9, weight decay of 0.0005, and where minibatches which contained 128 images each. After training, we obtained more than 99% accuracy on the test set. This 99% accuracy can be interpreted as the expected performance of the object classifier in a single-object grasping scenario.

5.6.2 Dense Clutter Experiments

To characterize the performance of our approach in dense clutter, we created a set of nine cluttered scenes which involve nine of the eleven objects. We excluded the screwdriver because it was difficult to grasp with the Baxter gripper and the rocket because it was nearly indistinguishable from the link roller handle for the 120x120 window size that we used. For each of the nine scenes, we acquired three point clouds, for a total of 27 point clouds. We used GPD to create a set of object proposals for each point cloud, for a total of 1197 object proposals. We manually labeled all proposals and then evaluated object classification accuracy for this set. Our system predicted the correct object in 1023 out of the 1197 proposals (85.4% accuracy). Table 5.4 shows the corresponding confusion matrix.

	A	B	C	D	E	F	G	H	I
A	17	0	0	0	0	0	0	0	0
B	0	18	0	0	5	0	1	0	0
C	0	0	136	0	14	0	0	0	0
D	1	0	0	228	0	0	0	0	0
E	0	0	0	0	206	0	3	0	15
F	8	4	28	0	35	150	3	0	16
G	0	0	0	0	0	0	76	0	0
H	0	0	8	0	31	1	0	136	1
I	0	0	0	0	0	0	0	0	56

Table 5.4: Confusion matrix for the nine objects used to test combined grasp and object detection. A: green castle, B: plush drill, C: Fig Newtons box, D: blue flashlight, E:lintroller, F: Neutrogena box, G: ranch dressing bottle, H: red pepper bottle, I: rocket.

The primary reason why object classification accuracy decreases in clutter relative to the single-object scenario is that the object proposals sometimes contain adjacent objects in addition to the objects that would be grasped. If segmentation was used to remove these adjacent objects from the proposals, we expect that it would be possible to improve these results.

5.7 Discussion

In this chapter, we proposed a new approach for the detection of 6-DOF grasp poses that can deal with both novel objects and objects presented in clutter. Compared to the method presented in the previous chapter, we replace the support vector machine with a convolutional neural network that can obtain state-of-the-art classification accuracy given images as input. Our results show that the choice of grasp representation is important because a different representation can improve classification accuracy by up to 10%.

Our robot experiments suggest that it is important to have a point cloud created by obtaining views from multiple different positions. The grasp success rate is about 9% larger for a point cloud obtained with a metric SLAM software from a trajectory that tracks the object from locations on a hemisphere than it is for one obtained by sensors which are located at fixed positions on the robot. In addition, our robot experiments indicate that the grasp selection strategy is important. Even simple heuristics to select a grasp, such as selecting the one that is highest in position relative to the table, can improve the grasp success rate significantly, bu up to 20%. Finally, grasp classification is extremely important because without it, we do not know which grasps are expected to be antipodal and the grasp success rate drops significantly, by up to 40%.

The object detection method presented in this chapter could be used to improve grasp detection by training a neural network by using data from a specific object instance or

category of interest. Given a novel scene, we might generate grasp candidates for all visible objects and then predict object identity on an instance or category level for each grasp. Given this prior, we would then use the appropriate classifier to detect grasps on the object. This approach could be particularly useful in scenarios where the goal is to grasp an object based on its category label. In this case, grasp candidates which wwere not predicted to have the desired category label could be eliminated and the grasp classification could focus on the remaining candidates.

5.8 Related Work

Compared to other robot grasping approaches, grasp detection attempts to directly detect local grasp surfaces from sensor data rather than first detecting objects and then planning grasps using that information. This idea originates from Saxena et al. who used machine learning to detect grasp points in an image given a corpus of hand-labeled training data [101]. In the following three sections, we elaborate on related work according to a categorization of grasp detection approaches into three types: 3DOF grasp detection, grasp templates, and kernel density estimation based approaches.

5.8.1 3-DOF Grasp Detection

Several grasp detection methods detect grasps in a three-dimensional space, e.g., the space of x, y, θ configurations in an image, where x is a horizontal coordinte and y is a vertical coordinate in the image and θ is the orientation about the grasp approach axis. For example, Jiang et al. model a graspable geometry as an oriented rectangle in an RGBD image [49]. To perform such a grasp, the gripper must approach the grasp target from a direction roughly orthogonal to the image. Several other approaches fall into this category as well, including [64, 90, 96, 29]. In particular, Redmon and Angelova [96] use the same dataset as Lenz et al. [64], but pose grasp detection as a regression problem and solve it using a CNN. Pinto and Gupta obtain online experience from the robot during an automated experience-gathering phase instead of hand-labeled grasps [90]. Fischinger et al. go beyond 3-DOF detection by running a 3-DOF detector on a set of different planes [28]. This enables 6-DOF grasp detection because the scene is "seen" from different angles.

5.8.2 Grasp Templates

The template-based approach of Herzog et al. [42, 43] might be the most similar prior work to the approach presented in this chapter. The authors segment the object roughly from the background and calculate a convex hull around the segmentation. At each facet in the convex hull, grasp candidates are generated for a dicrete number of orientations about the grasp approach axis. A set of nearby points that are projected onto a plane and categorized as either object, background, occluded, or void, are associated with each candidate. Kappler

et al. [52] adopt this approach by replacing the convex hull with a bounding box around the object and by adding translations to each rotation on each face of the bounding box. For each face of the bounding box, there is a total of 16 potential candidate grasp poses. Both methods encode a grasp as a multi-channel image where each pixel stores the depth along the approach vector and the category (as mentioned above).

Our method is distinguished from these methods by a few key differences. First, our method does not segment the object from the background. Neither does it calculate a convex hull or bounding box. Second, our method can generate grasp candidates on any visible surface of the object, whereas those methods only consider candidates at the center of a bounding box face or a convex hull facet. Third, our method encodes a grasp using surface normals and multiple views, whereas those methods encode the grasp using a height map and a single viewpoint. As indicated by the experiments in this chapter, the second and third difference make our method advantageous in comparison to those methods.

5.8.3 Grasp Representations Based on Kernel Density Estimation

Another type of approach related to ours uses kernel representations to encode the local geometry of object surfaces. For instance, Detry et al. search for local object surface which are similar to one or multiple members of a set of grasp prototypes [23]. The inner product between a kernel density estimator over points on the local object surface and the same estimator for each prototype is used to measure their similarity. Those local object surfaces which are similar to the prototypes are considered to be likely to be good grasps. Kopicki et al. extend this approach to multi-fingered robot hands [60]. Kroemer et al. predict various types of manipulation affordances using kernel density estimation [62]. A key drawback of those approachs is their high computational cost. For example, Detry et al. report that it took 18s on average to match a single prototype against candidate on a single object [23]. In comparison, methods based on deep learning, such as the one we proposed in this chapter, are much faster because they only perform one forward pass of the network per candidate.

Chapter 6

Deep Learning for Grasp Candidate Generation

In this chapter, we present a supervised learning approach to grasp candidate generation that uses convolutional neural networks (CNNs). This approach aims to improve the quality of the candidates and the speed at which they are generated. We replace the geometric generator in GPD (see Chapter 5) with a learned model, i.e., a CNN. To efficiently represent grasps to this CNN, we introduce a simple image representation based on planar projections of the points in a fixed size bounding cube in the vicinity of the grasp. In simulation experiments, we demonstrate that the learned model outperforms the geometric one in both qualitatively and quantitatively. Our robot experiments show that this method can be used to grasp objects presented in isolation and in dense clutter.

6.1 Problem Statement

We define a robot, $\mathcal{R}$, as a robot hand that can move to an arbitrary pose in its workspace. The **robot state** is defined by the pose of the hand, $h \in SE(3)$, and the configuration of the fingers. In this work, we assume the hand is a parallel jaw gripper whose jaws are actuated by a single degree of freedom. The **world state** $\mathcal{W} \in \mathbb{W}$ is a description of the state of the environment and objects around the robot, where $\mathbb{W}$ is the set of all possible world states. A **point cloud**, $\mathcal{C} \in \mathbb{C}$, is a finite set of points in the robot's environment which are obtained by one or more depth sensors, where $\mathbb{C}$ denotes the space of possible point clouds that can be generated by the sensor arrangement of the robot. The world state is a latent variable that is observed only via the point cloud. In particular, we model the depth sensor(s) as a function $\Lambda : \mathbb{W} \rightarrow \mathbb{C}$ that encodes aspects of world state as a point cloud. Given a robot $\mathcal{R}$ and a point cloud $\mathcal{C} = \Lambda(\mathcal{W})$ for some hidden world state $\mathcal{W}$, the problem of **grasp pose detection** is to find a robot hand pose, $h \in SE(3)$, such that if the robot hand is moved to h and the fingers are closed, then some object in the world $\mathcal{W}$ can be grasped.

We approach grasp pose detection as a two step process. First, we generate a set of grasp

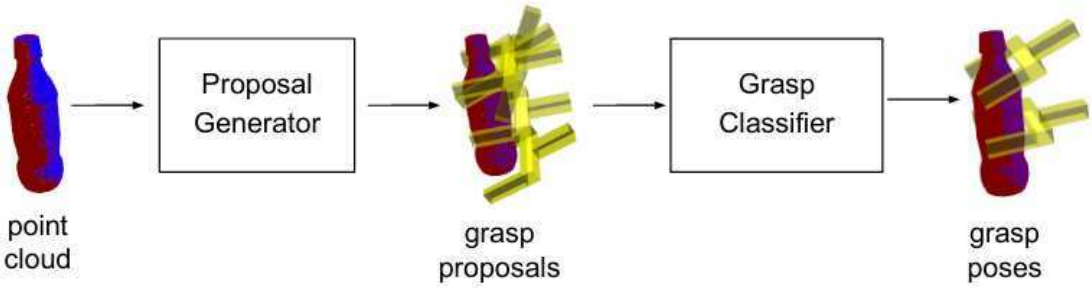


Figure 6.1: Overview of our approach. Given a point cloud, we generate grasp pose proposals using a convolutional neural network. We then classify the proposals as actual grasps using another convolutional neural network.

pose proposals. Second, we evaluate the probability that each proposal is an actual grasp. In this chapter, we are mainly concerned with improving the first step. We formulate the problem of generating grasp proposals as learning a function, $f : \mathcal{C} \rightarrow \text{SE}(3)$, that maps from the space of possible point clouds to the space of possible robot hand poses.

6.2 Approach

Figure 6.1 shows an overview of the end-to-end grasp detection system. There are two components: the proposal scoring network that generates a large set of grasp candidates quickly and the grasp classification network that makes a high quality binary prediction about each grasp candidate. Since the creation of images for the grasp classification network is computationally expensive [86], the proposal scoring network is essential because it focuses attention on promising grasp candidates.

6.2.1 Grasp Candidate Generation

Consider grasp detection in the plane, i.e., detecting (x, y, θ) grasp poses, where (x, y) is the grasp position in the plane and θ is the rotation about the grasp approach axis. In this case, the simplest approach to grasp candidate generation would be a single forward pass through a fully convolutional neural network, as, e.g., in Mahler et al. [74]. However, we cannot use this approach here because we are doing grasp detection in SE(3). Instead, we do the following. We sample k points from a region of interest in the point cloud. For each of the k points, we will do a single forward pass through the proposal scoring network predicting which hand orientations about the sampled point are likely to be a good grasp.

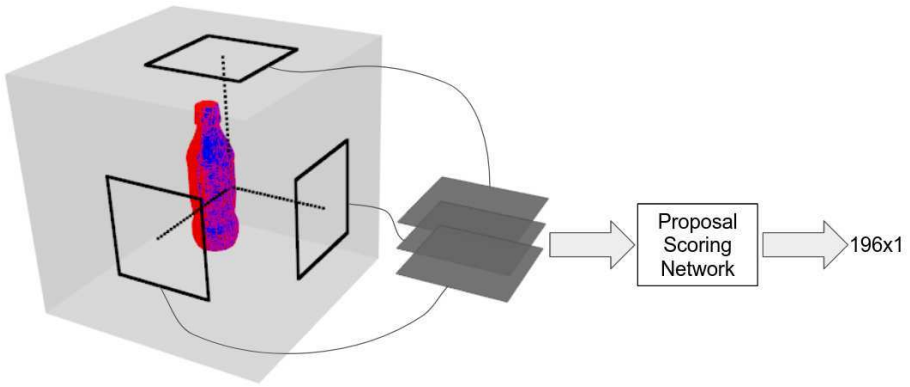


Figure 6.2: Grasp proposals are predicted with a proposal scoring network that takes a 3-channel image and produces a score for each grasp pose represented by the image. Each channel is an orthographic representation of a subset of the point cloud.

6.2.1.1 Input to the proposal scoring network

The proposal scoring network will take as input information about the local point cloud near the sampled point by extracting a bounding cube centered on the sampled point. Since we plan to do one forward pass through the proposal scoring network per point sampled (k forward passes), it is critical that this bounding cube is encoded efficiently. We do the following. First, we create a three-view orthographic representation of the entire point cloud expressed in the reference frame of the camera and centered on a point of interest (e.g., the estimated object center). Each of the three orthographic views is associated with a height map that describes the scene when viewed from that direction. These three orthographic projections are illustrated in Figure 6.2 as the sides to a cube that encloses the point cloud. Second, we crop a fixed-size rectangle in each of the three orthographic projections centered on the sampled point. Each of these crops encodes information about the local neighborhood of the sample (the three orthogonal boxes outlined in black in Figure 6.2). Finally, the height maps contained in these three rectangles are stacked and input to the grasp proposal network as a three-channel image. Notice that the approach above is very efficient per sampled point. For each point, we simply crop a rectangle from each of three images and copy them into the network input as a three-channel image (one channel for each crop).

6.2.1.2 Output of the proposal scoring network

Our baseline architecture for the proposal scoring network consists of two convolutional layers (no zero padding, stride 1, kernel size 5) followed by two FC layers followed by a sigmoid layer with m outputs. Each output is interpreted as the probability that a grasp exists at a particular orientation when the closing region of the hand is centered on the sample point and the hand is "pushed" forward until some part of it contacts the point

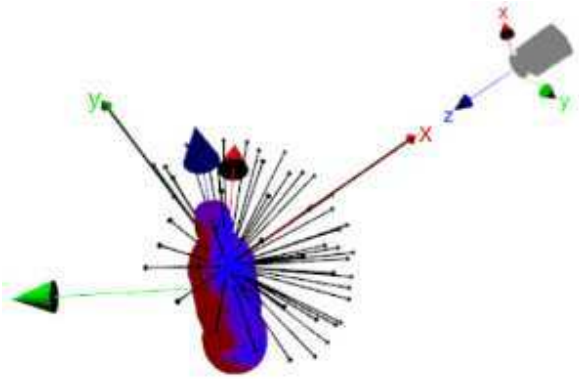


Figure 6.3: The robot hand orientations considered by our method. Each black arrow corresponds to a hand approach direction. For each direction, there are four possible orientations about the approach axis.

cloud[1]. While our method should generalize to an arbitrary number of orientations, we focus on the case where there are $m = 196$ orientations in this chapter. This is a much larger number of orientations than considered by GPD (GPD uses eight orientations per point, see Chapter 5). Note that since the output layer is a sigmoid instead of a softmax, the network makes predictions about grasps at multiple orientations for each sampled point. Figure 6.3 illustrates the $m = 196$ orientations about which our network makes predictions. Each black line is one of 49 approach vectors, about which there can be 4 rotations ($49 \times 4 = 196$). These orientations cover roughly a half dome of orientations pointed in the direction of the camera and are expressed in the reference frame of the camera. To summarize, the proposal scoring network takes as input the visible point cloud geometry in the vicinity of the sample point and outputs predictions about which orientations around the sampled point are likely to be grasps.

