ABSTRACT

Title of dissertation: SEEING BEHIND THE SCENE:
USING SYMMETRY TO REASON ABOUT
OBJECTS IN CLUTTERED ENVIRONMENTS

Aleksandrs Ecins, Doctor of Philosophy, 2017

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science

Rapid advances in robotic technology are bringing robots out of the controlled environments of assembly lines and factories into the unstructured and unpredictable real-life workspaces of human beings. One of the prerequisites for operating in such environments is the ability to grasp previously unobserved physical objects. To achieve this individual objects have to be delineated from the rest of the environment and their shape properties estimated from incomplete observations of the scene. This remains a challenging task due to the lack of prior information about the shape and pose of the object as well as occlusions in cluttered scenes. We attempt to solve this problem by utilizing the powerful concept of symmetry. Symmetry is ubiquitous in both natural and man-made environments. It reveals redundancies in the structure of the world around us and thus can be used in a variety of visual processing tasks.

In this thesis we propose a complete pipeline for detecting symmetric objects and recovering their rotational and reflectional symmetries from 3D reconstructions

of natural scenes. We begin by obtaining a multiple-view 3D pointcloud of the scene using the Kinect Fusion algorithm. Additionally a voxelized occupancy map of the scene is extracted in order to reason about occlusions. We propose two classes of algorithms for symmetry detection: curve based and surface based. Curve based algorithm relies on extracting and matching surface normal edge curves in the pointcloud. A more efficient surface based algorithm works by fitting symmetry axes/planes to the geometry of the smooth surfaces of the scene. In order to segment the objects we introduce a segmentation approach that uses symmetry as a global grouping principle. It extracts points of the scene that are consistent with a given symmetry candidate. To evaluate the performance of our symmetry detection and segmentation algorithms we construct a dataset of cluttered tabletop scenes with ground truth object masks and corresponding symmetries. Finally we demonstrate how our pipeline can be used by a mobile robot to detect and grasp objects in a house scenario.

SEEING BEHIND THE SCENE: USING SYMMETRY TO REASON ABOUT OBJECTS IN CLUTTERED ENVIRONMENTS

by

Aleksandrs Ecins

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Professor Nikhil Chopra, Dean's Representative
Professor Ramani Duraiswami
Professor Matthias Zwicker

Acknowledgments

I am grateful to my advisors Prof. Yiannis Aloimonos and Dr. Cornelia Fermüller for introducing me to the fascinating world of machine perception. You have taught me to think about computer vision problems in terms of their utility for active agents, that move and act in the world. I clearly remember the day when Yiannis came to the lab one Friday afternoon, picked a mug from my table, and pointed out that he had no trouble placing his fingers on the 'back' of the mug, even though he didn't see it. "You can use symmetry to do it. Look around you, everything is symmetric!". This was the defining moment of my PhD, and from that day onward I knew exactly what was going to be the topic of my thesis. While Yiannis inspired, Cornelia was there to help nurture these ideas and translate them into working code and papers, to be shared with the rest of the world. I am glad that I got a chance to work with both of you.

I would like to thank my colleagues that shared the lab with me over the 7 years of my PhD: Austin Myers, Yezhou Yang, Ching Lik Teo, Francisco Barranco, Kostas Zampogiannis, Kanishka Ganguly, Greogory Kramida, Chetan Mysore, Nitin Sanket and Snehesh Shreshtra. I will remember the time we spent in and outside of the lab, talking science and having fun.

I am happy that two of my fellow students, whom I met right after arriving to Maryland, remained my closest friends throughout the rest of my PhD. We shared a similar kind of craziness with Kotaro Hara, who is officially the best roommate ever. And I enjoyed chatting about our graduate life during long walks with Aishwarya

Thiruvengadam.

My parents, Elena and Sergejs, have always believed in me and supported me in my endeavors. I am thankful to you for giving me the encouragement and freedom to pursue my interests.

Finally, I want to thank my dear Sachoques for keeping me going throughout all these years. You were there for me, during the moments of doubt, during the moments of joy, and I can't imagine completing this journey without you.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Robots of The Future

Since ancient times humanity has dreamt of artificial contraptions, mechanical or otherwise, that would serve and help us with our daily lives. We live in the age where technology might be finally catching up to our dreams. The 20th century has seen great progress in automation due to the introduction of small and efficient electric motors and digital electronics. Industrial robots can be programmed to execute predefined motions with extreme precision and repeatability. However, due to their limited perceptual capabilities, they can not adapt to dynamic changes in the environment. Specialist programming is required to "reprogram" them for different narrowly defined tasks. As a result industrial robots are only used in large-scale manufacturing settings where operating space can be tailored to the specific needs of the robot. The robots of the future will instead be encountered in the "real-world" workspaces of our houses, shops, schools and hospitals, where they will interact and work together with humans. This capability requires three major components: 1) **Navigation** - robots need to move around in space without running into static or dynamic obstacles such as sofas and humans; 2) **Human robot interaction** - robots will need to understand human actions and intent, and at the same time be

(a) **(b)**

Figure 1.1: Robot operating environments. (a) Today's robot operate in environments where shapes and positions of all of the manipulable objects and obstacles are known in advance. (b) In the future robots will work in messy and unpredictable human environments.

able to communicate its own state and intent back to the humans in a natural way; 3) **Manipulation** - to do useful work, robots need the ability to alter the world through physical contact. While progress in all three of these areas is required to build a robotic butler, this thesis focuses on the latter one. Specifically we are addressing the problem of developing perceptual capabilities required for robotic manipulation of everyday objects.

1.2 Current Approaches to Perception For Manipulation

Like any perceptual process our pipeline begins with sensing the environment. Usual sensor choices are 2D cameras capturing RGB information or 3D range sensors that capture depth images or pointclouds. Once the data is captured, individual objects must be delineated from the rest of the environment. Next object shape properties are estimated in order to choose appropriate contact points between the

gripper and the object and to plan a suitable grasp strategy. This has to be done in typical messy real life scenarios where the objects are not placed in an orderly manner but instead are piled on top of each other. This remains an extremely challenging problem. There are countless variations in possible object shapes and appearances. This problem is compounded by the fact that cluttered scenes contain heavy occlusions, so only incomplete object observations are available in the data captured by the sensors. As a result it is difficult to segment the scene (i.e. to figure out which parts of the scene belong to different objects) and reconstruct the missing shape information for the objects in the scene.

