# Integrating Vision and Physical Interaction for Discovery, Segmentation and Grasping of Unknown Objects

Zur Erlangung des akademischen Grades eines

**Doktors der Ingenieurwissenschaften**

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

genehmigte

**Dissertation**

von

**Dipl.-Inform. David Schiebener**

aus Siegburg

| | |
|---|---|
| Tag der mündlichen Prüfung: | 21. Juni 2016 |
| Referent: | Prof. Dr.-Ing. Tamim Asfour |
| Korreferent: | Prof. Dr.-Ing. Aleš Ude |

# Acknowledgements

First of all I want to thank my doctoral supervisor Prof. Tamim Asfour for giving me the opportunity to do research in this fascinating field. His unlimited enthusiasm about everything concerning humanoid robotics, his valuable advice, his great commitment and his support during the time of my thesis have been invaluable. The research group he has built up, with its robotic platforms as well as excellent researchers, is a great environment for the work I have been doing. I also want to thank Prof. Aleš Ude who supervised me in the first nine months of my beginning scientific development. He introduced me to the world of robotics research, taught me how to write papers, and gave me just the right mixture of freedom and good advice that I needed to develop and realize my first own ideas.

Still before that, Dr. Kai Welke supervised me when I wrote my study and diploma thesis. He and Dr. Pedram Azad taught me all the basics of computer vision and how to program robots, and it was that great time of working with Dr. Kai Welke that lead me to do my PhD in this area. To conclude the list of people who were a kind of mentor to me, I would like to thank Prof. Rüdiger Dillmann and Dr. Markus Przybylski, who gave me the first opportunity to take on responsibility in teaching by letting me co-organize the lecture and exams on cognitive systems.

The time here would not have been the same, and I would not have gotten so far, without my friends and colleagues at the institute. There is in particular Mirko Wächter, with whom I spent countless hours working on the robot and still enjoyed it even when the robot tried to make us go mad, and Simon Ottenhaus, with whom I worked together on the topic of reactive grasping, as well as with Julian Schill. I would like to thank Dr. Niko Vahrenkamp who, with his calmness and experience, is always an anchor of stability and happy to give good advice. He also provided the image in Figure 3.6. My office mates Dr. Eren Aksoy, Markus Grotz and Dr. Natsuki Yamanobe created a great atmosphere that I enjoyed to work in. I would like to thank Dr. Eren Aksoy in particular for tuning the LCCP segmentation for the comparative evaluation in section 6.1. My time in Ljubljana was made a pleasant experience in particular by Dr. Andrej Gams and (soon Dr.) Miha Deniša, who immediately offered me their help and friendship when I arrived and made sure I never felt lost in that wonderful country.

I am also thankful to the bachelor and master students that chose me as their supervisor in the last years. It was always a pleasure to share my knowledge and enthusiasm with them and watch them develop their own perspective and insights on all the interesting problems we discussed. In particular, I want to thank Andreas Schmidt, who implemented the object shape completion described in chapter 4 during his bachelor thesis ([Schmidt, 2015]), and who continued to work on that topic as a student assistant. The original versions of the images in Figure 4.1 and 4.2 were created by him. I also want to thank Sarah Puch who helped a lot in transferring the code for the interactive object segmentation to our new software framework during her master's thesis ([Puch, 2015]), and always made sure that I was happy (with cake, if necessary).

Finally, I want to thank my parents, who have always without hesitation supported me in everything I did, and have never left a doubt that they love me and are there for me. And I want to thank my sister, my wife Natalie, and all my wonderful friends that have given me their love and friendship throughout my life, and who have made me the lucky person I am today.

# Deutsche Zusammenfassung

## Überblick

In dieser Arbeit werden Verfahren der Bildverarbeitung und die Fähigkeit humanoider Roboter, mit ihrer Umgebung physisch zu interagieren, in engem Zusammenspiel eingesetzt, um unbekannte Objekte zu identifizieren, sie vom Hintergrund und anderen Objekten zu trennen, und letztendlich zu greifen. Im Verlauf dieser interaktiven Exploration werden außerdem Eigenschaften des Objektes wie etwa sein Aussehen und seine Form ermittelt.

Der Umgang mit Objekten, über die a priori keinerlei Information verfügbar ist, stellt eine besondere Herausforderung für technische Systeme dar. Ein unbekanntes Objekt zu segmentieren, d.h. als eigenständige physische Einheit zu identifizieren und von seiner im ungünstigsten Fall ebenfalls unbekannten und möglicherweise sehr komplexen Umgebung zu unterscheiden, ist allein aufgrund visueller Daten wie Kamera- oder Tiefensensorbildern im Allgemeinen nicht möglich. Durch die physische Interaktion mit unbekannten Objekten kann ein Roboter wichtige Informationen bezüglich dieser Objekte gewinnen. In dieser Arbeit wird gezeigt, wie dadurch die visuelle Segmentierung von nichtverformbaren Objekten selbst in vollständig unbekannten und extrem komplexen Szenen möglich wird.

Wenn durch die Segmentierung bekannt ist, welcher Teil der wahrgenommenen Umgebung ein eigenständiges Objekt ist, kann diese Information genutzt werden, um visuelle und Formdeskriptoren des Objektes zu späteren Wiedererkennung zu lernen. Darüber hinaus kann nun, da die Existenz eines bewegbaren Objektes und seine ungefähre Form bekannt sind, der Roboter versuchen, es zu greifen. Da die meisten Greifplanungsalgorithmen ein

vollständiges 3D-Modell benötigen, wird in dieser Arbeit ein Verfahren vorgestellt, wie aus dem wahrgenommenen Teil des Objektes ein vollständiges Modell geschätzt werden kann. Hierbei ist jedoch zu beachten, dass aufgrund des teilweise unsicheren Wissens über das Objekt Fehler während des Greifvorgangs auftreten können. Zur visuellen Detektion dieser Fehler sowie zu Strategien zum reaktiven Greifen liefert diese Arbeit ebenfalls einen Beitrag.

## Beiträge der Arbeit

Die Segmentierung von Objekten in Bildern, d.h. die Unterscheidung, welche Teile eines Bildes zu einem Objekt gehören und welche nicht, ist eines der klassischen Probleme der Bildverarbeitung. Es ist ein hilfreicher und teilweise unverzichtbarer Vorverarbeitungsschritt für Erkennung, Lokalisierung, visuelles Lernen und Manipulation von Objekten. Rein bildbasierte Segmentierungsverfahren können jedoch im Allgemeinen nicht sicher entscheiden, welche Teile einer aus unbekannten Objekten bestehenden, nichttrivialen Szene zum selben physischen Objekt gehören und welche nicht. In dieser Arbeit wird die Fähigkeit humanoider Roboter, mit ihrer Umgebung physisch zu interagieren, genutzt, um diese Beschränkung zu überwinden und Mehrdeutigkeiten in komplexen Szenen gezielt aufzulösen. Zu diesem Zweck werden zunächst anhand der Kamerabilder des Roboters Objekthypothesen erstellt. Eine dieser Hypothesen wird ausgewählt und der Roboter versucht, sie durch vorsichtiges Schieben zu bewegen. Wenn es sich bei dem angestoßenen Gegenstand tatsächlich um ein bewegbares Objekt handelt, so bewirkt diese Aktion des Roboters eine Veränderung der Szene. Durch diese Veränderung werden zusätzliche Informationen wahrnehmbar, die sich für die Verifikation und Korrektur der Segmentierung nutzen lassen. Dass sich Teile gleichartig bewegen, ist ein starker Indikator dafür, dass sie zum selben Objekt gehören. Deutlich unterschiedliche Bewegungen hingegen sprechen für die Zugehörigkeit zu verschiedenen Objekten. Eine konsistente Bewegung einer der ursprünglichen Objekthypothesen bestätigt, dass sie tatsächlich mit einem realen Objekt korrespondiert.

Wenn sich eine Objekthypothese bewegt hat und demnach zu einem realen Objekt gehört, kann sie als Grundlage einer Segmentierung dieses Objektes verwendet werden. Die einzelnen Bildpunkte können dann darauf überprüft werden, ob sie sich bei der Bewegung des Objekts konsistent mit der Gesamthypothese verhalten haben. Ist das der Fall, gelten sie als verifiziert, anderenfalls werden sie aus der Segmentierung entfernt. Angrenzende Punkte dagegen, die sich zusammen mit der Hypothese bewegt haben, werden als möglicherweise zum selben Objekt gehörend neu in die Hypothese aufgenommen.

Durch mehrfache Wiederholung des Schiebens eines Objektes und die darauffolgende Bestätigung bzw. Aussonderung sowie Ergänzung der zur Hypothese gehörenden Punkte kann eine korrigierte Segmentierung gewonnen werden, die zudem durch physische Interaktion als mit einem realen Objekt korrespondierend verifiziert ist. Diese Segmentierung erlaubt beispielsweise das Lernen visueller und formbasierter Objektdeskriptoren zur zukünftigen Wiedererkennung und Lokalisierung. Während ähnliche Arbeiten aus diesem Bereich relativ starke Annahmen über das Objekt machen (zumeist signifikante Texturierung) oder einen vergleichsweise einfachen Aufbau der Szene annehmen (einige Objekte auf einem ansonsten leeren Tisch), wird in dieser Arbeit nur vorausgesetzt, dass das Objekt durch Schieben bewegt werden kann, sich näherungsweise als starrer Körper bewegt und dabei das Aussehen seiner Oberfläche nicht ändert (also nicht beispielsweise transparent oder stark spiegelnd ist).

Im Anschluss soll das neuentdeckte und segmentierte Objekt gegriffen werden. Hierzu wird aus der Segmentierung ein vollständiges 3D-Modell geschätzt, auf das dann ein Greifplanungsalgorithmus angewandt werden kann. Aufgrund der unvollständigen und ungenauen Information über das neue Objekt ist damit zu rechnen, dass der geplante Griff nicht perfekt ist. Fehler beim Greifen äußern sich üblicherweise dadurch, dass ein Teil der Hand mit

dem Objekt kollidiert, bevor sie die vorgegebene Greifpose erreicht. Dadurch wird das Objekt verschoben oder die Hand erreicht nicht die beabsichtigte Pose, was zum Scheitern des Greifversuchs führen kann. Solche verfrühten Kollisionen muss der Roboter bemerken und darauf reagieren.

Klassischerweise erfolgt die Detektion durch taktile oder Kraftsensoren, die allerdings für leichte Objekte nicht sensitiv genug sind und im Fall von taktilen Sensoren meist nur einen Teil der Hand bedecken. Daher wurde in dieser Arbeit ein neues Verfahren zur Kollisionsdetektion entwickelt. Es detektiert die Bewegung des Objektes visuell und ergänzt so die kraftbasierten Sensoren. Hierbei wird der optische Fluss, d.h. die Bewegung der Bildpunkte von einem Kamerabild zum nächsten, während des Greifvorgangs beobachtet und zu Clustern mit ähnlicher Bewegung gruppiert. Dadurch kann festgestellt werden, ob in der Bildregion nahe der Hand in deren Bewegungsrichtung ein Cluster existiert, das sich anders bewegt als der Rest der Szene. Das lässt darauf schließen, dass die Hand eine Bewegung des Objektes verursacht hat. Die Hand und insbesondere die Fingerspitzen werden visuell verfolgt, und indem bestimmt wird, welcher Finger dem Objekt am nächsten und damit vermutlich für die Kollision verantwortlich ist, kann eine gezielte Korrektur der Handpose durchgeführt werden. Auf diese Weise wird die Griffkonfiguration während der Ausführung so lange reaktiv angepasst, bis die Hand das Objekt kollisionsfrei erreicht hat.

## Zusammenfassung

In dieser Arbeit wird also die visuelle Objektentdeckung und -segmentierung durch gezielte physische Interaktion des Roboters mit seiner Umgebung unterstützt und verifiziert. Die Segmentierung erlaubt, Aussehen und Form des Objektes, soweit es sichtbar ist, zu lernen. Mithilfe des unvollständigen Wissens über seine Form wird ein Griff geplant und ausgeführt, wobei während der Annäherung der Hand an das Objekt Kollisionen visuell detektiert und reaktive Korrekturen ausgeführt werden, bis der Griff erfolgreich ist.

Durch das enge Zusammenspiel von visueller Wahrnehmung und Interaktion mit der Umwelt können so Probleme, die für sich genommen äußerst schwierig sind, durch den Roboter robust gelöst werden.

# Abstract

This thesis integrates computer vision and the ability of humanoid robots to physically interact with their environment, in order to discover new, unknown objects, segment them from the background, and grasp them. During this interactive exploration, object properties like its visual appearance and its shape are learned.

Dealing with objects about which no prior knowledge is available is a very challenging task for artificial systems. Segmenting an unknown object, i.e. identifying it as an individual physical entity and separating it from its unknown and possibly very complex environment may, in general, be impossible based only on visual input from cameras or depth sensors. By means of physical interaction with unknown objects, a robot can obtain additional helpful information. This thesis shows how that allows the visual segmentation of rigid objects even in completely unknown and extremely complex scenes.

The result of the segmentation is the information which part of a perceived scene constitutes an object. This can be used to learn visual or shape descriptors of the object. Aditionally, as the robot now knows about the existence, location and shape of a movable object, it can attempt to grasp it. Most grasp planning algorithms require a complete model of the 3D shape of the object. Therefore, this thesis proposes a method for estimating a complete model based on the perceived parts of the object. However, as the knowledge about the object is partly uncertain, grasping may fail during execution. In this thesis, a method for visual detection of such errors and appropriate corrective reactions is presented.

# Contents

# 1 Introduction

## 1.1 Motivation

Over the last decade, robots have gained momentum expanding into more and more areas of application. After having been restricted to controlled factory environments for a long time, there is now an increasing range of tasks that robots fulfill in open, real world situations, from simple vacuum cleaning robots to autonomous cars. At the same time, there is continuous progress in the improvement of the skills needed by humanoid and service robots that are supposed to interact and work together with humans in their everyday environment, or support them in disaster scenarios like the one considered in the newest DARPA robotics challenge.

One key to enabling robots to act in an open, uncontrolled environment is their ability to handle new, unexpected situations that they were not explicitly prepared for. An important special case is when the robot encounters new, unknown objects. Without any prior knowledge about them, only the perceived information can be used when the robot has to interact with them. In such a situation, the robot should be able to autonomously explore the objects and familiarize itself with them, which means that it should learn e.g. to recognize them and how they can be manipulated.

This thesis contributes to the ability of humanoid robots to autonomously interact with new, unknown objects, learn about them, and grasp them. The visual perception of the robot plays a key role in those tasks, and is used in close interplay with the physical actions. This close integration of perception and explorative physical interaction is essential for identifying individual objects in uncontrolled, complex scenes, as well as for manipulating them with only uncertain and incomplete knowledge.

## 1.2 Overview

The goal of this thesis is to enable humanoid robots to autonomously discover and segment unknown objects in realistic, complex scenes, learn about them, and grasp them. Figure 1.1 gives a schematic overview of the approach that was developed within this thesis in order to achieve these goals.

The first crucial step when attempting to learn about a new object is to segment it from its environment, i.e. to determine which parts of the perceived scene belong to one specific object and which belong to something else. This is an extremely difficult task when the scene is complex and the robot has little or no knowledge about the objects in it. The reason why it is so difficult to realize this - very fundamental - competence in an artificial system is that the definition of what is an object is hard to formalize, and depends on semantics and context. Segmentation approaches which are based only on passive visual perception are unavoidably restricted to visual criteria, which are not always sufficient to correctly understand the physical structure of an observed scene.

This thesis proposes a novel approach in which the robot interacts physically with the objects in order to support their visual segmentation. This introduces additional information by making physical properties of the explored scene apparent to the visual perception. Initially, object hypotheses are generated based on the visual input. By careful pushing, one or more objects are caused to move. This motion is used to verify or discard the initial hypotheses, and thus confirm the existence of actual physical objects corresponding to some of these hypotheses. The motion is then used as an additional visual cue for object segmentation, as it allows to correct and improve the confirmed object hypotheses. The developed approach is presented in chapter 3.

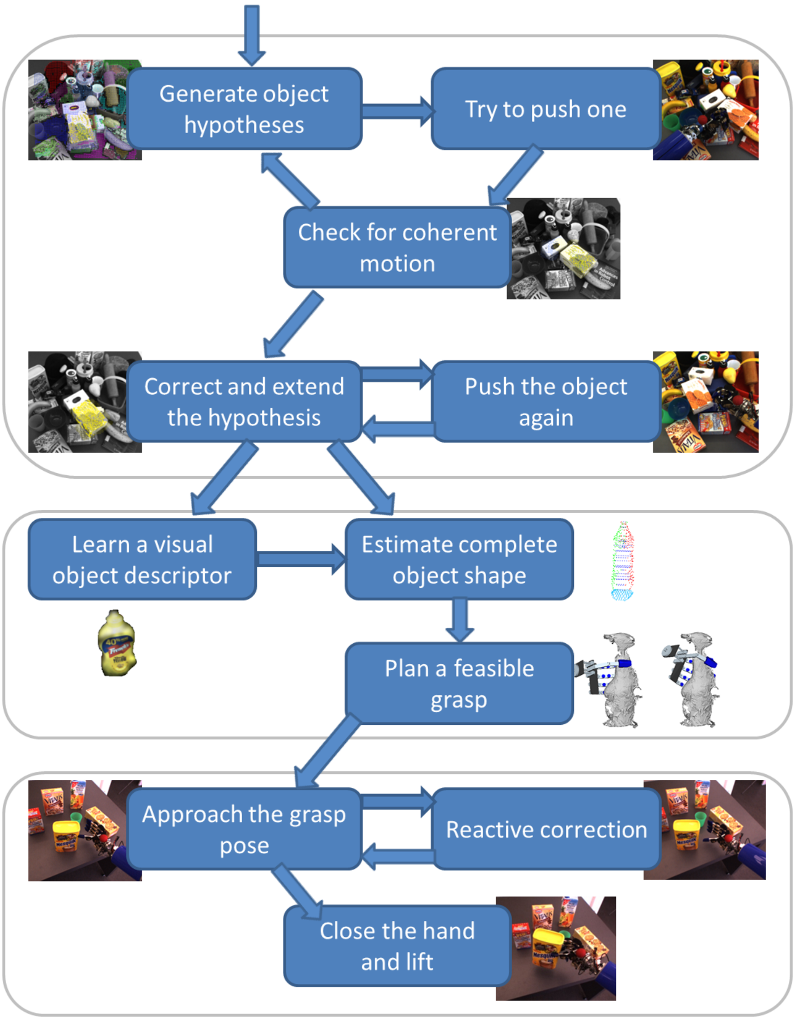

Generate object hypotheses
Try to push one
Check for coherent motion
Correct and extend the hypothesis
Push the object again
Learn a visual object descriptor
Estimate complete object shape
Plan a feasible grasp
Approach the grasp pose
Reactive correction
Close the hand and lift

Figure 1.1: Overview of the process for discovery, segmentation and grasping of unknown objects developed in this thesis.

When an object has been segmented, the robot knows which part of the perceived scene belongs to it, which allows to learn a visual descriptor for recognition. The segmentation also allows to estimate the shape of the object, which can be used to determine possible grasps. As most grasp planners require a complete shape model, it is necessary to estimate the entire shape of the object based the perceived part. To this end, a symmetry-based approach has been developed, which is detailed in chapter 4. Grasp planning on the resulting object model can then be performed with any state-of-the-art grasp planner.

It may happen though that the resulting planned grasp is not entirely suitable for the actual object, as the shape model was created based on real sensor data which may be noisy and incomplete. Additionally, imprecisions in perception and motion may lead to failure during the grasp execution. In such a case, it is necessary that the robot detects the failure, which is usually a premature collision of the grasping hand with the object. Such a collision can often be detected by tactile or force sensors, but in practice those are not always available on all parts of the robot. Furthermore, they may not have a sufficient sensitivity to detect collisions with e.g. very light objects that are easily pushed away without resisting.

To complement these sensors, a visual approach for collision detection is presented in chapter 5. By observing the hand of the robot and the optical flow in the camera images, this method detects when the robot causes the object to move. When that happens, a corrective movement is performed, and after one or more corrections the grasp can usually be concluded successfully.

# 2 State of the art

This chapter gives an overview of the fields that are related to this thesis. The first section provides an introduction to visual object segmentation and gives some chosen examples of recent publications on this topic. The second section covers the state of the art in interactive object segmentation, which is the core topic of this thesis. The third section describes different approaches to completing the three-dimensional shape of objects where only partial shape information is available. The fourth section gives an overview of the field of reactive grasping and reactive grasp correction.

## 2.1 Visual object segmentation

### 2.1.1 Problem statement

Visual object segmentation is the task of separating one or more objects from the other objects and/or the background in a given scene. The scene might be observed or given in the form of an image, a depth map, or a point cloud where each point has a 3D position and an RGB value. Traditionally, the problem has been considered almost exclusively for the case of camera images. In the last years, the increasing availability and popularity of combined RGB and depth cameras has caused a shift towards the usage of point clouds, which can be obtained by these sensors or by dense stereo matching when using stereo camera images. Given an image or a point cloud capturing a scene, the aim of object segmentation is to decide which of the pixels or points belong to the same real-world object.

### 2.1.2 Common approaches

While humans acquire the skill of visual object segmentation at a very early age ([Fitzpatrick et al., 2008]) and usually have no difficulties discerning objects from the background even in highly complex and cluttered scenes, recreating this ability in artificial systems has turned out to be extremely difficult. This is due to the very broad and often semantics- and context-dependent meaning of the word "object" ([Feldman, 2003]). Most approaches based on passive observation try to apply one or more geometric or perceptual principles as criteria to decide which elements of the perceived scene belong to the same real object.

These principles include e.g. connectedness ([Palmer and Rock, 1994], [Mishra et al., 2012]), bilateral symmetry ([Li and Kleeman, 2011]), coplanarity and co-linearity of contours ([Kraft et al., 2008]), convexity ([Stein et al., 2014]), visual similarity ([Alpert et al., 2012], [Kootstra et al., 2010]), smooth curvature ([Richtsfeld et al., 2012]) etc. Often, these approaches are realized in the form of a bottom-up region growing where neighboring regions are fused as long as they fulfill the respective criterion, or they formulate the segmentation as a graph-cut problem where the pixels/points or regions are nodes which, if they are adjacent, are connected by edges that are weighted with some kind of similarity measure. Both concepts implicitly presume that the objects fulfill the connectedness property.

The following two subsections explain in more detail an approach based on bottom-up region growing ([Alpert et al., 2012]) and on graph cut ([Mishra et al., 2012]), respectively. After that, a recent publication based on local convexity that makes explicit use of 3D information ([Stein et al., 2014]) is described as an example for the most recent research trends.

### 2.1.3 Image Segmentation by Probabilistic Bottom-Up Aggregation and Cue Integration

In [Alpert et al., 2012], image pixels and later regions are gradually merged together in a sequence of steps to create larger coherent regions. The question whether two regions should be merged is answered based on different similarity cues which are weighted following a *mixture of experts* formalism. This is a probabilistic framework that dynamically assigns weights to different cues depending on how meaningful they are given the current observation.

The image is considered to be divided into a set of regions

$$R = \{R_1, R_2, ..., R_n\}$$

together with a set of observations $\vec{\mathtt{H}}_i \in R_d$ for each region $R_i$. Following the *mixture of experts* formalism, the decision whether or not two regions $R_i$ and $R_j$ should be merged together is indicated by a binary variable $s_{ij}$ which takes on the value $s_{ij}^+$ if the regions should be merged and $s_{ij}^-$ otherwise. Based on the observations $H_i$ and $H_j$, i.e. the visual cues calculated in the two regions, the probability $P(s_{ij}^+|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j)$ is calculated as

$$\begin{aligned} P(s_{ij}^+|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j) &= \sum_k P(s_{ij}^+, c_k|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j) \\ &= \sum_k P(s_{ij}^+|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j, c_k) P(c_k|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j) \end{aligned}$$

Thus, the probability that $R_i$ and $R_j$ should be merged is the sum over the merge probabilities $P(s_{ij}^+|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j, c_k)$ for the individual cues $c_k$ weighted with a term $P(c_k|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j)$ which encodes the local and dynamic influence of the cues. The cues $c_k$ can be color, intensity, texture, boundary continuity etc. In this publication, only intensity and texture are used.

Figure 2.1 demonstrates how different cues can have different discriminative power which is incorporated by the dynamic influence $P(c_k|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j)$.

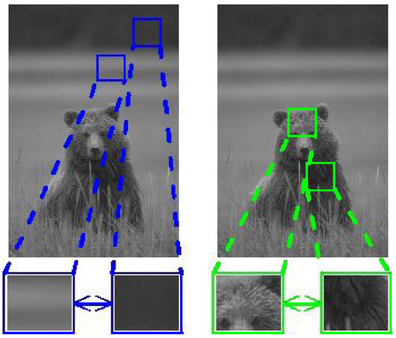


Figure 2.1: An example of a scene where different similarity cues should be used when comparing different region pairs (from [Alpert et al., 2012]).

.

While the blue patches highlighted in the left picture can be clearly distinguished by their intensity, the green patches in the right image differ in texture. Hence, the intensity cue should have a higher weight when deciding whether or not to merge the two regions marked on the left, and the texture should have a higher weight in the case of the regions highlighted on the right side.

As the authors only use intensity and texture, the weighting of these two cues can be determined by quantifying the texturedness of the patches that are considered. For each region a 256-bin histogram of local gradient magnitudes $G^i$ is computed. The histograms of textured regions will be rather dense, while untextured regions yield sparse histograms.

The sparseness $S_i$ of the histogram $G^i$ is quantified by the quotient of its $l_1$ and $l_2$ norm. For the individual regions, the values

$$P(c_1|\vec{\mathtt{H}}_i) = \frac{1}{(1 - e^{-(aS_i+b)})}$$

are calculated, and for the two examined regions $i, j$ the weight $p(c_1|\vec{\mathtt{H}}_i, \vec{\mathtt{H}}_j)$ of the intensity cue is the minimum of $(P(c_1|\vec{\mathtt{H}}_i)$ and $P(c_1|\vec{\mathtt{H}}_j))$.

To compute the merge probability contributed by each cue $c_k$, Bayes' formula can be applied to the posteriors such that

$$P(s_{ij}^{+}|\vec{\mathtt{H}}_i,\vec{\mathtt{H}}_j,c_k) = \frac{L_{ij}^{+}P(s_{ij}^{+}|c_k)}{L_{ij}^{+}P(s_{ij}^{+}|c_k)+L_{ij}^{-}P(s_{ij}^{-}|c_k)}.$$

$L_{ij}^{\pm} \triangleq p(\vec{\mathtt{H}}_i,\vec{\mathtt{H}}_j|s_{ij}^{\pm},c_k)$ denote the likelihood densities given $s_{ij}^{\pm}$ respectively which are estimated from the local conditions of the surrounding regions, and $P(s_{ij}|c_k)$ is a prior (see further below).

For the intensity cue, its corresponding likelihood can be modeled as

$$L_{ij}^{\pm} = p(\Delta_{ij}|s_{ij}^{\pm}) = \mathtt{N}(0,\sigma_{ij}^{\pm}),$$

i.e. it is given by a zero-mean Gaussian over the average intensity difference $\Delta_{ij} = \bar{I}_i - \bar{I}_j$ of the regions $R_i$ and $R_j$. $\sigma_{ij}^{\pm}$ is defined as $\sigma_{ij}^{\pm} = \sigma_{local}^{\pm} + \sigma_{scale}$. Here, the local term $\sigma_{local}^{\pm}$ is estimated using the intensity differences of adjacent regions. $\sigma_{scale}$ accounts for the effects of noise that is relatively big for small regions but small for larger ones.