6.2.1.3 Loss Function

Learning the parameters of the proposal scoring network is a multi-label classification problem where the labels are multi-hot vectors, $y \in [0, 1]^m$, where m is the number of orientations considered by the proposal scoring network. Generally, given some object part, multiple robot hand orientations can usually lead to a successful grasp. We treat this problem as if there are m independent binary classification problems, i.e., the binary cross entropy loss function is calculated separately for each orientation.

[1]While "pushing forward" of the hand was implemented as a brute force search over a fixed set of increments in the method for Chapter 5, we here directly calculate the maximum distance by which the hand can be pushed before a collision occurs. This is much more computationally efficient.

6.2.2 Grasp Classification

To classify grasp proposals, we use a grasp classification network that is similar to the one presented in our earlier work [86]. The network takes as input information about the local point cloud that would be contained in the closing region of the robot hand at the grasp pose. The points in that region are orthographically projected onto a plane parallel to the hand's approach axis. A height map of these points makes up the first channel and the average surface normal at each of these points makes up three more channels of a four channel image. We use this image type as it is much faster to compute than ones with a larger number of channels [86]. The output of the network is a binary label that is one if the grasp is predicted to be successful and zero otherwise. The architecture for the network is the same as for the proposal scoring network. We use the cross entropy loss to train the network.

6.3 Network Training

We implemented our networks in PyTorch 1.4 [87]. For point cloud processing, we used Open3D 0.9 [117]. To generate synthetic scenes, we used Pyrender [77]. All of our networks were trained on a computer with an Intel i7-9700K 3.60GHz CPU with eight physical cores, 64GB of system memory, and an Nvidia GeForce GTX 1080 graphics card with 8GB of memory.

6.3.1 Ground Truth Grasps

We consider a grasp to be successful if it (1) is collision-free and (2) has force closure. We assume a parallel jaw gripper with soft contacts so that an antipodal grasp is a sufficient and necessary condition for force closure [81]. We consider all points within a fixed distance from each finger as possible contact points. We then check if the surface normals of those points lie within the friction cone with a fixed friction coefficient. This is the same procedure as in our earlier work [86].

6.3.2 Generating Data for Training

Our training set consists of a total of 300 objects, ten objects for each of thirty categories in 3DNet [113] (see Figure 6.4. To fix holes and other issues with the 3DNet object meshes, we apply a preprocessing method that produces watertight meshes with a 2-manifold topology and vertices distributed about uniformly on the object surface [44]. We obtain point clouds from a mesh by placing it at the origin of a fixed world frame and scaling it by a randomly chosen factor such that the object's extents are within $[0.01\text{m}, 0.07m]$ We then obtain single-view, partial point clouds by sampling camera locations uniformly on a sphere that has a radius of 0.5m and that is centered on the world frame's origin. For our training set, we sampled 20 camera locations. From each of the resulting point clouds, we uniformly sampled

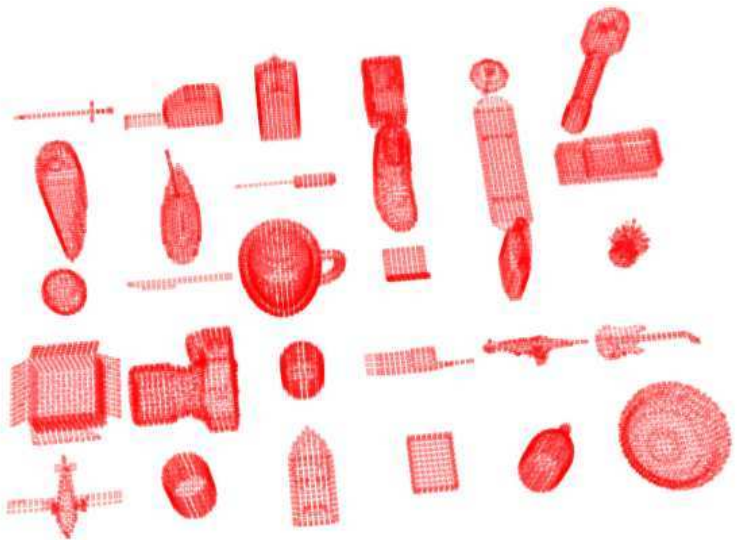


Figure 6.4: Example instances from 3DNet object categories used for training.

100 points and evaluated grasps with each method used for the simulation experiments against the corresponding ground truth mesh. In total, we evaluated about 117 million grasp poses to train the proposal scoring network and 600,000 poses for the grasp classification network.

6.3.3 Training the Networks

We use a batch size of 64 and images of width 60 and height 60. All models are trained with stochastic gradient descent with a momentum of 0.9. The learning rate starts from 0.01 and is exponentially decreased after each epoch by a factor of 0.96. We only use the synthetic data (described above) for training.

6.4 Antipodal Grasp Detection Experiments

In this section, we evaluate how well out method can detect antipodal, i.e., two finger force closure, grasps for objects presented in isolation. We obtain synthetic point clouds in the same way as described in Section 6.3.

6.4.1 Object Test Set

We evaluate on novel object instances from the same thirty object categories in 3DNet which were used for training. Objects are randomly scaled and viewpoints are generated in the

same way as for the training set (see Section 6.3.2). For each viewpoint, we sample $k = 100$ points uniformly at random from the point cloud.

6.4.2 Comparisons

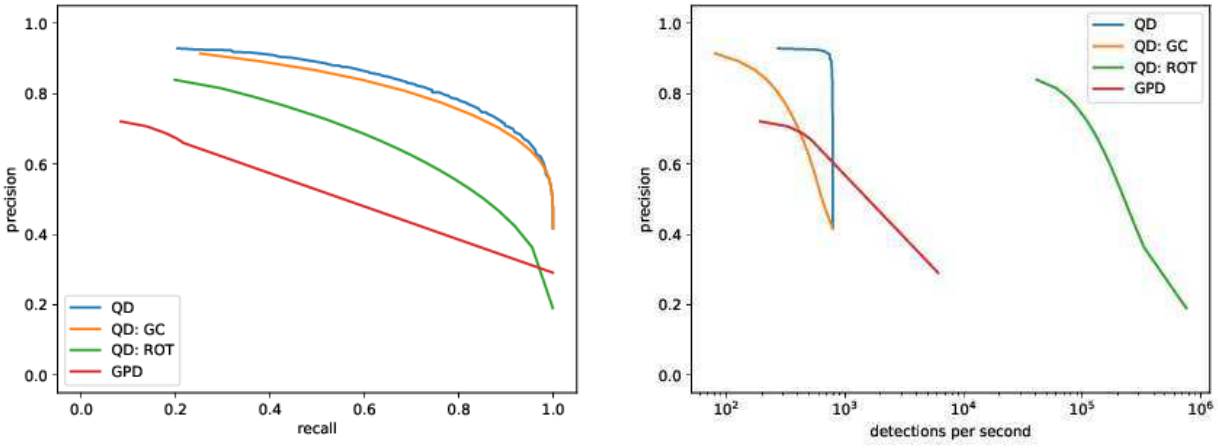
We compare our method, called QD (for the *quarter dome* of orientations considered), against two ablations, QD:GC and QD:ROT, and one baseline GPD. Here, GC stands for grasp classifier, and ROT stands for the proposal scoring network (as in *rotations*). QD:GC is an ablation of QD that consists of only the grasp classification network (second half of Figure 6.1). QD:ROT is an ablation that consists of only the proposal scoring network (first half of Figure 6.1). Both of these ablations are one-stage grasp detectors which evaluate all potential grasp poses. The GPD baseline is the 2-stage grasp detector from [86], as described in Chapter 5. It is structurally similar to the method in this chapter, but uses a geometric proposal strategy instead of a learned proposal generator. To enable a fair comparison, the grasp classifiers in QD, QD:GC, and GPD all use the same input shape. Similarly, the proposal scoring networks in QD and QD:ROT use the same input shape. All four methods have been trained on the dataset described in Section 6.3.2.

6.4.3 Evaluation Metrics

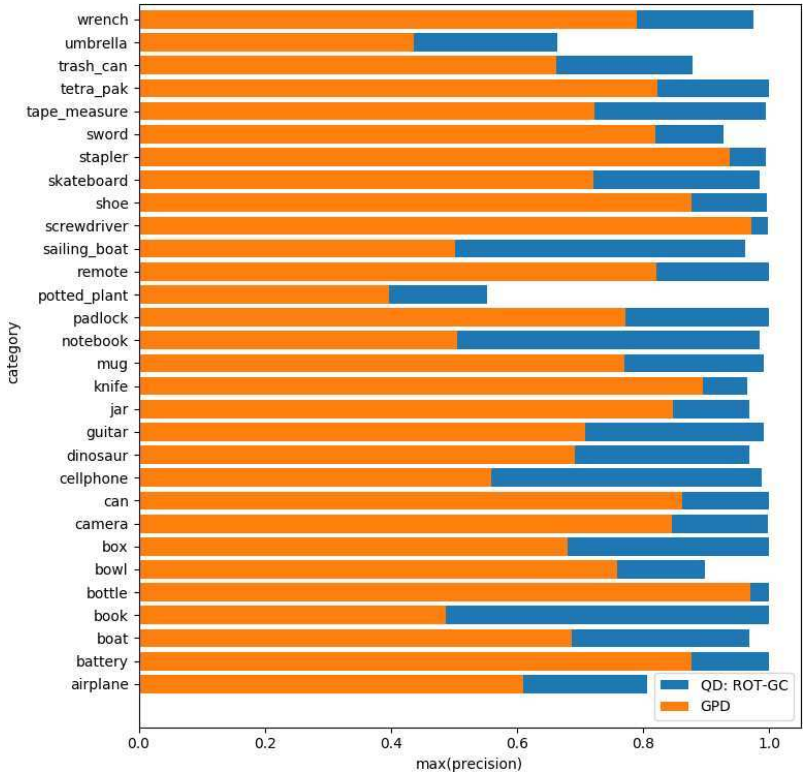
We evaluate our approach with three metrics: precision, recall, and detections per second (DPS). Precision and recall have the standard definitions. DPS is the number of grasp predictions produced by our method (both stages of Figure 6.1) per second. In addition to the standard precision-recall plot (see Figure 6.5a) we also show precision-DPS (see Figure 6.5b). Precision-DPS shows the tradeoff between precision and the number of predictions the grasp detection system makes. Ideally, our system would achieve a point in the upper right corner of this plot, i.e., lots of high-precision grasps produced per second.

6.4.4 Results: Isolated Objects

Figures 6.5a and 6.5b compare QD against QD:GC, QD:ROT, and GPD for the objects in the *3DNet* test set in terms of precision-recall and precision-DPS. First, notice that QD outperforms the GPD baseline by a large margin in both plots. While the QD:GC ablation is very similar to QD in terms of precision-recall, notice that it is much slower than QD, especially at high precision values. At 90% precision and above, QD:GC is approximately an order of magnitude slower than QD. We attribute this speedup to the fact that our candidate generation strategy allows us ignore a large number of candidates that the GC network would otherwise need to consider. Whereas QD:GC has to exhaustively calculate grasp images for every possible grasp proposal while QD only considers a subset. Looking at QD:ROT, notice that it is much faster than all other methods, but its precision-recall curve is significantly lower than either QD:GC or QD. This confirms that the proposal scoring network is fast but that it is not as accurate as the grasp classifier. These results emphasize that both



(a) (b)



(c)

Figure 6.5: Comparison between our method, the GPD baseline, and two ablations on novel 3DNet object instances. (a) Precision vs recall. (b) Precision vs detections per second. (c) Per-category maximum precision for GPD and QD.

components of QD are important for its performance: ROT allows us to subsample the set of grasp proposals to reduce runtime and GC allows us to make more accurate predictions because of the more informative grasp descriptor representation and the one-to-one relation between grasps and descriptors for GC compared to the many-to-one relation for ROT.

Figure 6.5c shows the precision that can be achieved by QD and GPD for each of the 30 object categories considered in our experiments. Notice that while QD outperforms GPD for all categories, the amount of outperformance is significant for some categories, like books. We believe that this simply reflects the difficulty the geometry based GPD proposal generator has for certain objects.

6.5 PyBullet Simulation Experiments

In this section, we investigate the performance of our method in a PyBullet physics simulation of robot grasping. While point clouds are still obtained synthetically in the same way as before, we measure actual performance of the grasp in the simulator under a simple physics model. This is a step closer to the evaluation on an actual physical robot.

6.5.1 Simulating Grasps

After obtaining a point cloud as described in the last section (using Pyrender) and performing grasp detection, we simulate a grasp as follows. This process is designed to mimic the simulation used in 6DoF GraspNet [79], which is not publicly available. We place a free floating Robotiq 2F-85 parallel jaw gripper in a pose relative to the object that corresponds to the selected grasp and close the fingers. Figure 6.6 shows examples of simulated grasps. After the grasp is formed, we turn on gravity such that it points in the direction that the

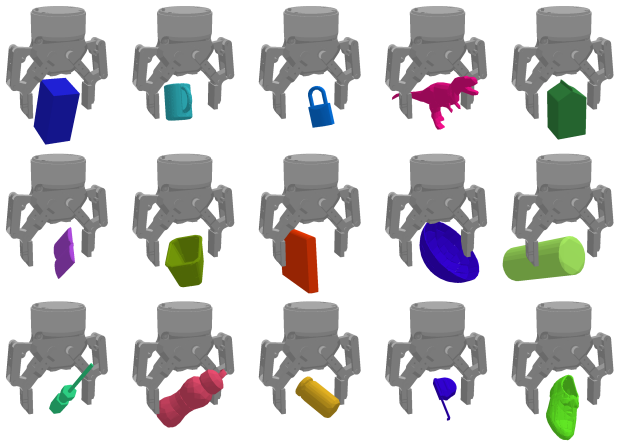


Figure 6.6: Examples of grasps simulated in PyBullet.

gripper is pointing (it is as if the gripper is pointed down). We then perform a predefined shaking motion. After this motion, we wait for 0.5s in simulator time, and evaluate if the grasp was successful. To check collisions between the hand and the object, we use FCL [83]. The shaking motion takes 8s in simulator time. The behavior of the simulation under gravity is such that any object that is not grasped or slips from the fingers quickly falls away from the gripper. If the object falls below a certain distance from the gripper, we label the grasp as a failure. During this process, the friction coefficient between the gripper and objects is kept constant, and all objects are simulated with a uniform mass of 0.5kg.