Current approaches to this problem can be largely divided into two classes: model-based and shape primitive based. Model-based methods use a priori knowledge about object appearance in order to recognize them in the scene. During the training stage a full 3D CAD object model is rendered from multiple views and visual/shape descriptors are computed and associated to each of the views. Object 6DOF pose in the scene is then recovered by extracting the descriptors from the input data and finding the best matching set of the training descriptors. This is often followed by a fine-tuning step that refines object pose by fitting the 3D model to the observed data using the Iterative Closest Points algorithm [2]. Numerous descriptor representations were proposed including contour models [3], feature constellations [4] [5] and appearance templates [6]. More advanced methods are capable of tracking the pose of multiple objects from multiple cameras by ensuring global consistency between all object hypotheses using a simulation framework [7]. These methods are capable of accurately estimating object pose in highly cluttered scenes.

Moreover, the object shape can be fully reconstructed by aligning 3D models of the object to the scene using the estimated pose. The downside of model-based methods is that they are not capable of detecting objects that were not available during the training stage. Manipulation of previously unseen objects is a key feature of truly autonomous robots.

A more general family of methods finds objects by fitting parametric solid shape primitives to the scene. Object shape and pose are obtained by simultaneously estimating primitive pose and parameters. The choice of the primitives used defines the trade-off between the accuracy of shape approximation and the robustness of the fitting process. For cluttered scenes, boxes, cylinders and spheres can be fitted to the input pointclouds using robust RANSAC method [8] [9]. These methods work well for objects that are well approximated by the primitives used (i.e. boxes and mugs) but are unsuitable for more complex objects (teddy bear). Several methods use a more general family of primitives - superquadrics [10] [11]. These methods are limited to single isolated objects and can not tolerate clutter since superquadric fitting is very sensitive to noise and outliers [12].

1.3 Symmetry is Key!

Wherever there is structure - there is symmetry. In its most general sense symmetry is repetition of some element or pattern. Lack of repetition implies lack of structure. Hence symmetry and structure are inseparable concepts. While symmetry is not limited to geometry, from the point of view of perception we are interested

(a) Butterfly **(b)** Flower **(c)** Human face

(d) Taj Mahal **(e)** Car **(f)** Teapot

Figure 1.2: Examples of symmetry in nature and in man-made objects.

in symmetries in geometrical shapes. Symmetrical structures have two appealing properties. Firstly, symmetrical structures are inherently stable. Whether it's an animal standing on its feet, or the the arrangement of atoms in a crystal lattice, or the aerodynamics of a car - symmetrical structures achieve a certain balance of forces that results in an overall stability of the system. Secondly, symmetric shapes can be represented very efficiently, which allows for certain efficiencies in replication. A fern expands as it grows by repeating the same pattern over an over again. An artist making a pot rotates a piece of clay around an axis and shapes only one side of the vessel. Due to these properties symmetry is pervasive both in natural and man-made artifacts. In nature symmetrical structures can be encountered at all scales, from the double helix of the DNA, to plant and animal bodies, all the way to the galaxies. Man-made made objects are almost universally symmetric: chairs,

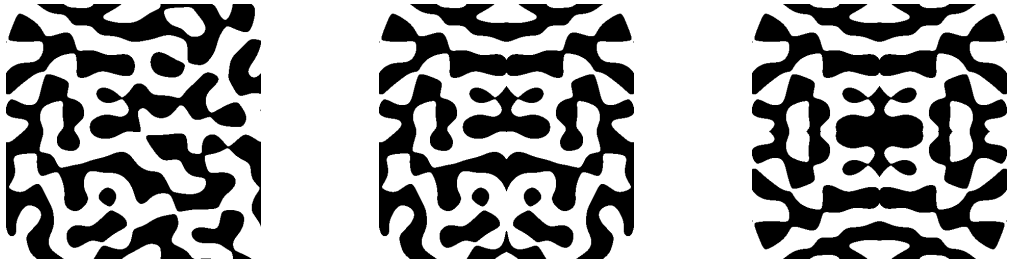


Figure 1.3: Human perception of symmetry (adapted from Treder [1]). (a) A random blob pattern is perceived as disorganized. (b) The symmetry of a pattern is immediately apparent to a human observer. (c) The sense of structure is increased if pattern has multiple symmetries.

cups, screens, buildings, cars are all symmetric. In fact, if you take a look around you right now, you will be hard-pressed to find asymmetrical objects.

Symmetry is so pervasive in natural environments that both human and animal visual systems have evolved to actively exploit it for perception [13] [14]. Symmetry plays a crucial role in the process of figure-ground organization i.e. the process of figuring which parts of the environment form a whole [15]. This is illustrated intuitively in Figure 1.3. A previously unseen pattern is perceived as a whole if it exhibits symmetry. Multiple symmetries strengthen this effect, while a pattern with no symmetry is perceived as lacking structure. Moreover, it was shown that humans identify symmetries more quickly when it belongs to a single object as opposed to being shared between two objects [16]. This finding suggests that object segmentation and symmetry detection are tightly interlocked processes. Additionally, it was found that local symmetry can be used to predict human eye movements when looking at images of novel scenes [17]. This implies that symmetry is used as a cue to guide our visual attention.

Symmetry information is particularly important for robotic grasping applications. Given a partial 3D scan of an object, symmetry can be used to reconstruct the occluded parts of the object. Furthermore, symmetry is directly linked to the object's center of mass. Assuming uniform density, center of mass of an object always lies on the symmetry plane/axis of an object, or at their intersection in case of multiple symmetries. Both of these properties can be used to predict a stable grasp for previously unseen objects [11] [18].