When considering the texture cue, a bank of edge filters is applied to the regions $R_i$ and their absolute responses are stored in a histogram $\mathbf{h}_i \in \vec{\mathtt{H}}_i$. The difference between two histograms $\mathbf{h}_i$ and $\mathbf{h}_j$ is measured using the $\chi^2$ distance

$$D_{ij} = \sum_k \left(\frac{\mathbf{h}_i(k)-\mathbf{h}_j(k)}{\mathbf{h}_i(k)+\mathbf{h}_j(k)}\right)^2,$$

and the likelihood $L_{ij}$ corresponds to the probability of this difference given a certain random noise.

The prior $P(s_{ij}|c_k)$ is determined by means of the geometrical properties of the regions and is relative to the length of their common boundary, i.e. it is independent of the cue $c_k$. With $\tau_{ij}$ being the length of the common boundary of two regions $R_i$ and $R_j$, the prior is estimated by

$$P(s_{ij}^{\pm}) = \frac{\tau_{ij}}{\min(\sum_k \tau_{ik}, \sum_k \tau_{jk})}.$$

This probabilistic framework can be used within any merge algorithm for segmentation. The authors chose the *Segmentation by Weighted Aggregation* algorithm ([Galun et al., 2003]), which creates a hierarchy of increasingly coarser graphs starting with single pixels as nodes.

### 2.1.4 Active Visual Segmentation

In [Mishra et al., 2012], the segmentation problem is simplified to a distinction between a single object and the background. It requires an initial starting point on the object to be segmented which they call a fixation point. With these restrictions, the segmentation task can be transformed into a well-posed binary labeling problem. The central idea is to transform the image into a polar space representation with the fixation point as its origin, and apply a graph cut algorithm to find an optimal boundary contour.

The main means to find optimal object boundaries are the edges detected in the image. Here it is crucial that the found edges should correspond to actual object contours and not to internal edges. Thus, applying e.g. the traditional Canny edge detector to the intensity or color channels is insufficient, as it returns the inner boundaries caused by texture as well as the object contours. A gradient-based edge filter would suffer the same problem. Thus, the authors use the Berkeley edge detector ([Martin et al., 2004]) which was trained with labeled samples of inner and outer object boundaries in color images and almost never returns edges in textured image areas. Besides this, the authors report that the additional use of disparity discontinuities or motion helps significantly in obtaining actual object boundaries when combined with that edge detector.

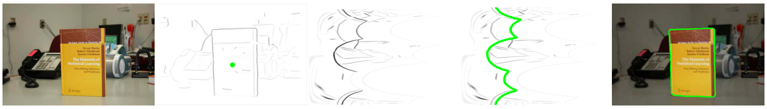


Figure 2.2: The different stages of the segmentation process: Original image, edge image, edges in polar space, result of the graph cut, and final segmentation (from [Mishra et al., 2012]).

The algorithm also requires a seed or fixation point on the object that is to be segmented. This point can be provided manually or by automatically selecting a salient point, i.e. a point in a part of the image that stands out visually, which can be calculated using e.g. the method from [Achanta et al., 2009]. The edge image is then transformed to polar coordinates with the fixation point as center. In the resulting polar edge image, each row represents a ray emerging from the fixation point at a certain angle, and the $n$-th column contains the pixels at a distance of $n-1$ pixels from the fixation point in the different directions. All pixels in the first column correspond to the fixation point and thus are in the segmentation, while all points in the last column are defined to be outside.

A graph is constructed from the polar image, in which each pixel is a node that is connected with its direct neighbors. The edges of the graph are weighted with the average value of the two pixels it connects. While enforcing that all nodes in the first column are inside the segmentation and all nodes in the last one are outside, the graph cut algorithm from [Brox et al., 2004] is used to find an optimal cut which results in a boundary that separates the polar edge image from top to bottom. Figure 2.2 visualizes this process: An edge image is computed from the input image, transformed to polar space around a seed point (green dot in the second image), and a graph cut is calculated. The determined boundary is transformed back into the original image and defines a closed contour around the fixation point, which is the final segmentation result.

### 2.1.5 Object Partitioning using Local Convexity

With the spread of cheap and solid sensors that provide images together with depth values, the use of 3D information has turned out to be helpful in tackling the problem of object segmentation. Depth discontinuities usually indicate object boundaries, and the work of [Stein et al., 2014] is based on the observation that object surfaces can most of the time be described as a set of convexly connected smooth patches.

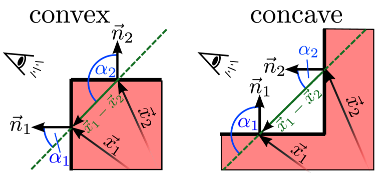


Figure 2.3: The decision whether two adjacent surface patches are convexly connected can be made by considering the angles between their normals and the line connecting their centers (from [Stein et al., 2014]).

The first step of the proposed algorithm is to oversegment the scene, which is given in the form of a set of 3D points, into a set of small surface patches which the authors call supervoxels, in analogy to superpixels. The patches are generated starting from a regularly sampled grid of seeds, which are extended using region growing that takes into account the distance to the seed, the local surface normals, and color. The most important property of the supervoxels is that they are designed to adhere to object boundaries. The resulting supervoxels form an adjacency graph in which they are connected to their immediate neighbors.

Each connection between adjacent supervoxels is analyzed on being convex or not. Given two supervoxels with centers $x_1, x_2$ and normals $n_1, n_2$, the vector $d = \frac{x_1 - x_2}{\|x_1 - x_2\|}$ indicates the direction from one center to the other. The connection between the supervoxels is convex if $\angle(d, n_1) < \angle(d, n_2)$ (see Figure 2.4). This is the case if $\cos(\angle(d, n_1)) > \cos(\angle(d, n_2)) \Leftrightarrow n_1 \cdot d > n_2 \cdot d$.

To account for noise, a connection is also considered convex when the angle between the two normals is below a certain threshold.

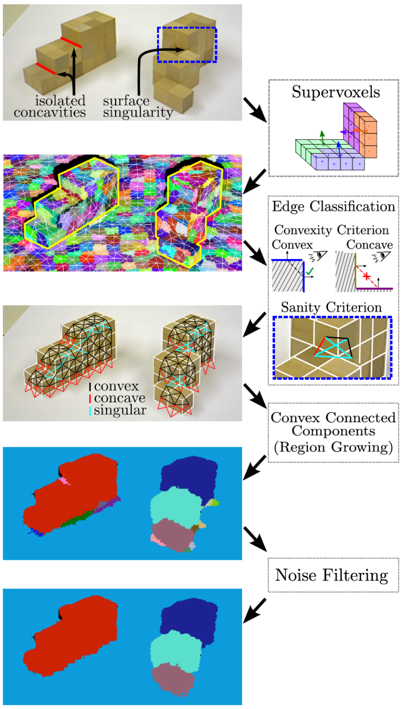


Figure 2.4: Overview of the algorithm for segmenting the scene into convexly connected regions (from [Stein et al., 2014]).

To increase reliability, the algorithm demands an extended convexity between adjacent supervoxels which requires them to be convexly connected and additionally have a common neighbor that is also convexly connected to both. Using region growing, all sets of supervoxels that are connected by this extended convexity are clustered together, i.e. the whole scene is segmented into convexly connected regions.

### 2.1.6 Discussion

This section has given an introduction to the problem of visual object segmentation. Three different attempts on solving it have been presented in some detail to give an impression of the different directions from which it has been approached.

The work of [Alpert et al., 2012] tackles the segmentation problem in its most general form, resulting in an algorithm that splits an image into a set of segments. Depending on the similarity cues that are used here, their parameters and the weights when combining them, very different segmentations can result from the same input. An important feature of this approach is that different cues can be integrated in a manner that allows different weighting depending on the image regions that are considered to be fused. This allows for great extensibility and potentially high robustness when using a multilateral set of cues. It is, however, very unclear how a weighting function can be designed that generates good weights for many different cues.

In [Mishra et al., 2012] the problem has been simplified by assuming that only one object is considered and a point on it is already known. The effectiveness of the algorithm depends entirely on the quality of the used edge detection. With a solid edge filter and when depth edges are available too, the authors report very good results. The fact that an initial point of interest can be set allows for a more goal-oriented segmentation without trying to segment the whole scene, and context knowledge can be integrated in this way.

Exploiting 3D information is the main idea of [Stein et al., 2014]. The assumption that objects are always completely connected by convex surfaces is of course not true in general; the authors state however that the segmentation into such parts still makes sense from the point of view that objects can again be divided into functional or semantic parts by their method. Indeed the resulting segments usually correspond to simple objects or intuitively reasonable parts of more complex ones and are therefore certainly interesting and useful.

As already mentioned, all these approaches suffer from the fact that they can easily be fooled: None of them would separate two relatively similarly looking objects that are placed next to each other. To overcome this fundamental weakness, physical interaction is necessary, which will be surveyed in the next section.

## 2.2 Object segmentation by physical interaction

Robots are not restricted to passive observation of a scene, but can act in a purposeful way to improve their understanding of it. This can be done e.g. by moving the camera in a helpful manner ([Connolly, 1985], [Aloimonos et al., 1988], [Bajcsy, 1988]), or by interacting physically with the environment. In the case of object segmentation, this is extremely helpful to overcome the limitations and ambiguities of visual perception. When objects move, this is an extremely helpful cue for their distinction from the background. By intentionally causing motion in the scene, the robot can greatly support its efforts on visual segmentation.

### 2.2.1 Early work

An early notable implementation of physical interaction to support visual segmentation was realized by [Tsikos and Bajcsy, 1991]. They consider an unordered heap of parcels and letters that the robot is supposed to separate. The robot is equipped with a laser range scanner and different manipulators that can push objects or "grasp" them using a suction device. Depending on an estimation of the possible objects and their spatial relation, the robot selects an action to either remove an object from the heap or to move one in order to change an ambiguous arrangement.

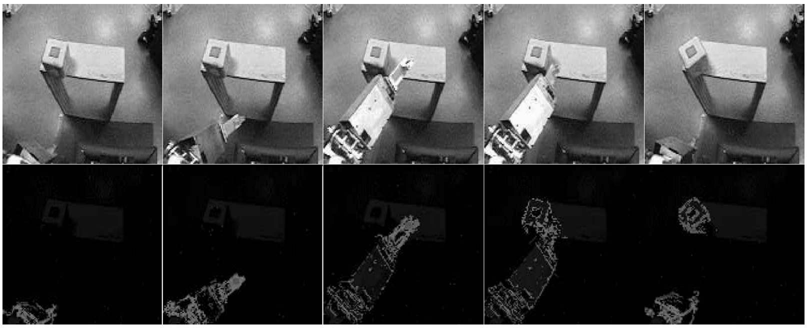


Figure 2.5: Approach of the robot arm towards the object, contact and retreat. Outlines of the moving arm and object are generated by simple image differencing (from [Metta and Fitzpatrick, 2003]).

Another conceptual study was done by [Metta and Fitzpatrick, 2003]. Here, the focus was to demonstrate how experimental manipulation by a robot can enable it to discover and segment individual objects. A static camera is observing a table with an unknown object placed on it, which is pushed by a robotic arm. In the camera images, the optical flow is calculated, which is the 2D motion of each pixel from one image to the next. While the arm is approaching the object, this allows to determine the regions covered by the arm and learn its appearance. When it touches the object, it starts to move and the nonzero optical flow suddenly spreads over a significant image region next to the arm. This can be detected, and in the moment of first contact the image region covered by the object is thus revealed. This allows for a rough object segmentation in this relatively simple setup and shows the huge potential benefit of using motion as a cue for this task.

A relatively similar approach was followed in [Kenney et al., 2009]. As in [Metta and Fitzpatrick, 2003], a robot arm approaches and touches an object while being observed by a static camera. The difference between consecutive images is determined, which during the approach is essentially caused by the moving arm. When the object is touched and starts moving, the amount of changing pixels increases significantly which indicates that the collision occurred. The new region in which the pixels differ from the last image frame is then used as an object segmentation.

Another relatively early attempt to exploit physical interaction for object segmentation and learning is [Li and Kleeman, 2009] (extended in [Li and Kleeman, 2011]). The robot finds symmetric object candidates in stereo camera images and applies a careful push to the object over a small distance. The symmetric outlines of the object are tracked during motion. The difference of the images before and after push contains the two overlapping regions covered by the object before and after pushing. Using the symmetry axis of the object, the overlap can be removed to obtain a segmentation (see Figure 2.6). SIFT features ([Lowe, 1999]) are used to learn visual object descriptors for recognition.

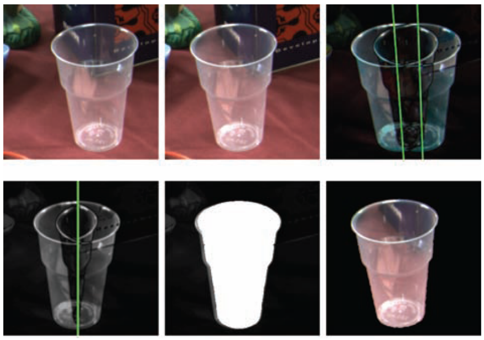


Figure 2.6: Top: The object before and after pushing, and the segmentation using the image difference. The two green lines mark the symmetry lines of the object. Bottom: Removal of the overlap, leading to a correct segmentation (from [Li and Kleeman, 2011]).

Since about 2011, the idea of physical interaction for object segmentation has gained increased popularity, and a number of different approaches has been proposed. In the following subsections, they are grouped by the perspective they take on the problem. One possible approach direction is to try to separate objects that lie on an unordered heap. Once the objects are singulated, the segmentation problem is essentially solved. Another way to look at the segmentation task is to assign probabilities to image pixels or regions that quantify the uncertainty about which object they belong to. Here the motion caused by the interaction is a cue that helps reduce this uncertainty. Lastly, one can try to track or relocalize the object during or after motion. This requires an initial idea what the possible objects in the scene are, and their models have to be updated and improved continuously to make their relocalization possible. From this perspective, the object learning is an integral part of the segmentation process.

### 2.2.2 Separation-oriented approaches

This kind of approaches tries to let a robot separate objects that form a clutter, i.e. are standing next to each other or form a heap or pile. The desired result is that each single object lies separated from the others, which implicates that it is segmented, and also facilitates grasping. These works assume that the scene contains a mostly empty table with one or more clusters of objects on it. This means that the background is known and simple, and that only a finite set of foreground objects needs to be considered.

#### Interactive singulation of objects from a pile

In [Chang et al., 2012], a robot equipped with an RGBD sensor has the task to singulate the objects lying in a cluttered arrangement on a table. To this end, the scene is presegmented by first finding the dominant plane which is assumed to be the table and removing it from the perceived point cloud, and then cluster the remaining points. This results in having one or more sets of points which the authors call "spatial units" that each correspond either to a single object or to a cluster of objects touching each other.

The robot now has to decide whether the spatial units consist of one or more objects, which is achieved by physical interaction. The smallest one is selected and pushed into a direction which is free of objects to avoid merging it with others. The camera is static, thus a simple point cloud subtraction can be used to find which spatial units moved. If in the resulting scene there is more than one unit that moved, this indicates that a separation has occurred. If only one unit moved, it could be either a single object or several objects that were not separated.

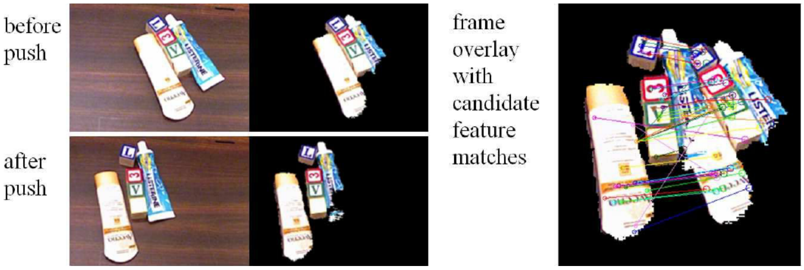


Figure 2.7: A cluster of objects before and after pushing. Second column: Remaining regions after subtraction of the table. Right: The feature correspondences that are used to estimate the rigid body transformation for evaluating whether one or more objects were moved (from [Chang et al., 2012]).

In this case, the system has to examine whether the pushed spatial unit consists of one or more objects. Its transformation due to the push is estimated using visual features at Harris corner points. Using RANSAC ([Fischler and Bolles, 1981]), up to five candidate transformations are determined. If there are insufficient visual features for this approach (e.g. on single-colored objects), the mean and the principal axes of the point cloud are used to estimate the transformation, and more candidates are created by small random perturbations.

Each candidate transformation is evaluated by measuring the ratio of matches in the dense point cloud. The quality of the dense point matches indicates whether it was one single rigid object that underwent the transformation, or if it were two or more objects that moved somewhat differently.

By pushing the same spatial unit several times and accumulation the matching scores, the evidence becomes increasingly reliable. When the certainty of the spatial unit being a single object is high enough, the robot picks it up and removes it from the table. Then the same procedure is repeated again for the smallest object cluster until the table is cleaned.

### Using Manipulation Primitives for Brick Sorting in Clutter

In the work of [Gupta and Sukhatme, 2012] (continued in [Gupta et al., 2015]), the goal is to sort a set of Duplo bricks by color, where initially the bricks may be lying on a chaotic heap. In order to be able to pick them up one by one and sort them, they first need to be separated. As in [Chang et al., 2012], they work on point clouds from an RGBD camera, and initially find and subtract the table plane from the scene point cloud. The remaining points are clustered into connected regions.

The authors exploit the fact that the used Duplo bricks have different colors to decide whether the connected regions consist of one or more bricks. Based on this and on the height of the region they classify it as being uncluttered, cluttered or piled. The latter means that bricks are lying on top of each other. Depending on that classification, they apply different manipulation primitives on that region: If it is piled, the robot places its finger next to the topmost point and pushes it horizontally to make the pile tumble.

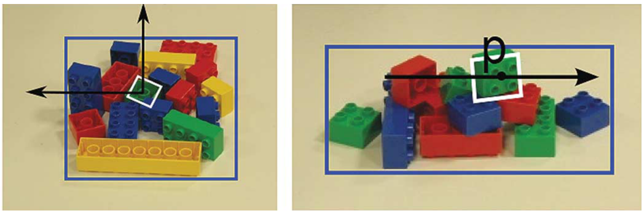


Figure 2.8: Visualization of the two manipulation primitives for spreading out cluttered objects (from [Gupta and Sukhatme, 2012]).

If the region is cluttered but all bricks are lying on the table and not on top of another one, the robot tries to separate one or more bricks from the rest. This is done by moving the finger down into the center of the region and then pushing horizontally first along one axis and then along the other one. In this way, the authors hope that several objects that may have been pushed together in the first movements will be separated when pushing in the perpendicular direction. Figure 2.8 visualizes the two proposed manipulation primitives.

When singulated objects are found, the robot grasps them and puts them into a glass depending on a sorting criterion like color or size. In this manner, the robot removes single objects and separates cluttered regions until the whole table is cleared.

### Clearing a Pile of Unknown Objects using Interactive Perception

Another approach to clearing away a pile of objects was proposed by [Katz et al., 2013]. They first form object hypotheses based on visual input from an RGBD sensor, push one of them, and if it can be re-localized afterwards, they consider it verified and grasp it in order to remove it from the table.

The first step is to segment the perceived scene into object candidates. This is done by finding facets, which the authors define as areas that are not disconnected by depth discontinuities or significant changes in normal orientation. To this end, they apply a filter on the depth image to find discontinuities, and estimate local normals for every image point. By overlaying two images that represent the depth discontinuities by grayscale values and the normal orientation by colors, they get an image that represents both criteria and is then segmented using Mean-Shift ([Fukunaga and Hostetler, 1975], [Silverman, 1986]). The resulting smooth, continues surfaces are used as object hypotheses.

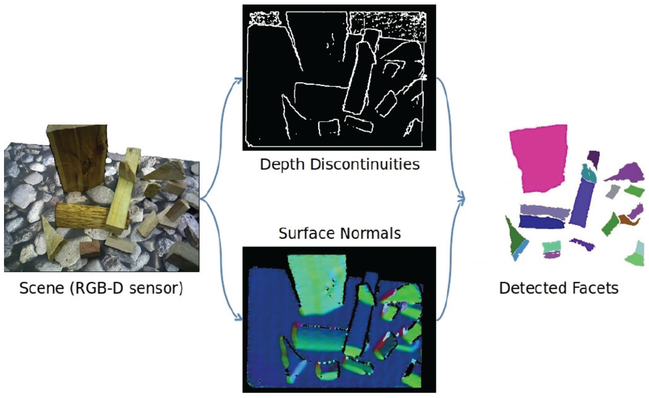


Figure 2.9: Object facets are determined by finding regions that are not separated by depth discontinuities and have relatively similar surface normals (from [Katz et al., 2013]).

One of them is chosen randomly and pushed by the robot. In the resulting scene, facets are again determined in the same manner. Now the facets of the old scene are matched with those in the new scene based on their size, color histogram and SIFT features. This may lead to ambiguous associations when several relatively similar facets exist in the scene. Therefore, the authors construct a graph containing the old and new facets as nodes that are connected if their similarity score is above a threshold, and the edge is weighted with it. Ambiguity in the form of multiple edges connected to one vertex is removed by computing a weighted bipartite matching on the graph.

With the correspondences between old and new facets established, those that moved and match well are considered verified. One of them is selected at random and grasped by the robot, so the object can be removed from the heap. Both the grasping and the pushing are realized using compliant controllers so that unexpected collisions of robot and objects do not cause damage to either.

### Guided Pushing for Object Singulation

In [Hermans et al., 2012], the authors try to singulate objects that initially are standing next to each other on a table. Here, the focus is on devising a deterministic process that systematically interacts with the clutter to make sure that all objects are actually separated.

The authors work on the data obtained from an RGBD sensor. Again, the first step is to determine the table plane in the scene point cloud and remove it, so that only the cluster of objects on it remains. For a given cluster, the visual and depth edges in it are considered candidates for object boundaries that need to be checked by physical interaction.

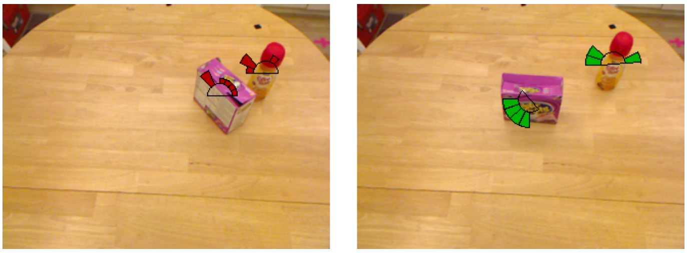


Figure 2.10: Left: Histogram of candidate boundary edge orientations. Right: Accumulated push history (from [Hermans et al., 2012]).

Instead of testing each edge, which would require a lot of effort especially in the case of textured objects as well as a tracking of all edges over several pushes, they quantize the edges into histogram bins by their orientation in the table plane. A similar histogram is created for the directions from which cluster has already been pushed. Using those histograms, a rough tracking of the change in orientation due to the pushes is sufficient.

A push direction is selected using one of the candidate edges. The cluster is hypothetically split into two parts along that edge, and the push goes through the center of one of them in the direction of the edge. Afterwards, ICP is used to determine the object transformation. When the resulting alignment has a high mean error, this indicates that the cluster contains more than one object.

Therefore, the push history is only updated when a good ICP match indicates that the cluster moved consistently. When a cluster is separated, their push history is reset. This process is continued until for every remaining cluster there is sufficient certainty that it consists of only one object.

### 2.2.3 Probabilistic approaches

The approaches presented in this subsection formulate the segmentation problem in a probabilistic manner. This allows to quantify the certainty of the obtained segmentation and to reduce the uncertainty by integrating information from several physical interactions.

#### Scene Understanding through Autonomous Interactive Perception

In [Bergström et al., 2011], physical interaction is used to check visually created segments on whether they contain one or two objects. To this end, the segment is separated into two by clustering, and one of those two hypothetical objects is pushed. In this way, the authors hope to cause different motion of the two parts if they actually correspond to two different objects.

Feature points are extracted in the initial segment and relocalized after pushing. The pairwise distance between points is calculated in the old and new scene, with the intuition that a change in distance would indicate that the two points lie on different physical objects. A matrix with the differences in pairwise distance is created, where element $(i, j)$ is the difference in the distance between points $i$ and $j$. If there are two objects that moved differently, two columns of that matrix will have zeros in the same places when their points lie on the same object.

Consequently, the two dominant eigenvectors of that matrix correspond to two subspaces containing the points lying on the two objects. By projecting all columns into the 2D space spanned by those eigenvectors, they can be clustered easily. By comparing the inter- and intra-class variance of the resulting point sets in image space it can be decided whether they form actually separated regions in the image, which would indicate that they belong to different objects, or whether they overlap and thus the differences in pairwise distances were just due to noise. The ratio of inter- and intra-class variance can be used to quantify the belief about the segment consisting of one or two objects. By accumulating the information of several pushes, the uncertainty of that belief can be reduced further.

### Probabilistic models for robot-based object segmentation

[Beale et al., 2011] propose a probabilistic framework for object segmentation based on the motion caused by pushing with a robot arm. They work on a single camera image which is first segmented by color, and the resulting small regions are tracked during the push using the method from [Paris, 2008].

During the execution of the pushing motion, the optical flow is calculated from frame to frame. The motion of the arm is calculated using forward kinematics and transformed into the image coordinate system. The motion of each image segment is then compared to it, and in the time before the first contact between arm and object the segments belonging to the arm can be determined. After contact, the segments belonging to the object move relatively similar to the arm.

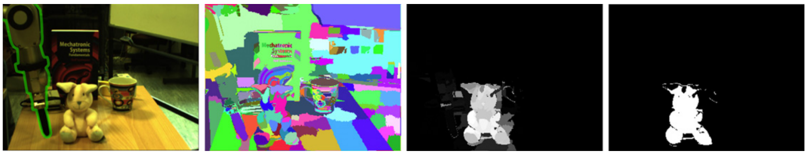


Figure 2.11: Camera image with the robot arm highlighted in green, the initial segments, their probability of belonging to the object that is being pushed, and the thresholded segmentation result (from [Beale et al., 2011]).

Based on the difference between the calculated hand motion and the observed motion of a segment, the probability that it is being pushed is quantified. By integration over several frames, the probability values for each segment tend to become very disambiguous. Using a threshold, a final binary segmentation mask can be obtained (see Figure 2.11).

### Probabilistic Segmentation and Targeted Exploration of Objects in Cluttered Environments

In the work of [van Hoof et al., 2012] (continued in [van Hoof et al., 2014]), the segmentation problem is formulated by assigning probabilities to parts of the scene that indicate which of them belong to the same object. Based on those probabilities, the robot chooses actions that are expected to reduce the uncertainty of the segmentation.

First, the authors create a set of parts from the observed scene by defining part centers on a regular 3D grid and assigning everything within a radius of 6 cm to them. These parts may overlap, and they are tracked during the interactions based on the SIFT features that exist within their radius. Their 3D motion caused by a push of the robot is determined based on those features using RANSAC. The determined transformation is used to estimate whether parts moved similarly or not.

The approach establishes a probability distribution over the possible partitionings of those parts into groups that correspond to actual objects. The motion is used as a cue to gain information that allows to improve that distribution towards the true segmentation. As it is computationally intractable to calculate the probabilities for each possible segmentation, a sampling method is used to approximate the distribution.

By sampling estimations of possible new distributions resulting from the motion of the parts, the one is chosen for pushing that is expected to minimize the entropy of the distribution. This allows to select interactions that maximize the expected information gain.