To make the ground truth robust against errors caused by imprecise robot kinematics and sensor noise, we simulate a small number of randomly perturbed poses for each grasp pose (i.e., in addition to the original proposal). The perturbed grasp poses are generated by applying a uniformly distributed random rotation between 0 and 5 degrees about a random axis, and a uniformly distributed random translation between 0 and 3mm. We decide the final label by a majority vote over the labels of the perturbations and the original grasp pose, i.e., if 3 out of the 5 poses are successful, we label the grasp as positive.

We compare our method to Graspnet [79], a recently published state-of-the-art 6-DOF grasp pose detection method. After obtaining a point cloud using Pyrender, we run Graspnet with the following default settings: 200 samples for the generative network and 10 refinement steps using Metropolis-Hastings sampling. We then label the resulting grasp poses using the PyBullet simulator as described above. We evaluate on unseen object instances from the same object categories that Graspnet has been evaluated on: bottles, bowls, boxes, cylinders and mugs (*BBBCM*). Cylinders and boxes are generated using Open3d with random dimensions. Bowls, bottles, and mugs are taken from the 3DNet object set.

6.5.2 Results: Physics Simulation of Grasps

Figure 6.7 shows the comparison with Graspet [79]. Our method is faster and produces grasp detections with higher precision. We do not perform the precision/recall comparison here because the Graspnet proposal generator does not have a discrete hypothesis set and we therefore cannot calculate recall. However, the precision/DPS shows that our method is much faster with no worse precision. It is important to point out here that Graspnet is at a disadvantage relative to our detector. The problem is that Graspnet cannot be retrained on our training set without access to high-end GPUs. As a result, we are using the version of pretrained Graspnet weights found using the labels generated in [79]. Although both methods were trained using the same object categories, it is likely there are slight differences between the labels in the training set and that this puts Graspnet at a disadvantage. However, given what is publicly available, this is the closest comparison that is possible.

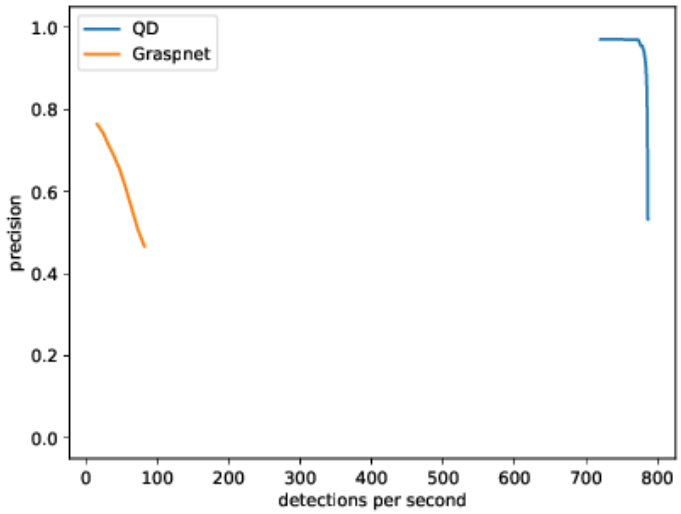


Figure 6.7: Precision-DPS comparison between QD and Graspnet. QD was trained using a subset of objects from the 3DNet dataset and we use the version of Graspnet trained by Mousavian et al. [79].

6.6 Robot Experiments

In this section, we investigate the performance of our method on a physical robot. We conduct experiments for objects presented in isolation as well as in dense clutter. We evaluate on an object set that is much larger and more diverse in shape, sizes, and object materials than the ones considered in earlier chapters. Besides, we obtain point clouds only from a single, fixed camera viewpoint.

6.6.1 Setup

We conducted all our robot experiments on a Universal Robots' *ur5* robot with six DOFs and a Robotiq *2F-85* parallel jaw gripper with one DOF. A *Structure IO* depth camera is mounted on the wrist of the robot arm (see Figure 6.8a). The robot is mounted to the same table on which the objects to be grasped are placed.

Our object set is shown in Figure 6.8b. All of the 60 objects in our set are unknown to our robot. We chose objects for which at least one side fits into the robot hand (preferably more than one). All objects weigh less than 0.5kg. Compared to the object sets used for evaluation in earlier chapters, this object set contains a much larger number and there is a much larger variety of shapes, sizes, and materials in this object set.

The computer used for these experiments has an Intel i7-7800X 3.50GHz CPU, 64GB system memory, and an Nvidia GeForce GTX 1080 graphics card with 8GB of memory. All robot experiments were run on ROS Melodic [95]. For inverse kinematics, an analytic

solution was used [41]. Motion planning and collision checking was done in OpenRAVE [26] and trajopt [102]. We use toppra [89] to produce a trajectory from a path under joint velocity and acceleration constraints.

To generate grasp poses, the robot moves its hand to a single viewpoint and obtains a point cloud from its wrist-mounted depth camera. The viewpoint is chosen such that the system can observe as much as possible of the scene. Our algorithm then takes the point cloud and produces a set of grasp poses. We sample $n = 500$ points from the point cloud. From the $n \times m$ outputs of ROT (n sampled points, m hand orientations), we select the top-k ($k = 20$) scoring orientations for each of the n points, and then uniformly subsample 300, and calculate grasp images for those. Then, we select the top-k ($k = 150$) scores from GC.

During initial testing, we found that grasps found using our method sometimes generated collisions with the object due to small kinematic errors in robot arm positioning. We believe these errors occur because the training set we used does not take proximity to collision into account. In order to correct for this problem on the robotic system, we followed the following procedure. First, we translate each detected grasp pose in the hand closing direction such that the visible points are centered in the grasp. Then, for grasp poses where the hand is further than a threshold distance from the closest point in the hand closing region, we translate the hand forward along its approach direction as far as possible without generating collisions. Next, we find sets of geometrically aligned grasps using a similar procedure as in [85] and generate one additional grasp per cluster that corresponds to the geometric center of the cluster. These centers are often close to object (part) centers. These centers

(a) (b)

Figure 6.8: Robot experiments setup. (a) A point cloud is acquired from a single viewpoint. (b) Object set for the isolated grasping experiment.

(a) (b) (c) (d)

Figure 6.9: Setup for isolated object grasping. (a-d) Examples of an isolated object scene with the four different configurations.

and all grasps produced previously are then checked for collisions with the point cloud. We select a grasp to be executed in a hierachical manner by considering grasps in the following order: cluster centers, cluster inliers, grasps with a score above 0.5, and grasps with a score below 0.5. For each group, we solve inverse kinematics for all grasps in the group and then compare them using heuristics that take into account the height of the grasp, how vertical the approach direction is and how wide the hand needs to be opened (similar to [86]). The former step reduces the number of inverse kinematics problems to be solved, and the second step can improve grasp success on in cluttered scenes because it mitigates the problem of collisions with objects located below other objects in a pile. If no grasp pose can be reached by the robot arm, the next group of grasps is considered.

6.6.2 Isolated Object Grasping

Here, we characterize how well our method is able to grasp objects that are placed in isolation on a table in front of the robot. For each object, we attempt a grasp for up to four distinct configurations: one to three horizontal orientations and one upright (see Figure 6.9)[2]. We generate a point cloud using a single depth sensor viewpoint and select a grasp using the procedure described in Section 6.6.1. Table 6.1 (first column) shows the results. The robot achieved an overall grasp success rate of 90.27% (204 successes out of 226 attempts). Grasps on box corners and the edges of objects were the most common failure modes in this experiment.

6.6.3 Dense Clutter Grasping

Here, we evaluate our method in a dense clutter setting similar to the one described in [86]. First, we place ten randomly selected objects from the object set into a box. The object set for this experiment is the same as shown in Figure 6.8b, except for the three bowls. We then turn the box upside down on a table in a fixed location, shake the box, and remove it.

[2]The bowls have one configuration that is repeated four times, and the wooden cube has two (a and c).

(a) (b) (c) (d)

Figure 6.10: Setup for dense clutter grasping. (a-d) Examples of dense clutter scenes.

Finally, the robot tries to grasp the objects one by one until no objects remain on the table. Examples of clutter scenes are depicted in Figure 6.10.

The second column of Table 6.1 shows the results of this experiment. Overall, the robot attempted 176 grasps out of which 144 were successful (81.82% success rate). The most common failure modes were edge grasps and corner grasps on boxes some of which may have failed due to noise in the robot's calibration. Furthermore, the robot removed 144 out of 150 objects from the table (96% success rate). The reason for the six objects which were not removed is that they rolled out of the robot's reach or view because of collisions which occurred while grasping another object.

6.7 Discussion

We proposed a method for predicting grasp poses which were represented as height maps of an orthographically projected point cloud. In simulation experiments, we found that our method is both fast and precise, in particular compared to geometric grasp proposal generation like in our prior work [85, 86]. We also showed that we can learn grasps based on a physics simulation where ground truth labels are assigned based on a simulation of the grasp with a robot hand on an object mesh instead of evaluating geometric conditions for an antipodal grasp. Finally, we demonstrated that our method can effectively grasp objects in isolation and in dense clutter on a robot.

In future work, we could extend the method proposed in this chapter to learn collisions

Table 6.1: Results of robot grasping experiments.

	Isolated objects	Dense clutter
Num grasp attempts	226	176
Num grasp successes	204	144
Grasp success rate	90.27%	81.82%
Object removal rate	/	96.00%

with an object's support plane, like a table, and with its surroundings, like in cluttered scenes. Another interesting direction is to learn task-dependent grasp candidates, e.g., to find grasp poses along the handle of a drill. Typically, task dependency is learned separately from grasp stability and collisions, but could be learned simultaneously.

6.8 Related Work

The work presented in this chapter is mainly related to two areas of research in the computer vision and robotics literature. The first area deals with the generation of proposals for object detection. The second area is concerned with the detection of 6-DOF grasp poses for a robot hand.

6.8.1 Object Proposals in Computer Vision

The ideas for proposing grasp poses algorithmically are analogous to proposing object locations and bounding boxes in computer vision. Girshick et al. laid the groundwork [33] in which a number of candidate object regions is proposed and individually evaluated by a convolutional neural network (CNN). Their work was later extended with region of interest pooling which improves object detection accuracy and reduces computational cost [32]. Ren et al. [97] further improved object detection by predicting candidate object bounding boxes with a region proposal network (RPN), extracting features for each box with region of interest pooling, classifying the box content and regressing the box parameters. Features for the RPN and the box evaluation are shared to further reduce computational inference cost. By learning to predict offsets from fixed reference bounding boxes, called *anchors*, the authors predict bounding boxes of different aspect ratios and sizes. Contrastively, Liu et al. [70] used a single network that makes predictions about a discretized space of default bounding boxes whose aspect ratios and scales are fixed. Similarly, our approach learns to predict 3D rotational "offsets" from a fixed set of reference grasp poses.

6.8.2 6-DOF Grasp Pose Detection

The state-of-the-art in grasp pose detection are deep learning based methods. Much work has focused on 3- or 4-DOF grasp configurations [65, 74, 51, 66, 100, 9]. Typically, such approaches keep two orientation DOFs fixed and only learn the orientation for one axis, usually the grasp approach axis. This severely restricts the ways in which an object can be grasped and thus restricts the manipulation tasks that such grasps can be used for. Another avenue of recent research explored 6-DOF grasping.

Mousavian et al. used a variational auto-encoder to sample a diverse set of grasp poses for an object [79]. These poses are then iteratively evaluated with a separate network and refined using the evaluator network's gradient. All of their networks are based on the PointNet++ architecture [93]. This approach is extended by Murali et al. [2] to grasping a particular

object in clutter. Qin et al. [94] used a single-shot grasp proposal network that predicts grasp poses by regression directly from the complete visible scene. Their network is based on the PointNet++ architecture [93]. In prior work, we detected grasp poses by first generating grasp proposals based on local geometry and then classifying multi-view projected features for each of the proposals [86]. Liang et al. extended this approach to directly work with point cloud data by training a network based on the PointNet [68] architecture. Gualtieri and Platt learn to grasp as part of pick and place modeled as an MDP with abstract state and action representations, and solve this MDP with a hierachical method that tells the robot where to look at in the scene [36]. Zhou and Houser [118] learn to predict a grasp score from a depth image and use the score for grasp pose refinement. Yan et al. [115] use generative 3D shape modeling to learn scene reconstruction that is then used to predict grasp outcomes. Merwe et al. [109] learn to reconstruct the object geometry and predict a score for a multi-fingered robot hand configuration that consists of a 6-DOF hand pose and finger joint positions. Lu et al. [72] train a 3D CNN on voxels to predict grasp success for a 16-DOF hand. Wu et al. [114] train a network to iteratively decide whether to zoom in or to execute a grasp for multi-fingered hands. Varley et al. [110] train a 3D convolutional network to perform object shape completion and then find grasps for the completed object. In contrast to the literature, we directly predict grasp poses based on the point cloud geometry and do not try to complete objects. Instead of generating grasp proposals in a heuristic fashion, we use a neural network to evaluate a fixed set of grasp poses.

Part III

Applications

Chapter 7

Open World Assistive Grasping Using Laser Selection

In this chapter, we present a robotic system for assistive grasping in open worlds. The system is comprised of a robot arm from a Rethink Robotics Baxter robot mounted to an assistive mobility device, a control system for that arm, and a user interface with a variety of accessmethods for selecting desired objects. The system uses grasp detection to allow previously unseen objects to be picked upby the system. The grasp detection algorithms also allow for objects to be grasped in cluttered environments. We evaluate our system in a number of experiments on a large variety of objects that characterize its performance for grasping objects in isolation from a table top and in a mobile scenario where objects are located on a table or in a shelf and the user must approach the target by controlling the system before a grasp can be attempted.

7.1 System Overview

Our system is composed of one robotic arm from a Rethink Robotics Baxter robot mounted on a Golden Avenger mobility scooter[1] as illustrated in Figure 7.1. The Baxter arm is a 7-DOF arm with a 1-DOF parallel jaw gripper. The mian difference between our Baxter arm and a standard Baxter arm is the fact that it has been separated from the rest of the body. The mobility scooter is off-the-shelf equipment that is designed for users with good upper-body mobility, but limited strength and endurance. The scooter cannot drive itself, but autonomous mobile manipulators have been explored by, e.g., Choi et al. [18]. A commercially available 1000W pure-sine Samlex AC inverter is used to convert power from the 24v batteries in the scooter for the Baxter arm and the control computer for the arm.