1.4 Contributions of This Thesis

Symmetry encodes structural redundancies of our world. This thesis explores approaches to revealing these redundancies and using them to enable general robotic manipulation in complex and unpredictable environments. Specifically, we present a complete system for segmenting objects and extracting their symmetries in densely cluttered scenes. Unlike CAD model-based methods, our system doesn't require any object specific prior knowledge and can deal with previously unseen objects. Our core assumption is that the three dimensional shape of objects is either reflectional or rotational symmetric. Although this assumption might seem restrictive at first, as discussed earlier, it holds true for a vast majority of rigid man-made objects. Our approach can be seen as a generalization of shape primitive fitting approaches, where instead of fitting shapes to the 3D data, symmetries are fitted instead. This allows our approach to avoid the problem of approximating complex object shapes with simple geometric primitives. Our pipeline consists of three major steps:

- Obtaining a multiple view 3D surface reconstruction of a scene
- Detecting candidate object symmetries
- Segmenting individual objects based on the candidate symmetries

The rest of this thesis is organized as follows. Chapter 2 describes the data collection procedure used to acquire 3D reconstructions. In addition to the traditional pointcloud-based reconstruction of the scene surfaces we collect a voxel-based occupancy map of the scene that is required for the later processing stages. We discuss the sensor calibration procedure for a commodity structured light depth sensor that is necessary to achieve high-quality reconstructions. We also present a dataset of densely cluttered tabletop scenes used for testing and evaluating the efficacy of our perception pipeline.

Chapter 3 addresses the problem of symmetry detection in 3D pointclouds. We propose two approaches. A curve-based approach detects reflectional symmetries by extracting and matching surface normal edge curves in the pointcloud. A surface-based approach can be used to detect both rotational and reflectional symmetries by fitting symmetry axes/planes to the geometry of the smooth surfaces extracted from the scene. This step acts as an attention operator that finds candidate symmetries in the scene.

In chapter 4 we discuss the algorithms for obtaining high quality object segmentation masks given a set of candidate symmetries in the scene. The goal of the segmentation stage is to find points in the scene that belong together according to the local grouping principles of convexity, and at the same time are geometrically

consistent with a given symmetry candidate. Our segmentation algorithm can be used with slight modifications to segment both reflectional and rotational symmetric objects. We demonstrate how a simplified version of our pipeline can be used by a collaborative robot to pick up objects in a kitchen scenario.

We conclude this thesis by discussing the avenues for future research.

Chapter 2: Data Collection

In this chapter we discus the data acquisition process as well as introduce the dataset used to test and evaluate our perception pipeline.

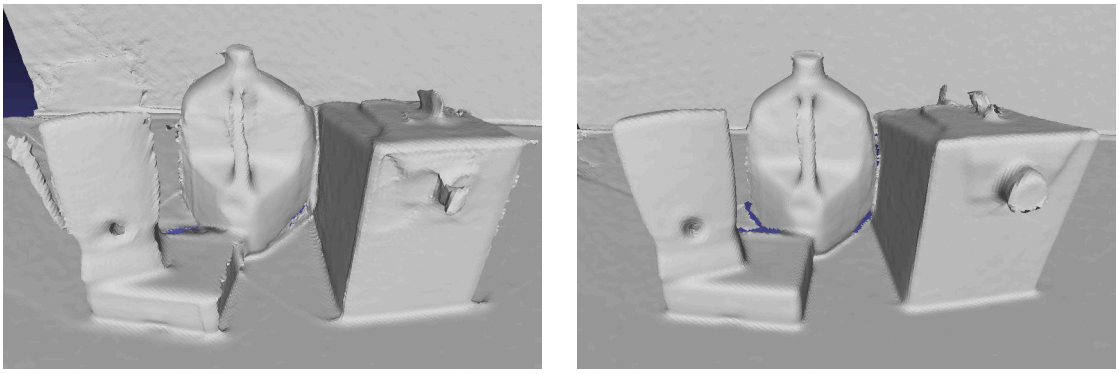
2.1 Reconstruction pipeline

Since our goal is to use symmetry as a constraint on an object's three dimensional shape, we are interested in collecting 3D geometric data. Unlike a static camera, a robot equipped with an imaging sensor fits the paradigm of an active observer [19]: it can actively change the position of its sensors in the world in order to collect additional information about the environment. Hence, our data collection process should not be limited to single views. Instead we want to collect data from multiple views and combine it into a single 3D reconstruction of the scene. In addition to surface reconstruction, we also want to get an occupancy map of the scene that identifies the free and occupied space. This information is used by symmetry detection and segmentation stages to evaluate the compatibility of scene points with candidate symmetries.

2.1.1 Surface reconstruction

The problem of 3D reconstruction from multiple images, known as Structure from Motion (SFM), is one of the oldest problems in computer vision [20]. Although the field has seen steady progress over the last several decades [21] [22], image based methods still struggle to produce highly accurate, dense and noise-free models. The main limitation of these methods is that both scene geometry and camera frame-to-frame poses (ego-motion) need to be estimated simultaneously. This problem can be alleviated by the use of 3D range sensors. 3D sensors directly measure the surface geometry of the scene, thus greately simplifying geometry and ego-motion estimation problems. For our pipeline we use the KinectFusion algorithm [23], that uses Kinect sensor to deliver extremely accurate dense 3D reconstructions while running in real time.

The primary idea behind Kinect Fusion is to fuse measurements from a sequence of depth images into a single model. The fused model is maintained using a volumetric Truncated Signed Distance Function (TSDF) [24]. Given a voxelized volume TSDF represents a surface within the volume by storing for each voxel the signed distance d to the nearest observed surface. Given a new frame, Kinect Fusion algorithm starts by back-projecting the depth image to a pointcloud. Camera pose in is then estimated by aligning the pointcloud to the TSDF using the ICP algorithm. The depth map is fused into the TSDF volume by updating voxels that project into the current depth map. The final TSDF is the weighted average of the TSDF's from individual depth maps. This achieves the effect of smoothing noisy



(a) **(b)**

Figure 2.1: Surface reconstruction of a simple tabletop scene. (a) Using default calibration parameters provided by the camera driver. (b) Using calibrated depth camera intrinsics as well as depth distortion calibration.

measurements from multiple depth maps which allows recovering dense noise-free scene models. Once the data is captured, the final mesh model can be extracted by running the Marching Cubes algorithm [25] on the TSDF.