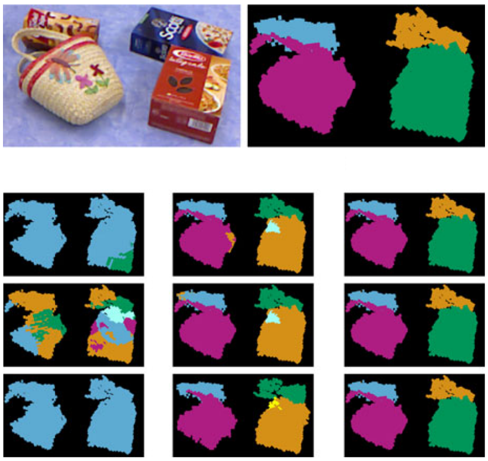


Figure 2.12: Top: Input image and ground truth. Bottom: Segmentations as estimated by three different samples from the probability distribution after 0, 5, and 15 pushes (from [van Hoof et al., 2014]).

Figure 2.12 shows an observed scene and the ground truth segmentation. In the three rows underneath, the segmentations are estimated by three different samples from the probability distribution, where the first, second and third column are generated after 0, 5 and 15 pushes respectively.

### 2.2.4 Tracking-based approaches

The following approaches focus on tracking parts of potential objects. This means that individual visual features are tracked continuously during the push, and their trajectories indicate whether they belong to the same object or not.

#### Segmentation of cluttered scenes through interactive perception

The approach presented in [Bersch et al., 2012] tries to segment objects in a cluttered scene by pushing them in order to make them move differently.

First, the table is detected and removed from the perceived scene point cloud, and the cluster consisting of possibly several objects remains. The 2D outline of that cluster projected to the table plane is determined, and concave edges in that outline are found. They are considered to be potential object boundaries, and therefore a push is executed through one of them parallel to the table plane. In this way, the cluster is split up and the objects are caused to move differently.

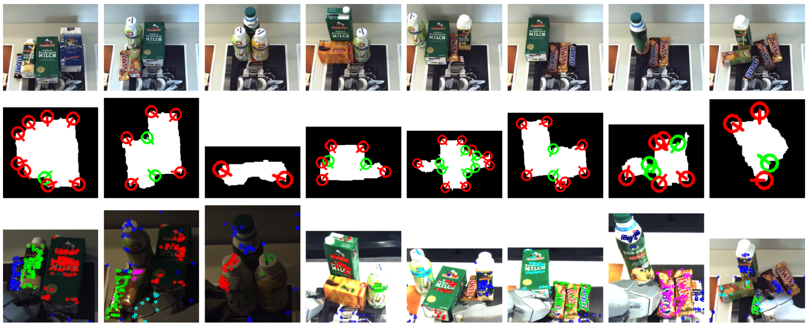


Figure 2.13: Different example scenes: The second row shows the outline of the object cluster with convex and concave corners and the corresponding push vectors through them. The third row shows the segmented subsets of features after pushing (from [Bersch et al., 2012]).

Before the push is executed, Shi-Tomasi features ([Shi and Tomasi, 1994]) are detected in the object cluster, and they are tracked during the push. Afterwards, random pairs of these features are chosen, the 2D transformation that they underwent is calculated for each tracking step, which corresponds to an arm motion of 1 cm. The trajectories of all other features are checked for conforming with that trajectory and, if they are, removed from the overall set. This is repeated iteratively, and the result are subsets of features that moved similarly during the push.

It is important to note here that the trajectories have to be consistent throughout the whole motion, and that not only the overall transformation is taken into account. A downside is that the authors report problems when features get lost during the pushing due to occlusions caused by the hand of the robot. While this approach is limited to textured objects which contain enough features for robust tracking, the authors hint at the upcoming work of [Hausman et al., 2013] which is presented in the next subsection.

### Tracking-based Interactive Segmentation of Textureless Objects

[Hausman et al., 2013] can be seen as a transfer of the ideas in [Bersch et al., 2012] to textureless objects. Instead of using visual features, shape descriptors are used for object tracking during the push. Point cloud data is used, and the first step is to oversegment the objects that are placed on a table. All possible combinations of up to eight connected segments are calculated, and considered object hypotheses.

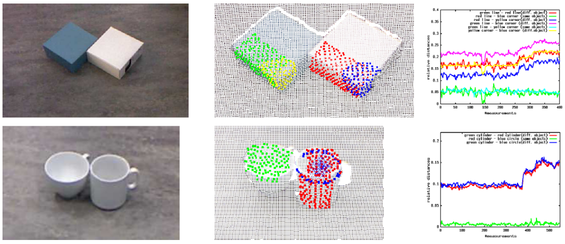


Figure 2.14: Two example scenes, the geometrical features being tracked, and their pairwise distances (from [Hausman et al., 2013]).

The parts are classified as belonging either to flat/box-shaped or to round/-cylindrical objects. The authors find empirical values for the expected number of parts forming one object, and based on those decide whether verification by interaction is necessary. If it is, the same strategy for finding pushing points and directions as in [Bersch et al., 2012] is used.

For tracking during the push, geometric features are used. In the case of box-shaped objects, straight edges and corners are determined and the point cloud in a 5 cm radius around them is used as a feature descriptor.

For cylindrical objects, circular edges are determined. These geometrical features are tracked using a particle filter. The trajectories of the tracked features are clustered, and in the final step they are used as seeds for region growing based on normal directions that yields the completed object point cloud as the segmentation result.

Figure 2.14 shows two examples for this: The second column visualizes the geometrical features being tracked, and the graphs in the third column shows pairwise distances between these features during pushing. A significant change in this distance indicates that the features lie on different objects.

### From passive to interactive object learning and recognition through self-identification on a humanoid robot

In [Lyubova et al., 2015] the focus is on autonomous learning of visual object models by a robot. Here, motion is used first to direct the attention of the robot and subsequently to segment the object from the background. Proto-objects are detected using difference images, and they are then tracked using Good Features to Track ([Shi and Tomasi, 1994]). Features are clustered by similarity of the observed motion, and the visual appearance of objects is learned using SURF features ([Bay et al., 2006]) as well as color descriptors for textureless regions.

A notable characteristic of this work is that the robot also learns the appearance of its own arms using the correspondence between proprioception and observed motion, and the appearance of a human teacher that shows objects to it. The discrimination between parts of the human and mere objects is made based on the fact that the human moves on his own, while objects only move when in contact with something else. By a combination of autonomous interaction with objects and human demonstration, an incrementally growing database ob multi view object representations is learned.

### 2.2.5 Discussion

The presented approaches show the potential benefits of having a robot intentionally cause motion in a scene to support object segmentation. While some works are rather a proof of concept that leaves parts of the problem unsolved, others try to achieve very general and robust solutions.

The first step is to decide if and how to create initial object hypotheses and how to push them. [Metta and Fitzpatrick, 2003], [Kenney et al., 2009] and [Beale et al., 2011] use preprogrammed motions. [Li and Kleeman, 2009] and [Lyubova et al., 2015] choose objects for pushing that are "interesting" in a certain way by being symmetric or by moving. The approaches that intend to separate objects try to move them away from one another ([Bergström et al., 2011], [Hermans et al., 2012], [Bersch et al., 2012], [Hausman et al., 2013]). The other approaches either choose a hypothetical object randomly ([Katz et al., 2013]), choose the smallest one ([Chang et al., 2012]), or push from the center of a cluster outwards ([Gupta and Sukhatme, 2012]). [Bergström et al., 2011] and [Hermans et al., 2012] try to split a cluster by pushing one half of it, while [Bersch et al., 2012] and [Hausman et al., 2013] have the opposite intuition and want to achieve the same by trying to push in between the two halves. Systematic approaches are proposed by [Hermans et al., 2012], who pushes from all directions that might split the cluster along an internal edge, and [van Hoof et al., 2012] who choose interactions such that an expected information gain is maximized.

Tracking of the objects during the push, or re-localization afterwards, is realized using optical flow ([Metta and Fitzpatrick, 2003], [Beale et al., 2011]), difference images ([Kenney et al., 2009]), point cloud matching ([Hermans et al., 2012]), geometric features ([Li and Kleeman, 2009], [Hausman et al., 2013]), or local visual features ([Chang et al., 2012], [Bergström et al., 2011], [van Hoof et al., 2012], [Bersch et al., 2012], [Katz et al., 2013], [Lyubova et al., 2015]). While local visual features are probably the most reliable choice, they also restrict the algorithm to sufficiently textured objects. On

| | object appearance | | | | scene setup | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | textured | single-colored | nontextured | arbitrary shape | many objects | any background | touching objects | stacked objects | autonomous interaction |
| [Metta and Fitzpatrick, 2003] | ✓ | ? | ✓ | ✓ | ✓ | ✓ | | | |
| [Kenney et al., 2009] | ✓ | ? | ✓ | ✓ | ✓ | ✓ | | | |
| [Li and Kleeman, 2009] | | ✓ | ✓ | | | ✓ | | | ✓ |
| [Beale et al., 2011] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| [Bergström et al., 2011] | ✓ | | | ✓ | | | ✓ | | ✓ |
| [Bersch et al., 2012] | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ |
| [Chang et al., 2012] | ✓ | ? | ? | ✓ | ✓ | | ✓ | | ✓ |
| [Gupta and Sukhatme, 2012] | | ✓ | | ? | ✓ | | ✓ | ✓ | ✓ |
| [Hermans et al., 2012] | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | | ✓ |
| [van Hoof et al., 2012] | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| [Hausman et al., 2013] | ✓ | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ |
| [Katz et al., 2013] | ✓ | ? | ? | ✓ | ✓ | | ✓ | ? | ✓ |
| [Lyubova et al., 2015] | ✓ | | | ✓ | ✓ | ? | ✓ | ? | ✓ |

Table 2.1: Overview of the presented related work on interactive object segmentation. The first four columns indicate for which types of objects the approaches are applicable. The following four columns show how versatile they are with respect to the complexity of the scene. The last column indicates whether the physical interaction is performed by the robot autonomously, or whether it is manually preprogrammed.

the other side, using optical flow or difference images works also for non-textured objects, but can not easily distinguish whether different objects are being moved at the same time.

A large part of the presented approaches explicitly makes the very specific assumption that the background is a table which is sufficiently empty for being detected reliably ([Chang et al., 2012], [Gupta and Sukhatme, 2012], [Katz et al., 2013], [Hermans et al., 2012], [Bersch et al., 2012], [Hausman et al., 2013]). Others are only applicable to very simple scenes ([Li and Kleeman, 2009]) or would probably fail when more than one object is moved at once ([Metta and Fitzpatrick, 2003], [Kenney et al., 2009], [Beale et al., 2011]). [Bergström et al., 2011] seems to have only been applied to scenes with two objects. Only [Beale et al., 2011] explicitly claim robustness to changes in the background, and [Lyubova et al., 2015] learn a model of the human which may be active in the observed scene. [Beale et al., 2011], [van Hoof et al., 2012] and [Hausman et al., 2013] initially oversegment the scene, and it is unclear what happens when those segments stretch over more than one object. Almost all of the presented approaches that can deal with several objects moving at once make limiting assumptions about the object. Most require texturedness, or they are restricted to specific shapes or singlecoloredness. None is applicable in unrestricted clutter where the background is unknown and complex, and objects are touching each other and may be stacked. Table 2.1 gives a compact overview of the applicability and restrictions of the presented approaches with respect to the considered objects and scene setup complexity.

## 2.3 Object shape completion

Most algorithms for grasp or manipulation planning, collision avoidance etc. require 3D models of the objects that are relevant for the respective task. However, in reality robots can never perceive their environment completely.

In the case of manipulable objects as considered in this thesis, their segmentation from the environment is the crucial prerequisite for model generation. But even with a perfect segmentation, a robot will not be able to perceive an object from all sides without grasping, lifting and turning it. To plan such a manipulation, in turn, requires a 3D model of the object[1].

Thus, it can often be helpful to generate a complete model from the partial object shape that has been perceived by the robot. This model can then be used by a grasp planner, and if desired the grasped object can be observed further from different sides, as e.g. described in [Welke et al., 2010]. Different approaches for shape completion based on partial information are described in the following subsections.

### 2.3.1 Shape approximation with geometric primitives

One line of research aims at approximating the shape of perceived objects by geometric primitives. An example is the work of [Marton et al., 2011] who try to fit boxes or cylinders to the objects, depending on which of the two shape classes seems to approximate the visible object part better.

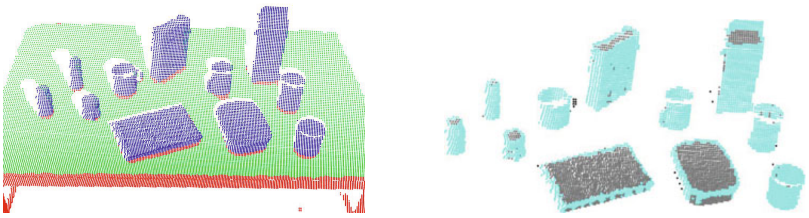


Figure 2.15: Left: Point cloud of a table with objects on it. Right: Objects remaining after removal of the table plane, with the edge points highlighted in blue (from [Marton et al., 2011]).

[1] There are grasp planning approaches that do not require an explicit 3D model of the object, e.g. [Kootstra et al., 2012], [Fischinger and Vincze, 2012] or [Lenz et al., 2015]. Most algorithms do however presuppose that a complete and exact model is available.

They assume to have a mostly empty table with some objects standing on it separately, which is perceived by a depth camera. They detect and remove the table surface, and project the remaining points onto it. By clustering the projected points, the objects are obtained, and the points on the edges of the clusters are determined (see Figure 2.15). These edge points are used to decide whether the object will be approximated by a cube or by a cylinder. First, a 2D principal component analysis (PCA) is performed to determine a bounding box. The authors then determine how well the bounding box approximates the object shape: They calculate the average distance of the edge points of the cluster to their bounding box, and normalize the distance by the size of the bounding box in that direction. If the mean normalized distance of the edge points to the bounding box is above a threshold, the object is considered to be cylindrical rather than box-shaped.

If the object is to be approximated by a box, pairs of parallel lines are fitted to the edge points. The pair with the maximal number of points within an interval near it (or two pairs that are roughly orthogonal to each other) is used to create the 2D base of the box. In the case of cylindrical objects, circles are fitted (see Figure 2.15). If there are two circles where one contains the center of the other, they are merged as they probably describe parts of the same object.

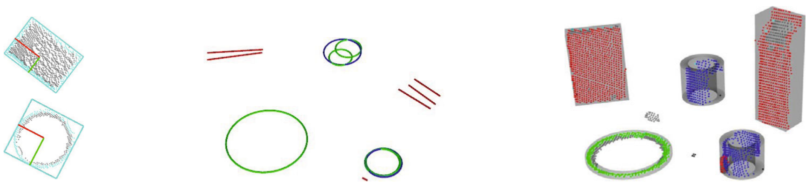


Figure 2.16: Left: Examples for the edge points and their bounding box in the case of a box-shaped and a cylindrical object. Center: The determined straight lines and circles. Right: The reconstructed primitive shapes overlayed with the perceived point cloud (from [Marton et al., 2011]).

The minimal and maximal height of the 3D points of the cluster are used to determine the height of the respective box or cylinder. Finally, the generated model is tested for its accordance with the perceived object: Points are generated along the surface of the assumed shape, and the generated model is rated based on whether they coincide with existing real points (which would support the shape hypothesis), lie in an invisible area behind the perceived points, or lie in a visible area far from the existing points (which would contradict the shape hypothesis).

Another popular way to approximate shapes is to use superquadrics or superellipsoids. An axis-aligned superellipsoid can be defined implicitly by the equation

$$F(x) = \left( \left( \frac{x}{a_1} \right)^{\frac{2}{\varepsilon_2}} + \left( \frac{y}{a_2} \right)^{\frac{2}{\varepsilon_2}} \right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left( \frac{z}{a_3} \right)^{\frac{2}{\varepsilon_1}} = 1$$

where $a$, $b$ and $c$ scale the shape along the main axes, $\varepsilon_2$ influences the shape of the cross section of the superellipsoid with a plane orthogonal to the z-axis and $\varepsilon_1$ influences the shape of the cross section with a plane that contains the z-axis. Depending on those parameters, it may take the shape of e.g. cubes, cylinders, spheres or bicones.

Including the 6 degrees of freedom for position and orientation in space, a superquadric or a superellipsoid has 11 free parameters that need to be estimated when fitting it to a perceived point cloud. [Biegelbauer and Vincze, 2007] assume that the shape parameters are already known and only estimate the pose. To solve the full fitting problem, [Duncan et al., 2013], building on the work of [Solina and Bajcsy, 1990], formulate it as an optimization problem where, given a set of points $x_k$, the parameter set $\Lambda$ needs to be optimized to minimize the sum

$$\sum_{k=0}^{n} \left( \sqrt{a_1 a_2 a_3} \left( F^{\varepsilon_1}(x_k; \Lambda) - 1 \right) \right)^2$$

applying the iterative Levenberg-Marquardt algorithm ([Moré, 1978]). The term $\sqrt{a_1 a_2 a_3}$ makes sure that smaller shapes are preferred, and by exponentiating with $\varepsilon_1$ the error metric becomes independent of the shape factor which is supposed to accelerate convergence. The parameters are initialized such that the shape of the superellipsoid is that of an ellipsoid along the principal axes determined by principal component analysis (PCA) of the object point cloud.

The optimization in the 11-dimensional parameter space requires huge computational effort, in particular when the point cloud contains a large number of points. To reduce the effort, the point cloud is downsampled using a voxel grid. They start with a coarse grid, which reduces the point cloud significantly and thus allows quick optimization, and refine the grid iteratively until the error reduction is below a threshold or a minimal voxel size is reached.

### 2.3.2 Symmetry-based shape completion

Another popular assumption that can be made about the partially known object is that it possesses some kind of symmetry. Early work in that direction was done by [Grupen et al., 1988]. They approximate the perceived object surface by planar patches and estimate their centroid, size and normal. The volume of the object is subdivided into discretized bins, and for each surface patch, the normal is followed from the centroid through the object and the value of each bin is increased by a weight equal to the size of the surface patch. In this way, a discretized estimation of the mass distribution is created. By means of a principal component analysis (PCA) of that mass distribution, its principal axes are determined. Based on their corresponding eigenvalues, the object is classified as line-shaped, planar or spherical, and the corresponding symmetry type (axial, planar or spherical) is thus deduced.

The authors of [Thrun and Wegbreit, 2005] propose a hierarchy of symmetry types that they try to detect efficiently, starting with simple ones and possibly combining them to more complex ones. Figure 2.17 shows the basic symmetry types they consider. On the left are reflectional symmetries, where each point can be mirrored either on a plane, a line, or a point. On the right are rotational symmetries where each point can be rotated around an axis or a point.

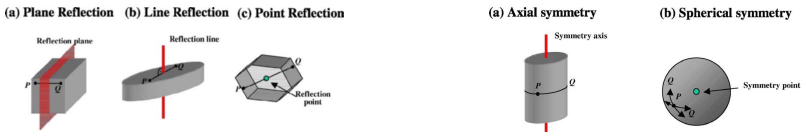


Figure 2.17: Different kinds of symmetry that exist in 3D space (from [Thrun and Wegbreit, 2005]).

The main challenge is to detect these basic symmetries in the point cloud. The authors segment the scene into compact point sets, which serves to segment it into candidates for symmetric objects or object parts. In each of these subsets, the optimal parameters for each basic symmetry type are determined using optimization with a rating function based on visibility criteria similar to [Marton et al., 2011], where the mirrored points should lie amongst the perceived points or otherwise should not be visible. If a basic symmetry type with a good final rating can be found, it is accepted.

From these basic types, composite symmetries can be constructed. For such a higher order symmetry to exist in a point cloud, the constituting basic symmetries must be present in it. The search for the composite symmetries can thus be restricted to the types that can exist given the detected basic symmetries. Additionally, the parameters of the composite types are restricted by those of the constituting basic symmetries which significantly speeds up and facilitates the search. Figure 2.18 gives a schematic overview of the considered symmetries and their hierarchy.

Another approach to object shape completion based on symmetry was proposed by [Bohg et al., 2011]. Here the explicit aim is to create object models

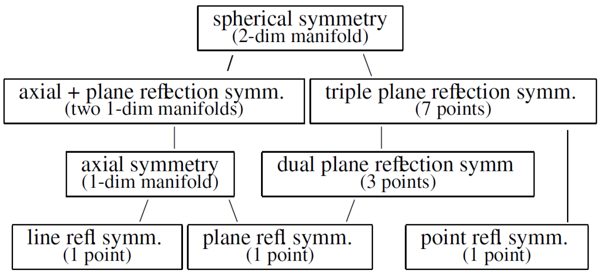


Figure 2.18: The hierarchy of symmetry types, where the basic types on the bottom constitute the higher order types above. The schema also gives the number of points or the space to which a point is reflected by this symmetry type (from [Thrun and Wegbreit, 2005]).

for grasp planning based on the point cloud of the object that was perceived from a single view. Their work is the basis for the approach developed within this thesis and described in chapter 4.

The assumption made in this work is that the object has a planar symmetry. In order to restrict the search space and to enforce that a back side is generated to complete the object shape that was only seen from the front, the symmetry plane is assumed to be orthogonal to the table on which the object is placed, and its orientation around the vertical axis is also restricted so that it is not parallel to the view axis of the camera.

The remaining space of possible orientations is sampled in regular intervals. For each possible orientation, symmetry plane candidates are generated at regular translational intervals in the direction perpendicular to the plane within the extent of the perceived object point cloud.

The symmetry plane candidates are rated somewhat similarly to [Thrun and Wegbreit, 2005] and [Marton et al., 2011]: All perceived points are reflected at the symmetry plane. If a point lies in a location where it would have been visible to the camera, this contradicts the symmetry plane hypothesis and thus leads to a bad rating. Points that coincide with perceived points support the candidate plane, and those that lie behind the existing points and can not be perceived by the camera give additional information about the object shape. The symmetry plane candidate with the best rating is finally used to mirror all perceived points and thus create the back side of the object.

### 2.3.3 Shape modeling with extrusions

An extrusion is a 3D shape that is created by moving a 2D silhouette along a trajectory. This potentially allows to model a large variety of shapes, but it is unclear how to determine the silhouette and the trajectory in the general case. Therefore, [Kroemer et al., 2012] restrict the trajectory to be either a straight line or a circle. In that case, the resulting shape is symmetric. They find the symmetry with a voting-based approach ([Mitra et al., 2006]), where the normal at each point is estimated and pair-wise symmetries are searched and vote for symmetry planes with corresponding parameters.

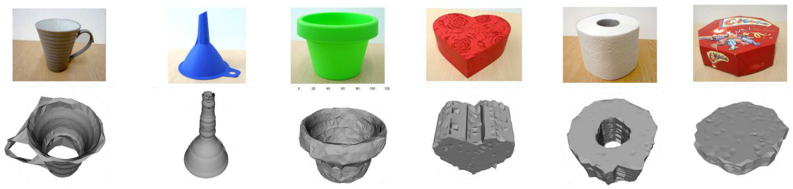


Figure 2.19: Examples for shape completion using extrusions (from [Kroemer et al., 2012]).

Given a symmetry plane, for a linear extrusion, the object point cloud is divided into the two parts on both sides of it. Iterative closest point (ICP) is used to shift one half over the other, and the resulting translation is the initial estimation of the linear path of the extrusion. For a rotational extrusion, the point cloud is split in two parts based on the normal directions of the points, and the result of ICP aligning the two parts gives the rotation for the extrusion path.

Based on these initial estimations of the path, the 2D shape is determined that defines the extrusion by moving along the path. An iterative refinement process on both is performed to fit well to the perceived object points. Finally the extrusion is rated based on visibility criteria as in the approaches described above.

Figure 2.19 shows some results of this algorithm. The objects in the top row were perceived with a depth sensor from one perspective, the bottom row shows the resulting extrusions. Another approach that uses extrusions as well as approximation by superquadrics to generate object models for grasp planning was proposed by [Huamán Quispe et al., 2015].

### 2.3.4 Discussion

When aiming at completing the shape of an object that has only been perceived partially, making assumptions about the shape of the invisible part is unavoidable. Under these assumptions, one then has to find the reconstruction parameters that give the result which fits best to the available information. Consequently, the proposed methods differ mainly in the assumptions they make about the object shape, how they restrict the parameter search space, and how they decide which of the completions possible under their assumptions is considered the best.

[Marton et al., 2011] try to fit rather simple geometric primitives to the objects, and propose efficient preprocessing steps that probably allow a very quick determination of their parameters. Their results are however restricted to be boxes or cylinders, which would be insufficient for e.g. grasp planning when the real object has a significantly more complex shape. Somewhat related is the approach of [Schnabel et al., 2009] who try to fill holes in object models with geometric primitives, but require relatively complete initial shape knowledge.

Superquadrics allow for a larger variety of shapes, but that comes at the cost of an 11-dimensional parameter space in which good values have to be searched. In order to keep the computational effort manageable, [Biegelbauer and Vincze, 2007] assume that some parameters are known, which is inappropriate when dealing with unknown objects. [Solina and Bajcsy, 1990] and [Duncan et al., 2013] use optimization techniques to search the parameter space. To limit the necessary effort, [Duncan et al., 2013] down-sample the object point cloud, and iteratively increase its size again in the later refinement phase of the optimization process.

Extrusions are a comparably versatile shape class, however to estimate them given a partial point cloud further assumptions have to be made as was done by [Kroemer et al., 2012] and [Huamán Quispe et al., 2015]. With these restrictions, the extrusions become a subset of the objects that have at least a planar symmetry. Thus despite the more complicated modeling and estimation process, using such restricted extrusions for shape estimation is more limiting than the symmetry assumptions made by [Bohg et al., 2011] or [Thrun and Wegbreit, 2005]. The parametric character of the extrusions may however be exploited for grasp planning as done by [Huamán Quispe et al., 2015].

The approach for shape completion which is presented in this thesis takes the work of [Bohg et al., 2011] as its starting point. The assumption of planar symmetry allows for a large variety of shapes and is fulfilled or approximately fulfilled by most artificial and many natural objects. Of course there are objects that contain significant asymmetries, so when a grasp is planned with an object model that was created based on the symmetry assumption, prudence is advisable when actually executing the grasp.

## 2.4 Reactive grasping

Visual perception can not always provide all information necessary for successful grasping, and due to imprecisions in perception as well as motion a robot may fail when executing a grasp. The ability to detect such failure and correct the grasping behavior is crucial for the real-world aptness of robots. The preferred sensing modalities for contact detection during grasping are tactile and force sensors. An extreme case is blind grasping, where little or no visual information is used and the grasping is based almost solely on haptic feedback. The application that is of interest in this thesis is reactive correction behavior during grasp execution. The contribution made here is a method for visual collision detection. Other approaches related to these topics are presented in the following subsections.

### 2.4.1 Blind grasping

The motivation behind approaches for blind grasping is that haptic information can be sufficient for grasp execution. When contact with the object is detected, this is used to adapt the hand pose or finger configuration in a way that is expected to lead to a successful and stable grasp. [Ozawa et al., 2006] propose a controller for multi-finger pinch grasps using tactile contact information. Based on the tactile feedback, the fingertips exert forces in a way that stabilizes the object between them (see Figure 2.20). It seems though that the controller has only been tested in simulation.

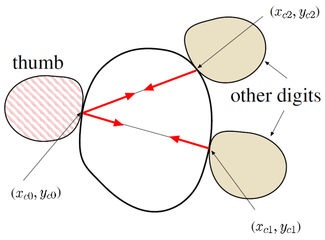


Figure 2.20: Example of a three-finger pinch. The red arrows visualize the contact forces generated by the controller (from [Ozawa et al., 2006]).

[Dang et al., 2011] train a Support Vector Machine (SVM) classifier to recognize stable grasps based on tactile sensor information and the kinematic configuration of the hand. They simulate a large number of grasps and the corresponding tactile feedback to get training samples for their SVM, and test it with simulated grasps on different objects.