In our system, the Baxter robot, the laser pointer based user interface, and the grasp control subsystems are all implemented as ROS [95] nodes on computers located on board the scooter. The basic operations of the arm are controlled by the computer that is typically

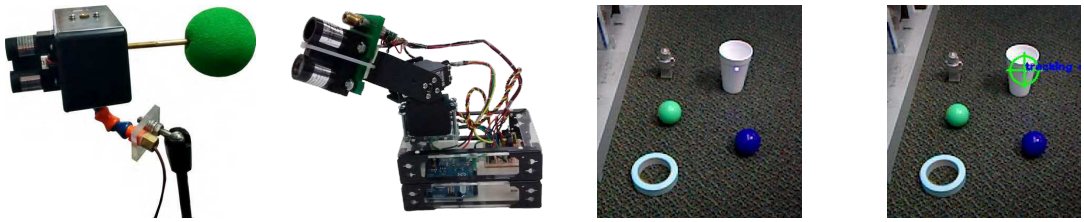
[1] https://www.goldentech.com/scooters/offroad-scooters/avenger/

Figure 7.1: Illustration of our robotic system for people with motor disabilities to perform the activities of daily living. Our system is comprised of a single arm from a Rethink Robotics Baxter robot that has been mounted to the front of a Golden Avenger mobility scooter. Novel user interface and grasping capabilities enable this system to autonomously grasp objects that the user points to using a laser pointer.

built into Baxter robots, running the same controllers and ROS nodes as in the standard Baxter robot. The grasp selection, laser detection, and kinematic control nodes are run on a separate laptop that is directly connected to the Baxter PC with a network cable. This laptop can run for approximately 1.5 hours on its internal battery, but it can also be powered from the scooter-mounted inverter for longer runtime. Adding the arm and laptop to the power draw on the scooter batteries will reduce the amount of time that the scooter can be operated between charges, but the system as a whole can operate for multiple hours without recharging. During manipulation experiments, with the scooter repeatedly driving small distances and powering both the laptop and the robot arm, the system was run for approximately five hours, in which it used less than half of the available charge from the scooter battery, as indicated by the scooter's built-in power display.

7.2 User Interface

The goal of the laser pointer based user interface is to enable people with motor or cognitive disabilities to operate the grasping system. Our goal is for these users to be able to indicate easily which objects in the surrounding environment are to be grasped. Since this system is to operate in open world environments which lack structure, it is important not to require prior



(a) (b) (c) (d)

Figure 7.2: Our user interfaces to control the laser pointer based grasping. (a) Manual laser interface. The foam ball can be replaced by other hand grips according to the needs of individual users. (b) Servo-based laser interface. This device can be controlled by a variety of computer access methods. (c) The laser is pointed at the cup in an example of a possible scene. (d) The laser point is detected on the cup.

models of the objects to be grasped. Ideally, such a system would be able to grasp anything that the user points at, regardless of the exactl type of object to be grasped. Nevertheless, the system is limited by the capability of the installed gripper and the arm. For example, the standard Baxter gripper is restricted to a fixed minimum and maximum opening width.

While indication of a target object with a laser pointer appears to be an easy-to-use interface, we must take into account the actual abilities of the user. We generally follow the same approach as Kemp et al. who used a laser pointer to enable users to indicate objects in an environment that should be grasped by a robot [56]. Choi et al. extended this approach to use an ear-mounted laser for patients with limited upper-limb motor control, but good head movement [18]. This enables people with severe motor disabilities to use the system.

We developed two different interfaces to the laser pointing system. The first is a manual interface that is designed for users with tremors which can affect their hand and arm motions. The laser pointer is mounted on a frictional ball joint. The user points the laser by manipulating a foam ball (see Figure 7.2c). The second interface is designed for users who have more severe upper body impairments. The laser pointers are mounted on a pan-tilt system controlled by a pair of servos (Figure 7.2d). The servos enable to control the position of the laser using a joystick or mouse. Alternatively, it could be controlled by sip-and-puff or single-switch scanning to accommodate users with quadriplegia.

The user interfaces are mounted to an articulated arm which is in turn mounted to the handlebars of the scooter. The articulated arm permits the interface to be placed where the user can operate it, and where the path of the laser is minimally obstructed by the arm and other hardware. The interface itself contains two lasers (5mW 650nm): one red and one green. These lasers are used to indicate whether a point is within the workspace of the robot's end-effector or not. When the point is outside the workspace of the robot arm (we approximate this as within a two meters envelope of the laser), the interface illuminates it with the red laser. When it is within the manipulator workspace, it is illuminated with the green laser. This indication provides clear visual feedback to the user about whether the

targeted object can be grasped or not.

7.3 Perception

In our system, there are three perceptual components: laser pointer detection, point cloud acquisition, and grasp detection. These components are described in the upcoming sections. Finally, we propose an automatic grasp selection strategy.

7.3.1 Laser Pointer Detection

We detect the position of the point indicated by the laser by pulsing the laser at 5Hz and looking for differences in successive camera frames. The area in front of the robot is observed by a PrimeSense Carmine sensor that is mounted near the base of the Baxter arm (labeled as "RGB-D Camera" in Figure 7.1). By differencing successive frames and looking for large intensity changes, areas of interest are calculated. Areas of high brightness after automatic exposure correction based on [78] are also used for area of interest calculation. Since motion in the environment can also cause such differences, the system filters the resulting regions based on size and color. We reject detections which are inconsistent over several frames. An example of a laser detection is given in Figure 7.2(c-d). After correctly pointing the laser, the user indicates to the system to go ahead with the grasp and the detected point is provided to the grasp sub-system.

7.3.2 Point Cloud Acquisition

Our grasp detection system takes as input a point cloud and outputs grasp configurations. The quality of this point cloud has a significant impact on the performance of grasp detection. As described in Chapter 5, our grasp detection method can achieve a 9% higher grasp success rate (93% instead of 84% grasping in dense clutter) when using a point cloud obtained using dense metric SLAM instead of a point cloud obtained directly from the depth sensors [86]. This result suggests that a similar approach would improve the accuracy of the scooter system's grasping. In our prior work, we mounted the depth sensor close to the robotic hand and moved the hand through a pre-programmed trajectory around the objects of interest (see Figure 5.9c). Unfortunately, that strategy cannot be used in the current situation because the objects to be grasped can be found in different places around the workspace, e.g., on shelves, tables, and so forth. Futhermore, the user could drive the scooter into a location where the pre-programmed trajectory would collide with objects in the environment, such as a closet entrance or a narrow pantry. To avoid this problem, we gave the system the ability to plan a sensor trajectory that avoids obstacles while scanning the relevant portion of the workspace.

We refer to the trajectory that the sensor takes in order to observe the object to be grasped as the *information gathering trajectory*. To successfully plan this trajectory, several

constraints need to be satisfied. The first constraint is that the view of the point of interest must not be occluded by obstacles in the environment. The second constraint is that all viewpoints need to lie beyond the minimum range of the depth sensor. The third constraint is to minimize the length of the information gathering trajectory. The reason for this constraint is that SLAM packages, such as InfiniTAM [92] that we use for this work, can lose track of the environment relative to the voxelized grid that is created when there is not enough structure in the viewed environment to enable the matching algorithm (e.g., iterative closest point [6]) to function robustly. In particular, this is a risk at close range because there is typically less structure in the environment than in room-scale scenarios.

In order to satisfy these three constraints, the system plans the information gathering trajectory as follows. First, we assume that the system is given a point of interest. In our case, this is the point indicated with the laser. The system constructs a sphere about the point of interest with a radius of 42cm, slightly larger than the 40cm minimum range of the Structure sensor. Pairs of points are then sampled uniformly at random from the sphere until a pair is found such that the points are less than 22 cm apart and have collision-free inverse kinematic (IK) solutions that permit the sensor to point directly at the point of interest. To generate IK solutions, we use IKFast [25]. OpenRAVE [26] is used to check collisions with known obstacles, such as the scooter. Once a feasible pair of points is found, we use TrajOpt [102] to generate a trajectory that is collision-free with respect to the point cloud obtained from the Carmine sensor, remains outside the minimum range of the depth sensor, causes the sensor to point directly toward the point of interest throughout, and travels between the two sampled points.

7.3.3 Grasp Detection

Grasp detection is a key element of our system because it enables us to grasp novel objects. Traditional approaches to robotic grasping are based on localizing and grasping specific known objects [57, 17], or grasping based on behaviors that take advantage of common features of environments built for humans [47]. While these methods can work well in structured environments, or where object geometries are known ahead of time, they are less suited for unstructured or cluttered real world environments.

We use GPD, the method presented in Chapter 5, for this work. Note that there is no notion of "object" required for GPD. A detected grasp is simply a hand configuration from which it is expected that the fingers will form a grasp when they are closed. GPD is particularly well suited to open world environments because it does not require object models and because it has shown to achieve high grasp success rates for object presented in dense clutter [86].

7.3.4 Grasp Selection

If grasp detection runs successfully, it typically finds ten or hundreds of hand configurations which are likely to be good grasps. We filter those grasps as follows. First, we eliminate

grasps which have a position that is further than 6cm away from the laser point. Such grasps are unlikely to be grasps on the object of interest. Next, we remove from consideration all grasps which have no collision-free IK solutions A key drawback of grasp detection in its current form is that it only takes into account the geometry of the object surfaces to be grasped. It ignores other relevant information such as distance to the center of gravity of an object, whether an object is near the top of a pile or not, and how heavy the object is likely to be. Currently, we compensate for this by encoding simple heuristics into a cost function that is used to select grasps. The cost function gives preference to grasps with large z coordinate values (i.e., height relative to the ground plane). If objects are located in a pile, such grasps are likely to be near the top of the pile. We also give preference to grasps which are closely aligned with the gravity vector. We call these grasps "top grasps" because the gripper approaches the object from a top-down direction, as opposed to "side grasps", where the gripper approaches the object from one of its sides. Empirically, we have found that top grasps succeed more often and are less likely to cause collisions with nearby objects. However, the top grasp heuristic is not used if the grasp position is less than 34cm beneath an obstacle in the environment. This allows the robot arm to reach objects on a shelf that would not be reachable with top grasps because the robot arm will not fit between the object and the shelf above it. Futhermore, our cost function penalizes grasps which would require the robot arm to reach configurations close to its joint limits. Grasps which require the hand to be opened or closed completely are also penalized. Finally, our cost function encodes a preference for grasps which can be reached by moving the arm the shortest distance in configuration space from its final viewing configuration. We select the grasp with the lowest cost to be executed, and use TrajOpt to generate a motion plan to the grasp configuration and for moving the hand safely back to a position over the basket where the grasped object can then be deposited.

7.4 Experiments

To characterize the performance of our robotic system, we performed two types of experiments. First, we performed a table top experiment where we evaluated the grasping system in isolation from the rest of the system. Second, we performed an experiment that evaluated system performance in scenarios close to our targeted use-case scenarios, i.e., where the user moves the mobility scooter around in a room and uses it to pick up objects from a number of different locations such as a shelf or a table.

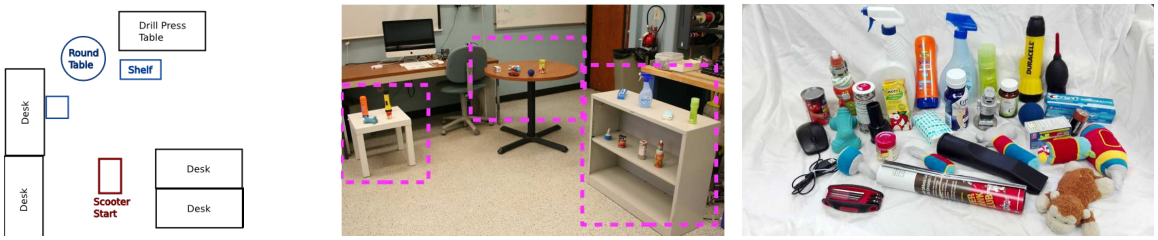
7.4.1 Evaluating Grasping in Isolation

Since grasping is the core functionality of this system, it is important to characterize grasp performance as accurately as possible. In the standard configuration where Baxter is mounted on its stock stand, it is possible to achieve a 93% grasp success rate for novel objects presented in dense clutter such as that shown in [86]. However, the current scenario is somewhat differ-

Figure 7.3: Experimental scenario for evaluating the grasping system by itself.

ent because the object to be grasped is now identified by a laser. Nevertheless, we should be able to achieve a similar grasp success rate in the current scooter configuration. To evaluate our system's performance for this scenario, we performed a series of 15 trials for the tabletop scenario shown in Figure 7.3. At the beginning of each trial, six objects were randomly selected from a set of 30 novel objects (see Figure 7.4a(c)) and placed at random locations on the table within the workspace of the Baxter arm. The system was then directed with the laser pointer to each of the six objects in a random order. If the selected grasp belonged to the object targeted by the user, then we considered this an object detection success. Otherwise, this was an object detection failure. Similarly, we measured the grasp success rate. Once a grasp was selected for execution (whether or not that grasp was on the target object), we checked whether the robot was able to successfully grasp, transport, and drop the object into the basket. If the robot was successful in performing all of these steps, then the grasp was considered a success. Otherwise, it was considered a failure.

Overall, we performed 15 trials with 6 objects on each trial (a total of 90 objects to be detected and grasped). There were 123 attempts to detect an object using the system. This number exceeds 90 because detection and grasp errors caused us to attempt detection multiple times for some objects. Only 108 detections were successful out of the 123 attempts, resulting in an 87.8% object detection success rate. Out of the 90 objects, there were 87 grasp attempts. The remaining objects rolled or fell out of the robot's workspace because of direct or indirect collisions with the robot arm or hand. Out of the 87 grasp attempts, 78 were successful, resulting in an 89.6% grasp success rate. On the whole, we consider these to be good results. The 89.6% grasp success rate is close to what we have obtained in the lab under ideal conditions. The 87.8% object detection success rate is lower than desired, but it could likely be improved using segmentation methods from the literature.



(a) (b) (c)

Figure 7.4: Setup for in-situ experiments. (a) Room layout. (b) Locations where objects to be grasped can be found: a low table, a high table, a middle shelf, and a top shelf. (c) The 30 novel household objects to be grasped.

7.4.2 Evaluating Grasping In-Situ

Our main goal is to to characterize whether our system can help people with motor or cognitive impairments to perform ADLs. Therefore, we performed an experiment to evaluate the system in a more realistic setting: a user would sit in the scooter, drive it around a domestic environment, and attempt to grasp objects of interest. Ultimately, we plan to evaluate our system with patient populations at a collaborating rehabilitation facility. For this work, we only evaluate the degree to which the system is capable of performing these tasks under ideal conditions. Therefore, all of the following experiments were performed by able-bodied users operating the system.

Our experimental setup is illustrated in Figure 7.4. The scooter was placed in a room with two tables and a shelf. Figure 7.4a shows the layout of this room. The blue items in Figure 7.4a indicate the two tables and the shelf. These same pieces of furniture are shown in the magenta boxes in Figure 7.4b. We performed a series of five trials. At the beginning of each trial, 10 objects were randomly selected from a set of 30 novel objects, shown in Figure 7.4c. We selected these objects according to two criteria: (i) they represent objects which might typically be used in daily life, and (ii) the Baxter parallel jaw gripper is mechanically capable of grasping them. We placed the 10 selected objects on the three pieces of furniture in randomly assigned positions. Three objects were placed on each of the two tables and two objects were placed on each of the two shelves. The top shelf was 90 cm above the ground and the middle shelf was 60 cm above the ground. In total, there were $5 \times 10 = 50$ objects to be grasped.