For our experiments we used an Asus Xtion Pro sensor (similar to Kinect) for data capture and an open-source implementation of the KinectFusion algorithm [26] with a voxel size of 3.1 millimeters. KinectFusion has no bundle adjustement or loop closure mechanism, so errors in frame-to-frame pose estimates accumulate over multiple frames leading to inconsistencies and tearing in the reconstructed model. Hence, a high quality sensor calibration is required to achieve precise reconstructions. This can be done by calibrating the intrinsics and the radial distortion of Kinect's infrared camera [27]. Additionally, it was found that Kinect sensors exhibit complex systematic residual errors in their depth measurement values. In a robotic manipulation scenario, these can result in the robot not being able to pick up the object, since it is perceived as either being too far or too close to it's true

Figure 2.2: Pointcloud reconstruction of a tabletop scene.

position. To correct these errors we constructed a series of pixel-wise lookup tables that store the multiplicative values for the depth measurements at different depth ranges [28]. Figure 2.1 shows the output of our surface reconstruction process on a sample tabletop scene.

After obtaining the scene mesh we convert it to a pointcloud and remove all points that do not belong to the tabletop objects. This is done by detecting the table plane and extracting points lying directly above it. Finally we donwsample the pointcloud using a voxelgrid filter with voxel size of 5 millimeters. Our experiments show that this resolution is sufficient for our needs and at the same time significantly reduces the processing time for the further processing stages of our pipeline. An example of the final scene pointcloud is shown in Figure 2.2.

2.1.2 Occupancy mapping

To construct an occupancy map of the scene we use the OctoMap algorithm [29]. Given a sequence of depth maps and corresponding camera poses, it generates a 3D

volumetric map, where each voxel is labeled as either occupied, occluded or free space. At runtime each voxel stores the probability of the corresponding space being occupied. Voxel occupancy probabilities are updated by casting rays defined by the depth maps into the volume. If a voxel is traversed by a ray, it's occupancy probability in decreased. Conversely if a ray endpoint falls within a voxel, it's occupancy probability is increased. After integrating all of the depth maps, the occupancy probabilities are thresholded to classify voxels as either free or occupied. Voxels that were never traversed by a ray are considered occluded. The resulting map is expected to have very few tightly localized occupied voxels and large chunks of free and occluded voxels. OctoMap takes advantage of this data sparsity by storing the map in an octree datastructure. This significantly reduces the memory footprint of the algorithm and allows storing of large maps.

For our experiments we used the open-source implementation of the OctoMap algorithm [30]. Since OctoMap has no mechanism for frame-to-frame camera pose estimation, we used poses estimated by KinectFusion instead. The occupancy map was constructed with voxel size of 5 millimeters. In our experiments, we have found that lower resolutions negatively affected the performance of the symmetry detection and segmentation stages of our pipeline. One of the downsides of the OctoMap algorithm is that rays are integrated into the volume sequentially on a CPU, which results in slow performance. To alleviate this bottleneck we only processed every 5th frame of our sequences and for each depth map use every 10th point.

Finally, to reason about scene occlusions we compute the three dimensional Euclidean Distance Transform (EDT) of the occupancy map. In the context of

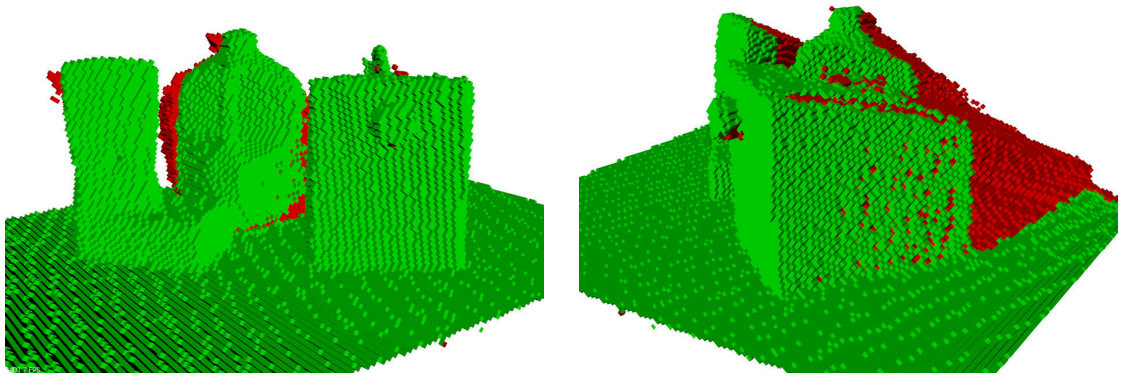


Figure 2.3: Visualization of the occupancy map. Green shows the interface (boundary) between the occupied and free space, while red shows the interface between occupied and free space. In the right image, notice the trailing "tail" behind the objects, that corresponds to the occlusions behind the objects.

robotics EDT is usually used for path planning and collision avoidance. Given a binary spatial map where obstacles are labeled with ones, an EDT computes the Euclidean distance from each voxel in the map to the nearest obstacle voxel. EDT allows quickly evaluating the distance from an arbitrary point in space to the nearest obstacle. Given an occupancy map, we construct the EDT by treating both occluded and occupied voxels as obstacles. As will be discussed in the following chapters, we will use the EDT to judge which points of the scene are compatible with a given symmetry by reflecting/rotating the point and checking it's distance to the occluded/occupied space. We will use $EDT(P_i)$ to denote the distance from point P_i to the nearest occluded/occupied voxel in the scene. The distance is clamped to a maximum of d_{max}^{occl} in order to reduce the influence of outliers. We used the value of $d_{max}^{occl} = 3cm$ throughout our experiments.