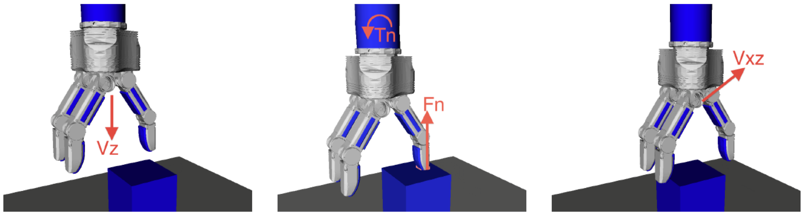


Figure 2.21: The robot hand approaches the object from the top. When contact occurs, the torque measured in the wrist is used to decide in which direction the hand is moved in order to correct the grasp position (from [Felip et al., 2012]).

[Felip et al., 2012] realized a complete reactive grasping pipeline: The robot hand approaches the object, or in their experiments a basket with some objects inside, from the top. Contact between the hand and the object is detected using a force-torque sensor in the wrist. Depending on the detected torque, the finger that is pressing onto the object is determined. The hand is lifted slightly and a horizontal translation is performed in the direction of that finger with the intention of bringing it besides the object. The hand then moves down again, and if another contact is detected, another corrective reaction is performed (see Figure 2.21). This is repeated until all fingers touch the supporting surface below the object, which causes a roughly vertical force and no torque, in which case the fingers are closed to grasp the object.

### 2.4.2 Reactive grasp correction

During grasp execution, tactile sensing can be used to correct the grasping motion by recognizing the occurrence of premature collisions. An example for this is the work by [Hsiao et al., 2010]: The object is approached for grasping with a two-finger gripper. When a collision is detected on a fingertip, the gripper retreats a bit and its position is changed into the direction of contact. If another contact is detected, then the same reaction is executed if it was the same finger. If it was the other one, then the correction is performed in the other direction but only with half of the step size. Contacts on the inner side of the fingers are ignored, and contacts on the palm indicate that the object was reached, thus the approach is stopped and the fingers are closed. Tactile feedback is also leveraged while closing the fingers: When contact is detected on only one finger while closing the gripper, the controller is changed such that that finger does not move anymore to avoid displacing the object. When both fingers have contact with the object, the tactile sensors indicate whether the grasp will be stable: If the contact is only on the edge of the fingers, the gripper is opened and repositioned to achieve more robust contact locations.

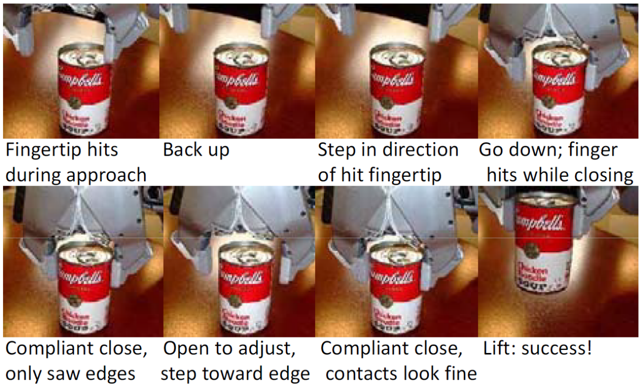

Fingertip hits during approach
Back up
Step in direction of hit fingertip
Go down; finger hits while closing
Compliant close, only saw edges
Open to adjust, step toward edge
Compliant close, contacts look fine
Lift: success!

Figure 2.22: Different stages of grasp execution with reactive corrections during the approach of the hand as well as closing the fingers on the object (from [Hsiao et al., 2010]).

Other work where tactile sensors are used to estimate the stability of a grasp before lifting and, if necessary, modify the grasp, includes [Dang and Allen, 2014], [Platt, 2007] and [López-Coronado et al., 2002].

## 2.4.3 Visual collision detection

All approaches based on force or tactile feedback require that the object resists the robot hand sufficiently so that a force can actually be measured. This is not always the case: When e.g. a light object is grasped from the side, it might be moved without causing sufficiently clear sensor values that would indicate that a collision occurred. One way to circumvent this is to use proximity sensors as in [Hsiao et al., 2009], another way is to use visual information.

This idea is followed by [Bernabe et al., 2013] who attempt to visually detect collisions during grasping. They obtain an RGBD point cloud from a static depth camera observing the scene which consists of a table surface with only the object on it and the robot arm, of which a geometric model is available.

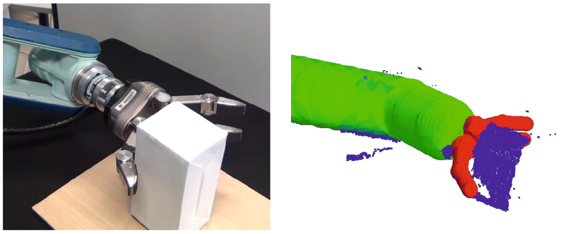


Figure 2.23: While the robot is grasping an object, the perceived point cloud is divided into arm (green), hand (red), and object (purple) (from [Bernabe et al., 2013]).

The object point cloud is segmented in each frame by removing the table surface, and tracked using ICP from one frame to the next. If the translation or rotation returned by ICP is above a threshold, the object is considered to have moved. The arm is also tracked in the perceived point cloud, and when the object moves while the arm is near it, a collision is assumed to have occurred. The most probable part of the hand to have caused the collision is determined based on an occupancy grid map of the object point cloud and the geometric model of arm and hand. This information can be used to implement a reaction strategy, although this is not described in the paper.

### 2.4.4 Discussion

Monitoring grasp execution for premature collisions that may cause failure is essential when designing a robotic grasping skill that is supposed to work reliably under realistic conditions. Checking for failures is particularly important when dealing with high uncertainties, e.g. when grasping unknown objects. Intuitively, tactile sensors seem like the appropriate modality for detecting collisions of the hand of the robot with the object, as is done by [Ozawa et al., 2006] and [Hsiao et al., 2010]. Simple force-torque sensors can also be helpful and were used successfully in [Felip et al., 2012].

There are, however, two practical limitations on the use of these sensors: The first is that on currently existing robots that have tactile pads, they cover only a part of the hands. This may become less of a problem as these sensors become cheaper and more flexible to install. The second, more serious problem is that for light contacts the sensitivity of current sensors is insufficient, and even if it were sufficient a very light contact might be indistinguishable from sensor noise, or the effects of small accelerations when using force-torque sensors.

A complementary sensing modality is vision, which can be applied in exactly the situations in which the force-based modalities may fail: When there is a light object that starts moving when touched and does not offer significant force resistance. Detecting such a movement is the idea behind the work of [Bernabe et al., 2013] as well as the approach presented in this thesis.

# 3 Object segmentation by physical interaction

The main contribution of this thesis is an approach for segmenting unknown objects in complex, cluttered environments by exploiting the ability of humanoid robots to interact purposefully with their environment. Initially, the robot generates object hypotheses based on the visual input following some heuristic criteria. These initial object hypotheses may correspond to actual objects, but they may also cover only parts of an object or contain two or more different objects. In order to verify and correct these hypotheses, the robot interacts with the objects by pushing them to cause motion. This motion is then used as an additional cue that helps to verify or discard the object hypotheses, to remove parts of them that do not belong to the actual object and extend them to cover the object completely.

The approach is designed to work in extremely cluttered scenes where many unknown objects may be touching each other, be stacked, or lie in a

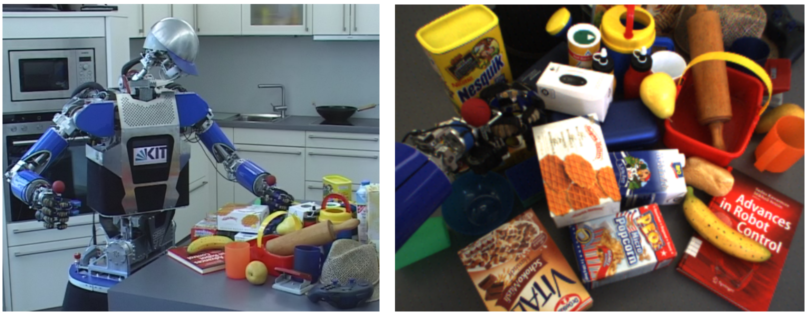


Figure 3.1: The robot ARMAR-III pushing an object in a cluttered scene.

completely unordered heap. No assumptions are made about the background, the only crucial assumptions made are that the object that will be segmented can be moved by the robot, fits into the camera images, and moves as a rigid body, i.e. does not undergo significant deformations. The result is a segmentation of usually very high quality, that is verified by physical interaction to correspond to an actual object. The steps of the proposed algorithm are described in detail in this chapter, and the results are evaluated in section 6.1.

## 3.1 Creation of initial object hypotheses

Initial object hypotheses are required to guide the physical interaction. The robot needs an idea where movable objects may be located in order to be able to perform a targeted pushing action, which in turn causes actual objects to move and thus allows the visual segmentation. This means that the initial hypotheses do not need to constitute good segmentations, but rather they should only indicate the presence of objects.

Three different and somewhat complementary heuristics are used to generate initial object hypotheses. They are described in the following subsections.

### 3.1.1 Single-colored regions

An intuitive indicator for potential objects are connected image regions of a single color. While only a small subset of all actual objects has only one color, such regions are an effective clue for this class as well as for single-colored parts of larger objects.

Single-colored regions are determined in the camera images using a variant of maximally stable extremal regions (MSER, [Matas et al., 2002]). The original algorithm finds regions in greyscale images that have similar brightness values and stay constant when the brightness threshold used to create them is varied. Thus these regions are significantly different from their direct surroundings, and robust to variations in illumination and threshold parameters. They are determined efficiently based on a watershed algorithm. In

[Forssen, 2007], this idea is extended to color images to find regions of similar color that are robust. It is implemented e.g. in OpenCV ([Bradski, 2000]).

### 3.1.2 Geometric primitives

Another heuristic that is particularly useful for artificial objects is to look for geometric primitives like planes, spheres and cylinders. Such simple geometric structures often correspond to a part of, or a complete object. Most artificial objects can be described by one or more of these simple geometric structures, and also many natural objects can be approximated by them. Thus, such a primitive is a strong indicator for a physical object. In the 3D data obtained from stereo cameras or depth cameras, three types of primitives are determined and used as initial object hypotheses: Planes, spheres, and cylinders.

#### Planes

Given the perceived point cloud of a complex scene, finding a plane in it is difficult, as the vast majority of the points will be outliers with respect to any plane contained in the point cloud. In order to find the dominant planes, the Random Sample Consensus algorithm (RANSAC) proposed by [Fischler and Bolles, 1981] is used. The idea of RANSAC is to draw minimal random samples from the point cloud that can define the desired geometric primitive, and then count the number of inliers for it in the complete set.

A plane is uniquely defined by three non-collinear points $p_1, p_2, p_3$. With $v_1 = p_2 - p_1$ and $v_2 = p_3 - p_1$, the surface normal is $n = v_1 \times v_2$. The plane is then defined by the equation $n^T x + d = 0$ with $d = -n^T p_i$ for any $p_i \in \{p_1, p_2, p_3\}$. When $n$ is normalized, the equation $n^T x + d$ gives the distance of a point $x$ from the plane.

For a plane defined by a random sample, the distance of each point in the scene to that plane is calculated and the number of inliers within a margin around the plane is determined. The plane with maximal support is retained, all inliers are removed from the point set, and the search is repeated. This is done iteratively until no more plane with at least a minimal support is found.

The determined planes may correspond to object surfaces, but they may also extend over two or more objects that are arranged in a way such that their surfaces are coplanar. To reduce the occurrence of this problem, clustering is performed on the planes to split them up if necessary. After this clustering it is still possible that two neighboring objects are subsumed in one plane. However, if such an object hypothesis is pushed, then the motion will most likely split it up.

### Spheres

Finding spheres in the scene point cloud is done in the same way as detecting planes. A sphere is uniquely defined by four points which are not coplanar. The sphere's center $c$ and radius $r$ can be calculated by solving the determinant equation

$$|M| = 0,$$

where $M$ is defined as

$$M = \begin{pmatrix} x^T x & x^T & 1 \\ p_1^T p_1 & p_1^T & 1 \\ p_2^T p_2 & p_2^T & 1 \\ p_3^T p_3 & p_3^T & 1 \\ p_4^T p_4 & p_4^T & 1 \end{pmatrix}.$$

With $M_{ij}$ denoting the submatrix of $M$ formed by leaving out row $i$ and column $j$, the solution is given by

$$c = \frac{1}{2} \begin{pmatrix} \frac{|M_{12}|}{|M_{11}|} \\ -\frac{|M_{13}|}{|M_{11}|} \\ \frac{|M_{14}|}{|M_{11}|} \end{pmatrix} , \quad r = c^T c - \frac{|M_{15}|}{|M_{11}|} .$$

If $|M_{11}| = 0$, the four points are coplanar and there is no solution. To find a sphere, RANSAC is used as described above. In contrast to planes, subsequent clustering did empirically not seem to be necessary.

## Cylinders

Finding cylinders is more difficult than finding planes or spheres. In [Beder and Förstner, 2006], methods are presented that allow the calculation of cylinder parameters given 5 - 9 points on its surface, but only the methods that require 7 or 9 points return a unique solution. In all cases, equation systems must be solved that require a relatively high computational effort. Moreover, the probability of randomly selecting 7 points belonging to a cylinder is rather small if the overall set consists mainly of outliers, which is usually the case in nontrivial scenes. Therefore, many iterations of RANSAC would be necessary to find good cylinder parameters.

Therefore, instead of using the direct RANSAC approach, the idea presented in [Chaperon and Goulette, 2001] is followed. Two nested RANSAC loops are executed which are both computationally inexpensive. In the outer loop, promising candidates for the cylinder axis are discovered. In the inner loop, the points are projected onto a plane that is orthogonal to the candidate

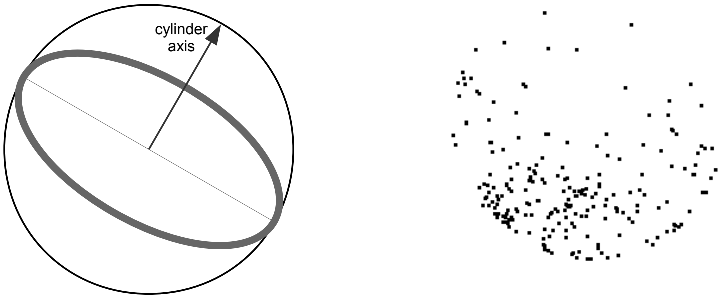


Figure 3.2: Gaussian image of a perfect cylinder, and of the real 3D data perceived from a cylindrical object.

cylinder axis, which reduces the problem of finding the other cylinder parameters to the search for the best two-dimensional circle within the projected points.

Finding cylinder axis candidates requires the calculation of the Gaussian image of a 3D point cloud. The Gaussian image consists of the surface normals at each of the 3D points. With surface normals having unit length, this is equivalent to a set of points on the unit sphere. The Gaussian image is calculated by approximating the surface normal at every 3D point using its nearest neighbors.

The surface normals of a cylinder form a great circle on the unit sphere (see Figure 3.2), therefore great circles in the Gaussian image that contain a large number of estimated normals are of interest. The axis of the cylinder is perpendicular to the corresponding great circle. The search for great circles is simplified by the fact that it is equivalent to the intersection of the unit sphere with a plane through its center. Such a plane is uniquely defined by two normals, i.e. two points on the unit sphere, and its center. This allows efficient use of RANSAC for planes to find great circles with maximal support. Figure 3.2 shows on the left how the Gaussian image of a perfect cylinder would look like. The right image shows the Gaussian image resulting from a set of 3D points in a scene containing a cylindrical object. Only the front side of the object is visible for the cameras, therefore only half of the great circle is present in the Gaussian image.

Given a cylinder axis, the next step is to find a radius and offset such that the maximal number of points lie on the cylinder surface. This can be simplified by projecting all points to a plane perpendicular to the axis and searching for the circle that contains the maximal number of the projected points. The search can be further sped up by considering only those points that have contributed to the great circle in the Gaussian image which defined the cylinder axis.

A 2D circle is uniquely defined by 3 non-collinear 2D points $(x_i, y_i)$ with $i \in \{1,2,3\}$. The coordinates of the center of the circle $(x_c, y_c)$ are then given by

$$\begin{aligned} x_c &= \frac{(y_3-y_2)(x_1^2+y_1^2)+(y_1-y_3)(x_2^2+y_2^2)+(y_2-y_1)(x_3^2+y_3^2)}{2\delta} \\ y_c &= \frac{(x_3-x_2)(x_1^2+y_1^2)+(x_1-x_3)(x_2^2+y_2^2)+(x_2-x_1)(x_3^2+y_3^2)}{2\delta} \end{aligned}$$

where

$$\delta = x_1(y_3-y_2)+x_2(y_1-y_3)+x_3(y_2-y_1).$$

The cylinder radius is $r = \sqrt{(x_1-x_c)^2+(y_1-y_c)^2}$, and the offset is an arbitrary point on the line that results from the backprojection of the circle center into 3D space.

## 3.1.3 Salient regions

The heuristics for single-colored and simply-shaped objects cover a wide variety of possible appearances. In order to detect objects that do not fall into one of those categories, a third cue was implemented. Salient image regions that are not yet covered by one of the other heuristics are also used to create initial hypotheses. The term *salient* is used in the context of cognitive

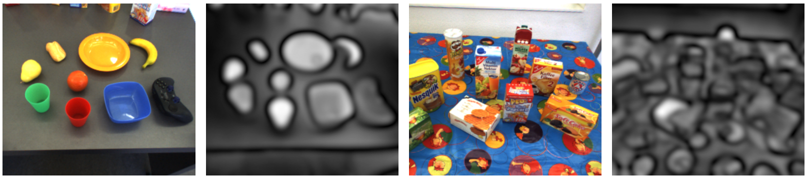


Figure 3.3: A relatively simple and a confusing scene with their respective saliency images, generated with the algorithm from [Achanta et al., 2009].

sciences and means that something is visually "sticking out", i.e. it catches an observers attention (see e.g. [Frintrop et al., 2010], [Itti et al., 1998]). A more practical definition is that it should look significantly different compared to its nearby environment.

To calculate a saliency map of the camera image that quantifies the saliency of each pixel, the approach proposed in [Achanta et al., 2009] is used. In that work, saliency is defined as the difference of an image region to its neighborhood, which is calculated at different scales using band-pass filters. The filters are realized using a Difference of Gaussian (DoG) filter $G(x,y,\sigma_j) - G(x,y,\sigma_k)$ with $\sigma_j > \sigma_k$. Summing up all edge images at different scales is equivalent to using a filter that is the sum of all filters, which can be simplified as follows:

$$\sum_{n=1}^{N} G(x,y,\sigma_n) - G(x,y,\sigma_{n+1}) \;=\; G(x,y,\sigma_1) - G(x,y,\sigma_N)$$

Thus the resulting saliency image is calculated as

$$S = |G(\sigma_1) * Img - G(\sigma_N) * Img| \, ,$$

i.e. the difference of the image after being filtered with a Gaussian kernel with the lowest and highest desired standard deviation. This is done for all three color channels of the RGB image, and the results are added. Good results with the used cameras were obtained by choosing $\sigma_1 = 80px$, which limits the size of detected regions to a size that corresponds to the maximal extent an object is expected to have in the image, and $\sigma_N = 10px$ which smoothes out the fine textures. Two examples for saliency images can be seen in Figure 3.3.

### 3.1.4 Filtering of the initial hypotheses

In a complex scene, the combined use of all three heuristics yields a large number of initial object hypotheses. Figure 3.4 shows two examples for the kind of scenes considered here, and the initial hypotheses that are generated in them. Having many object hypotheses is not a fatal problem, as the robot could just systematically try all hypotheses, including those that result e.g. from the tablecloth. But it can save a lot of time to filter out the dispensable hypotheses beforehand.

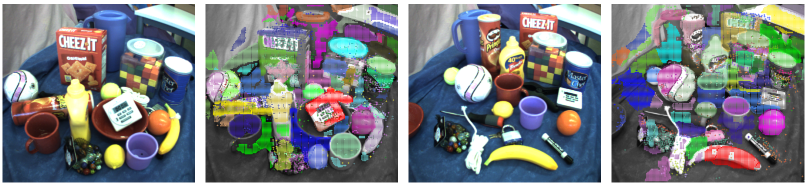


Figure 3.4: Two example scenes, and the initial object hypotheses generated in them.

Therefore, as only those objects that can be moved by the robot are relevant, an additional filtering is applied in order to keep only those hypotheses that seem to allow pushing. A simple criterion for estimating if this is the case is to check whether a candidate object is higher than its direct neighborhood. The camera image is subdivided into regular bins, and in each of them the average height of the contained 3D points is calculated. Every bin is compared to its eight direct neighbor cells. By doing this at different scales and adding up the results, a map is created that gives a value for the relative local height of the image regions. This map is used to filter the object hypotheses and keep only those that lie in a region which is higher than its direct surroundings.

### 3.1.5 Representation of the object hypotheses

Camera images as well as 3D information are used throughout the approach. When working with a stereo camera pair, the 3D position of a pixel in the camera image is obtained using dense stereo matching with the semi-global block matching algorithm proposed by [Hirschmuller, 2008]. When using a camera with active depth perception like the Kinect, the distance of each point from the camera is immediately available.

In both cases, the available visual information consists of a camera image in which each pixel is associated with a 3D position. This is usually referred to as RGBD (RGB+depth) data. After the initial object hypotheses have been generated, each one is represented by the RGBD points in the image region that it occupies. Thus, an object hypothesis is represented as a set of 3D points with associated color information. When needed, the location of such an object point in the camera image is obtained by projecting it into the image plane using the known camera parameters. In this way, the image region occupied by the object can be determined. In the rest of this chapter, for most steps of the segmentation algorithm the color-annotated point cloud is used though.

## 3.2 Induction of object movement by pushing

One of the hypothetical objects is chosen to be moved in order to verify its objectness and create an additional cue for its segmentation. Unless there are reasons to choose a specific hypothesis, it makes sense to select an object that is well within the area that is reachable by the robot, and has an appropriate

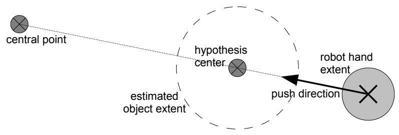


Figure 3.5: Key points of the pushing motion, depending on the estimated object position and size.

size with respect to the physical manipulation and perception components of the robot.

### 3.2.1 Generation of a pushing trajectory

As a priori nothing certain is known about the object, attempting to plan and perform complex manipulation actions is not very promising. Instead, a simple pushing action is executed to cause the object to move, and thus gain more reliable information about it. The motion to be induced on the object must be significant enough to be clearly distinguishable from noise and misperceptions. On the other hand, the resulting object motion should not be so large that it completely changes the object's appearance, which would make it difficult or impossible to relocalize and verify the hypothesis after the push. In particular, the object needs to stay within the robot's field of view and inside the area that the robot can reach with at least one of its arms. The latter aspect becomes increasingly more important when the object is pushed several times.

The object position is assumed to be the mean of the points contained in the hypothesis. Although this is not the true position of the object center, it is a sufficient approximation for the generation of a simple pushing action. To direct the push, a central point is chosen, which is positioned right in front of the robot at the same height as the hypothetical object. This point is well within the field of view of the robot and is also easy to reach with both arms. The starting point of the pushing movement is taken to be on the side of the hypothesis that is opposed to the central point. The main difficulty is to choose the right offset from the object center, as the size of the object is unknown. It is estimated by taking the maximal distance of hypothetical object points from the center of the hypothesis. To this value, the size of the robot hand and a safety margin are added (see Fig. 3.5). The end point of the pushing movement is determined by calculating a point at a fixed distance from the starting point towards the central point.

Consequently, the object is expected to be pushed into the direction of the central point, and over a fixed distance. In the long term, that means that over a large number of pushes its position will be oscillating around the central point. Thus it will be in a central region within the field of view, well reachable by the robot, and each pushing action will move it roughly over the intended distance. While in the long term it is interesting to reveal different sides of the object for learning, it is initially not helpful to intentionally cause larger rotations, as they increase the difficulty of relocalizing the object.

### 3.2.2 Execution of the pushing action

Before and after executing the push, the robot is in an initial pose in which its arms and hands are outside the field of view to avoid occluding the object. Starting from this initial pose, the hand is moved in straight lines to four keypoints:

- to a position above the starting point for the pushing movement
- downwards to the starting position for pushing
- to the end position of the intended push trajectory
- to a position above the end position for pushing

In the end, the robot moves back to its initial pose with the hand outside the field of view. By first moving the hand to a position above the starting point for the pushing motion, unexpected collisions with other objects in the scene can mostly be avoided. It is therefore not necessary to use sophisticated approaches for collision-free path planning, which would be difficult to accomplish in scenes with many unknown objects.

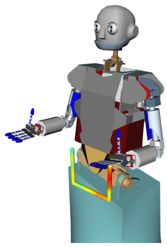


Figure 3.6: Reachability values on a pushing path for the left hand of ARMAR-III. Dark red dots indicate low, bright dots high reachability values.

One of the two hands of the robot has to be chosen for executing the push. A useful criterion for this decision is to analyze the reachability of both hands as described in [Vahrenkamp et al., 2012a]. The reachability is encoded as a discretization of the Cartesian space around the robot that gives information about the probability that an inverse kinematics solution exists for a given 6D pose. This allows to determine whether a position can be reached by one of the robot hands and how flexible the selected hand is in this area. The robot should use the hand that is more versatile during the whole course of the pushing motion, therefore the reachability along the whole pushing trajectory is integrated. Figure 3.6 shows an example of such a path. The hand that accumulates higher reachability over the complete trajectory is selected for pushing. A much simpler criterion that works as well is to choose the left hand when the push direction is from left to right, and the right hand when it is from right to left.

In cluttered scenes, the hand may still collide with other objects, especially during the phase when it is lowered to the starting position for pushing. Such collisions are detected by the force-torque sensor in the wrist of the robot. If such a collision occurs, the hand is raised again and lowered a bit closer to the estimated object position. This is repeated until it could be lowered without collision. If the hand collides with an obstacle several times, the correcting movements eventually bring it very close to the object. When the hand is so close to the estimated object center that it is probably positioned above the object, the robot moves it down until contact with the object and executes a sliding movement instead of a push. This reactive strategy enables the robot to move the objects even in very difficult situations.

Figure 3.7 shows example trajectories of the hand during pushes with and without collisions. The trajectory represented by the continuous red line was recorded during an undisturbed push, while the dashed green one was generated when the hand hit an obstacle twice before it could be lowered besides the object. The forces arising due to collisions with the obstacle are shown by the dotted blue line.

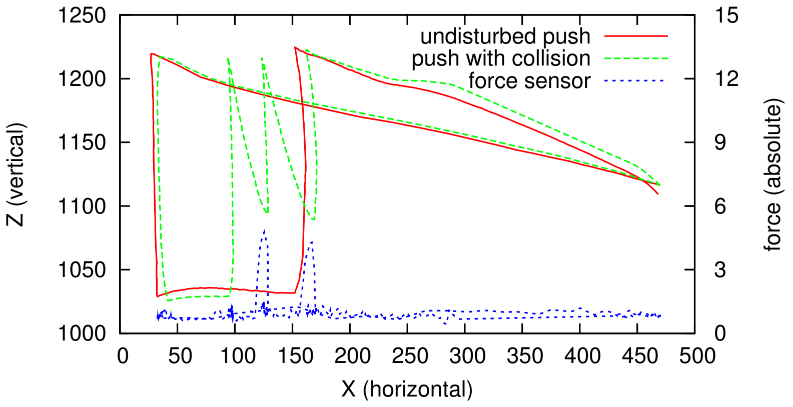


Figure 3.7: Trajectories of the robot hand during two pushes.