Initially, the scooter was placed in a starting configuration as shown in Figure 7.4a(a). The arm was moved to an initial configuration that minimizes interference with the laser beam and the user's view of the environment. The user was allowed to choose the order in which the objects were to be grasped. After the user instructed our system to grasp a particular object, the user drove the scooter up to the corresponding object, e.g., in front of the shelf, illuminated the object using the laser, and activated the automatic grasping process. Then the robot arm would scan the region around the laser point to acquire a point

cloud from the wrist-mounted depth sensor, compute a grasp, and attempt that grasp. We measured the object detection success rate and the grasp success rate as in the previous experiment. Figure 7.5 illustrates a grasp executed by our system for this scenario.

Out of the 50 objects available to be grasped, we attempted to detect an object 66 times. Note that we attempted to grasp some objects multiple times because of grasp failures. Out of those 66 detection attempts, 7 were failures, resulting in an 89.4% object detection success rate. This result is similar to the 87.8% detection success rate that we observed for the previous experiment. Out of the 50 objects available to be grasped, there were a total of 61 grasp attempts, as some objects were attempted multiple times. Out of 61 attempts, there were 17 grasp failures, resulting in a 72.1% grasp success rate. This grasp success rate is significantly lower than the 89.6% success rate observed in the previous experiment. There were two primary failure modes that account for most of the additional failures. The first failure mode is collisions between the arm and the environment. In 4 out of the 17 grasp failures, the sensors failed to "see" obstacles in the environment that the arm subsequently collided with. This failure mode did not occur in the previous experiment because the scooter was placed in a configuration that was largely obstacle-free. However, the main failure mode (13 out of the 17 grasp failures) was incorrect grasp detections. Here, the robot detected a grasp on the edge of the table or shelf that was very close to the detected laser point. Since grasp detection does not detect objects *per se*, there is no intrinsic reason why it cannot detect a "good" grasp of the edge of a table or shelf. Such failures did not occur in the previous experiment because the table location was known, and grasps on the table were excluded using workspace limits.

We also evaluated the task in terms of the time required to drive the scooter up to an object and grasp it. Out of the 50 objects, the average time to grasp an object was 128s, the minimum was 44s, and the maximum was 374s. While this is a fairly large variation in time-to-completion, several of the trials were completed in close to the minimum time. Essentially, our system achieves the minimum time when each step of the grasp works as

Figure 7.5: Our system grasping an object in-situ from a shelf.

intended the first time, i.e., when our SLAM package does not lose track, the arm does not collide with unseen objects in the environment, and the grasp works on the first try. If one of these elements fails, then the trial can still succeed, but it takes more time. In order to put these time measurements in context, we also performed a series of teleoperated grasp trials that were exactly the same as the automated trials, except that the user directly controlled the motion of the arm using a keyboard interface. The average time to complete a grasp in teleoperation mode was 40s. Although this is less than the 44s minimum time taken by our automated system, it is close enough to suggest that our automated system could be a practical approach if we are able to correct for some of the failures described above. Moreover, note that even when the automated system takes longer to grasp than would be required by an able-bodied person to teleoperate the system, the automated capability is still extremely important for our target users, for many of whom it would be extremely challenging to teleoperate a robotic arm.

7.5 Discussion

This chapter described a robotic system for assistive grasping that uses the GPD algorithm described in Chapter 5. The system consists of a number of off-the-shelf components such as a Golden Avenger mobility scooter, a single Baxter robot arm, and RGBD cameras. A laser pointer is used to indicate an object of interest and the robot picks up that object autonomously.

While the experimental results obtained for this work are promising, the system can be improved in a variety of ways. A key thing to improve is the time required for a grasp. Waiting two minutes to grasp a single object could be frustrating to users. One way to speed up the process would be to eliminate the information gathering trajectory and instead rely on a point cloud from a single view. With improvements in grasp detection or in choosing the correct viewing angle for the object, the same reliability might be achieved as with the SLAM method. Another major failure mode was arm collisions with unseen portions of the environment. In this case, we expect that adding additional depth sensors to the mobile base will eliminate most of these failures. The laser detection method could be improved to reduce its failure rate. Grasp detection failures where a grasp is detected on a table or a shelf could be eliminated to some degree by point cloud segmentation or object recognition and subsequent object detection in the point cloud. Furthermore, GPD could be trained specifially for the scenario under consideration, e.g., objects placed on a shelf, and the system could identify the particular scenario that it is encoutering based on visual sensor data.

Chapter 8

Open World Assistive Pick and Place

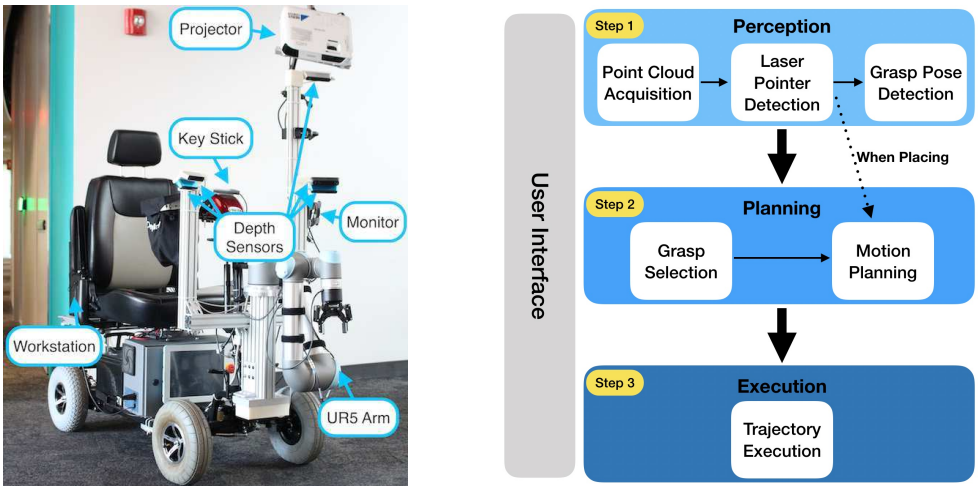
In this chapter, we describe a robotic system for assistive pick and place in open world environments. The basis of the system is similar to the one presented in the previous system: a robot arm mounted to a mobility scooter and a laser pointer to indicate an object to be grasped. However, we replace the Baxter robot arm from the previous system with a UR5 robot arm that is more compact and more accurate. We evaluate our system in both pick and pick-and-place tasks. For picking, we use a similar experimental setup as in the previous chapter to allow a comparison with the previous robotic system. We characterize task success and runtimes for grasping in isolation from other components and for grasping in situ (i.e., with driving the scooter). We also characterize performance for simple pick-and-place tasks.

8.1 System Overview

An overview of our system is shown in Figure 8.1a. A UR5 robotic arm (6-DOF), equipped with a Robotiq 2-finger 85 gripper, has been mounted on a Merits Pioneer 10 mobility scooter[1]. Five Occipital Structure depth sensors provide visual data to our system: four are mounted near the handlebars (two on the left and two on the right) and the fifth is mounted higher up. The sensors on the handlebars enable our system to see the environment in front of the scooter, whereas the other sensor allows us to view elevated surfaces like a tall table top or the top of a shelf. This sensor setup permits to detect objects in a one meter tall area in front of the scooter. The sensors, their supporting structures, and the robot arm are mounted on the scooter such that the front view of the user is not blocked. It also gives the sensors a nearly occlusion-free view of the robot's workspace (see the nearly complete point cloud in Figure 8.2c).

The physical components of our user interface, shown in Figure 8.2b, are: a dual laser pointer device, a ten-inch monitor, an X-keys XK-4 stick, and an EPSON LCD H801A projector. A computer is mounted on the back of the scooter and is connected to the robot,

[1] http://www.meritsusa.com/page/product_38.html

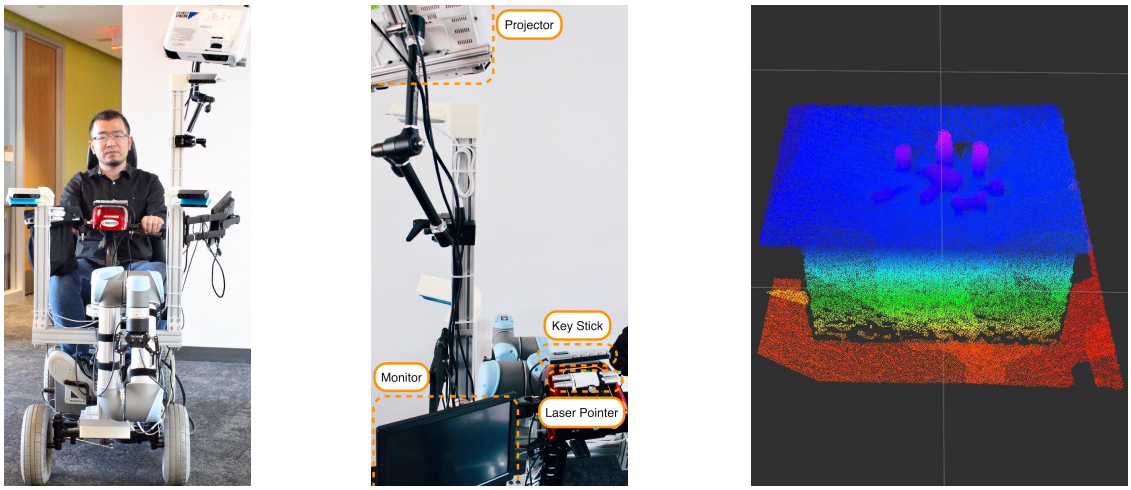


(a) (b)

Figure 8.1: Our system's mobile base is a Merits Pioneer 10 mobility scooter with a Universal Robotics UR5 robot arm mounted to its front. Perception capabilities are provided by five Occipital Structure depth sensors. The user interface consists of a monitor, projector, key stick, and dual laser pointer device.

sensors, and user interface. It has a 3.6 GHz Intel Core i7-6850K CPU with six physical cores, 32 GB of system memory, and an Nvidia GeForce GTX 1080 graphics card.

Our system's software has four subsystems: perception, planning, execution, and user interface. An overview of our software workflow is given in Figure 8.1b. The perception subsystem consists of three parts. First, we acquire a point cloud from each of the five sensors and concatenate them into a single, larger point cloud. Second, we use the infrared radiation (IR) images of the sensors to detect the 3D position of the laser pointer. Third, we use Grasp Pose Detection (GPD), the method described in Chapter 5 to detect potential grasp poses. Once grasps have been detected, the planning subsystem selects a grasp based on either automatic grasp selection or manual grasp selection provided by our system. Then, we try to find a feasible trajectory for the robot arm. The trajectory is executed by the execution subsystem. When the system performs placing, grasp pose detection and grasp selection are skipped and the planning subsystem directly generates a trajectory to the position that the laser pointer indicated. During the whole process, the user interface subsystem provides feedback to the user and interacts with the user. All subsystems are implemented as ROS [95] nodes running on the computer mounted on the system. Figure 8.3 illustrates the process of our system picking up an object.



(a) (b) (c)

Figure 8.2: User interface and perception. (a) The front view of our system. (b) The physical components of our user interface. (c) A point cloud generated by combining data from the five depth sensors. Note that the point cloud is nearly unoccluded.

(a) (b) (c) (d) (e)

Figure 8.3: The picking workflow of our system. (a) The user drives to a pick location. (b) The user identifies the target object using the laser pointers. (c) The projector illuminates the region where grasps will be detected. (d) Our system detects grasps, selects one to execute and calculates a motion plan. (e) The robot arm executes the grasp.

8.2 User Interface

We developed a simple graphical user interface (GUI) that, using RViz [95], visualizes the various outputs of our system: the perception results, the status of the system, and the current actions available to the user. By using the key stick (see Figure 8.2b), the user interacts with this interface.

The user selects the next target, i.e., an object for picking or a location for placing, with the dual laser pointer. Two 5mW 650nm red lasers make up the laser device. The lasers are mounted in a parallel configuration so that the user can illuminate the desired target with one laser and then confirm the selection with the second laser.

Additional information is provided to the user by a dynamic spatial augmented reality (DSAR) system for which a projector has been mounted on the scooter (see Figure 8.1a). This system can highlight the area around the next pick or place target with a beam of light. For grasping, the area around the object to be grasped can be highlighted, as shown in Figure 8.3c. For placing, the position where the object is to be placed is highlighted.

To enable highlighting of the next target, we first create a virtual camera that has the same intrinsics and extrinsics as the projector. We then project the segmented point cloud into the virtual camera's frame and create a black-and-white image. Pixels are white if the corresponding point belongs to the segmentation and pixels are black otherwise. We finally send the image to the projector and use it to illuminate the white pixels.

Since the intrinsics of the projector were unknown, we modeled it as a pinhole camera and determined the intrinsic parameters by mounting it at a fixed known distance away from a wall and projecting an image onto the wall. The focal length and principal point, which comprise the intrinsic parameters for this case (we assume the lens has no distortion), were then determined using the standard projection equation [40] .

8.3 Perception

In our system, there are three perceptual components: laser pointer detection, point cloud acquisition, and grasp detection. These components are described in the upcoming sections. Next, we propose an automatic grasp selection strategy, outline how manual grasp selection interacts with the user, and describe how the placements are found given a target location indicated by the system's user.

8.3.1 Laser Pointer Detection

To identify the target for the next pick or place action, a dual laser pointer device is used. We detect the positions of the two laser points in the IR images acquired from the depth sensors. Hierachical clustering [98] is used to find nearby high-value areas in each image in order to find the potential location of each laser point. To increase detection accuracy, we

use the IR images of all five depth sensors to detect potential locations and map them to 3D to arrive at a matching position.

A dual laser pointer device is used to identify the target for the next pick or place action. We detect the position of the laser pointers by reading the infrared radiation (IR) image from the depth sensors. Since our device consists of two parallel laser pointers, each potential location is detected by finding two nearby high-value areas within each IR image using hierarchical clustering [98]. To increase the accuracy, we detect potential locations in the IR image of each of the five sensors and map them to 3D to find a matching position.

8.3.2 Point Cloud Acquisition

Point cloud information is used as the primary input for both grasp detection and motion planning in our system. Therefore, it is important to obtain a point cloud of the robot's workspace that is as complete as possible. The four sensors on the handlebars are mounted 114cm above the ground. These sensors are separated into two mirrored groups with a horizontal distance of 63cm between the two groups. In each group, one sensor points 25 degrees downward, and the other sensor points 55 degrees downward. The former configuration covers most of the workspace, while the latter supplements a view of the area that is lower and closer to the robot arm. The fifth sensor is mounted 156cm above the ground and points 44 degrees downward. This enables to see elevated surfaces such as a shelf top. All five sensors point 20 degrees inward to the robot's center.

As shown in Chapter 5, observing the workspace with multiple views can signficantly improve both obstacle avoidance and grasp detection. Here, we try to have at least two views of the robot's workspace for our system. Each sensor group on the handlebars gives one view and the higher sensor gives another view. An example of a point cloud created by our sensors is shown in Figure 8.2c. This point cloud covers all of the workspace reachable by the robot arm, as suggested by the complete table being visible in front of the robot.