Figure 2.4: Sample scenes from the Cluttered Tabletop Dataset.

2.2 Cluttered Tabletop Dataset

To evaluate our pipeline we propose a new Cluttered Tabletop Dataset, which contains 3D reconstructions of 89 cluttered tabletop scenes. The set of objects used in the dataset ranges from simple objects like boxes to highly non-convex objects such as bowls and stuffed toys. Scenes have varying complexity, from single objects to up to 15 objects put side by side and stacked on top of each other, as shown in Figure 2.4. Scenes were captured by manually moving a depth sensor in approximately 120° horizontal arc around the center of the scene. This strategy was chosen to simulate the environment sensing procedure that could be executed by a robot with a camera mounted on its manipulator. All of the scenes were labeled with ground-truth pointwise segmentation masks for the individual objects,

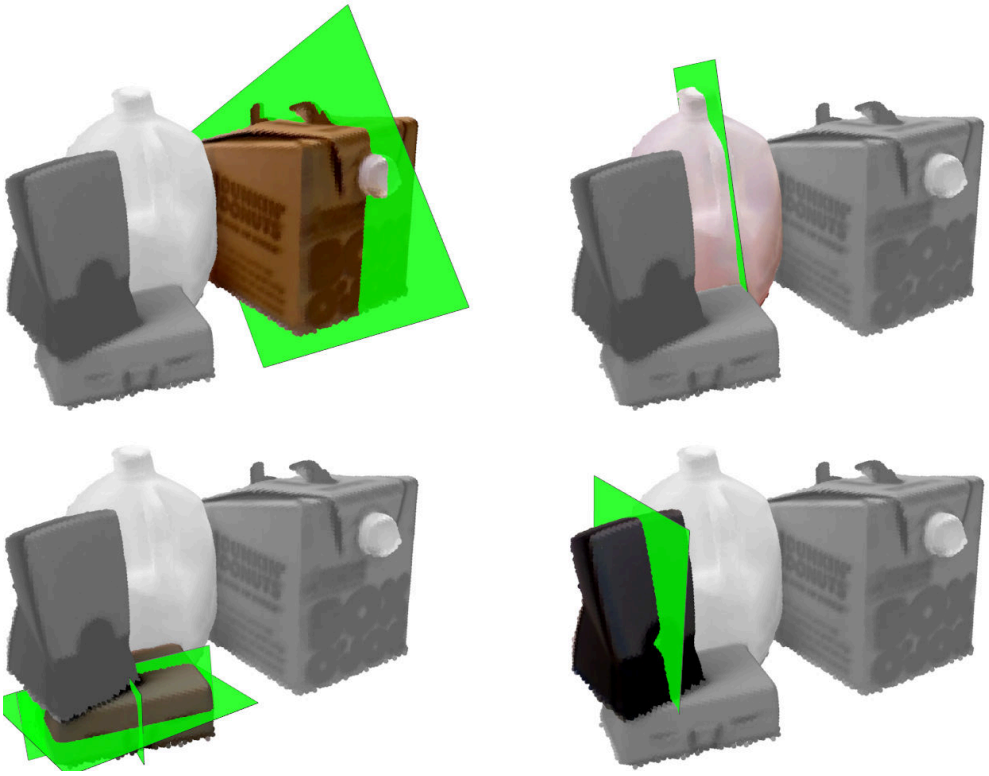


Figure 2.5: Labeling for one of the scenes in the Cluttered Tabletop Dataset. Colored points denote individual segments and green planes show corresponding symmetries.

as well as corresponding object rotational and reflectional symmetries, as illustrated in Figure 2.5.

Chapter 3: Symmetry Detection in 3D Pointclouds

3.1 Introduction

Symmetry is ubiquitous in natural and man-made objects. Gestalt psychologysts have recognized the role of symmetry in the human visual system as an important cue that serves as input to the higher level visual processes such as segmentation [15]. This observation has been mirrored in Computer Vision applications, where symmetry is used as a preprocessing step for a variety of object-centric perception tasks such as object recognition [31], retrieval [32], reconstruction [33] as well as robotic manipulation [11].

Unlike most of the previously proposed approaches that operate on high quality 3D models of single objects, our aim is to detect symmetries of multiple objects in incomplete pointclouds. Our approach is to first extract stable geometric features and then find symmetries within single or multiple feature instances. In order to handle cluttered scenes, the features used have to satisfy two properties: a single feature must belong to a single object and at the same time it must capture the geometry of the underlying object. Although orientable 3D point features may seem like an obvious choice, they are not suitable for our task. Due to the limitations in the resolution of the pointcloud and the nature of the shape of the object, ex-

tracting a sufficient number of features may be problematic. For example, a box only has 6 corners that can serve as discriminative feature points and only few of those may be visible due to occlusions. A bowl has no unique feature points at all. Alternatively we propose two families of approaches that use curve and surface features instead. Our curve-based approach detects reflectional symmetries by extracting surface normal edge curves and then searching for pairs of curves that are "aligned" by a reflectional symmetry. In our surface-based approach we first extract smooth surfaces from the pointcloud and then fit reflectional/rotational symmetry planes/axes to the segments. For rotational symmetry, we define a novel measure of fitness between an oriented point and a symmetry axis that allows us to fit a symmetry axis directly to the segment points. Reflectional symmetries are detected using an ICP-like technique that alternates between finding correspondences between symmetric points and refining the candidate symmetry plane given the correspondences. For each of the three approaches we define a set of confidence measures that can be used to control the precision-recall tradeoff of the method.