## 3.3 Estimation of the object motion

The robot's push may have moved one or more objects in the scene. Tracking an unknown number of unknown objects during the pushing motion, where they may at times be occluded by the robot's hand and arm, would be extremely hard. Therefore, the objects that moved are relocalized after the push instead, using the initial hypotheses and the change in the scene that was caused by the physical interaction.

### 3.3.1 Finding changed image regions

The first step is to determine whether the pushing action had any effect and which parts of the scene were changed. This is done by comparing the camera images before and after the push. Just calculating the difference of the two images would be very sensitive to image noise and changes of illumination. Therefore, before the push, $n = 5$ images are recorded, and for each pixel the mean and covariance matrix of its hue and saturation value are determined. That means that for every pixel there is a two-dimensional mean $\mu_i$ and covariance matrix $\Sigma_i$ of its hue and saturation value. Assuming that the camera image is subject to normally distributed random noise in each pixel, the probability that a pixel $i$ takes on the value $x$ is

$$P_i(x) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_i)}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i (x-\mu_i)}.$$

In the new camera image after pushing, this probability is calculated for each pixel, and if it is below 0.5 the image is considered to have changed in this pixel. The result is a binary mask that indicates in which parts of the image the scene has changed. By applying morphological operations, spurious pixels caused by noise are removed and contiguous changed image regions are closed.

Using the resulting binary mask, each of the initial object hypotheses is checked for lying in a part of the scene that changed. If more then half of the points belonging to it are in such a region, this indicates that the underlying object has moved and thus a relocalization will be attempted. Otherwise no additional information about the underlying object can be gained, so the hypothesis is discarded. If the whole scene is unchanged, the push obviously failed to move an object, and the process is restarted by generating new initial object hypotheses.

The detected change in the scene is also used to generate more object hypotheses. In the altered regions, geometric clustering is used to create at least two new hypotheses - if an object moved, the camera images differ at its old and new location. If a single object was moved significantly, the clusters often form extremely good hypotheses. If several objects move, their quality varies strongly.

## 3.3.2 Iterative closest point (ICP)

The relocalization of the objects that were moved by the robot's push is primarily based on the Iterative Closest Point (ICP) algorithm proposed by [Besl and McKay, 1992] (a somewhat similar idea was presented by [Chen and Medioni, 1991]). The aim of this algorithm is to align two point clouds $C_1$ and $C_2$ by finding the rigid transformation that minimizes the mean square distance of each point $p_i \in C_1$ to its nearest neighbor $N_{C_2}(p_i)$ in $C_2$, i.e. that minimizes

$$d(C_1, C_2) = \frac{1}{|C_1|} \sum_{p_i \in C_1} (p_i - N_{C_2}(p_i))^2.$$

Starting from an initial alignment, the algorithm iteratively applies rigid body transformations, i.e. a rotation and translation, to the points in $C_1$, that reduce the mean square distance $d(C_1, C_2)$ of $C_1$ to $C_2$ (note that this distance measure is not symmetric). The following steps are repeated until a convergence criterion is fulfilled:

- For each point $p_i \in C_1$, its nearest neighbor $N_{C_2}(p_i)$ in $C_2$ is determined. This can be done in expected time $O(|C_1| \log |C_2|)$ using a KD-tree.
- Based on these correspondences, the 3D transformation that minimizes the mean squared distance between all the pairs is calculated. This is done using the method described in [Horn, 1987].

- The determined rotation and translation are applied to all $p_i \in C_1$.

These iterations are repeated until the reduction of $d(C_1, C_2)$ is below a threshold or a maximal number of iterations has been performed. In each step, the mean square distance is reduced, until the algorithm converges. However, it converges to a local optimum depending on the initial position and orientation of the two clouds, and thus does not necessarily find the globally optimal alignment (see Figure 3.8).

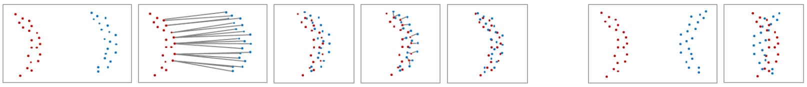


Figure 3.8: Examples of ICP in 2D: On the left is a successful alignment, on the right the unfortunate initialization leads to a bad local minimum.

### 3.3.3 Object relocalization

For the relocalization of a pushed object in a complex scene, ICP in its original form is not suitable. While it works well for aligning two relatively similar point clouds, finding a small (object) point cloud within a large and complex (scene) point cloud is much harder. There are three main reasons causing this difficulty:

- The algorithm converges to the nearest local optimum. That means that if the start position for the alignment is not very close to the true new position of the object, it will most probably just end up in a nearby part of the scene that has a shape more similar to the object than its direct neighborhood. Therefore, a global search approach is necessary.

- Most object shapes are not very unique. In particular, artificial objects are dominated by planar surfaces, and scenes containing artificial objects and structures have lots of planes to which the object point cloud can be aligned. In general, any smooth surface with only modest curvature can be aligned well with any other smooth surface. Using only shape information is therefore insufficient for reliable matching.

- These two difficulties are aggravated by the fact that the initial object hypotheses are often far from being correct. These point clouds usually contain only a part of the actual object, and often enough points that do not belong to it. This much more than the sensor noise (which of course is also present) makes the matching of the initial hypothesis with the moved object much harder.

Two measures are taken to overcome these difficulties. The first is a slight extension of the original ICP algorithm to incorporate color information. This is done by representing each point not only by its 3D position, but also by a normalized RGB vector and a brightness value. For finding the nearest neighbor of each point in the first step of each ICP iteration, the distance between two points is defined as the weighted sum of their cartesian distance, their distance in normalized RGB space, and difference in brightness.

The weighting allows to balance the relative importance of color and shape matching. The weight of the color component should not be too small to avoid mismatching due to similar shapes. On the other hand, if it is set too high, the risk of mismatches due to similar color rises. It is set to such a value that the maximal possible color and brightness distance is equivalent to a cartesian distance of 10 cm. Empirically, values between 5 and 30 cm produced reasonable behavior. The choice may also depend on the precision of the 3D sensor and the point cloud density. By using both shape and color information, the problem of mismatching in case of similar shapes, which would otherwise occur frequently, can be reduced significantly. There are extremely few publications on work in which color information has been used to improve the classical ICP algorithm, one example is [Johnson and Kang, 1999].

The second measure is aimed at overcoming the local character of point cloud matching with ICP. This is done by starting the alignment at several different initial locations.

However, first the old object position is checked, i.e. ICP is started there. If a good match is found, i.e. the resulting mean distance in cartesian and color space is small, and the determined transformation indicates that the hypothesis did not move significantly, this means that the object is most probably still roughly at its old location. In that case the motion is insufficient to provide robust information for segmentation, and the hypothesis is discarded.

If the determined transformation is large, or only a bad match was found, the object has obviously moved. In this case, a global search is performed. Even if a good match was found when starting ICP from the old object position, this global search often leads to better results, and avoids mistakes due to credibly-looking mismatches.

Thus, ICP is executed several times with different initial estimates of the new object pose, and the resulting transformation that yields the best match is kept. As the object may have been moved over a large distance, finding it again requires an appropriate choice of the initial poses for ICP. To this end, a color histogram of the object hypothesis is created. Regular windowing on different scales is performed over the whole image to find regions with color histograms that are similar to that of the object hypothesis. In each of these locations, and at each location in different orientations, ICP is started. The necessary number of different initial positions can be reduced by taking into account the direction of the push, which must not be done in a too restrictive manner as the caused object motion is rather unpredictable.

The best transformation returned by the differently initialized registration attempts is refined by another execution of ICP on a reduced point set, where all those points are left out that still have a large distance to their nearest neighbor. If the hypothesis contains points that do not belong to the object on which the main part of the hypothesis is lying, this last step leads to an alignment that is not affected by them.

### 3.3.4 Verification of the object transformation

Some plausibility checks are performed on the determined transformation in order to avoid using it if it can be detected to be unreliable or plainly wrong. Probable mismatches are detected by checking the mean cartesian and color distance of the object points to their nearest neighbor in the scene at the new location. If this distance is too large, i.e. no good match was found, the hypothesis is discarded. This happens in most cases when the initial hypothesis did not correspond to a real object, and may also sometimes occur when the object's appearance changed too much due to rotation.

Another criterion is that at its new location, the object has to lie in a part of the scene that has changed due to the push. If the new location is in an unchanged part, even if the match seems good, the object hypothesis was obviously aligned with something that was already there before. To give an example, this occurs quite frequently when the objects are arranged on a unicolored table. If there was an object hypothesis on a part of the table surface and this region is changed because an object was moved there, the hypothesis is usually matched to another part of the table. This leads to a very good match quality, but can easily be recognized and discarded by noticing that the new location of the hypothesis is in an unchanged region.

Finally, the estimated object motion has to be large enough to give reliable information for segmentation. The threshold for the motion must be high enough to allow certain distinction between actual motion and noise in the visual perception. On the robot used in this work, given the precision of the depth perception and a usual distance of 50-80 cm between camera and object, a threshold of 3 cm for the object translation turned out to be definitely safe. If all these criteria are met, the object hypothesis is considered to be verified.

## 3.4 Improving the verified object hypotheses

Those initial hypotheses for which a valid transformation has been determined do most likely belong to an actual object that has been moved by the robot. However, most probably they do not cover the object completely, and they often contain points that do not belong to the object.

### 3.4.1 Correction and extension of the hypotheses

Points that do not belong to the actual object are removed by checking each point of the hypothesis: After applying the estimated object motion, a point must match its nearest neighbor in the scene point cloud well with respect to cartesian and color distance. It also has to lie in a region that changed due to the push. If both of these criteria are met, the point is considered to be verified, otherwise it is removed from the hypothesis.

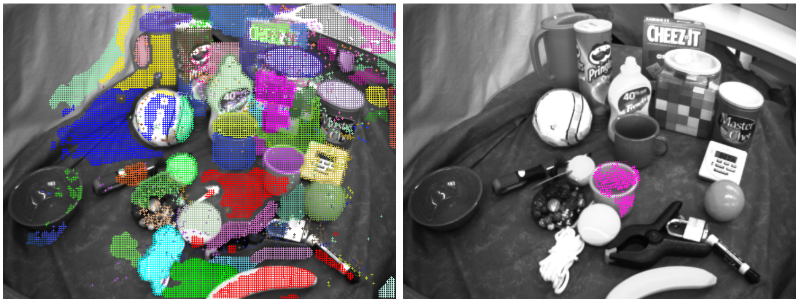


Figure 3.9: Initial object hypotheses, and the confirmed hypothesis after pushing.

In this way, those points from the initial hypothesis that do not lie on the same real-world object as the main part of the hypothesis are identified: If the transformation of the object is applied to them, they will end up somewhere near the new object location. As the other object they actually belong to did not move, or moved differently, there will most probably be no good correspondence for them in their new location: Either there is no scene point at all nearby, or if there is, it has a different color.

The point may also end up in a region that was not changed at all by the pushing action, and is thus discarded. The remaining hypothesis points accord with the overall transformation, and are thus considered verified. Figure 3.9 shows an example scene with many initial hypotheses, out of which one is confirmed and corrected after being pushed.

However, as the initial hypotheses may only have covered a part of the object, a confirmed hypothesis often does not cover the whole object, and needs to be extended. To this end, more points are added that may belong to the same object. All scene points that lie close to the verified points and within the image region that changed are promising candidates for this. A new point is added to the hypothesis if it fulfills four criteria: It has to be close to one of the verified points, but not so close that it coincides with it. The point must lie in a changed image region, and its old position before the push must also have been in a changed region. This hypothetical old position is calculated by applying the inverse of the transformation of the hypothesis to the point.

The last check helps to filter out some of the points that belong to other objects that were also moved. However, it is not very restrictive, and by explicitly not checking that it has a good correspondence in the old scene before the push, it is possible to add points that were not visible before, e.g. because they were on the back side of the object.

By pushing the object again and repeating the steps described before, these new candidate points can be verified or discarded, and new candidates can be added. In this way, the hypothesis can grow in all directions and holes can be filled. It is important that only the verified points are used to estimate the object's transformation, so that even by adding a lot of new candidate points no harm can be done. However, allowing for new points to be too far from the verified points may sometimes lead to some of them being verified by chance, which may affect the estimation of the transformation in later iterations. Depending on the maximal permitted distance, the object size and the quality of the initial hypothesis, it usually takes two or three pushes until the whole object is contained in the hypothesis and thus segmented completely.

### 3.4.2 Improving a hypothesis over several pushes

Usually, more than one object hypothesis is verified by the first push and the subsequent analysis. This happens in particular when several actual objects are moved, but also when more than one hypothesis is lying on the same physical object. For the second push, the hypothesis containing the maximal number of confirmed points is chosen. It is probably the most reliable one, and if there are several hypotheses on the same object it is the one covering the largest part of it.

In the default case considered in this work, the robot only wants to segment and learn one object at a time. Therefore, after the second push, the hypotheses that did not move are discarded, and from the remaining ones the one with the maximal number of confirmed points is selected for further pushing. If the robot does indeed move several objects, all of them can be segmented, but for the sake of simplicity only one is considered in the descriptions here.

Figure 3.10 gives an example for the improvement of the segmentation over several pushes. The first image shows the initial hypotheses, and each of them is represented by crosses of the same color. The following images show the confirmed hypotheses after one to five pushes. Here, the crosses denote verified points, while the small dots indicate new points that were added to the hypothesis as new candidates. After the first push, two hypotheses that both cover a part of the bowl that was moved are verified. There is also a small verified hypothesis on the edge of the bowl and a neighboring object that moved slightly. After the second push, the hypotheses cover most of the object, and after three pushes they cover it completely. As they now overlap

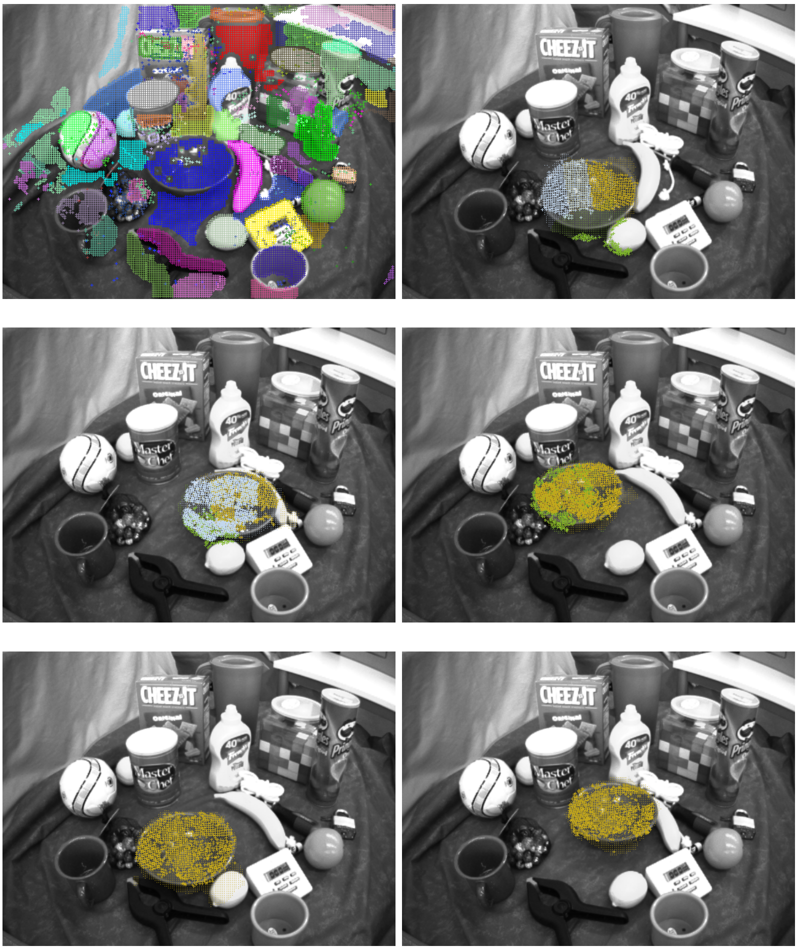


Figure 3.10: An example for the object segmentation by physical interaction: The initial hypotheses, and the confirmed hypotheses after one to five pushes (row-wise from left to right).

significantly, the smaller ones are discarded, and after four pushes only one is left that covers the object completely with confirmed points.

When several objects are moved by a push, as long as they undergo different 3D transformations, they can be separated based on their different motion. It may happen though that two touching objects move exactly alike, in which case they are subsumed in one hypothesis. Most likely they are separated when pushed several times from different directions. Heuristics for systematical pushing to this end have been proposed in [Chang et al., 2012] and [Hermans et al., 2012], as described in section 2.2. When two objects contained in one hypothesis are separated, the hypothesis will follow the object that is matched better after the motion, which is usually the bigger one. Pushing an object several times often reveals different sides of it, thus the creation of a multi-view object descriptor is possible, although some sides (in particular the bottom) will probably never be observed.

Figure 3.11 shows an interesting example where a hypothesis extends over two objects, which are both moved by the first push. The first image shows the initial object hypotheses, and the second image shows the verified hypotheses after the first push, which has moved the cookie box and the tea box in the bottom left part of the image. The hypotheses represented by blue, turquoise and dark green crosses on the cookie box have been verified, as well as the one represented by green crosses on the tea box and the yellow one that stretches over both objects. The other initial hypotheses either did not move or could not be relocalized after the push. As can be seen, most points of the hypotheses that lie on only one of the two objects were verified, while only some of the points belonging to the yellow hypothesis that stretches over both were verified and most were discarded. Neighboring points in the changed image regions were added to all hypotheses as new candidates.

After the third push, only the tea box has moved, so the hypotheses on the cookie box are discarded. The yellow hypothesis that was stretching over both objects is also discarded, because the larger part of it was lying on the cookie box and therefore it did not move. The hypothesis represented by the

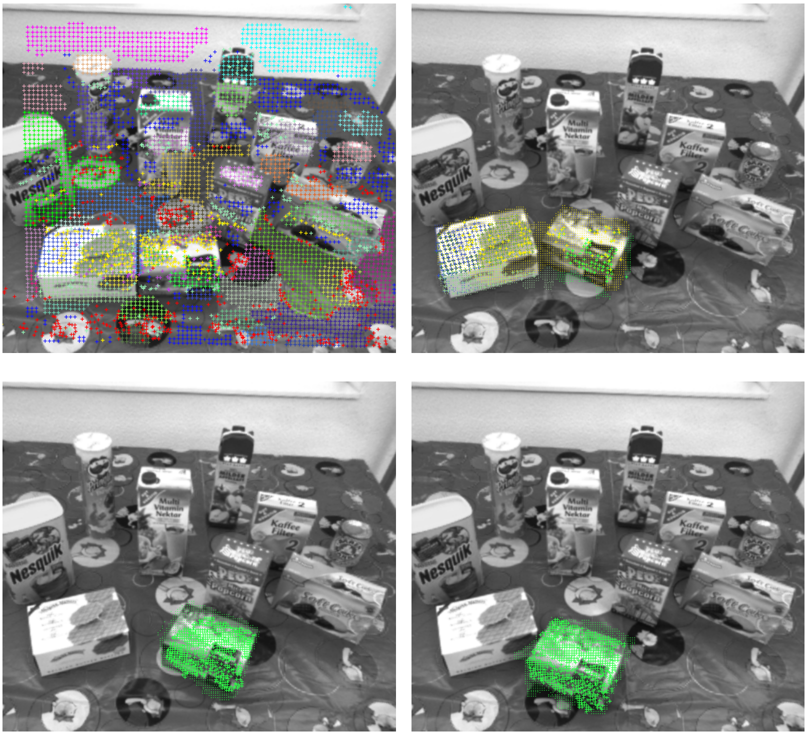


Figure 3.11: An example for the object segmentation by physical interaction, where an initial hypothesis covers two different objects.

green crosses on the tea box is extended significantly, as the new candidate points added after the first push have been confirmed now.

In the fourth image, which shows the remaining verified hypothesis after the third push, the whole tea box is covered by verified hypothesis points. Around it are new candidate points which would be discarded though after another push, so the hypothesis would not change significantly over further pushes, except for parts of the object that would become visible or invisible due to rotation.

## 3.5 Object learning

After two to three pushes, the resulting object segmentations usually cover the actual object almost completely and rarely contain any outliers (see evaluation in section 6.1.1). This knowledge about what part of the perceived scene constitutes the object is an essential prerequisite for learning its visual appearance or other properties. An obvious use of the segmentation is to learn a visual object descriptor that can be used for recognition and localization.

Object recognition is arguably the most classical task of computer vision, and has accordingly been subject to intensive scientific efforts over the last decades. It is, however, still not a solved problems, and a recognition system that works reliably on any kind of object in cluttered scenes or under occlusion still does not exist. Two surveys that give an overview of commonly used 2D and 3D descriptors for object recognition are [Mikolajczyk et al., 2005] and [Guo et al., 2014]. Any kind of 2D or 3D descriptor can be used on the segmentations obtained here, as both image and point cloud information of the object are available.

In order to demonstrate the suitability of the segmentations for learning visual object descriptors, a simple object localizer was implemented. The descriptor is simply the point cloud of a confirmed object hypothesis that was saved. The localization is performed in exactly the same way as the relocalization of an object hypothesis after pushing: All image regions with a similar color histogram are determined, and around each of them several instances of ICP are started. The result with the smallest mean cartesian and color distance is selected, and if the error is below a threshold, it is accepted and the object is considered to be recognized and localized there.

As the object has been pushed several times, different confirmed hypotheses are available that represent the object from different points of view. Using all of them simultaneously for localization, and choosing the best match as the result, allows for multi-view object recognition. It also increases the reliability of the recognizer, which is evaluated in section 6.1.3. For other types of recognizers, it might be helpful to fuse the different segmentations using the determined 3D transformations caused by the pushes.

Besides learning the visual appearance of the object, observing its motion may also serve to study physical properties of the object like its mass distribution or surface properties. Here, the forces and contacts perceived during pushing should also be taken into account.

Finally, the shape of the explored object is a particularly important property when more complex manipulation actions are intended. An approach for extending the perceived partial object shape to make it usable for manipulation planning is presented in the following chapter.

# 4 Object shape completion

When an unknown object has been segmented, e.g. using the approach described in the previous chapter, the robot knows which parts of the perceived scene belong to the object, and thus visual and shape information about it is available. However, although in the course of pushing it a few times different sides of the object may have been revealed, it can usually not be observed from all directions without being picked up.

Grasp planning, as well as most other robotic manipulation tasks, requires or at least benefits greatly from information about the object shape. Most traditional grasp planning algorithms require a complete and precise object model, which in general is not available, and in particular when dealing with unknown objects. In this chapter, a method is proposed for completing shapes that are only partially known, which is a common situation when a robot perceives a new object, and the typical result of the segmentation presented in the previous chapter.

The approach presented in this thesis is an extension of the work of [Bohg et al., 2011], and is similarly based on the assumption that most objects are roughly symmetric. Therefore, symmetry plane candidates are determined and rated in order to estimate the hidden parts of the object. The starting point for this approach is a point cloud, i.e. a set of 3D points, of the new object that the robot has segmented. In the worst case, the object has only been observed from one perspective, which means that only its front side towards the camera is available. Additionally, the point cloud of the scene around the object is used. By finding possible supporting planes in this immediate neighborhood, the search space for symmetry planes is restricted, and the bottom part of the object is added. Gaps along the sides in the direction of the view axis are closed by linear interpolation.

## 4.1 Symmetry plane hypotheses

The main assumption of this approach is that the object has a planar symmetry, thus the main challenge is to find this symmetry based on the available partial shape information. The perceived scene surrounding the object is used to restrict the search space that needs to be sampled.

### 4.1.1 Supporting plane hypotheses

Unless it is held by someone, any object must be standing on some kind of supporting structure. The robot is assumed to know the direction of gravity, which can easily be achieved using an inertial measurement unit. It is assumed that the structure supporting the object must be roughly perpendicular to this vertical direction.

The immediate neighborhood of the object is searched for candidates for the surface that the object is standing on. The neighborhood is obtained by removing the object from the scene point cloud and considering only those points that lie near it: With the object radius $r_{obj}$ being defined as half of the maximal diameter of the object point cloud, all points that are at most $3 \cdot r_{obj}$ away from the object center are considered to belong to its neighborhood.

In this neighborhood point cloud, the dominant planes are determined using random sample consensus (RANSAC, [Fischler and Bolles, 1981], see also subsection 3.1.2). The plane with maximal support is determined, and removed from the point cloud. This is repeated until no more planes with a minimal number of inliers are found.

From these planes, those which are reasonable candidates for being the supporting surface of the object are selected. There are two criteria that have to be met by each surface $S_i$. Firstly, the surface should be roughly horizontal, i.e. its normal $n_i$ should not differ too much from the direction of gravity $g$.

Consequently, the angle $\alpha = \arccos\left(\frac{|n_i \cdot g|}{|n_i||g|}\right)$ between $n_i$ and $g$ should be small. Secondly, as the object is assumed to be standing on it, there should not be too many object points lying below the plane. For each point $p_j$, its projection $p'_j$ onto the surface is calculated. The sign of the scalar product of $(p_j - p'_j)$ and $n_i$ indicates whether the point is lying above or below the plane. The rating of the surface candidate is the weighted sum of those two cues: the ratio of object points below the surface, and the angle between its normal and the direction of gravity. To limit the computational effort later, only the 10 supporting plane hypotheses with the best rating are considered in the subsequent steps.

### 4.1.2 Generation of symmetry plane candidates

Comparable to [Bohg et al., 2011], the symmetry plane of the object is assumed to be perpendicular to the supporting surface, with the important difference that in this work there are several hypotheses for that surface, which removes the restriction that the object has to stand on an empty table. For each supporting surface hypothesis, symmetry plane candidates are created by uniform sampling. A plane can be defined by a support vector and two orthogonal vectors that span it. Initially, the center of the object point cloud $O_{obj}$ is used as the support vector. With $O^P_{obj}$ being the projection of $O_{obj}$ onto the supporting surface, the principal axis of $O^P_{obj}$ which is more orthogonal to the view direction is chosen as the first spanning vector, and the normal of the supporting plane as the second one.

Starting from this initial symmetry plane candidate, more candidates are generated by rotating and shifting the plane over regular intervals. The translation axis $a_t$ is the projection of the view direction onto the supporting plane. The translational sampling along this axis is restricted to an interval between the 0.1 and 0.9 quantile of the object point cloud along that axis. For each position, different orientations are sampled by rotating the plane around the normal of the supporting surface, with a maximal rotation of 45° in both directions.

The required computational effort to analyze and rate all symmetry plane hypotheses is proportional to the product of the number of supporting plane hypotheses, translational samples per plane and rotational samples per translation. Thus, restricting the sampling space efficiently is the key to having a high chance of good results within a reasonable computation time.

By testing different potential supporting surfaces, this extended approach allows for a higher flexibility compared to [Bohg et al., 2011], for the price of higher effort - or a lower sample density when keeping the effort similar. In section 6.3, the effect of different sampling densities is evaluated.

### 4.1.3 Calculation of points mirrored on a plane

Given a symmetry plane candidate with support vector $c$ and normal $n$, a point $p_i$ is mirrored to the point $p_i'$ by adding twice the vector from the point to its orthogonal projection onto the symmetry plane. With

$$p_i^P = p_i - \frac{(p_i - c)\cdot n}{n\cdot n} n$$

being the orthogonal projection of $p_i$, the mirrored point is

$$p_i' = p_i + 2(p_i^P - p_i) = 2p_i^P - p_i = p_i - 2\frac{(p_i - c)\cdot n}{n\cdot n} n$$

### 4.1.4 Rating of symmetry planes

In order to choose the best symmetry plane candidate, a rating of its apparent plausibility is necessary. The rating is calculated based on the locations of the points that are generated by mirroring the originally perceived object points on the symmetry plane. A mirrored point falls in one of four categories, depending on its position:

1. If it is very close to an original point, this indicates that the symmetry appears reasonable.

2. If it lies behind the original points, and thus would be invisible for the camera, it may be a correct estimation that introduces information about the back side of the object.