8.3.3 Grasp Detection

To facilitate grasp detection, we eliminate from consideration those parts of the environment which are not relevant for grasping. First, we remove all planes from the point cloud which are roughly horizontal to the scooter using RANSAC [30] with a 0.015m distance threshold and a maximum angle of five degrees relative to the horizontal. Planes which contain at least a threshold number of inlier points (we use 13k) are eliminated. Once horizontal planes are removed, we use Euclidean clustering [98] with a distance threshold of 0.005m in an attempt to segment individual objects. We remove all clusters except for the one that is closest to the point of interest that was found with our laser pointer detection and that contains at least 500 points. If the closest cluster has less than the required number of points, we instead keep all points within a ball of 0.1m radius that is centered at the detected laser point.

Once the point cloud has been reduced to the relevant parts of the environment, we detect feasible grasps using GPD. We feed two different point clouds as input to GPD: the

segmented cloud as described above and the complete cloud. The former is used by GPD to seed grasp candidates and the latter to ensure that the detected grasps are not in collision with the environment. Only grasps in the vicinity of the target object are detected because the candidates are seeded only from the segmented point cloud.

8.3.4 Grasp and Place Selection

This section describes our grasp and place selection strategies. After grasps are detected, our system prunes all grasps for which there are no collision-free inverse kinematics (IK) solutions [41, 25]. To select a grasp for execution from the remaining grasps, the user can either let our system automatically choose one based on a built-in set of heuristics or manually select one.

8.3.5 Automatic Grasp Selection

To select grasps automatically, we use the following objective function:

$$C = C_w C_j C_a C_s C_p, \tag{8.1}$$

where C_w, C_j, C_a, C_s, and C_p are cost functions which make up a set of heuristics. The goal of these heuristics is to choose a grasp that is easy for the robot to execute, unlikely to cause collision, and that minimizes the motion time for the robot arm.

Grasps which are near the gripper width limits are penalized by C_w. Grasps which are close to the maximum gripper width are difficult to execute because even small errors in perception or kinematics can make them fail. Grasps which are close to the minimum width often correspond to small parts of an object which are prone to be mechanically unstable. We calculate C_w as:

$$C_w = 1 - \frac{\max(0, W_d - \min(|G_w - W_{\min}|, |G_w - W_{\max}|))}{W_d}, \tag{8.2}$$

where G_w is the grasp width, i.e., the extent of the points in the closing direction of the gripper, $W_{\min}$ is the min gripper width, $W_{\max}$ is the max gripper width, and W_d is a minimum acceptable distance from the gripper extrema (0.015m in our experiments).

Because grasps with larger joint distance require more time to be executed, C_j gives preference to grasps which can be reached by moving a shorter distance in configuration space. We calculate C_j as:

$$C_j = \max(0, 1 - \frac{1}{J_m} \|J_i - G_j\|), \tag{8.3}$$

where J_i is the initial arm configuration, G_j is the arm configuration which reaches the grasp pose, and J_m is the maximum joint distance.

Top-down grasps are easier for the robot arm to execute, while side grasps which approach an object from the front are less likely to collide with obstacles blocked by the target object and which might not be observable by the sensors. We calculate C_a such that top grasps are preferred over side grasps and side grasps are scored higher when they approach an object from the front:

$$C_a = \begin{cases} 0.5 + 0.5|G_a[1]|, |G_x[2]| > 0.8 \\ 1, \text{otherwise} \end{cases} \tag{8.4}$$

where G_a is the approach vector and G_x is the axis vector of the grasp.

A preference for grasps which are nearby the segmented point cloud is given by C_s:

$$C_s = 1 - \frac{\min(l, \min_{\forall p \in S} \|p - G_b\|)}{l}, \tag{8.5}$$

where G_b is a fixed point relative to the grasp position, l is the gripper length, and S is the segmented point cloud.

Finally, we penalize grasps which are far away from the detected laser point by C_p as:

$$C_p = \exp(-10 * \max(0, \|p - G_b\| - \text{th})), \tag{8.6}$$

where p is the position of the laser point and th is a distance threshold (0.05m in our experiments).

8.3.6 Manual Grasp Selection

Even though automatic grasp selection usually works well, there are few potentially problematic scenarios. The first scenario is densely cluttered scenes where the segmented point cloud might contain multiple objects and the selected grasp is not on the desired object. The second scenario is a grasp detected on a table, shelf, or some other part of the environment. The third scenario considers pick-and-place where the user might prefer to let the robot grasp an object in a particular way. To overcome these problems, manual grasp selection allows the user to specify how to grasp the object. Here, the user selects one of the grasps detected by GPD instead of relying on the built-in heuristics described previously.

During manual grasp selection, we visualize the detected grasps on a small monitor that is mounted near the handlebars of the scooter (see Figure 8.2b). Using the keystick, the user can go through those grasps and select one to be executed. To simplify this process, we cluster the detected grasps to eliminate from consideration those which are nearby duplicates using hierarchical clustering [98] on the 3D position of the grasps. We set the maximum Euclidean distance inside each cluster to 0.02m. Each cluster is then visualized on the small monitor.

8.3.7 Place Selection

When our system is used for pick-and-place, grasps are detected, selected, and executed in the same way as outlined above. The difference is that once a grasp has been selected, we

measure the distance between the grasp and the surface below the object that the grasp is on to determine at which vertical position to place the object. Once the grasp has been executed, the user can freely move the scooter to a position in front of the surface onto which they want to place the object. For the placement pose, the orientation is set to the same orientation as in the pick pose, and the position is calculated from the laser pointer detection on the placement surface and the distance measured previously at the pick.

8.4 Experiments

We conducted four experiments to evaluate our proposed system. The first two experiments compare the performance of the system presented in this chapter to the one in the previous chapter, in terms of grasping in isolation and grasping in-situ. The third experiment investigates pick-and-place capabilities in a more challenging scenario. The fourth experiment compares our automatic and manual grasp selection.

We use OpenRAVE [26] for collision checking and motion planning. To generate a motion plan, we first attempt a linear trajectory. If this fails, we use a trajectory optimization method, TrajOpt [103]. For grasping, our system iterates through the grasps in order of the value of our cost function from Equation 8.1 and tries to compute a collision-free motion plan for the complete grasping task, i.e., reaching, grasping, lifting, and dropping. We execute the first grasp for which such a plan can be found.

Our object set consists of the 30 novel objects shown in Figure 8.4. These objects are chosen to be similar to the ones used for the experiments in the previous chapter. Note that the opening width limits of the Robotiq 2F-85 gripper enable to grasp objects with a much larger variety of sizes than the Baxter gripper used in the previous chapter.

Figure 8.4: The 30 novel objects used in our experiments.

8.4.1 Evaluating Grasping in Isolation

This experiment compares the grasping capabilities of our system to our prior work, as described in the previous chapter. We performed 15 trials in a tabletop scenario on a 46 cm tall table while the scooter was stationary. For each trial, six objects out of the 30 test objects were randomly selected and placed on the table in randomly selected positions. The objects were required to be placed such that they were at least two centimeters away from each other and such that they were in an upright orientation. These trials were run with automatic grasp selection in *pick-and-drop* mode by an expert user. The expert user determined the order in which the objects were grasped. We evaluated the detection success rate, grasp success rate, and the required time. We define detection success as the selected grasp being on the desired object. We define grasp success as the robot grasping and transporting the desired object to a handlebar-mounted basket. For time measurements, we only average over successful detections, grasps, plans, etc. We do not include time spent by the user to interact with the system, e.g., time spent by the user to press a key.

The results of this experiment are given in Table 8.1. Over 104 attempts for the 90 objects to be detected and grasped, we obtained a 92.3% laser pointer detection success rate (96 out of 104) and a 93.8% grasp success rate (90 out of 96). Compared to the system presented in the previous chapter, this improves the detection success rate by 4.5% and the grasp success rate by 4.2%.

The runtime of our system is shown in Table 8.2. On average, it took 51.7s to perform a grasp. Note that half of the time was required by the robot to execute the reaching motion. This runtime is signifcantly lower than the one for our previous system.

8.4.2 Evaluating Grasping In-Situ

To characterize the performance of our system in a real-world scenario, we used the open kitchen area shown in Fig. 8.5a. We performed five trials. In each trial, ten objects were randomly selected from the 30 novel objects and placed in randomly chosen positions: three objects were placed on a 73 cm tall table, three on a 31 cm tall table, two on the top shelf of a bookshelf (100 cm high), and two on the middle shelf of the bookshelf (69 cm high). Certain objects were not allowed to be placed on the middle shelf as they could only be grasped from the top down and the arm could not fit inside the shelf. During each trial, we used automatic grasp selection in pick-and-drop mode and the system was operated by an expert user. The sequence in which the objects had to be grasped was randomly generated

	Chapter 7	**This Chapter**
Grasp Success Rate	89.6% (78/87)	93.8% (90/96)
Detection Success Rate	87.8% (108/123)	92.3% (96/104)

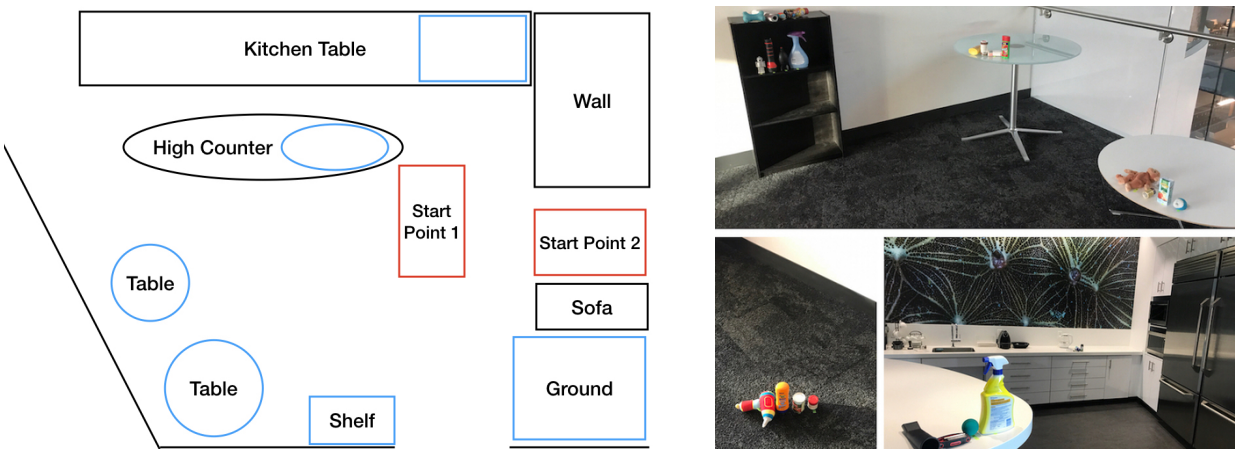
Table 8.1: Comparison between our two robotic systems for grasping in isolation.

	Avg.	Std.
Acquire Point Cloud	8.0s	2.4s
Detect Laser Pointer	5.8s	5.3s
Detect Grasp Pose	7.4s	2.3s
Grasp Filter	1.6s	0.6s
Plan Motion	2.4s	2.8s
Execute	24.2s	1.1s
Attempt Total	51.7s	7.2s

Table 8.2: Runtime for grasping in isolation.

for each trial. For each object in the sequence, the scooter was first driven to *Start Point 1* (see Fig. 8.5b) and then up to the desired object. We allowed the user to trigger the system several times (until the target object was retrieved). To compare to the system in the previous chapter, we use the same evaluation metrics for this experiment.

The results of this experiment are given in Table 8.3. Out of the 50 tasks (5 trials each with 10 objects), 48 succeeded, giving us a task success rate of 96%. The two failures were due to objects falling over on the middle shelf during the experiment, making it impossible for the system to grasp them. Our system has a 10% higher task success rate and a 24% lower failure rate than the system from the previous chapter. Out of the 58 attempts, the system failed 6 times to generate a feasible grasp to be executed (because of a failure in grasp detection, IK solving, or motion planning). This is a 6.2% improvement compared to



(a) (b)

Figure 8.5: (a) The kitchen layout for in-situ experiments. (b) Setup for one experimental trial in the kitchen environment.

	System from Chapter 7	**This Chapter**
Task SR	86.0% (43/50)	96.0%(48/50)

Failure Type	**System from Chapter 7**	**This Chapter**
Detect Laser	7.6% (5/66)	10.3% (7/68)
Lost Track	15.1% (14/93)	n.a.
Plan Grasp	16.5% (13/79)	10.3% (6/58)
Wrong Object	3.0% (2/66)	4.4% (3/68)
Execute Grasp	27.9% (17/61)	4.0% (2/50)

Table 8.3: Comparison between our two robotic systems for grasping in-situ.

	Chapter 7		This Chapter	
	Avg.	Std.	Avg.	Std.
Acquire Point Cloud	-	-	8.6s	2.7s
Detect Laser Pointer	-	-	6.0s	3.6s
Detect Grasp Pose	-	-	4.7s	1.7s
Filter Grasp	-	-	1.6s	0.9s
Plan Motion	-	-	2.9s	2.9s
Execute	-	-	24.3s	1.1s
Attempt Total	-	-	50.7s	5.7s
Driving	-	-	11.4s	4.7s
System Task Total	-	-	57.7s	15.3s
Task Total	128s	99s	69.0s	18.4s

Table 8.4: Runtime comparison between our two robotic systems for grasping in-situ.

the prior system. The prior system had a 15.1% track loss, while the current system has no track loss because it does not require metric SLAM. Our system has a similar performance as the baseline with respect to wrong object failures, i.e., the robot grasps an object that was not selected by the user.

The average runtime for this experiment is shown in Table 8.4. The total time to complete one succesful attempt was 50.7s. When failed attempts and driving the scooter are included, the time increases to 69s. This is a significant improvement compared to the 128s time reported for the prior system.

8.4.3 Evaluating Pick-and-Place In-Situ

This experiment investigates the pick-and-place performance of our system. We consider a more complex version of the kitchen environment from the previous experiment. Along with the shelf and the two tables, there was also a 108cm tall high table, a 87cm tall kitchen counter, and a ground area. On each of these seven surfaces we put four numbered tags 7cm away from each other. We performed three trials. In each trial, 28 objects were randomly sampled from the object set, labeled from 1 to 28, and placed on top of the corresponding tag. A coin flip determined the orientation of the objects: 1 = upright, 0 = not upright. A setup for one trial is depicted in Figure 8.5b. We assigned a random target surface to each object. The order in which the objects were picked up and placed was determined randomly. An expert user operated the system during these trials and manual grasp selection was used to choose a grasp. At the start of each pick-and-place task, the user drove the scooter to *Start Point 2* (see Figure 8.5a) and then up to the next object to be grasped. Once the object had been grasped, the user drove the scooter in front of the target surface and then the object was placed on top of it by the system. During the experiment, the user was permitted to perform pick-and-place actions to separate the target object from nearby objects. Figure **??** shows our system grasping an object in this experiment. We use the same evaluation metrics as for the previous experiment.