3.2 Related Work

A large amount of research in Computer Vision is devoted to symmetry detection in 2D and 3D data. Most of the 2D approaches rely on extracting and matching intensity discontinuities such as edges and point features. One of the first successful methods for detecting symmetry in 2D images is the "Generalized Symmetry Transform" of Reisfeld *et al.* [34]. The main idea of the method is that two image

points provide evidence for the existence of symmetry at a midpoint between them if they "reflect" onto each other. This principle was used to generate a continuous "symmetry map" that measures how symmetrical the neighborhood around an image point is. This idea was extended by Loy and Eklundh [35] who employed robust oriented SIFT features to improve the quality of symmetrical matching. Pairs of symmetrically matching feature points were used to generate reflectional and rotational symmetry hypotheses which were then aggregated in a Hough voting scheme to find the dominant symmetries in the image. Some approaches proposed symmetry for attention [36], and symmetry to aid segmentation of symmetric objects [37]. More recently Teo *et al.* [38] proposed to detect curved reflectional symmetries and used them to drive the segmentation process by embedding a symmetry term into the cost function of the Markov Random Field (MRF) used for segmentation.

For three dimensional data most of the existing methods focus on recovering symmetries from complete 3D models. Sun et al. proposed to detect global symmetries in a 3D model by analyzing its three dimensional surface normal histogram called the Extended Gaussian image (EGI) [39]. The key assumption is that an EGI of an object has the same global symmetries as the object itself. Thus they can be discovered by finding the transformation that maximizes the correlation between the original and transformed EGIs. This assumption however does not hold for noisy and incomplete object models. Podolak *et al.* [32] defined the "Planar Reflective Symmetry Transform" that captured the degree of symmetry of an object with respect to all possible planes passing through it. Exact object symmetries were then detected by extracting the transform maxima. In [40], Mitra *et al.* . relied on

orientable feature matching to detect partial symmetries in complete 3D meshes of single objects. Symmetry hypotheses were found by matching uniquely orientable keypoints on the surface of the mesh and filtering out the dominant ones using mean shift clustering. This approach was capable of detecting reflectional, rotational, translational and scaling transformations present in the mesh. Thrun and Wegbeit reasoned about occlusions of the scene to detect symmetries in partial 3D pointclouds [41]. Symmetries were found by searching through the space of possible transformations and finding the one that minimizes the number of points that reflect into the unoccluded space and maximizes the match between the original and the reflected clouds. Detected symmetries were then used to reconstruct occluded parts of the object.

3.3 Notation

We use the following notation to denote oriented points, symmetries and reflections/rotations of points throughout this and the following chapters:

- $P = \{p, n\}$ - an oriented point with coordinate p and a normal vector n.
- $S^{refl} = \{n^{refl}, d^{refl}\}$ - a reflectional symmetry plane defined by a normal n^{refl} and distance to origin d^{refl}.
- $S^{refl} = \{n^{refl}, p^{refl}\}$ - a reflectional symmetry plane defined by a normal n^{refl} and a point p^{refl} lying in the symmetry plane.
- $S^{refl}(\cdot)$ - a reflection of either a point or a normal by the reflectional symmetry

plane S^{refl}.

- $S^{rot} = \{d^{rot}, p^{rot}\}$ - a rotational symmetry axis defined by a point p^{rot} of the axis and the direction vector d^{rot}.

- $S^{rot}(\cdot, \theta)$ - a rotation of a point or normal by angle θ around the rotational symmetry axis S^{rot}

3.4 Curve-based symmetry detection

Consider a simple toy example of detecting reflectional symmetries of a box. The edges of the box are the straight lines connecting it's vertices. It is immediately obvious that edges provide sufficient information for discovering the symmetries of the box: we can find pairs of edges for which there exists a symmetry plane that reflects them onto each other. Following this process we will find some of the symmetries that are not correct (between edges that don't lie on the same planar surface of the box), but we are guaranteed not to miss any of the true symmetries. This observation motivates our curve-based symmetry detection approach. Given a pointcloud with surface normal information we define surface normal edges as an ordered set of points where surface normal direction changes abruptly. After extracting the edges we find symmetric matches between pairs of such edges. Using a robust clustering method allows us to recover symmetries even from noisy edges that contain a high number of outliers.

3.4.1 Pointcloud edge detection

Our strategy for extracting normal edges is similar to that presented in [42] which utilizes the duality between segmentation and edge detection. We oversegment the cloud into patches whose borders are likely to lie along normal edges and then find the boundaries between them. We use the algorithm of Papon et al. [43] that segments pointclouds into supervoxels with high boundary adherence. To avoid extracting boundaries between supervoxels that belong to the same flat surface we merge all adjacent supervoxels that have collinear surface normals (Figure 3.2d). Superovxel boundary points are found by extracting points which have at least one neighbor that belongs to a different supervoxel (Figure 3.2b). Boundary points are then linked together using a Minimum Spanning Tree algorithm to form a set of disjoint edge curves $\mathbf{C} = \{C_1, ..., C_n\}$. For each of the curves, its pointwise tangent direction vectors are estimated by fitting lines to each of its points and their neighbors. Each curve $C_i = (p^1_{C_i}, ..., p^m_{C_i})$ is represented by an ordered set of oriented points and describes a surface normal edge in the cloud (Figure 3.2e).

3.4.2 Reflectional symmetry detection

Once the edge curves are extracted, pointcloud symmetries can be recovered by finding symmetrical pairs of edge curves. The basis of this process is the idea of symmetrical correspondence between two oriented points. Consider two points in space P_i and P_j. The coordinates of the points uniquely define a reflectional symmetry plane that is perpendicular to the line connecting the points and contains

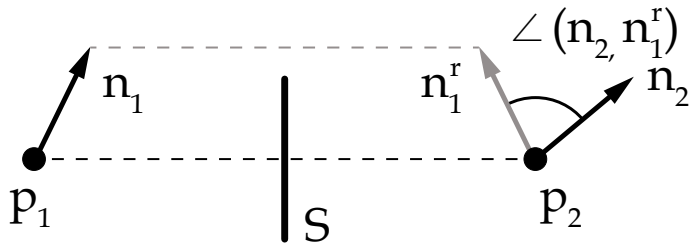


Figure 3.1: Symmetry detection from a pair of oriented points.

the midpoint between them. Representing the symmetry plane $S^{refl} = \{n^{refl}, d^{refl}\}$ by its normal and distance to the origin, its parameters can be expressed as:

$$n^{refl} = \frac{p_i - p_j}{\|p_i - p_j\|} \tag{3.1}$$

$$d^{refl} = \frac{(p_j - p_i) \cdot n^{refl}}{2} \tag{3.2}$$

We define the matching score of the symmetric correspondence between P_i and P_j as the angular difference between the normal of one of the points n_i and the reflected normal of the other point $S^{refl}(n_j)$:

$$M_S(P_i, P_j) = \angle(n_i, S^{refl}(n_j)) \tag{3.3}$$

By putting a threshold on the maximum allowed matching score, we can reject noisy symmetric correspondences. This mechanism allows us to decide if a valid symmetry plane exists for a pair of points and to recover that symmetry.