3. If, in contrast, the point lies besides or in front of the original object points, it would be visible to the camera and, if it belongs to the object, should be already included in the original point set. Therefore, it indicates that the symmetry plane hypothesis may be incorrect.

4. If a mirrored point lies underneath the estimated supporting surface, this also reduces the credibility of the candidate symmetry plane. (As the symmetry planes are perpendicular to the supporting plane, these points are the same for all symmetry hypotheses based on the same supporting surface candidate. However, they are helpful as a rating cue amongst symmetry hypotheses originating from different potential supporting planes.)

Figure 4.1 shows the different kinds of positions of the mirrored points with relation to the original object points (black). If the new points lie amongst the original ones, they support the symmetry plane hypothesis. If they lie behind them (gray), they give additional speculative information about the object shape. If they are in front of or beside the original points (red), they would have been visible before, hence indicating that the symmetry plane might be incorrect. The same holds for points that lie under the estimated supporting surface (orange).

To rate a symmetry hypothesis, each mirrored point $p_i'$ is classified according to these categories. The first check is for the point to be above or below the supporting plane - if it is below, its distance to the plane is added to the rating with a negative weight. For the remaining points, it is first checked whether they are beside the original point cloud when seen from the camera.

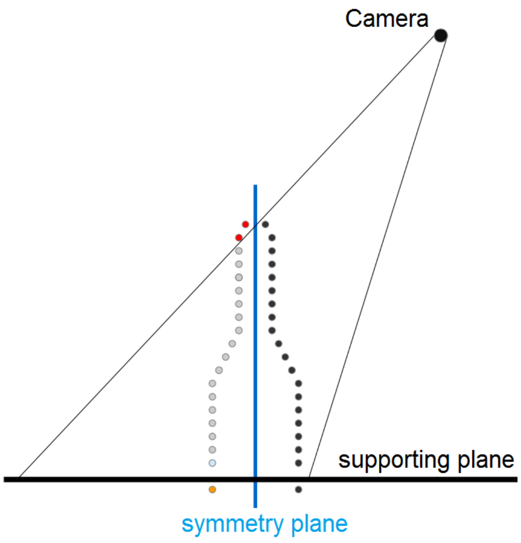


Figure 4.1: Visualization of the different regions where the mirrored points may lie with relation to the original object points.

This is done efficiently using a binary segmentation mask created by projecting the object point cloud into an image and closing it using morphological operations. If the point is projected outside that segmentation area, it is beside the object, otherwise it may be in front of, behind, or coinciding with the original points.

In any case, the nearest neighbor of $p'_i$ within the original point cloud is determined. If $p'_i$ is projected within the segmentation mask, the distances of $p'_i$ and its nearest neighbor from the camera are compared to decide whether it lies in front of, within, or behind the original point cloud. If it lies in front of or beside of it, the distance between $p'_i$ and its nearest neighbor is used for the rating with a negative weight. If it lies within or behind the original points, a positive value is added to the rating. These two positive weights can be used to balance the preference of symmetry hypotheses that either seem plausible for having many points that coincide with the original ones, or generate more speculative information about the back side of the object. In this work, the same weight is given to both categories.

The overall rating of a symmetry plane candidate is the weighted sum of the positive and negative ratings for all mirrored points. The rating is normalized by the number of points, which makes no difference for choosing the best symmetry plane but allows for comparison between different objects (or different perceptions of the same object). The symmetry plane candidate with the best rating and its corresponding supporting plane hypothesis are selected, and all original points are mirrored on it.

## 4.2 Additional shape completion steps

When testing the shape completion using only mirroring, it turned out that there were often significant holes left in the result. Usually, an object perceived by the robot is seen well from the front and from the top. The back side, and sometimes the rear part of the top when it is not flat, are reconstructed by the mirrored points. But the sides, when they are roughly aligned with the view direction, tend to be missing. This is due to the fact that with stereo cameras as well as active depth sensors, little reliable depth data can be perceived from these surfaces. Additionally, when an object is segmented, the borders of the segmentation are often slightly imperfect.

The bottom of the object is usually invisible to the camera, except when the object is bowl-shaped or open at the front. Little or no information can be added here by mirroring on a symmetry plane that is perpendicular to the supporting surface. As the goal is to estimate an object model that is as complete and free of gaps as possible to allow e.g. grasp planning algorithms to work with it, additional shape completion steps need to be performed in order to overcome these two frequent deficits.

## 4.2.1 Sides of the object

To define what are the sides of the object, its points are considered in the camera coordinate system. Here, the (approximately) vertical edges are of interest, i.e. the minimal and maximal values along the horizontal image plane axis for any existing value on the vertical axis. As the object is given in the form of a discrete point cloud and not as a filled spatial region, its outline has to be approximated.

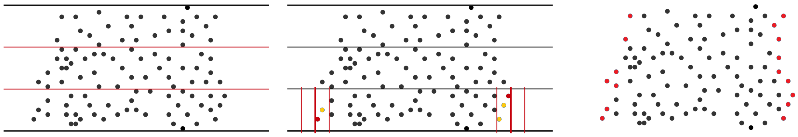


Figure 4.2: Determination of the sides of the object: The point cloud is divided horizontally, and in each segment the left- and rightmost points are determined. The points with similar horizontal positions form the edges.

First, all points are projected into the image plane, i.e. the two dimensions perpendicular to the view direction. One possibility would be to use the two-dimensional convex hull, but that would lead to great errors on concave shapes. Instead, the point cloud is subdivided along the vertical direction to get horizontal segments, in each of which the respective left- and rightmost points are determined. The size of the subdivisions is crucial, as too large intervals would eliminate details, while too small ones would carve into the object when no point on the actual border lies within the segment.

To choose an appropriate size for the segments, the mean distance $\bar{d}$ of a point in the original object point cloud $P_{obj}$ to its respective nearest neighbor is determined:

$$\bar{d} = \frac{1}{|P_{obj}|} \sum_{p_i \in P_{obj}} \min_{p_j \in P_{obj}/\{p_i\}} |p_i - p_j|.$$

The point cloud, projected into the camera plane, is subdivided vertically into equal intervals between its highest and lowest point, such that the height of the subdivision segments is greater or equal to $2 \cdot \bar{d}$. This gives a high certainty that actual edge points are contained in each segment, while still maintaining a reasonable degree of precision .

In each segment, the minimal and maximal points in horizontal direction are determined. These points, and all points in the segment that are within a distance of $\bar{d}$ from the vertical line through the minimal or maximal point, are considered to be part of the edge. Each of these edge points is connected with its mirrored counterpart on the back side of the object. Along the straight line connecting the original and the mirrored point, points are added in regular intervals of length $\bar{d}$.

Figure 4.2 visualizes this process: First, the point cloud is subdivided into horizontal segments. Within each segment, the left- and rightmost points are determined. They, and all points within the segment that are near the vertical lines through them, are considered to be edge points. This is shown in the central image for the bottom segment. In the right image, all resulting edge points are marked in red.

### 4.2.2 Bottom of the object

The bottom part of an object that is standing on something else is usually not visible, so it is a very common case that this part is missing in the perceived object point cloud. The estimated supporting plane is used to add a bottom part in the last step of the shape completion. The intuition is to estimate the area where the object touches the supporting surface, and close it there. Consequently, all points of the model generated so far, i.e. original as well as mirrored and side points, are checked for being very close to the supporting plane. The threshold here is set to $3 \cdot \bar{d}$.

Those points which seem to be close to the bottom of the object are projected onto the supporting plane. Their 2-dimensional convex hull is calculated and filled with a regular grid of points at intervals of $\bar{d}$. It is important

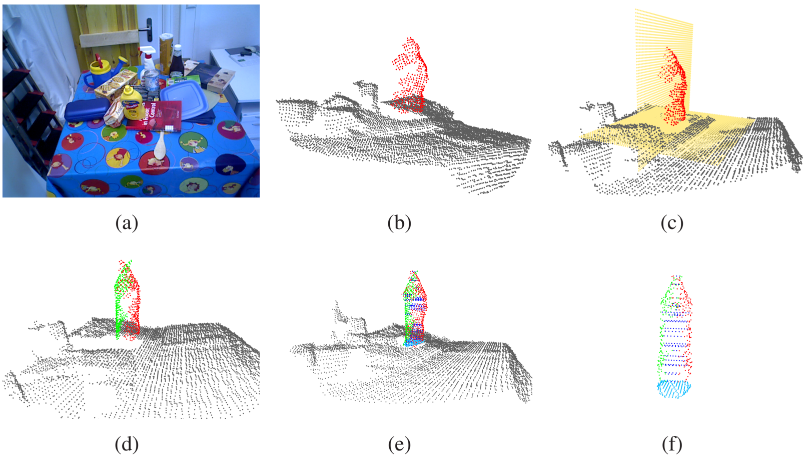


Figure 4.3: The steps of the object shape completion.

that only the low object points are used to to estimate the bottom area, because otherwise artificial planes would be created under object parts that spread out in higher parts of the object, e.g. the handle of a cup.

There are still cases in which this approach would create a too large bottom part. This occurs when the part of the object that is in contact with the supporting surface is not convex, or even unconnected, e.g. when the object has legs like a table. To limit this effect, all bottom points that would have been visible from the camera are removed, similar to what was done in the rating of the symmetry hypotheses in section 4.1.4.

Figure 4.3 shows the different steps of the shape completion algorithm. 4.3(a) is the camera image of the scene, in which the yellow mustard bottle has been segmented. 4.3(b) shows the point cloud of the object (red) and the part of the environment that is near it (gray). In 4.3(c), the estimated supporting plane and the symmetry plane that is used for shape completion are shown (both in yellow). It can be seen clearly that only the front side of the object was perceived. 4.3(d) shows the new points in green that were generated by mirroring of the original points on the symmetry plane. In 4.3(e) and 4.3(f), the final completed model is shown after adding the sides (dark blue) and the bottom (light blue).

## 4.3 Grasp planning with the completed object shape

The execution of a grasp in general comprises the motion of the arm, hand and fingers of the robot from an initial pose to a configuration in which the object is held firmly inside the hand. Collision-free motion planning in such a high-dimensional space is challenging ([Vahrenkamp et al., 2012b]). The most common approach is to separate the grasp and the motion planning: Possible grasps for the object are planned offline in advance, and at execution time one of them is selected considering the constraints given by the current scene, i.e. the robot must be able to reach the desired grasp pose relative to the object considering its kinematic limitations and possible obstacles (see e.g. [Chen et al., 2013] or [Asfour et al., 2013]).

For the task of grasping a new, unknown object however, both grasp and motion planning have to be performed after the object has been segmented and its shape estimated. Thus, it makes sense to simultaneously plan both, as proposed in [Vahrenkamp et al., 2012b], although of course the two planning tasks could also be solved separately if two different components are used for them.

The completed object shape that is generated as described in this chapter is represented as a point cloud. The planner described in [Vahrenkamp et al., 2012b] works on surface meshes, therefore the point cloud needs to be triangulated before grasp planning can be performed. For this, the method from [Marton et al., 2009] is used, and the triangle mesh that is created from the point cloud is provided to the grasp planner, which generates an appropriate grasp configuration. Finally, the planned grasp for the new object can be executed by the robot, which is described in the following chapter.

# 5 Visual collision detection for reactive grasp correction

When a new object has been discovered and segmented, its shape estimated and a grasp planned, this grasp finally has to be executed by the robot. It is in this stage that the effects of possible errors or imprecisions in all the perception and planning components may manifest themselves. When dealing with previously unknown objects as in this thesis, an additional source of errors is that the knowledge about the shape of the object is noisy and in parts uncertain. A grasp configuration that was planned based on an imperfect model like the completed object shape generated in chapter 4 might not be entirely correct given the real object shape.

Even when grasping known objects for which good grasp configurations are know, imprecision of the perception or the motion execution itself may cause problems, e.g. due to imprecise kinematic modeling or calibration. Thus, there is a huge variety of possible causes that may lead to failure of the grasp execution if it is performed without paying attention to arising problems.

Besides collisions with the environment, which are not considered in this work, the occurrence of a failure means that the hand of the robot does not succeed in grasping the object. This may either mean that the hand misses the object completely, or that parts of the hand collide with it before the grasp pose is reached. While a complete miss would only occur in the case of large errors, a premature collision of hand and object can happen easily, in particular when there is a low tolerance margin e.g. because the object is almost as big as the aperture of the hand.

Within this work, the focus is on detecting and reacting to such collisions in the final part of the approach of the hand towards the object. The assumption is that from a pregrasp pose somewhere near the object the hand approaches the final grasping pose on a straight trajectory. Figure 5.1 shows a schematic overview of the proposed grasp execution procedure. First, the robot moves the hand to a pregrasp pose relative to the object. Then the actual grasp pose is approached. During that critical last part of the approach, when the hand is near the object, the robot continuously checks for collisions. If a collision is detected, a corrective movement is executed and the grasp pose is adapted. When the grasp pose is reached, the fingers are closed.

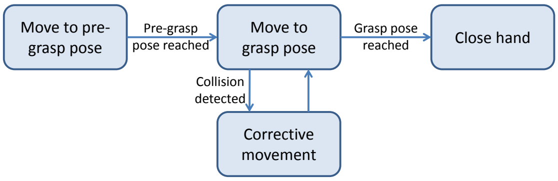


Figure 5.1: Schematic overview of grasping with collision detection and reactive corrections.

## 5.1 Visual collision detection

Collision detection is traditionally realized based on tactile or force sensors. They are however somewhat limited, as they require sufficient force feedback from the object to obtain significant sensor signals, which is not always the case when dealing with light objects that can be moved easily. In this thesis, a novel approach is proposed that complements these sensors by detecting collisions visually when they cause the object to move.

The main idea is to detect the motion of the object that is caused by the unintended collision with the hand of the robot. To this end, the hand is tracked in the camera images and the optical flow next to it is observed for irregularities. In contrast to [Bernabe et al., 2013], where the camera is static and the object stands freely on a table, this approach is designed to work with the cameras of the robot itself that move during the grasp execution, and makes no further assumptions about the scene except that the hand does not collide with any other parts of the environment besides the object that is being grasped.

### 5.1.1 Hand tracking

Visual tracking of the hand of the robot, and in particular the fingertips, is necessary for the collision detection as well as for estimating the part that caused the collision later. The tracking is realized based on a particle filter ([Isard and Blake, 1998]) which estimates the position, orientation and finger configuration of the hand. The end effector of ARMAR-III ([Asfour et al., 2006]) which is used here is a five-finger hand which is actuated pneumatically, i.e. with air pressure. It has two degrees of freedom in each finger and one in the palm. Tactile sensor pads are installed in each finger tip and the palm, and a force-torque sensor in the wrist[1].

A single-colored spherical marker is fixed at the wrist that allows a quick and robust visual localization of the hand position, but does not give any information about the orientation. Based on this marker and the fingertips, which have a distinctive blue color, the hand is localized. The particle filter estimates the position and orientation of the hand, and a reduced set of the finger DoF, which results in a 12-dimensional state space. Thus, each particle corresponds to a hypothetical hand configuration from that state space. The particles are initialized with the position from the localization of the spherical marker, the orientation from forward kinematics, and the measured finger joint angles.

[1] These sensors were used for reactive top-down grasping of unknown objects which was realized by Julian Schill and is described in [Schiebener et al., 2012]. They are ignored in the descriptions here, but for any practical application should definitely be used together with the visual collision detection.

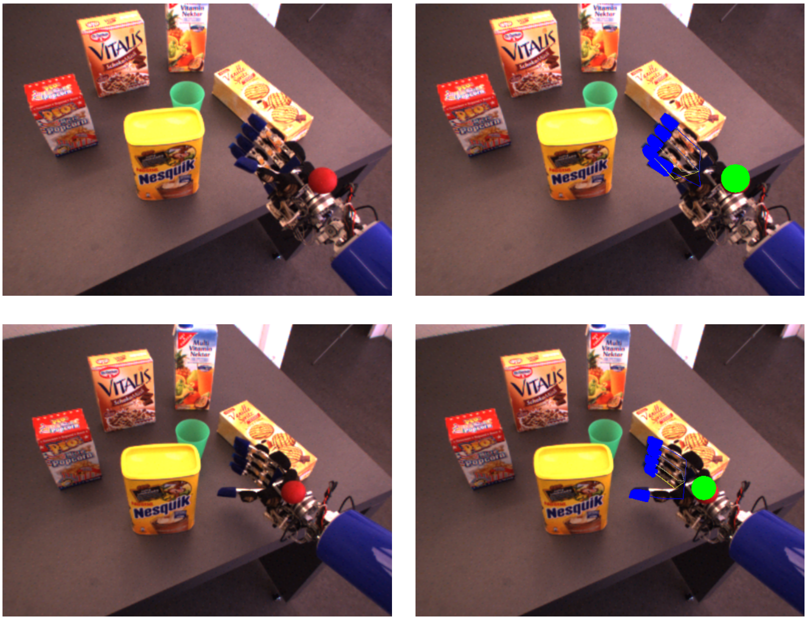


Figure 5.2: Localization of the hand of the robot, and in particular the fingertips, during grasping.

In each iteration of the particle filter algorithm, the particles are perturbed by adding random Gaussian noise, and then the plausibility of the hand configurations defined by the particles is evaluated based on the current camera images. In the next iteration, particles are redrawn with a probability proportional to their rating, the relative hand motion since the last iteration is applied to them, random noise is added and they are evaluated again. Robustness of the tracking is enforced by only allowing particle configurations that are within an empirically determined interval around the configuration obtained from forward kinematics and joint value sensor readings.

The key component of the particle filter is the rating function which estimates for each particle $s_i$ the conditional probability $p(z|s_i)$ that the input

$z$ (the camera images) was caused by the hand configuration defined by $s_i$. This probability is calculated based on five different cues, which are each determined in both of the stereo camera images. The first cue is $q_1(s_i) = 1/d$, where $d$ is the distance between the positions of the red spherical hand marker in the estimation of the particle and in the camera images.

The other four cues are based on the blue fingertips: They are projected into the images given the hand configuration of particle $s_i$. The cue $q_2(s_i)$ gives a rating proportional to the number of pixels in the area covered by the fingertips that have the correct color, and $q_3(s_i)$ rewards if a large part of the area has the correct color. This is done to avoid that the rating becomes too good if the projected fingertips are extremely small or large, which would be the case if only one of the two criteria was used. The cue $q_4(s_i)$ checks for intensity edges in the image that correspond to those of the projected fingertips, and $q_5(s_i)$ rewards similar edge directions.

The conditional probability of a given particle $s_i$ is then

$$p(z|s_i) \;=\; \vartheta\, e^{\sum_{j=1}^{5} \omega_j\, q_j(s_i)}$$

where $\vartheta$ is a scaling factor and the $\omega_j$ are weights for the different cues.

On each pair of stereo camera images, two iterations of simulated annealing (see [Kirkpatrick et al., 1983], [Deutscher et al., 2000]) are performed to enhance the precision of the final localization result, which is the average of all particles weighted with their probability.

### 5.1.2 Optical flow

Concurrently with the hand localization, the optical flow between the current camera image and the one taken at the last iteration of the collision check is calculated. The optical flow between two camera images estimates for each pixel how it moves from one image to the other. It is calculated using the algorithm proposed by [Farneback, 2003] which is implemented in OpenCV

([Bradski, 2000]). The idea of the algorithm is to approximate the neighborhood of each pixel by a quadratic function. If a quadratic function undergoes a translation, the displacement can be determined in closed form. By iteratively determining these translations first on a coarse and then on increasingly finer scales, larger displacements that exceed the direct neighborhood of the pixel can also be determined and refined. The algorithm provides a dense estimation of the optical flow between two images, although in larger monotone image regions it does not return any values. This is not a problem in this case, as only the image region around hand and object, which offers enough visual information for the algorithm, is relevant for the collision detection.

### 5.1.3 Collision detection

The idea for the visual collision detection is inspired by the interactive segmentation approach of [Metta and Fitzpatrick, 2003], where the moment of the collision between robot arm and object is recognized by the fact that an area of significant optical flow appears next to the hand. However, while in their case the camera is static, the cameras used here are mounted on the robot itself. Therefore, they are moving with the robot during the grasp execution, and consequently there is nonzero optical flow throughout the whole image. This poses the more general problem of discovering if an object next to the hand moves in a way that is inconsistent with the rest of the scene. It is also important to note that for the static part of the scene, its projected motion is not equal throughout the image but depends on the distance to the camera.

To overcome this problem, the pixels of the camera image are clustered by their optical flow values, i.e. their 2D motion vector from one frame to the next. This is done using a variant of k-means that automatically determines an appropriate number of clusters given a parameter that balances the number of clusters and their intraclass variance ([Pelleg and Moore, 2000]). The idea is to detect if there is a cluster of similar optical flow next to the hand which is different from the optical flow in the rest of the scene, which would indicate that an object is being moved by the hand.

The image area that is checked for such an outstanding cluster is determined by taking the hand position, adding a translation into the direction into which the hand is currently moving, and projecting this point into the image. A quadratic area around that point which has roughly the size of the object is then analyzed[2]. For each cluster of similar optical flow $c_i$, the number $n_i$ of pixels belonging to it in the whole image is determined, as well as the number $a_i$ of pixels belonging to it in the area in front of the hand. If for one of the clusters the ratio $\frac{a_i}{n_i}$ is more than 0.5, i.e. most of the pixels of the cluster occur inside that small area, this is a strong indication that this unique motion has been caused by an object that is being moved by the robot hand.

It is very probable (yet not certain) that the object will move in a similar way as the hand of the robot and parts of its arm. Therefore, the image area covered by the hand and the arm, which is determined based on the results of the hand tracking, is not taken into account when the values $n_i$ and $a_i$ are determined.

Figure 5.3 visualizes the optical flow, its clustering and the relevant image regions just before and during a collision. The left column shows the scene from the cameras of the robot immediately before and after the collision. The central column visualizes the optical flow, the right column the clusters of similar optical flow, where each cluster has been marked with a distinct color. The darker area is occupied by hand and arm and therefore ignored. The white box marks the area next to the hand where a possible collision is expected to occur. If there is a cluster of optical flow that exists mostly within this area but not outside of it, this observation indicates that the hand collided with an object and caused it to move.

Due to the restricted image area in which collisions are expected to happen, individual motion in the background (which most of the time moves as a

[2] The size of the object in the image can be estimated from its maximal extent and the distance to the camera.

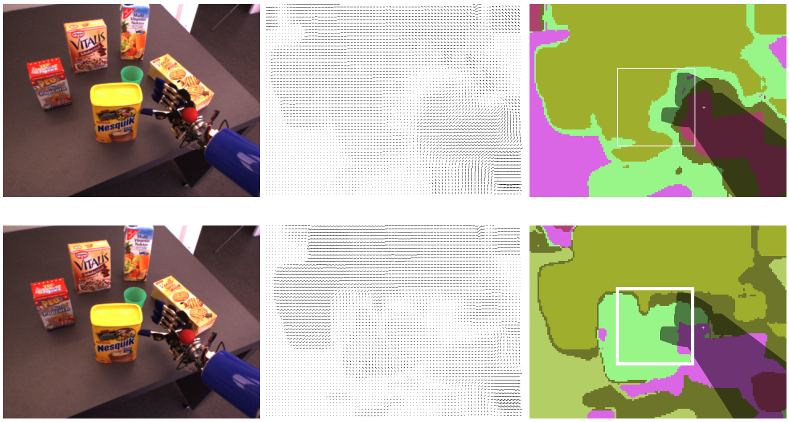


Figure 5.3: Visual collision detection immediately before and after the moment when the hand of the robot touches the object.

whole due to the motion of the camera) can theoretically cause false collision detections, but only when it occurs within the image area next to the hand in the direction into which it is moving as described above.

## 5.2 Corrective reaction

When the robot detects an unintended collision during the grasp execution, it should react in a way that allows to successfully complete the grasp. Optimally, the robot would have all relevant information about shape and pose of the hand and the object and could just re-plan a collision-free grasp trajectory. But in practice such comprehensive information is not available, otherwise the collision would not have occurred in the first place. Thus, robust heuristics are needed that can deal with incomplete and uncertain information and create a reaction that has a good chance of correcting the execution error that the robot committed.

### 5.2.1 Collision localization

One piece of information that is necessary for a reasonable corrective reaction is which part of the hand collided with the object. A random change of the hand pose may sometimes be successful, but as shown in the experiments in section 6.3 the informed reaction strategies are clearly superior to random modifications of the grasp.

The information which part of the hand touched the object is immediately available when using tactile sensors, but if the collision was detected visually it has to be determined in another way. Although this depends on the kind of hand that is used, it can be assumed that for a majority of grasping motions the fingers and in particular the fingertips are the primary causer of premature collisions. In the case of the conducted experiments, collisions are virtually always caused by the fingertips. Therefore, their positions are obtained from the hand localization, and the fingertip which is closest to the object is determined. This finger is considered to have caused the collision.

### 5.2.2 Reaction strategies

Different reaction strategies to correct the hand pose in the case of a premature collision were implemented and evaluated in this thesis. They are however limited to modifying the position and orientation of the whole hand, although in general there are cases in which it would be necessary to correct the configuration of individual fingers.

The general reaction scheme is the same for all proposed strategies: When a collision is detected, the hand retreats 2 cm into the direction that it came from with a straight motion to avoid disturbing the object any more. A corrective offset for the hand pose is calculated according to the respective strategy. The hand retreats another 2 cm during which half of the corrective offset is already applied, to make sure that the reaction has already taken effect before approaching the object again. The corrective offset is then permanently applied to the grasp pose definition. From that point on, the robot moves towards the intended grasping pose again.

If the robot collides with the object again, another corrective reaction takes places. Thus, the corrective offsets add up, and the robot repeatedly tries to grasp and corrects the hand pose as often as necessary until the grasp is successful. In practice it would probably make sense that if the grasp does not succeed after a certain number of corrections, a totally different grasp is planned.

Within this reaction scheme, the key to a helpful correction is to determine an appropriate corrective offset. As a baseline for the experiments, a strategy was implemented where the orientation of the hand is modified by a small random rotation. Such a purely exploratory approach would probably be the only possibility if there were no further information available about the collision, and there is a certain chance that the grasp will eventually succeed after one or more random modifications of the hand pose.

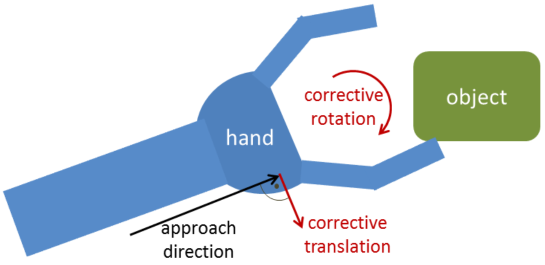


Figure 5.4: Depiction of the two basic reaction strategies: By translation or rotation, the colliding part is moved away from the object.

However, as the information which finger collided with the object is available, a more constructive correction offset can be determined. Two basic kinds of motion are useful to avoid another collision when approaching the object again: The first is to translate the hand into the direction of the finger that caused the collision, thus aligning the palm with the closest part of the object. The translation direction is from the hand center to the colliding finger, and perpendicular to the approach direction. The second possibility is to rotate the hand such that the colliding finger is turned away from the object. In the case of a hand where the thumb opposes the other fingers, or a simple gripper, the rotation would be around the axis that is perpendicular to the approach direction and to the direction from the thumb to the other fingers. This is depicted schematically in Figure 5.4. Of course, the two basic motions can also be blended. Both motions and a balanced combination of them were implemented in this thesis, and a comparative evaluation can be found in section 6.3.