Table 8.5 presents the results of this experiment. Out of 84 pick-and-place tasks (3 trials with 28 tasks each), 81 succeeded, resulting in a 96.4% task success rate. In 13 of the succesful tasks, it was necessary to move a nearby object out of the way to reach the desired object. Compared to the previous experiment, the laser detection success rate decreased. The main reason for this is that this experiment was run at night so that there was less interference from sunlight. Compared to the previous experiment, the grasp success rate is similar. However, our system suffered from a larger number of IK, planning or execution failures. The main reason for these failures is the larger complexity of the environment compared to the previous experiment. In particular, the high counter top was challenging because it is close to the workspace limits of the robot arm. Because of the manual grasp

Task SR	96.4%(81/84)

Failure Type	Grasping	Placing
Detect Laser	2.0%(3/150)	1.0%(1/104)
Detect Grasp	8.0%(12/150)	-
IK/Planning	20.0%(27/135)	7.8%(8/103)
Wrong Object	0%(0/135)	-
Execute	12.0%(13/108)	3.2%(3/95)

Table 8.5: Results for pick-and-place in-situ.

	Avg.	Std.
System Task Total	129.9s	75.6s

	Grasping		Placing	
	Avg.	Std.	Avg.	Std.
Acquire Cloud	8.3s	2.5s	9.5s	4.4s
Detect Laser Pointer	6.0s	2.2s	4.7s	3.0s
Detect Grasp Pose	3.7s	1.6s	-	-
Grasp Filter	4.9s	2.6s	-	-
Grasp Selection	2.5s	2.6s	-	-
Motion Planning	7.5s	6.4s	1.1s	5.0s
Execution	17.1s	3.9s	15.6s	1.5s
Attempt Total	53.3s	9.5s	35.1s	10.2s

Table 8.6: Runtime for pick-and-place in-situ.

selection, there were no wrong object failures.

Table 8.6 reports the runtimes for this experiment. The total time for a pick-and-place task was 130s, excluding the time for driving the scooter. Note that this time is longer than the sum of the times required for grasping and placing in Table 8.6 because sometimes it required multiple pick-and-place attempts to solve a task.

8.4.4 Automatic vs Manual Grasp Selection

In this experiment, we compare automatic and manual grasp selection. We use the same setup as in Section 8.4.1 and perform 50 trials. In each trial, we selected two objects randomly from our set of novel objects and placed both of them next to each other on the table. One of the two objects was then randomly chosen as the target object and an expert user then operated our system to perform one pick-and-drop task on the target object with automatic grasp selection and another one with manual grasp selection. One attempt to was allowed for each mode to pick up the target object. We measured the wrong object and grasp failure rates.

Table 8.7 gives the results of this experiment. The wrong object failure rate for manual grasp selection was 20% lower than for automatic grasp selection. The reason for tihs is that manual selection enables the user to choose on-target grasps when they are available. In contrast to that, automatic selection may decide for a grasp that is not on the target object but has a higher objective function value. Even so, there were three failures where our system only detected grasps on the wrong object for the case of manual selection. The grasp execution failure rate is 6% lower for automatic selection. The reason for this is that our system had to sometimes execute a challenging grasp, e.g., one that could likely result

	Automatic Selection	Manual Selection
Wrong Object Failure	26.0% (13/50)	8.0% (4/50)
Grasp Execution Failure	6.0% (3/50)	14.0% (7/50)

Table 8.7: Comparison of automatic and manual grasp selection.

in a collision, in manual mode because no other grasps were available. Comparatively, the automatic mode basically selected grasps on the wrong object which were easier to execute for the robot arm.

8.5 Discussion

In this chapter, we proposed a robotic system for assistive pick-and-place that makes several improvements over the system proposed in the previous chapter. With respect to hardware, we use a more accurate and smaller robot arm. Our vision system uses a set of fixed cameras which take a quick one-shot observation of the environment instead of having to perform a much slower arm trajectory to collect a number of point clouds from multiple viewpoints with a wrist-mounted camera and fusing them using a visual SLAM method.

With respect to software, we detect the position of two close-by laser points by combining the estimated detections from multiple images. This makes the detection more robust, as evidenced by the improvement in the laser detection success rate that we observed in our experiments for this chapter. With regard to grasping, we use segmentation to more clearly identify the object to be grasped and propose a cost function based on heuristics to select a grasp that is easy and fast to execute. As evidenced in our experiments, this results in an improved grasp success rate.

While the previous system was only able to perform picking, the current system is also able to perform simple pick-and-place tasks where the user selects a target position for the grasped object with the laser pointer. This provides a baseline for subsequent systems that build up on this work. However, the pick-and-place tasks we consider are simple: we only consider the target position of the placement and mostly ignore the object's orientation. An important capability to add to our system would be allow the user to choose a target orientation for the placement and enable our system to reorient the object accordingly. Besides, the robot currently cannot maintain a fixed object orientation while moving the arm once an object has been grasped. This could be a problem in some manipulation tasks, e.g., if the robot was supposed to transport a mug filled with coffee.

The two most common failure modes of our system are: (i) not finding a motion plan, and (ii) not detecting the desired object with the laser pointer. The first failure mode could be addressed by using a different planner. The second failure mode could be addressed by developing a better laser pointer detection algorithm. One way to do this might be to use

RGBD sensors, such as the Intel Real Sense, to be able to exploit color as an additional source of information.

While the runtime of our system has signficantly improved compared to our prior system, it could be reduced in multiple ways. Currently, point cloud acquisition, laser pointer detection, grasp pose detection, and robot arm motions take up most of the time. To avoid infrared interference that causes the point clouds to be very noisy and have a large amount of missing depth data, we currently switch the sensors on and off, which takes a lot of time. This problem could be tackled by using stereo cameras which do not use infrared and therefore are not required to be turned on and off. Using faster hardware, e.g., CPUs with more cores, would decrease grasp detection runtime.

Another avenue for improvement is our user interface. The projector could be used for visualizing more than just the current target. For example, it could display the workspace of the arm or the location of a planned grasp. At an extreme, the monitor could be removed and all visual feedback for the user could be provided by the projector through augmented reality.

A further avenue for future research are user studies. So far, we have only operated our system with an expert user. As this system's ultimate goal is to be operated by people with different motor abilities, a variety of user interfaces with different access methods could be explored. We would need to figure out which interfaces work generally well and which are best suited to specific user demographics. This could result in a robust assistive system that can be used by a large variety of users.

8.6 Related Work

Research on assistive robotic systems for manipulation tasks has mainly been focused on human-robot interaction (HRI). Typically, existing systems lack the ability to autonomously grasp and place unknown objects in an unstructured environment. Martens et al. developed a semi-autonomous robotic system, composed of a robot arm mounted on a wheelchair with a speech interface [75]. Visual servoing techniques to grasp known objects are used in their approach. Kemp et al. created a system with a laser pointer interface to allow a user to select an object in the world that their mobile robot would retrieve [56, 46]. They report an 88.9% (32/36) grasp success rate in experiments where objects can be located on two different surfaces. However, the evaluation was restricted to objects placed in isolation and top-down grasps. Arhic et al. combined a hybrid brain-computer interface and an electric wheelchair equipped with a robotic arm for navigation and manipulation tasks [1]. However, they only evaluted pre-programmed robot arm motions. Pathirage et al. presented a similar system but they rely on a database of object models for grasping [88]. Grice et al. developed an assistive robotic system where a PR2 robot was teleoperated for manipulation tasks via a web browser and a single-button mouse [35]. Their system required direct control of the robot's end-effector. This makes their system much harder to use than a system that is able

to autonomously grasp objects. Compared to the literature, the system described in this chapter is able to pick and place novel objects in unstructured environments.

Part IV

Conclusion

Chapter 9

Conclusion

We studied the robotic grasp perception problem under partial visibility and without access to object models. Four methods for 6-DOF grasp pose detection were presented, including methods based on geometry and supervised learning. We also described two robotic systems which use one of these methods to detect grasp configurations.

All algorithms presented in this dissertation follow the same algorithmic structure. First, we sample robot hand poses from a subset of $SE(3)$. Second, we classify the poses based on the geometry of the sensor data or with a machine learning model that was trained with supervision. The key challenge for the first step is how to sample efficiently from $SE(3)$ which is a continuous, high-dimensional space. We addressed this challenge by generating grasp poses for which the robot hand is oriented such that its forward axis is orthogonal to the minor curvature axis of the object surface to be grasped. For the second step, the key challenge is how to represent the hand poses such that we can infer if they correspond to a good grasp. We introduced different grasp respresentations for both steps. These representations enable the use of classification models to make accurate and efficient predictions about the corresponding grasps. Another challenge is how to generate data to train classifiers on these representations. To tackle this challenge, we introduced an approach to generate ground truth grasp labels from real world data and others to label grasps from synthetic, publicly available data.

In evaluations on physical robots for objects presented in isolation and in clutter, we characterized performance for detecting grasp poses from point clouds which were taken with one or multiple depth sensors. We demonstrated that our algorithms can achieve high grasp success rates on a large variety of objects. Furthermore, we showed that adding views can improve grasp performance. The high success rates which we observed for these experiments indicate that our methods can generalize to other robot arms and hands. Moreover, this result suggests that our method can be used for manipulation tasks that require grasping as a prerequisite.

Our development and evaluation of two robotic systems for assistive grasping, aimed at people with motor disabilities, emphasizes that our grasp detection methods are useful in practice. These systems are a stepping stone toward enabling people with motor or cognitive

disabilities to be assisted in activities of daily living. In particular, they can autonomously pick up unknown objects in household scenarios.

Finally, we consider it an important contribution to the robotics community that our methods are available to the public under open-source licenses. Basically, this allows others to use our algorithms on their own robots (given that they have a depth sensor, a robot arm, and a robot hand at their disposal). Besides, these implementations can serve as baselines for other work.

9.1 Opportunities for Future Research

While this dissertation provides a basis for robotic manipulation tasks which require grasping, there are many directions for future research. Here, we briefly outline a few of those opportunities which we consider to be most promising and relevant to advance the field of robotic grasping and which are closely related to the work done for this dissertation.

9.1.1 Multi-Fingered Grasps

The methods introduced in this work consider enveloping and two-finger grasp configurations. However, there is a large variety of other grasps that could be detected. In particular, multi-fingered robot hands may allow to perform many different grasps. While we do not explicitly study multi-fingered hands here, we expect that our algorithms can be adapted to such hands by replacing the two-finger hand model that we use with a multi-fingered model for the particular hand under consideration. Such an adaption might increase the size of the space of possible grasps because the degrees of freedom increase with the number of fingers. Deep learning for multi-fingered robotic grasping has been studied in the literature (see, e.g., Varley et al. [111], Lu et al. [73], or Wu et al. [114]).

9.1.2 Deformable Objects

While we sometimes used deformable objects for physical robot experiments, we implicitly made the assumption that all objects are rigid when designing our algorithms. As long as the deformation that happens to the object while it is being grasped does not change its shape in such a way that it slips out of the robot hand, our algorithms can provide grasp poses for such objects. Generally, however, deformation should be taken into account, in particular as it might change the object pose relative to the hand. Thhis can affect subsequent manipulation actions that one wants to perform the robot after the grasp.

9.1.3 Additional Sensor Modalities

No method proposed in this dissertation considers any other sensor modality than vision. We intuitively expect additional sensor modalities to improve grasp detection similar to how

additional views make visual sensor data more useful. This should decrease uncertainty about the geometry of the objects to be grasped. It can also enable grasping of objects that are not or barely visible with a depth sensor. An immediate extension would be color (RGB) data that is readily available together with aligned depth data for many modern 3D cameras. Channels with local color information could be easily added to the image representation of grasp poses that we introduced.

Another sensor modality that has been widely studied in the robotic grasping literature comes from tactile sensors. Recently developed tactile sensors such as GelSight [67], GelSlim [27], and Soft-bubble [3] can provide data about the geometry of an object surface in contact with much higher spatial resolution than traditonal tactile sensors which typically give low-dimensional and spare data.

9.1.4 Task- and Object-Specific Grasping

We do not consider object identity for most of the work done for this dissertation. This problem has been studied in the literature. Methods typically combine detection for task relevant object regions with grasp detection [24]. However, methods that learn both at the same time in an end-to-end fashion could be explored in context with this dissertation. Furthermore, only learning task relevant regions is not sufficient for the actual physical execution of most tasks. Finger arrangement and the type of grasp matters for many tasks, such as operating a drill, in particular for tasks which require a multi-fingered hand. Here, one could try to learn task-specific grasps for which the tasks are evaluated in a robot simulator.

Another promising direction is to grasp objects in specific ways, e.g., to always grasp specific objects from the same approach direction.

9.1.5 Learning to Cluster Grasps

We presented a number of strategies to cluster grasps which are spatially close to each other. This type of clustering can be used to eliminate outliers and increase robustness of grasp detection. Nevertheless, the strategies which we employed are purely geometric and relatively simple. This leaves an opportunity for advanced approaches to clustering grasps, e.g., RANSAC [30] or machine learning. Apart from filtering outliers, one could also use the cluster to infer additional grasps, e.g., for some part of an object that is not visible. To give a more detailed example, consider a partially visible handle of a screwdriver, for which grasps have been detected on the visible parts and we could use these to infer grasps on the not visible handle part by considering the geometry of the cluster that the grasps form.

Bibliography

[1] F. Achic, J. Montero, C. Penaloza, and F. Cuellar. "Hybrid bci system to operate an electric wheelchair and a robotic arm for navigation and manipulation tasks". In: *2016 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*. 2016, pp. 249–254.

[2] Arsalan Mousavian Adithyavairavan Murali, Clemens Eppner, Chris Paxton, and Dieter Fox. "6-dof grasping for target-driven object manipulation in clutter". In: *Int. Conf. on Robotics and Automation (ICRA)*. 2020.

[3] Alex Alspach, Kunimatsu Hashimoto, Naveen Kuppuswamy, and Russ Tedrake. "Soft-bubble: a highly compliant dense geometry tactile sensor for robot manipulation". In: *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. 2019, pp. 597–604.

[4] Ravi Balasubramanian, Ling Xu, Peter D. Brook, Joshua R. Smith, and Yoky Matsuoka. "Human-guided grasp measures improve grasp robustness on physical robot". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2294–2301.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: speeded up robust features". In: *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 404–417.

[6] Paul J Besl and Neil D McKay. "Method for registration of 3-d shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE. 1992, pp. 239–256.

[7] Antonio Bicchi and Vijay Kumar. "Robotic grasping and contact: a review". In: *IEEE International Conference on Robotics and Automation*. Vol. 1. 2000, pp. 348–353.

[8] A. Blake, M. Taylor, and A. Cox. "Grasping visual symmetry". In: *1993 (4th) International Conference on Computer Vision*. 1993, pp. 724–733.

[9] Cristian Bodnar, Adrian Li, Karol Hausman, Peter Pastor, and Mrinal Kalakrishnan. "Quantile qt-opt for risk-aware vision-based robotic grasping". In: *Proc. of Robotics: Science and Systems*. 2020.

[10] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. "A tutorial on the cross-entropy method". In: *ANNALS OF OPERATIONS RESEARCH* 134 (2004).

[11] J. Bohg, A. Morales, T. Asfour, and D. Kragic. "Data-driven grasp synthesis a survey". In: *IEEE Transactions on Robotics* 30.2 (Apr. 2014), pp. 289–309.