Given a pair of edge curves C_i, C_j we exhaustively search for symmetrical correspondences between their individual points $\{P_{C_i}, P_{C_j}\}$ using the symmetrical

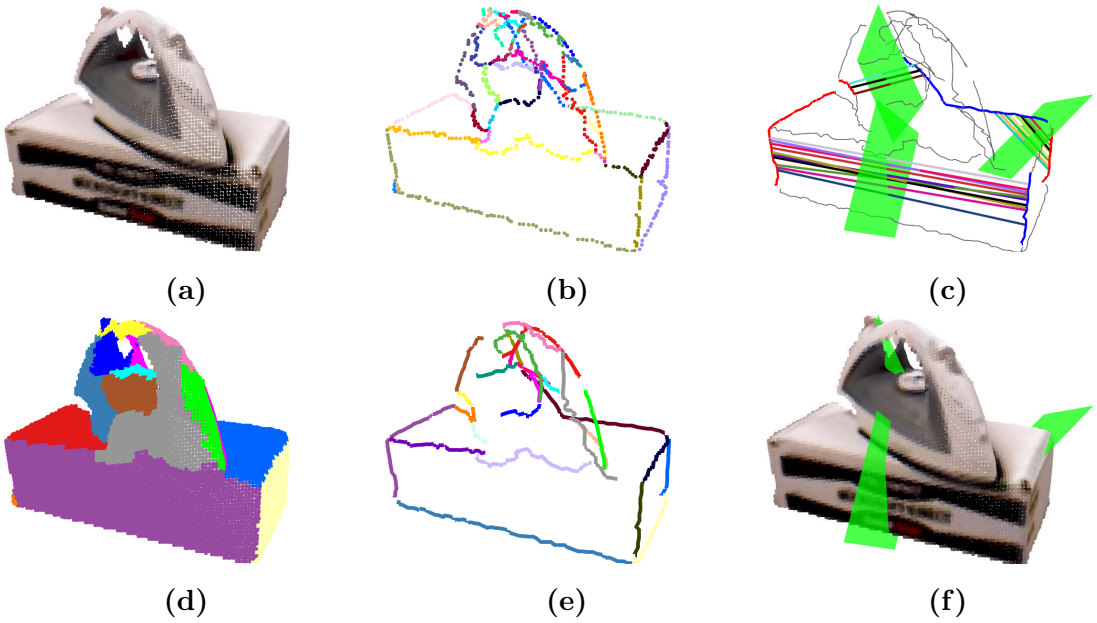


Figure 3.2: Curve-based symmetry detection. (a) Input pointcoloud (d) Segmentation (b) Segment boundary points (e) Linked edge curves (c) and (f) Symmetry correspondences and corresponding symmetry hypotheses.

correspondence matching approach described above. Curve tangent direction vectors are used as orientation vectors instead of surface normals, since surface normal estimates are inherently noisy at the points of high curvature. Matches are filtered by removing correspondences with a match score higher than a threshold $M_S(P_{C_i}, P_{C_j}) > \alpha_{max}$ and enforcing a one-to-one correspondence relationship. This results in a set of noisy candidates for the global symmetry plane aligning the two curves. To recover the global symmetry plane we take an approach similar to the one proposed in [40]. Symmetry candidates are treated as samples of the probability density function of the global symmetry plane. The problem of recovering the global symmetry plane becomes equivalent to clustering these candidates. To do the clustering we represent the candidates with points in three dimensional space by

weighting the symmetry plane normal by the distance to the origin i.e. n^{refl}/d^{refl}. We then use mean shift clustering [44] to find the dominant mode of the distribution and use its maximum as the symmetry plane aligning the two curves. This procedure is run on every pair of edge curves resulting in a set of 3D reflectional symmetry hypotheses for the input pointcloud $\mathbf{S} = \{S_1^{refl}, ..., S_k^{refl}\}$ (Figure 3.2c). Finally, we filter out symmetries for which the ratio of the number of valid symmetric correspondences to the total number of points in the corresponding curves is lower than a threshold n_{inlier}. This allows us to remove some of the false symmetry hypotheses that will inevitably be detected by our approach.

3.5 Surface-based symmetry detection

While the curve-based approach focuses on the high curvature features of the pointcloud, our surface-based approach makes use of the dense geometric information available from the smooth surfaces. It capitalizes on the fact that many objects contain large continuous surfaces which share the same symmetries as the complete object. First, we oversegment the pointcloud using a region growing algorithm that uses local surface normals and point connectivity to enforce a smoothness constraint. To detect scene symmetries, we employ a geometric fitting approach. Two separate algorithms are used to fit rotational symmetry axes and reflectional symmetry planes respectively to the scene segments. Since the segments are noisy, we ensure that both algorithms are robust to outliers and at the same time are sensitive enough to detect symmetries from limited support.