In the case of the hand of ARMAR-III, the thumb opposes the four other fingers, therefore it is sufficient to distinguish whether the thumb or one of the other fingers collided with the object. These results can directly be transferred to simple grippers or to precision grasps with two fingers. For more general hand configurations the reaction strategies have to be adapted to the individual hand geometry following the above principles.

# 6 Evaluation

All evaluations were performed using the humanoid robot ARMAR-III ([Asfour et al., 2006]). It is equipped with stereo cameras that have a resolution of $640 \times 480$ pixels, and has two 7 DoF arms with anthropomorphic hands. All experiments were performed using the images from the cameras of the robot.

For the segmentation, a point cloud was generated using dense stereo matching ([Hirschmuller, 2008]), and the results of the segmentation are used when evaluating the object shape completion. The algorithms were also tested with an active Asus Xtion camera that immediately produces colored point clouds (RGBD data) which are a bit more accurate than those obtained from stereo matching, but no significant differences in the results of the segmentation or the shape completion were found.

The objects used for the evaluation are from the YCB object and model set ([Calli et al., 2015]). Out of the overall set of 75 objects, 18 cannot be perceived well enough by the available cameras because they are transparent, reflective or too small. Another 5 objects are deformable, and 4 have the same shape but different sizes. Therefore, the experiments were performed using the remaining 49 unique and well perceivable objects. In the following sections, the three contributions of the thesis are evaluated in the order as they were described before: First the object segmentation by physical interaction, then the object shape completion based on those segmentations, and finally the visual collision detection for reactive grasp correction.

## 6.1 Object segmentation by physical interaction

The proposed method for object segmentation by physical interaction is evaluated in challenging scenes where objects are stacked in random heaps, i.e. they may touch each other, lay on top of each other, and no prior knowledge about them or the background is available. Each object is pushed 5 times, resulting in segmentations in the form of the 3D points belonging to the verified object hypothesis.

### 6.1.1 Evaluation of the segmentation quality

To evaluate the quality of the segmentations, they are compared to a manually annotated ground truth, where in the camera images those pixels are marked that belong to the actual object. To allow a comparison, the points belonging to the object hypothesis are projected back into the images. There, an image region for the segmentation is generated by marking the projected points as white pixels in an otherwise black image, applying a Gaussian filter to enlarge them, and binarizing the resulting greyscale image with a threshold. The overlap of this segmented image region with the ground truth is then used to rate the quality of the segmentation. For the segmentation, only the confirmed hypothesis points are taken into account, but not the new candidates.

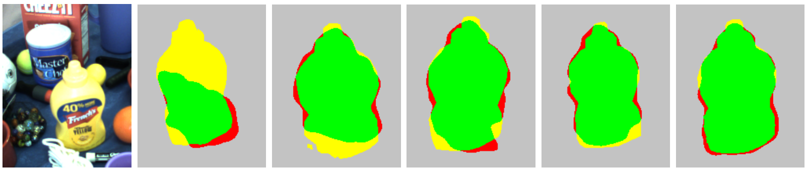


Figure 6.1: Segmentation of the yellow mustard bottle in the left picture compared to the ground truth after one to five pushes.

Figure 6.1 shows an example for this evaluation. The yellow mustard bottle in the left picture is segmented, and the results after one to five pushes compared to the ground truth. The green areas are true positives, i.e. correctly segmented regions, the red areas are false positives, and the yellow areas are false negatives, i.e. object parts that are not covered by the segmentation. As can be seen, after the first push, only a part of the object is covered by the segmentation. This is because the initial hypothesis only covered that part, so only in that area there can be confirmed hypothesis points. However, after the first push, new candidates are generated on the rest of the object, which are confirmed after the second push. After that, the object is mostly covered with confirmed points. Consequently, the segmentation after two pushes is already much better than after one. As the last three images show, the segmentation and the ground truth conform almost completely after three to five pushes.

The typically used values to quantify segmentation quality are *precision* and *recall*. *Precision* is the ratio of the number of true positives to the overall number of true and false positives, thus it indicates how reliable the algorithm is when claiming that a point belongs to the object. *Recall* is the ratio of the number of true positives to the number of true positives and false negatives, i.e. the overall number of points belonging to the object, so it indicates the total coverage ratio of the object by the segmentation.

In this evaluation, these values give a measure of the reliability of the points resulting from the segmentation, and of how completely the object is covered by them. The focus can to some degree be shifted between these two criteria. The confirmed 3D points of the hypothesis are projected into the image, where pixels are included in the segmentation depending on their distance to the projections of the hypothesis points. When only accepting exactly those pixels onto which the points are projected, a high precision is achieved, but many pixels in between are discarded which results in a low recall. In contrast, when the points are extended with a Gaussian filter and a low threshold is used for binarization, there will be no holes within the object region, but it may also stretch out further than the actual size of the object. This would result in a high recall, but a low precision.

Figure 6.2 shows the precision and recall values after one, two and three pushes depending on the binarization threshold. As expected, the precision increases with the threshold, while the recall decreases as the binarization gets more restrictive. An appropriate threshold should balance these two values depending on their relative importance. The $F_\beta$ score is a quality measure that incorporates precision and recall into one value, with $\beta$ as a weighting factor:

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

With $\beta = 1$, similar weight is given to both measures, and the $F_1$ score equals their harmonic mean:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

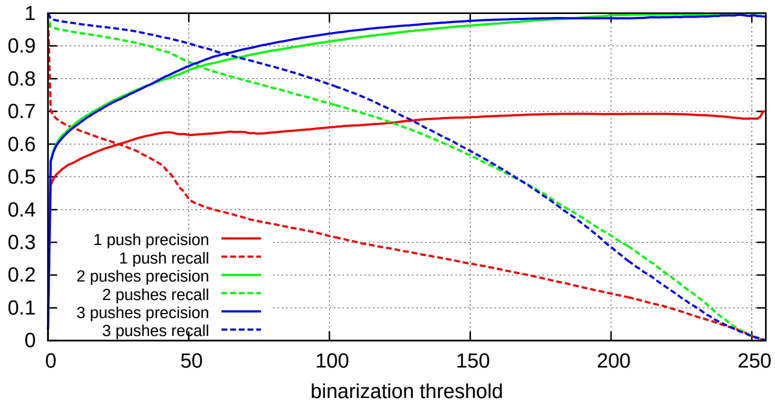


Figure 6.2: Precision and recall values of the segmentations after one, two and three pushes, depending on the binarization threshold.

Figure 6.3 shows the mean $F_1$ scores over the whole test set depending on the binarization threshold, and table 6.1 gives the $F_1$ scores together with their standard deviations when using the optimal threshold. It can be seen that after one push, the score is rather moderate and has a high variance. This is due to the fact that the initial object hypotheses may be anything between perfect and very bad, and only the correct points contained in them are confirmed after the first push. This means that the recall value here can be very low if the initial hypothesis only covered a small part of the object. However, new candidate points are added after the first push and can be verified after the second push. Consequently, the rating significantly improves after two pushes, and the small standard deviation shows that these good results are consistent over the whole test set (see also Figure 6.1).

There is another small improvement after the third push, but the fourth and fifth usually do not improve the results further. They may, however, unveil parts of the object that were not visible before and therefore be helpful for learning multiview object descriptors. The appearance of previously invisible object parts after each push is also the reason why the segmentations can

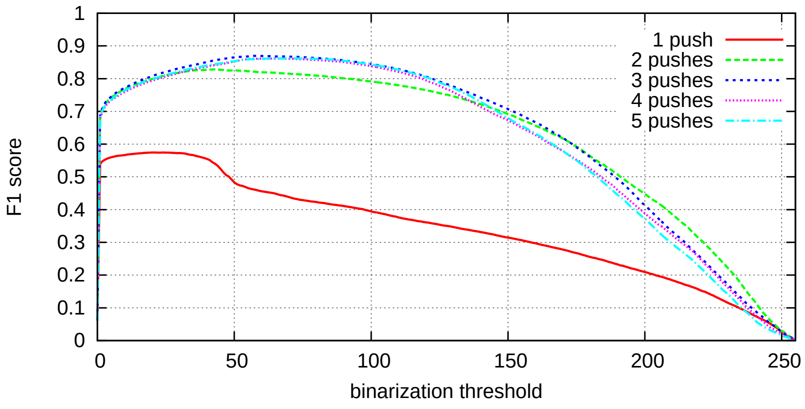


Figure 6.3: The mean $F_1$ scores of the interactive segmentation results after one to five pushes, depending on the binarization threshold.

Table 6.1: The mean $F_1$ scores for the interactive segmentation and the respective standard deviations over the complete test set, after one to five pushes.

| 1 push | 2 pushes | 3 pushes | 4 pushes | 5 pushes |
|---|---|---|---|---|
| 0.575 $\pm$0.279 | 0.829 $\pm$0.059 | 0.869 $\pm$0.045 | 0.861 $\pm$0.061 | 0.862 $\pm$0.085 |

usually not reach a perfect score, because there can be no confirmed points on newly appearing object parts.

It is also interesting to note that after the first push, when often only a part of the object is covered, the segmentation rating would profit from a rather low threshold that essentially just enlarges the segments. In contrast, the better segmentations resulting from two or more more pushes consistently have a higher optimal threshold that restricts the segmented region to the areas that are very close to confirmed points. As can be seen from Figure 6.2, the recall normally does not reach 1 for the reason mentioned above, but for strict thresholds the average precision is almost perfect, which means that there are extremely few and often no false positives at all in the segmentations.

### 6.1.2 Comparison to a state-of-the-art segmentation algorithm

In order to compare the results of the proposed approach to a segmentation algorithm that does not leverage physical interaction, the same evaluation was performed with the Locally Convex Connected Patches (LCCP) algorithm presented in [Stein et al., 2014] that is also described in section 2.1.5. It works on point clouds and segments them into parts that are connected by a convex path of small surface patches. Thus it is comparably well suited for the scenes considered here, as it does not rely exclusively on visual cues which are very ambiguous in complex arrangements of many objects that may or may not be textured, but also takes geometric information into account.

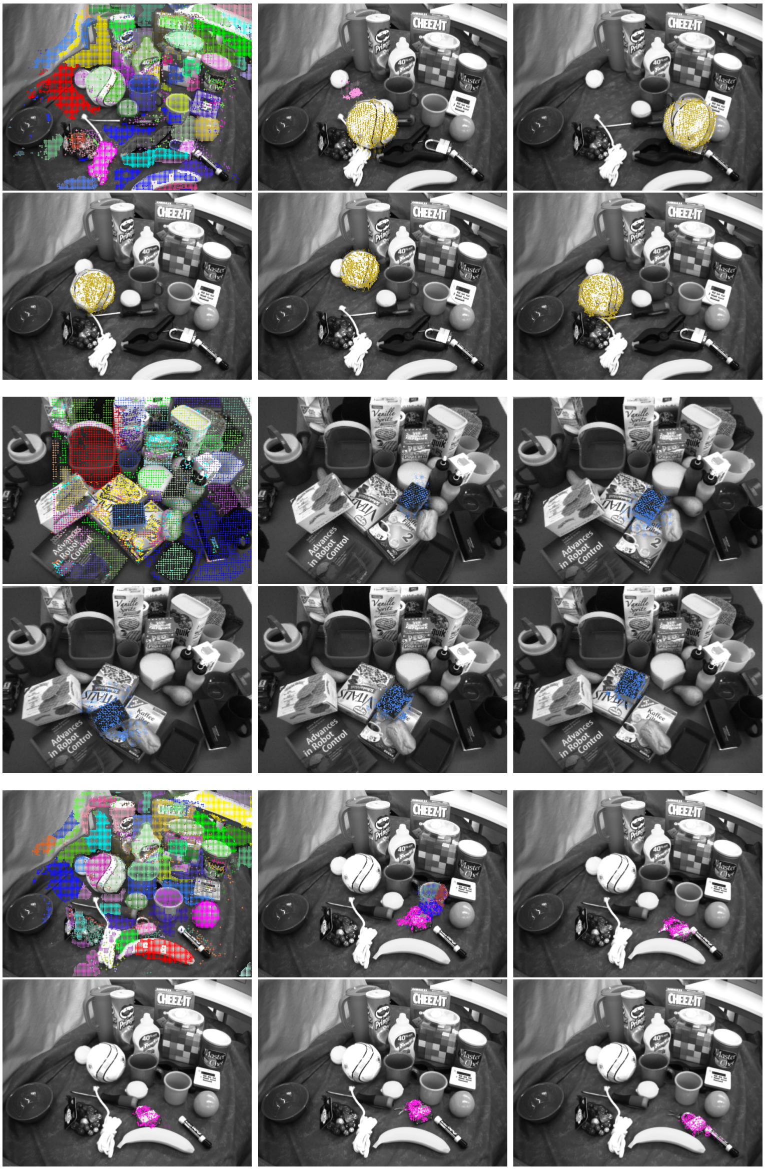


Figure 6.4: Examples of the results of the object segmentation by physical interaction: The initial hypotheses, and the confirmed hypotheses after one to five pushes.

Similar to the algorithm presented in this thesis, the resulting segments generated by LCCP are point clouds. As described above, the points are projected into the camera image, spread by using a Gaussian filter, and a segmentation mask is generated by binarization of the result. As the LCCP algorithm segments the scene into several convexly connected regions, the one with the best overlap with the ground truth is selected and used for the evaluation.

A member of the group that developed the algorithm set its parameters appropriately for the recorded scenes from the evaluation of the interactive segmentation. It turned out that one parameter needs to be set to different values, depending on the size of the object that is to be segmented. To accommodate for that, the algorithm is run with two different values for that parameter. In each scene, the segment with the best overlap with the ground truth is determined and used to calculate the rating.

The precision and recall are determined using different binarization thresholds for the Gaussian-filtered segmentation mask, and the $F_1$ score is calculated. In Figure 6.5 it is plotted depending on the binarization threshold, together with its standard deviation. For comparison, the $F_1$ score for the interactive segmentation presented in this thesis is also plotted, using the results after three pushes.

The mean score for LCCP at the optimal threshold value is $0.573$ with a standard deviation of $0.265$. This result is much worse than the $0.869(\pm 0.045)$ of the interactive segmentation, but the high variance also shows that the average has a very limited significance here. LCCP does not normally find mediocre segmentations - on the contrary, when the object can be segmented correctly, the precision as well as recall are usually very good. But there are also instances where LCCP completely fails to create a segment that corresponds to the object. This happens when e.g. it is positioned next to one or more other objects from which it can not be separated based on the applied geometric criteria. When, in contrast, the object is standing apart

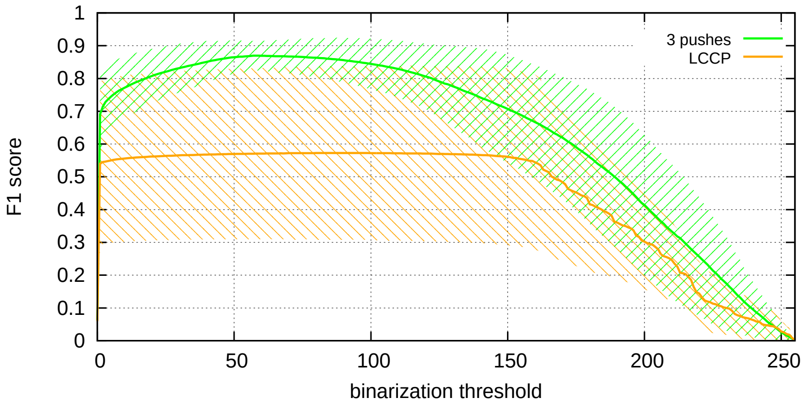


Figure 6.5: The $F_1$ scores of LCCP and the interactive segmentation after three pushes depending on the binarization threshold, together with their respective standard deviations.

from the other objects in the scene, the segmentation results are usually very good.

Over the whole test set, the interactive segmentation clearly outperforms LCCP. The latter tends to give slightly better results in the instances in which it is successful though: The objects are often covered more completely, i.e. the recall is a bit higher, as LCCP is not leaving out the previously unseen parts of the objects where the interactive segmentation can not have confirmed points yet. On the other side, it often fails when an object is in touch with other objects and does not significantly stick out of the scene. While the segments created by LCCP are rather speculative, the interactive segmentation returns object hypotheses that are verified and grounded by observation of the effects of physical interaction.

### 6.1.3 Evaluation of the object learning

To evaluate how suitable the obtained segmentations are for learning visual object descriptors, a simple object localizer was implemented. The segmentation results, i.e. the confirmed hypothesis point clouds, are saved. To localize an object, image regions with a similar color histogram are determined, and in them ICP is used to align the learned object point cloud with the scene. If the best match is below an error threshold, the localization is accepted.

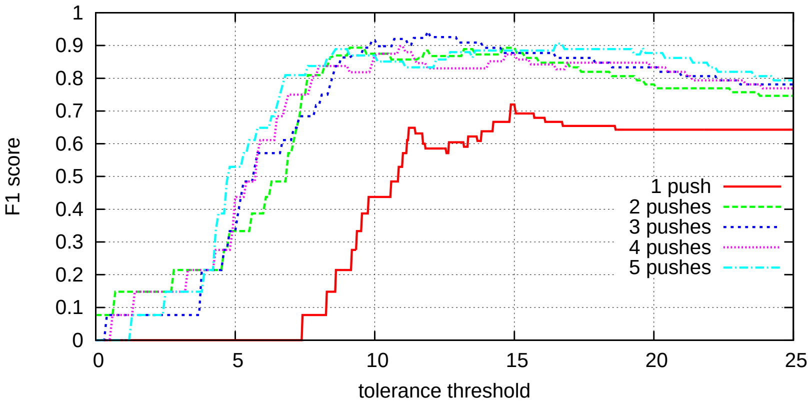


Figure 6.6: Recognition performance when using the segmentation results after one to five pushes.

For each object, recognition was attempted in five scenes that contain the object and five that do not. Figure 6.6 shows the $F_1$ score depending on the error threshold for accepting a match as a recognition. A smaller threshold tends to cause higher precision but lower recall than a high one. As can be seen, the results when using the segmentations after one push are rather bad, probably because at this point the hypotheses often cover only a part of the actual object. After two or more pushes, the segmentations usually cover the object well, and correspondingly the recognition performance increases significantly. Interestingly, there seems to be no improvement when using the descriptor after a higher number of pushes. Although the segmentation quality is a bit higher after three or more pushes than after two, it does not seem to make a big difference for the recognition performance. Only the range of thresholds where the best results are achieved seems to be a bit wider, which may indicate a higher reliability.

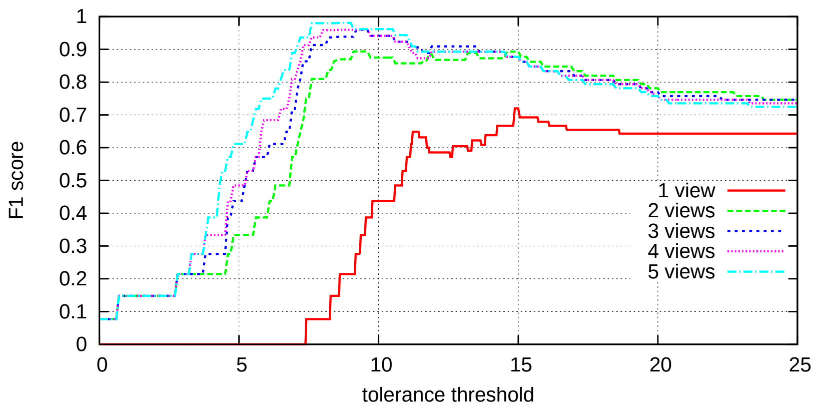


Figure 6.7: Recognition performance when using the best match of some or all of the learned descriptors per object.

Pushing the object several times allows to generate descriptors for multiple views of it. To make use of this, the recognition tries to localize an object using some or all of the available descriptors of it. The best match, i.e. the one with the smallest matching error, is used as the result. Figure 6.7 shows the recognition performance when using the first one, two, three, four, or all five learned views of the object. When using only the first one, there is obviously no difference, but when using the best result of several ones, the recognition performance increases.

In contrast to using single descriptors, it keeps improving with the number of used views. This was to be expected, as trying out multiple views increases the chance that one of them matches the object in the perceived scene well. When using four or five views, the recognition performance is clearly superior to that when using only a single view, and the $F_1$ score gets above 0.95 for a robust threshold range.

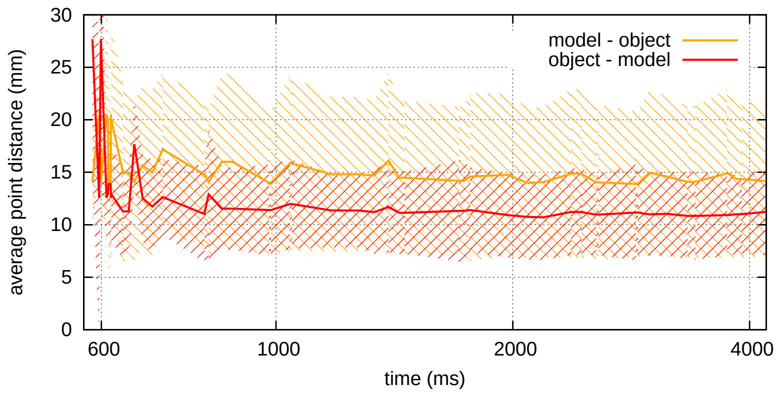
## 6.2 Object shape completion

The segmentation results obtained after 3 pushes were used to test and evaluate the approach for symmetry-based object shape completion. Based on the point cloud obtained from the segmentation, the shape completion is performed, and the results are compared to the ground truth, which is available in form of the scanned models of the YCB object and model set.

### 6.2.1 Quality measures

To quantify the similarity of the completed object shape with the model, they first need to be aligned. The model is loaded as a point cloud and matched with the completed point cloud using the Iterative Closest Point (ICP) algorithm ([Besl and McKay, 1992]). As this algorithm finds a local optimum, it is instantiated multiple times with different initial relative positionings: The two point clouds are placed so that their centers are in the same location, and one of them is rotated to all six possible orientations that can be reached by rotations of 180° around the three coordinate axes. The ICP result with the smallest mean error is selected, thus avoiding to use an alignment resulting from a local minimum far off the correct matching of the two point clouds. However, it is possible that the optimal alignment is not found. Therefore, the ratings for the completed point clouds may sometimes be worse than they would be with the optimal alignment, and thus underestimate the quality of the shape estimation.

One measure of shape similarity is the mean distance of each point in one point cloud to its nearest neighbor in the other point cloud. This score is calculated in both directions, i.e. the nearest neighbor for each point of the completed shape is found in the ground truth model, and vice versa, to make sure that the rating penalizes missing or added object parts. This gives a measure of the distance between estimated and real shape, however it is also affected by the pixel density and the precision of the camera. The results for this measure are shown in the top part of Figure 6.8.



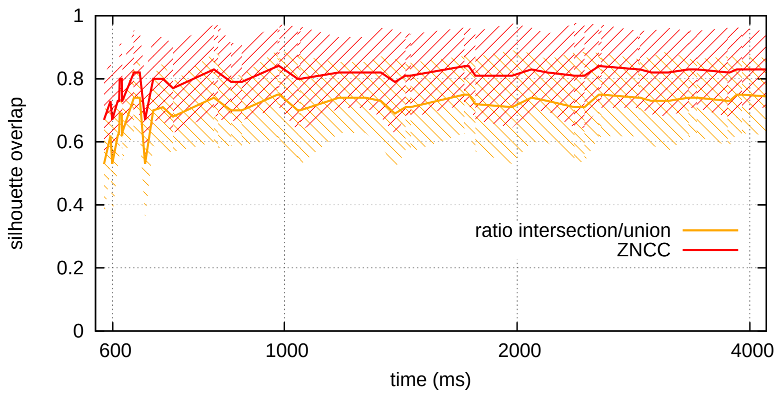


Figure 6.8: Top: Mean distances of points in the completed object shape to their nearest neighbors in the ground truth model, and vice versa. Bottom: Overlap of the silhouettes of completed model and ground truth.

Therefore, another measure is calculated that is intended to quantify the shape similarity in a way that gives a robust and dimensionless correlation value. The completed shape and the model are projected into an image, thus getting a binary silhouette. The projection is performed along all three coordinate axes. Two measures for the silhouette similarity are calculated: The zero-mean normalized cross correlation (ZNCC), and the ratio of the size of the region that is occupied by both to the size of the region occupied by at least one of the two silhouettes, i.e. $\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Both measures give a value in $[0,1]$ that is 1 for perfect correspondence and 0 for none. The average of these values for the three projections from different directions is used in the statistics. The results are shown in the bottom part of Figure 6.8.

### 6.2.2 Quality depending on the sampling density

The main tradeoff to be made in that approach is between the required computation time and the quality of the resulting completed object shape. With $|O|$ being the number of points in the original object segmentation, the expected effort is in $\Theta(|O|\log|O|)$ for the nearest neighbor search, and all other calculations are linear in $|O|$. The objects are segmented at a resolution of $320 \times 240$, and the original object point clouds contain between 100 and 900 points, on average 312. The resulting completed models contain on average 1251 points, which is about 4 times the initial size.

As downsampling the object point clouds would cause a significant loss of information, the most effective factor with which the computational effort can be influenced is the number of samples for the position and orientation of symmetry plane candidates. The number of potential supporting planes is set to be 10, but the number of positions and orientations of the symmetry plane candidates that are tested for each of the supporting surfaces is varied. Figure 6.8 shows the dependency of the shape quality on the invested computation time. The results indicate that the quality reaches saturation when more than one second is spent testing symmetry plane candidates. This corresponds to a sampling density of 20 different positions per supporting surface and 10 different orientations per position, i.e. 2000 different poses in total.

Table 6.2: Average computation times for the different steps of the shape completion algorithm (in ms).

| find supporting planes | generate and evaluate symmetry planes | add sides and bottom | total |
|---|---|---|---|
| 150.3 ±45.5 | 535.4 ±168.3 | 144.4 ±82.9 | 830.1 ±425.0 |

Table 6.2 shows the computation times for the different steps of the shape completion using this sampling density. The values are averages over the whole test set, together with their respective standard deviations. The computationally intensive steps are parallelized, and the experiments are run on a PC with an Intel Core i7 CPU with 8 virtual (4 physical) cores at 3.6 GHz. There is potential for some more speedup, but the required time is acceptable for the intended use as a prerequisite to grasp planning.

### 6.2.3 Discussion of the results

The results show that when sufficient computation time is invested, the shape completion creates estimations that robustly come close to the real object shapes. While it is hard to judge the absolute similarity between estimated and real shape given the average point distances, the silhouette-based measurements indicate that the concordance is high, but not perfect. The fact that the mean distance of a point in the ground truth model to its nearest neighbor in the completed shape is higher than that of a point in the completed shape to the model indicates that the estimations tend to lack parts of the real objects.