[12] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Association for Computing Machinery, 1992, pp. 144–152.

[13] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. "Collaborative grasp planning with multiple object representations". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2851–2858.

[14] B. Calli, M. Wisse, and P. Jonker. "Grasping of unknown objects via curvature maximization using active vision". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 995–1001.

[15] Lillian Chang, Joshua R. Smith, and Dieter Fox. "Interactive singulation of objects from a pile". In: *International Conference on Robotics and Automation (ICRA*. 2012.

[16] Roland T Chin and Charles R Dyer. "Model-based recognition in robot vision". In: *ACM Computing Surveys (CSUR)* 18.1 (1986), pp. 67–108.

[17] S. Chitta, E. Jones, M. Ciocarlie, and K. Hsiao. "Perception, planning, and execution for mobile manipulation in unstructured environments". In: *IEEE Robotics and Automation Magazine* 19.2 (2012), pp. 58–71.

[18] Young Sang Choi, Cressel D Anderson, Jonathan D Glass, and Charles C Kemp. "Laser pointers and a touch screen: intuitive interfaces for autonomous mobile manipulation for the motor impaired". In: *Int'l ACM SIGACCESS Conf. on Computers and Accessibility*. 2008, pp. 225–232.

[19] Alvaro Collet, Dmitry Berenson, Siddhartha S. Srinivasa, and Dave Ferguson. "Object recognition and full pose registration from a single image for robotic manipulation". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 48–55.

[20] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, pp. 886–893.

[21] Colin Davidson and Andrew Blake. "Error-tolerant visual planning of planar grasp". In: *Sixth International Conference on Computer Vision*. 1998, pp. 911–916.

[22] P. Delonge. "Computer optimization of deschamps' method and error cancellation in reflectometry". In: *IMEKO-Symposium on Microwave Measurement*. 1972, pp. 117–123.

[23] Renaud Detry, Carl Henrik Ek, Marianna Madry, and Danica Kragic. "Learning a dictionary of prototypical grasp-predicting parts from grasping experience". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 601–608.

[24] Renaud Detry, Jeremie Papon, and Larry Matthies. "Task-oriented grasping with semantic and geometric scene understanding". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017.

[25] Rosen Diankov. "Automated construction of robotic manipulation programs". PhD thesis. Carnegie Mellon University, Robotics Institute, Aug. 2010.

[26] Rosen Diankov and James Kuffner. *OpenRAVE: A Planning Architecture for Autonomous Robotics*. Tech. rep. CMU-RI-TR-08-34. Pittsburgh, PA: Carnegie Mellon University, July 2008.

[27] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. "Gelslim: a high-resolution, compact, robust, and calibrated tactile-sensing finger". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1927–1934.

[28] David Fischinger and Markus Vincze. "Empty the basket-a shape based learning approach for grasping piles of unknown objects". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 2051–2057.

[29] David Fischinger, Markus Vincze, and Yun Jiang. "Learning grasps for unknown objects in cluttered scenes". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 609–616.

[30] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395.

[31] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin Harcourt, 1979.

[32] Ross Girshick. "Fast r-cnn". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[34] J. Glover and S. Popovic. "Bingham procrustean alignment for object detection in clutter". In: *IEEE Int'l Conf. on Intelligent Robot Systems*. 2013, pp. 2158–2165.

[35] Phillip M Grice and Charles C Kemp. "Assistive mobile manipulation: designing for operators with motor impairments". In: *RSS 2016 Workshop on Socially and Physically Assistive Robotics for Humanity*. 2016.

[36] Marcus Gualtieri and Robert Platt Jr. "Learning 6-dof grasping and pick-place using attention focus". In: *2nd Annual Conf. on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proc. of Machine Learning Research. PMLR, 2018, pp. 477–486.

[37] Marcus Gualtieri, James Kuczynski, Abraham M. Shultz, Andreas ten Pas, Robert Platt Jr., and Holly A. Yanco. "Open world assistive grasping using laser selection". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4052–4057.

[38] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. "High precision grasp pose detection in dense clutter". In: *IEEE Int'l Conf. on Intelligent Robots and Systems*. 2016.

[39] Chris Harris and Mike Stephens. "A combined corner and edge detector". In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[40] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[41] Kelsey P. Hawkins. *Analytic Inverse Kinematics for the Universal RobotsUR-5/UR-10 Arms*. Tech. rep. Georgia Institute of Technology, 2013.

[42] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. "Template-based learning of grasp selection". In: *IEEE Int'l Conf. on Robotics and Automation (ICRA)*. 2012, pp. 2379–2384.

[43] Alexander Herzog, Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, Jeannette Bohg, Tamim Asfour, and Stefan Schaal. "Learning of grasp selection based on shape-templates". In: *Autonomous Robots* 36.1-2 (2014), pp. 51–65.

[44] Jingwei Huang, Hao Su, and Leonidas J. Guibas. "Robust watertight manifold surface generation method for shapenet models". In: *CoRR* abs/1802.01698 (2018). arXiv: 1802.01698. URL: http://arxiv.org/abs/1802.01698.

[45] Katsushi Ikeuchi, Berthold KP Horn, Shigemi Nagata, Tom Callahan, and Oded Feingold. *Picking up an Object from a Pile of Objects*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1983.

[46] Advait Jain and Charles C Kemp. "El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces". In: *Autonomous Robots* 28.1 (2010), p. 45.

[47] Siddarth Jain and Brenna Argall. "Grasp detection for assistive robotic manipulation". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2015–2021.

[48] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: convolutional architecture for fast feature embedding". In: *Proceedings of the 22Nd ACM International Conference on Multimedia*. MM '14. ACM, 2014, pp. 675–678.

[49] Y. Jiang, S. Moseson, and A. Saxena. "Efficient grasping from RGBD images: learning using a new rectangle representation". In: *IEEE Int'l Conference on Robotics and Automation*. 2011, pp. 3304–3311.

[50] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray. "Very high frame rate volumetric integration of depth images on mobile devices". In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (2015), pp. 1241–1250.

[51] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. "Scalable deep reinforcement learning for vision-based robotic manipulation". In: *Proc. of The 2nd Conf. on Robot Learning*. Vol. 87. 2018, pp. 651–673.

[52] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. "Leveraging big data for grasp planning". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4304–4311.

[53] I. Kåsa. "A curve fitting procedure and its error analysis". In: *IEEE Transactions on Instrumentation and Measurement* IM-25.1 (1976), pp. 8–14.

[54] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz. "Clearing a pile of unknown objects using interactive perception". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 154–161.

[55] Ben Kehoe, Akihiro Matsukawa, Sal Candido, James Kuffner, and Ken Goldberg. "Cloud-based robot grasping with the google object recognition engine". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 4263–4270.

[56] Charles C Kemp, Cressel D Anderson, Hai Nguyen, Alexander J Trevor, and Zhe Xu. "A point-and-click interface for the real world: laser designation of objects for mobile manipulation". In: *2008 ACM/IEEE Int'l Conf. on Human-Robot Interaction*. 2008, pp. 241–248.

[57] Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. "Real-time cad model matching for mobile manipulation and grasping". In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009, pp. 290–296.

[58] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib. "Grasping with application to an autonomous checkout robot". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2837–2844.

[59] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sendai, Japan, 2004, pp. 2149–2154.

[60] Marek Kopicki, Renaud Detry, Florian Schmidt, Christoph Borst, Rustam Stolkin, and Jeremy L. Wyatt. "Learning dextrous grasps that generalise to novel objects by combining hand and contact models". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 5358–5365.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: (2012), pp. 1097–1105.

[62] Oliver Kroemer, Emre Ugur, Erhan Oztop, and Jan Peters. "A kernel-based approach to direct action perception". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2605–2610.

[63] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*. Vol. 86. 11. 1998, pp. 2278–2324.

[64] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *Robotics: Science and Systems*. 2013.

[65] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724.

[66] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection". In: *Int. J. Robot. Res.* 37.4-5 (2018), pp. 421–436.

[67] Rui Li and Edward H. Adelson. "Sensing and recognizing surface textures using a gelsight sensor". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1241–1247.

[68] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. "Pointnetgpd: detecting grasp configurations from point sets". In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2019.

[69] Shuo Liu and Stefano Carpin. "A fast algorithm for grasp quality evaluation using the object wrench space". In: *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. 2015, pp. 558–563.

[70] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: single shot multibox detector". In: *ECCV*. Vol. 9905. 2016, pp. 21–37.

[71] D.G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, pp. 1150–1157.

[72] Q. Lu, M. Van der Merwe, B. Sundaralingam, and T. Hermans. "Multifingered grasp planning via inference in deep neural networks: outperforming sampling by learning differentiable models". In: 27.2 (2020), pp. 55–65.

[73] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. "Planning multi-fingered grasps as probabilistic inference in a learned deep network". In: *Robotics Research*. Springer International Publishing, 2020, pp. 455–472.

[74] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. "Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *Robotics: Science and Systems (RSS)*. 2017.

[75] C. Martens, N. Ruchel, O. Lang, O. Ivlev, and A. Graser. "A friend for assisting handicapped people". In: *IEEE Robotics and Automation Magazine* 8.1 (2001), pp. 57–65.

[76] MathWorks. *Train support vector machine (SVM) classifier for one-class and binary classification*. URL: https://www.mathworks.com/help/stats/fitcsvm.html (visited on 05/10/2021).

[77] Matthew Matl. *Pyrender*. 2019. URL: https://github.com/mmatl/pyrender.

[78] Matej Meško and Štefan Toth. "Laser spot detection". In: *Journal of Information, Control and Management Systems* (2013).

[79] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-dof graspnet: variational grasp generation for object manipulation". In: *Proc. of the 2019 IEEE Int. Conf. on Computer Vision (ICCV)*. Oct. 2019, pp. 2901–2910.

[80] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[81] Van-Duc Nguyen. "Constructing force-closure grasps". In: *The International Journal of Robotics Research* 7.3 (1988), pp. 3–16.

[82] Donald A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[83] Jia Pan, Sachin Chitta, and Dinesh Manocha. "Fcl: a general purpose library for collision and proximity queries." In: *ICRA*. IEEE, 2012, pp. 3859–3866.

[84] A. ten Pas and R. Platt. "Localizing handle-like grasp affordances in 3d point clouds". In: *Int'l Symposium on Experimental Robotics*. 2014, pp. 266–271.

[85] A. ten Pas and R. Platt. "Using geometry to detect grasp poses in 3d point clouds". In: *Proceedings of the International Symposium on Robotics Research*. 2015.

[86] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. "Grasp pose detection in point clouds". In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1455–1473.

[87] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "Pytorch: an imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[88] Indika Pathirage, Karan Khokar, Elijah Klay, Redwan Alqasemi, and Rajiv V. Dubey. "A vision based p300 brain computer interface for grasping using a wheelchair-mounted robotic arm". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (2013), pp. 188–193.

[89] H. Pham and Q. Pham. "A new approach to time-optimal path parameterization based on reachability analysis". In: 34.3 (2018), pp. 645–659.

[90] L. Pinto and A. Gupta. "Supersizing self-supervision: learning to grasp from 50k tries and 700 robot hours". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3406–3413.

[91] Florian T. Pokorny and Danica Kragic. "Classical grasp quality evaluation: new algorithms and theory". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 3493–3500.

[92] Victor Adrian Prisacariu, Olaf Kähler, Ming Ming Cheng, Carl Yuheng Ren, Julien Valentin, Philip HS Torr, Ian D Reid, and David W Murray. "A framework for the volumetric integration of depth images". In: *arXiv preprint arXiv:1410.0925* (2014).

[93] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "Pointnet++: deep hierarchical feature learning on point sets in a metric space". In: *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*. Dec. 2017, pp. 5105–5114.

[94] Yuzhe Qin, Rui Chen, Bernie Zhu, Meng Song, Jing Xu, and Hao Su. "S4g: amodal single-view single-shot se(3) grasp detection in cluttered scenes". In: *Conf. on Robot Learning*. Oct. 2019.

[95] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. *ROS: an open-source Robot Operating System*. 2009.

[96] Joseph Redmon and Anelia Angelova. "Real-time grasp detection using convolutional neural networks". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1316–1322.

[97] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 91–99.

[98] Lior Rokach and Oded Maimon. "Clustering methods". In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Boston, MA: Springer US, 2005, pp. 321–352.

[99] R.B. Rusu and S. Cousins. "3d is here: point cloud library (pcl)". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4.

[100] V. Satish, J. Mahler, and K. Goldberg. "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1357–1364.

[101] A. Saxena, J. Driemeyer, and A. Ng. "Robotic grasping of novel objects using vision". In: *International Journal of Robotics Research* 27.4 (2008), pp. 157–173.

[102] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. "Motion planning with sequential convex optimization and convex collision checking". In: *Int. J. Robot. Res.* 33.9 (2014), pp. 1251–1270.

[103] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. "Finding locally optimal, collision-free trajectories with sequential convex optimization". In: *Robotics: Science and Systems*. 2013, pp. 1–10.

[104] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. "Bigbird: a large-scale 3d database of object instances". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 509–516.

[105] G. Taubin. "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.11 (1991), pp. 1115–1138.

[106] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[107] Mostafa Vahedi and A. Frank van der Stappen. "Caging polygons with two and three fingers". In: *The International Journal of Robotics Research* 27.11-12 (2008), pp. 1308–1324.

[108] Nikolaus Vahrenkamp, Steven Wieland, Pedram Azad, David Gonzalez, Tamim Asfour, and Rudiger Dillmann. "Visual servoing for humanoid grasping and manipulation tasks". In: *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*. 2008, pp. 406–412.

[109] Mark Van der Merwe, Qingkai Lu, Balakumar Sundaralingam, Martin Matak, and Tucker Hermans. "Learning continuous 3d reconstructions for geometrically aware grasping". In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2020.

[110] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. "Shape completion enabled robotic grasping". In: *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 2442–2447.

[111] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. "Generating multi-fingered robotic grasps via deep learning". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 4415–4420.

[112] Dian Wang, Colin Kohler, Andreas ten Pas, Maozhi Liu, Holly A. Yanco, and Robert Platt. "A scooter-mounted robot arm to assist with activities of daily life". In: *International Symposium on Robotics Research*. 2019.

[113] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze. "3dnet: large-scale object class recognition from cad models". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 5384–5391.

[114] Bohan Wu, Iretiayo Akinola, Abhi Gupta, Feng Xu, Jacob Varley, David Watkins-Valls, and Peter Allen. "Generative attention learning: a "general" framework for high-performance multi-fingered grasping in clutter". In: *Autonomous Robots* (2020).

[115] Xinchen Yan, Jasmine Hsu, Mohammad Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. "Learning 6-dof grasping interaction via deep geometry-aware 3d representations". In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2018, pp. 1–9.

[116] Yu Zheng. "Computing the best grasp in a discrete point set". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2208–2214.

[117] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3d: a modern library for 3d data processing". In: *arXiv preprint arXiv:1801.09847* (2018).

[118] Yilun Zhou and Kris Hauser. "6dof grasp planning by optimizing a deep learning scoring function". In: *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact-Turning a Problem into a Solution*. Vol. 2. 2017.