By visually inspecting the results, two reasons for this can be identified: One is that already in the segmentation the visible side of the object is not always covered completely. This error is propagated through the shape completion, and consequently the resulting shape estimation will be smaller than

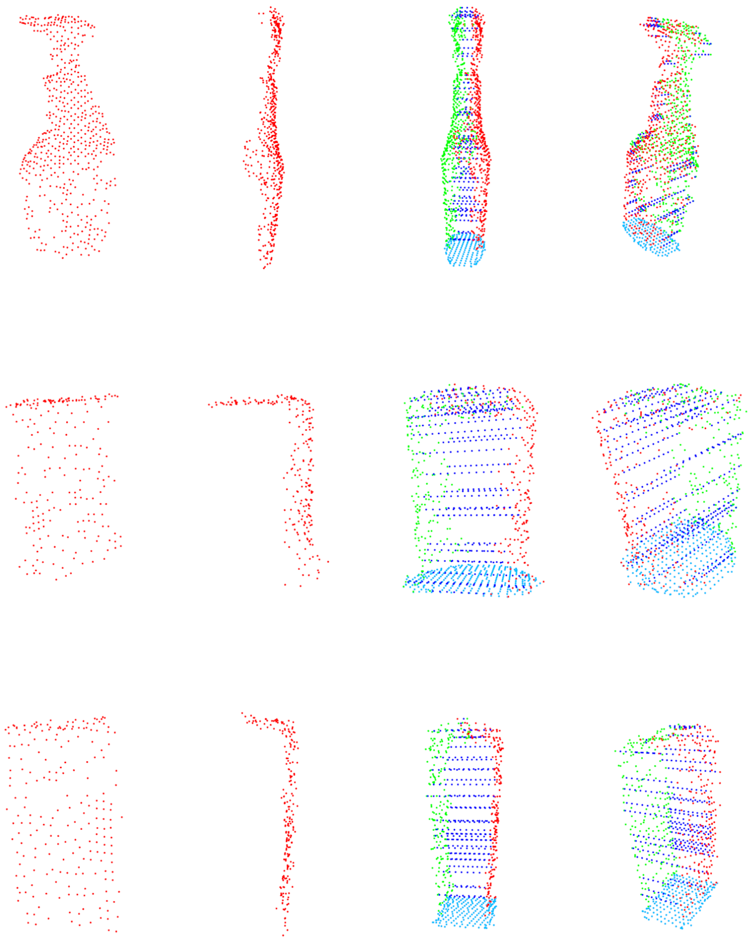


Figure 6.9: Original point clouds (seen from front and side) and completed shapes for a spray bottle, a cylindrical and a box-shaped object.

the actual object, missing the parts whose visible side was not included in the segmentation.

Another case where the approach tends to underestimate the volume is when dealing with rounded objects. Here, depth cameras as well as stereo cameras tend to give no values at the edges of the object where its curvature causes the surface to be nearly parallel to the view axis. In those cases, the shape completion method connects front and mirrored back sides by straight lines, which comes close to the real shape, but omits the bulge on the side. A good example to see this is the cylindrical coffee package in the second row of Figure 6.9. Here, the first and second column show the original point cloud, the third and fourth show the result. The original points are red, those generated by mirroring on the symmetry plane are green, the sides are dark blue and the bottom points are light blue.

Overall, the evaluation results as well as the subjective impression indicate that the shape resulting estimations approximate the real objects reliably. The mean error never exceeds 30mm, which is too much though to guarantee that grasps planned on these shapes can be executed successfully. The approach is robust to the objects being placed in nontrivial environments and on tilted supporting surfaces, which is an important improvement compared to previous work that only considers objects in isolation from their environment. However, the evaluation benefits from the fact that the objects in the test set were all at least roughly symmetrical. For a totally asymmetrical object, the shape completion error might be significantly larger.

In the evaluation, only the most problematic case was considered, in which the object has only been seen from a single point of view. In practice, a robot will usually be able to move its cameras and thus perceive more than only one side of the object, or when using the segmentation presented here fuse the different views obtained during pushing. Any additional points on the sides other than the front side of the object will make the estimation of the symmetry plane significantly easier and more stable. When different points of view are taken into account for the visibility criteria in the rating of the symmetry plane hypotheses, that rating becomes more informed and thus more robust. The potential volume of misestimations of the back side can then effectively be restricted to the remaining region that can not be seen from any of the points of view.

## 6.3 Visual collision detection for reactive grasp correction

The visual collision detection is first evaluated in isolation in order to determine its sensitivity, and then as part of a reactive correction strategy that is used during grasping with the robot ARMAR-III.

### 6.3.1 Sensitivity of the collision detection

The first test aims at quantifying the sensitivity of the visual collision detection in a very controlled setup. An object was moved manually over a fixed distance, while the robot hand is close to it, and therefore the detection should classify this motion as the result of a collision.

For each distance, the object was moved 25 times, and the respective detection rates are given in Table 6.3. Even if the object was shifted only by 2 mm, this was already detected most of the times, and when the translation was 5 mm or more the detection rate was clearly over 90%.

Table 6.3: Collision detection rate depending on the distance that the object has moved.

| 2 mm | 5 mm | 10 mm |
|---|---|---|
| 76 % | 92 % | 96 % |

Although these results suggest that a very small motion is sufficient to detect a collision between hand and object, in the execution on the real robot the objects were usually pushed about 1-4 cm. The two reasons for that are that the detection algorithm runs only at 2-3 frames per second on a standard

PC due to the relatively high computational intensity of hand localization, optical flow calculation, and clustering. More importantly, when a collision is detected, it takes some time until the hand of the robot actually stops moving forward.

Table 6.4: Collision detection rate depending on the angle between image plane and direction of movement (over a distance of 5 mm).

| 0° | 45° | 70° | 90° |
|---|---|---|---|
| 92 % | 96 % | 84 % | 72 % |

As the visual collision detection is based on optical flow, it is important to examine whether it is able to detect object motion that occurs in the direction perpendicular to the image plane. While motions within the two dimensions spanned by the image plane create significant optical flow, a motion straight away from the camera only causes the object to shrink in the image. Therefore, in the second experiment, the object is moved by 5 mm in different angles relative to the image plane.

The results can be found in Table 6.4. At 0° and 45°, there seems to be no difference. At an angle of around 70°, the detection rate drops slightly to 84%. At 90°, which is a motion exactly into the depth direction, the detection rate is still 72%. In practice however, 70° is the biggest angle to the image plane that can realistically be expected to occur. Usually, objects are placed in locations that are below the cameras of the robot, therefore a horizontal motion of the object e.g. on a table will never be exactly perpendicular to the image plane.

### 6.3.2 Performance of the different reaction strategies

Finally, the performance of the overall framework for reactive grasp correction based on visual collision detection was tested. In order to avoid distortion of the results by the effects of other algorithms, the grasping was performed on known, freestanding objects. The detection does however not require knowledge about the objects apart from their approximate size, and makes no assumptions about the background. It can therefore be applied in the same manner when grasping newly discovered and segmented objects.

Five different test objects were used, and for each of them a grasp was defined manually using a graphical tool that allows to position a model of the robot hand relative to a model of the object. These grasp definitions looked correct in their visualization, but their execution on the robot often failed due to imprecisions in perception and execution.

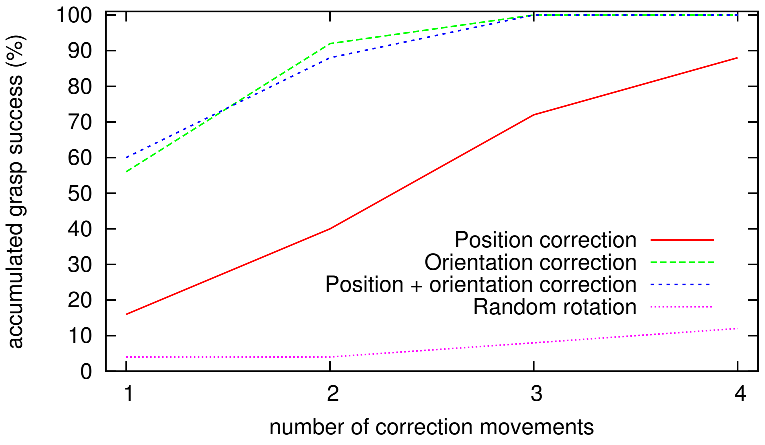


Figure 6.10: Percentage of successful grasps after a certain number of correction movements, depending on the applied strategy.

The aim of the experiment was to examine how well the different reaction strategies proposed in section 5.2.2 would be able to correct a failing grasp execution. Every reaction strategy was evaluated by placing each of the five objects at five different reachable poses in front of the robot. Grasp attempts that were immediately successful were ignored and repeated, so every strategy was tested with 25 grasps during which at least one collision with the object occurred. During grasp execution, visual servoing was employed to control the hand motion, i.e. both object and hand were localized visually to reduce the influence of imprecisions in the kinematic model of the robot.

Figure 6.10 depicts the results of these tests. It shows how many of the grasping attempts had succeeded after a given maximal number of corrections. The baseline strategy, which modifies the hand pose by a random rotation of 25° after a collision, performed rather badly, as was to be expected. In only one case a single correction movement lead to a successful grasp, and another attempt succeeded after three and four corrections respectively.

The proposed strategies that take into account which finger caused the collision performed significantly better than the uninformed random reaction. With the strategy which modifies the hand position by a translation of 25 mm towards the finger that caused the collision, the robot managed to successfully grasp the object after one correction in 16% of the cases, and in another 24% two corrective movements are sufficient. In 32% of the attempts, three corrections were necessary. The overall success rate after at most four corrections was 88%, which is already quite an achievement regarding the fact that without the reactions all those grasps would have failed. In two of the remaining three cases the object was still grasped after further correction movements, but in one case it was finally pushed out of reach of the robot.

However, the two reactive strategies which modify the orientation of the hand turned out to be even more effective. One modifies only the orientation by 25° to turn the colliding finger away from the object, the other changes the orientation by 15° and the position by 10 mm. With both strategies, the robot managed to grasp the object after one correction in about 60% of the attempts, and had an overall success rate of around 90% after one or two and 100% after at most three corrective movements.

The reason why the strategies that apply a rotational correction are more effective than the one that corrects only the position may lie in the way the arm movement during grasping is controlled on ARMAR-III: Visual servoing is used, but only the position of the hand is visually corrected, while its orientation is obtained from the forward kinematics of the robot. Thus, the orientation error during execution exceeds the position error. Furthermore, when localizing an object, its position is usually determined more reliably than its orientation. Therefore, it is entirely possible that on other robotic platforms, or when the visual servoing can also correct the hand orientation, the differences in the performance of the three strategies might be smaller.

# 7 Conclusion and future work

In this thesis, visual perception and the manipulation skills of humanoid robots were integrated to realize novel methods that enhance the ability of robots to correctly perceive new, unknown objects and interact with them. It was shown how, by purposeful manipulation actions, the extremely hard problem of visual segmentation in complex, uncontrolled scenes can be solved much more reliably than by passive perception alone. Object segmentation hypotheses are verified and corrected autonomously by the robot by physically interacting with them, thus making them much more meaningful and reliable compared to passive observations.

Having segmented an individual object from its environment enables the robot to study it further, e.g. by learning visual descriptors for recognition, or to plan more complex manipulation tasks like grasping. To support the latter, the complete shape of the object is estimated based on the visible parts. When performing a grasp, due to the uncertainties inherent in real data from the own perception of the robot, there is a risk of failure that can not be neglected. To detect problems when they occur, visual perception can give important information, in particular in situations where haptic sensing may be insufficient.

## 7.1 Contributions

To summarize, the main scientific contributions of this thesis are:

- **Object segmentation by physical interaction**
  The problem of segmenting individual objects out of a scene containing many unknown objects in a complex, cluttered arrangement without any prior knowledge about them is extremely hard to solve by passive observation only. The developed approach for solving this problem initially generates object hypotheses based on the perceived scenes using a set of different heuristics. By actively inducing motion to the underlying real objects, these hypotheses are verified. Depending on whether they move consistently or not, they are confirmed or discarded. The individual points that constitute the hypotheses are each verified or discarded, and the hypothesis is thus corrected. It is also extended by adding nearby points that appear to move with it, which are in turn verified after moving the object again. In contrast to other work in this field, the approach presented here makes only minimal assumptions about the objects and none about the background.

- **Object shape completion**
  The resulting object segmentations were used to estimate a complete 3D shape model, which is particularly important when the object is to be grasped. Here, a symmetry-based heuristic was developed that generates complete shapes, using also information from the scene near the object, and works in much less restricted scenes than comparable methods.

- **Visual collision detection for reactive grasp correction**
  To support grasping under uncertainty, which is particularly important when handling unknown objects, an extension for grasping with reactive corrections was developed. The proposed method for visual collision detection complements classical haptic sensing in the situations where it may fail - when dealing with light objects that are displaced easily without giving significant force feedback. This motion is detected visually, and used in a reactive grasping framework.

Notably, this approach works with the internal cameras of the robot that are moving themselves during grasping, and does not require prior knowledge about the object or the background.

All developed methods were evaluated with real data, and their practical applicability was demonstrated on the humanoid robot ARMAR-III[1]. The evaluation of the segmentation with objects from the YCB object set showed that it reliably segments objects even in extremely cluttered scenes, and clearly outperforms a non-interactive state-of-the-art segmentation algorithm. The object shape completion was shown to generate models that robustly come close to the real object shape even when the objects were only perceived from one point of view. The high sensitivity of the visual collision detection was ascertained experimentally, and it was evaluated how effectively different reaction strategies using it can successfully correct grasp attempts that would otherwise have failed.

The work on object segmentation by physical interaction has been published in [Schiebener et al., 2011], [Ude et al., 2012], [Schiebener et al., 2012], [Schiebener et al., 2013] and [Schiebener et al., 2014a], the work on object shape completion in [Schiebener et al., 2016], and the work on visual collision detection and reactive grasping in [Schiebener et al., 2012], [Schiebener et al., 2014b] and [Paikan et al., 2015].

## 7.2 Future work

The presented approaches offer several links to other fields of research related to visual perception, learning and grasping on humanoid robots. The results of the interactive segmentation are a perfect starting point for autonomous learning of different properties of the discovered new objects. The learning

[1] Videos of the execution of the object segmentation by physical interaction and the reactive grasping with visual collision detection can be found on
www.youtube.com/watch?v=Rul21hZFoKE
www.youtube.com/watch?v=MkNIFWth5D4

of visual multiview descriptors for recognition has already been investigated during this thesis, but could and should be explored much more in depth.

Another idea is to have the robot autonomously study the physical properties of the objects like e.g. their mass distribution. Some preliminary results in that direction were already achieved in a master's thesis ([Puch, 2015]). More generally, the affordances of the object, i.e. the ways how it can be manipulated or used, could be explored by the robot. The more complex manipulation actions necessary for this would certainly profit from a close integration of visual and haptic sensing, which would allow a multimodal exploration of the object.

Grasp planning on the segmented objects is an extremely interesting and relevant topic. Although the completed shapes are sufficient to allow grasp planners to find possible grasp configurations, it might be helpful to adapt those planners in order to deal with shape uncertainty, which is inherent to all models built from real sensor data, but is particularly high in the parts where the shape was not perceived but just estimated.

Reactive grasping and reactive grasp corrections are indispensable for enabling robots to reliably perform grasping or other manipulation tasks in uncontrolled real-world environments. The developed concept for reactive grasping with visual collision detection was also successfully applied in transferring grasp knowledge between different robotic platforms. It allowed the iCub robot to adapt the grasping skill and grasp configurations taken from ARMAR-III during execution, thus learning quickly how to grasp objects for which ARMAR-III already had the necessary knowledge. This knowledge was initially not appropriate for iCub due to its different size and kinematic structure, but by detecting collisions during grasping and performing corrective reactions until success, the grasp configurations could be adapted ([Paikan et al., 2015]).

Another interesting topic is the integration of different sensing modalities for contact detection, like force sensors, tactile pads and proprioception with the visual collision detection. In the current implementation, a contact is detected if any of these sensors perceives it. It might be possible to improve the reliability of the detection by integrating these sources in a probabilistically sound manner.

# List of Figures

# List of Tables

# Bibliography

R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1597–1604, 2009.

J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

S. Alpert, M. Galun, A. Brandt, and R. Basri. Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):315–327, 2012.

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

T. Asfour, N. Vahrenkamp, D. Schiebener, M. Do, M. Przybylski, K. Welke, J. Schill, and R. Dillmann. ARMAR-III: Advances in humanoid grasping and manipulation. *Journal of the Robotics Society of Japan*, 31(4):341–346, 2013.

R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

D. Beale, P. Iravani, and P. Hall. Probabilistic models for robot-based object segmentation. *Robotics and Autonomous Systems*, 59(12):1080–1089, 2011.

C. Beder and W. Förstner. Direct solutions for computing cylinders from minimal sets of 3D points. In *Computer Vision - ECCV 2006*, volume 3951, pages 135–146, 2006.

N. Bergström, C. H. Ek, M. Björkman, and D. Kragic. Scene understanding through autonomous interactive perception. In *Computer Vision Systems*, pages 153–162. Springer, 2011.

J. Bernabe, J. Felip, A. P. del Pobil, and A. Morales. Contact localization through robot and object motion from point clouds. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

C. Bersch, D. Pangercic, S. Osentoski, K. Hausman, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz. Segmentation of cluttered scenes through interactive perception. In *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, pages 9–13, 2012.

P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.121791.

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

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

G. Bradski. The OpenCV library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, pages 25–36. Springer, 2004.

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

L. Chang, J. Smith, and D. Fox. Interactive singulation of objects from a pile. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3875–3882, 2012.

T. Chaperon and F. Goulette. Extracting cylinders in full 3D data using a random sampling method and the Gaussian image. In *Proc. Vision Modeling and Visualization Conference*, 2001.

D. Chen, Z. Liu, and G. Wichert. Grasping on the move: A generic arm-base coordinated grasping pipeline for mobile manipulation. In *European Conference on Mobile Robots (ECMR)*, pages 349–354, 2013.

Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2724–2729, 1991.

C. I. Connolly. The determination of next best views. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 432–435, 1985.

H. Dang and P. Allen. Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 36(4):309–330, 2014.

H. Dang, J. Weisz, and P. Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5917–5922, May 2011. doi: 10.1109/ICRA.2011.5979679.

J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 126–133, 2000.

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

G. Farneback. Two-frame motion estimation based on polynomial expansion. In J. Bigun and T. Gustavsson, editors, *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. Springer Berlin Heidelberg, 2003.

J. Feldman. What is a visual object? *Trends in Cognitive Sciences*, 7(6), 2003.

J. Felip, J. Bernabe, and A. Morales. Contact-based blind grasping of unknown objects. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 396–401, 2012.

D. Fischinger and M. Vincze. Empty the basket - a shape based learning approach for grasping piles of unknown objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2051–2057, 2012.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, volume 24, 1981.

P. Fitzpatrick, A. Needham, L. Natale, and G. Metta. Shared challenges in object perception for robots and infants. *Infant and Child Development*, 17:7–24, 2008.

P. Forssen. Maximally stable colour regions for recognition and matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

S. Frintrop, E. Rome, and H. Christensen. Computational visual attention systems and their cognitive foundations: A survey. In *ACM Transactions on Applied Perception 7*, 2010.

K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *IEEE International Conference on Computer Vision (ICCV)*, pages 716–723, 2003.

R. A. Grupen, T. C. Henderson, and C. D. Hansen. Apparent symmetries in range data. *Pattern recognition letters*, 7(2):107–111, 1988.

Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11):2270–2287, 2014.

M. Gupta and G. Sukhatme. Using manipulation primitives for brick sorting in clutter. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3883–3889, 2012.

M. Gupta, J. Muller, and G. Sukhatme. Using manipulation primitives for object sorting in cluttered environments. *IEEE Transactions on Automation Science and Engineering*, 12(2):608–614, April 2015. ISSN 1545-5955.

K. Hausman, F. Balint-Benczedi, D. Pangercic, Z. Marton, R. Ueda, K. Okada, and M. Beetz. Tracking-based interactive segmentation of textureless objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

T. Hermans, J. Rehg, and A. Bobick. Guided pushing for object singulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012.

H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb. 2008.

B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal Optical Society America A*, 4, 1987.

K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng. Reactive grasping using optical proximity sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2098–2105, 2009.

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

A. Huamán Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor. Exploiting symmetries and extrusions for grasping household objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

A. E. Johnson and S. B. Kang. Registration and integration of textured 3d data. *Image and Vision Computing*, 17(2):135 – 147, 1999.

D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz. Clearing a pile of unknown objects using interactive perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 154–161, 2013.

J. Kenney, T. Buckley, and O. Brock. Interactive segmentation for manipulation in unstructured environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

G. Kootstra, N. Bergstrom, and D. Kragic. Fast and automatic detection and segmentation of unknown objects. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 442–447, 2010.

G. Kootstra, M. Popovic, J. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Krüger. Enabling grasping of unknown objects through a synergistic use of edge and surface information. *International Journal of Robotics Research*, 31(10):1190–1213, 2012.

D. Kraft, N. Pugeault, E. Baseski, M. Popovic, D. Kragic, S. Kalkan, F. Wörgötter, and N. Krüger. Birth of the object: Detection of objectness and extraction of object shape through object-action complexes. *International Journal of Humanoid Robotics*, 5(2):247–265, 2008.

O. Kroemer, H. Ben Amor, M. Ewerton, and J. Peters. Point cloud completion using extrusions. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 680–685, 2012.

I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

W. H. Li and L. Kleeman. Interactive learning of visually symmetric objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4751–4756, 2009.

W. H. Li and L. Kleeman. Segmentation and modeling of visually symmetric objects by robot actions. *International Journal of Robotics Research*, 30 (9), 2011.

J. López-Coronado, J. L. Pedreño-Molina, A. Guerrero-González, and P. Gorce. A neural model for visual-tactile-motor integration in robotic reaching and grasping tasks. *Robotica*, 20(01):23–31, 2002.

D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. Computer Vision*, Corfu, Greece, 1999.

N. Lyubova, S. Ivaldi, and D. Filliat. From passive to interactive object learning and recognition through self-identification on a humanoid robot. *Autonomous Robots*, pages 1–25, 2015.

D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

Z. C. Marton, R. B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy datasets. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

Z.-C. Marton, L. Goron, R. B. Rusu, and M. Beetz. Reconstruction and verification of 3d object models for grasping. In *Robotics Research*, pages 315–328. Springer, 2011.

J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference*, 2002.

G. Metta and P. Fitzpatrick. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 361(1811), 2003.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005.

A. K. Mishra, Y. Aloimonos, L.-F. Cheong, A. Kassim, et al. Active visual segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):639–653, 2012.

N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (TOG)*, 25(3): 560–568, 2006.

J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

R. Ozawa, J.-H. Bae, and S. Arimoto. Multi-fingered dynamic blind grasping with tactile feedback in a horizontal plane. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1006–1011, 2006.

A. Paikan, D. Schiebener, M. Wächter, T. Asfour, G. Metta, and L. Natale. Transferring object grasping knowledge and skill across different robotic platforms. In *International Conference on Advanced Robotics (ICAR)*, pages 498–503, 2015.

P. Palmer and I. Rock. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic Bulletin and Review*, 1:29–55, 1994.

S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *European Conference on Computer Vision (ECCV)*, pages 460–473. Springer, 2008.

D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proc. 17th Int. Conf. Machine Learning*, pages 727–734, San Francisco, CA, 2000.

R. Platt. Learning grasp strategies composed of contact relative motions. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 49–56, 2007.

S. Puch. Exploration unbekannter Objekte durch physische Interaktion. Master's thesis, Karlsruhe Institute of Technology (KIT), October 2015.

A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4791–4796, 2012.

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

D. Schiebener, A. Ude, J. Morimoto, T. Asfour, and R. Dillmann. Segmentation and learning of unknown objects through physical interaction. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Bled, Slovenia, 2011.

D. Schiebener, J. Schill, and T. Asfour. Discovery, segmentation and reactive grasping of unknown objects. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 71–77, 2012.

D. Schiebener, J. Morimoto, T. Asfour, and A. Ude. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior*, 21(5):328–345, 2013.

D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4959–4966, 2014a.

D. Schiebener, N. Vahrenkamp, and T. Asfour. Visual collision detection for corrective movements during grasping on a humanoid robot. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 105–111, 2014b.

D. Schiebener, A. Schmidt, N. Vahrenkamp, and T. Asfour. Heuristic 3D object shape completion based on symmetry and scene context. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

A. Schmidt. Heuristische Vervollständigung der 3D-Form von nur teilweise bekannten Objekten. Bachelor thesis, Karlsruhe Institute of Technology (KIT), May 2015.

R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, volume 28, pages 503–512, 2009.

J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990.

S. Stein, M. Schoeler, J. Papon, and F. Wörgötter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2014.

S. Thrun and B. Wegbreit. Shape from symmetry. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1824–1831, 2005.

C. J. Tsikos and R. K. Bajcsy. Segmentation via manipulation. *IEEE Transactions on Robotics and Automation*, 7(3):306–319, 1991.

A. Ude, D. Schiebener, N. Sugimoto, and J. Morimoto. Integrating surface-based hypotheses and manipulation for autonomous segmentation and learning of object representations. In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Pauls, Minnesota, 2012.

N. Vahrenkamp, T. Asfour, and R. Dillmann. Efficient inverse kinematics computation based on reachability analysis. *International Journal of Humanoid Robotics*, 9(4), 2012a.

N. Vahrenkamp, T. Asfour, and R. Dillmann. Simultaneous grasp and motion planning. *IEEE Robotics and Automation Magazine*, 19(2):43–57, 2012b.

H. van Hoof, O. Kroemer, H. Ben Amor, and J. Peters. Maximally informative interaction learning for scene exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5152–5158, 2012.

H. van Hoof, O. Kroemer, and J. Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209, 2014.

K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann. Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010.

K. Welke, D. Schiebener, T. Asfour, and R. Dillmann. Gaze selection during manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

# Publications

## *Journals*

**D. Schiebener, J. Morimoto, T. Asfour and A. Ude**
*Integrating visual perception and manipulation for autonomous learning of object representations*
Adaptive Behavior, Vol. 21, No. 5, pp. 328 - 345, 2013

**T. Asfour, N. Vahrenkamp, D. Schiebener, M. Do, M. Przybylski, K. Welke, J. Schill and R. Dillmann**
*ARMAR-III: Advances in Humanoid Grasping and Manipulation*
Journal of the Robotics Society of Japan, Vol. 31, No. 4, pp. 341 - 346, 2013

## *Conferences*

**D. Schiebener, A. Schmidt , N. Vahrenkamp and T. Asfour**
*Heuristic 3D Object Shape Completion based on Symmetry and Scene Context*
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016

**S. Ottenhaus, M. Miller, D. Schiebener, N. Vahrenkamp and T. Asfour**
*Local Implicit Surface Estimation for Haptic Exploration*
IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2016

**N. Vahrenkamp, H. Arnst, M. Wächter, D. Schiebener, P. Sotiropoulos, M. Kowalik and T. Asfour**
*Workspace Analysis for Planning Human-Robot Interaction Tasks*
IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2016

**A. Paikan, D. Schiebener, M. Wächter, T. Asfour, G. Metta and L. Natale**
*Transferring Object Grasping Knowledge and Skill Across Different Robotic Platforms*
International Conference on Advanced Robotics (ICAR), 2015

**D. Schiebener, A. Ude and T. Asfour**
*Physical Interaction for Segmentation of Unknown Textured and Non-textured Rigid Objects*
IEEE International Conference on Robotics and Automation (ICRA), 2014

**D. Schiebener, N. Vahrenkamp and T. Asfour**
*Visual Collision Detection for Corrective Movements during Grasping on a Humanoid Robot*
IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2014

**K. Welke, D. Schiebener, T. Asfour and R. Dillmann**
*Gaze Selection during Manipulation Tasks*
IEEE International Conference on Robotics and Automation (ICRA), 2013

**A. Ude, D. Schiebener, N. Sugimoto and J. Morimoto**
*Integrating Surface-based Hypotheses and Manipulation for Autonomous Segmentation and Learning of Object Representations*
IEEE International Conference on Robotics and Automation (ICRA), 2012

**D. Schiebener, J. Schill and T. Asfour**
*Discovery, Segmentation and Reactive Grasping of Unknown Objects*
IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2012

**D. Schiebener, A. Ude, J. Morimoto, T. Asfour and R. Dillmann**
*Segmentation and Learning of Unknown Objects through Physical Interaction*
IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2011

**K. Welke, J. Issac, D. Schiebener, T. Asfour and R. Dillmann**
*Autonomous Acquisition of Visual Multi-View Object Representations for Object Recognition on a Humanoid Robot*
IEEE International Conference on Robotics and Automation (ICRA), 2010