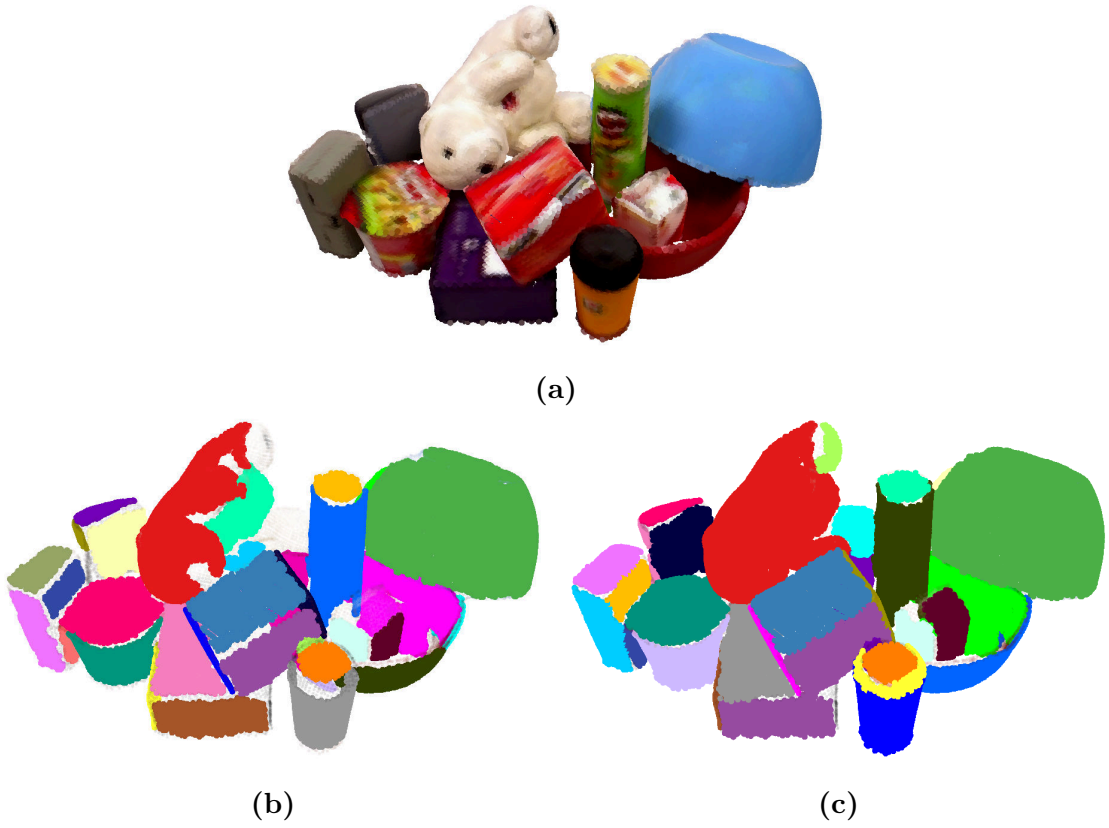


Figure 3.3: Scene pointcloud pre-segmentation. (a) Scene pointcloud. Pre-segmentation with (b) $\theta_{smooth} = 10°$, and (c) $\theta_{smooth} = 15°$.

3.5.1 Smooth surface segmentation

We employ a variant of the region growing algorithm with a smoothness constraint [45]. Pointcloud points are sorted in the order of increasing curvature and the point with the minimum curvature is selected as the seed point. At each step six of the nearest neighbors of the current seed point are examined. Segment smoothness is enforced by requiring that the angle between the normals of neighboring points is smaller than a threshold θ_{smooth}. Unlike the standard approach, in which all of the neighboring points satisfying the smoothness constraint are added as new

segment points, we only grow the segment if at least half of the neighbors satisfy the constraint. This prevents the segments from "leaking" over the surface normal boundaries due to noisy surface normal estimates. The newly added points are used as new seeds. The process continues until no more valid neighbors can be found or all of the pointcloud points belong to a segment. If there are any points remaining they are used to initialize a new segment.

To increase the likelihood of recovering the necessary segments, we run the above segmentation algorithm at several smoothness thresholds. Specifically we use $\theta_1 = 10°$ and $\theta_2 = 15°$. Figure 3.6 shows an example of a scene pre-segmentation. Note how the two segmentations complement each other. With $\theta_1 = 10°$ the front faces of the red and violet boxes are segmented correctly, while the teddy bear is over-segmented. On the other hand, with $\theta_1 = 15°$ teddy bear is segmented correctly while the box faces are merged.

3.5.2 Rotational symmetry detection

Rotational symmetry imposes a strong constraint on the shape of an object. We observe that for any point on the surface of a rotational symmetric object, the corresponding surface normal lies in the plane formed by the symmetry axis and the point itself, as shown on Figure 3.4. This allows us to formulate an optimization scheme that fits a rotational symmetry axis directly to the segment pointcloud.

Consider a pointcloud consisting of oriented points $P_i = \{p_i, n_i\}$ and a candidate symmetry axis $S^{rot} = \{p^{rot}, d^{rot}\}$ described by a point and a unit direction

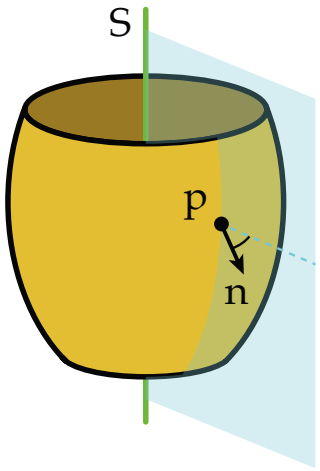


Figure 3.4: Constraint imposed on the points of a rotationally symmetric object. The surface normal n of any point of a rotationally symmetric object must lie in the plane formed by the symmetry axis S and the point p itself.

vector. Let $\angle(S^{rot}, P_i)$ denote the angle between the point normal n_i and the plane formed by the symmetry axis S_{rot} and point p_i. The candidate symmetry axis can be fitted to the pointcloud by minimizing the following function over the symmetry axis parameters:

$$\sum_i \min(\angle(S^{rot}, P_i), \alpha_{max}) \tag{3.4}$$

where α_{max} is the maximum angle used to limit the influence of outliers. In our experiments we use $\alpha_{max} = 45°$. This function can be minimized using a non-linear solver such as Levenberg-Marquardt.

To initialize the minimization we use PCA to construct three candidate symmetry axes that go through the center of mass of the pointcloud and are aligned to the segment principal axes. We have found this initialization procedure ensures that at least one of the three axes will converge to the true symmetry axis. Once all

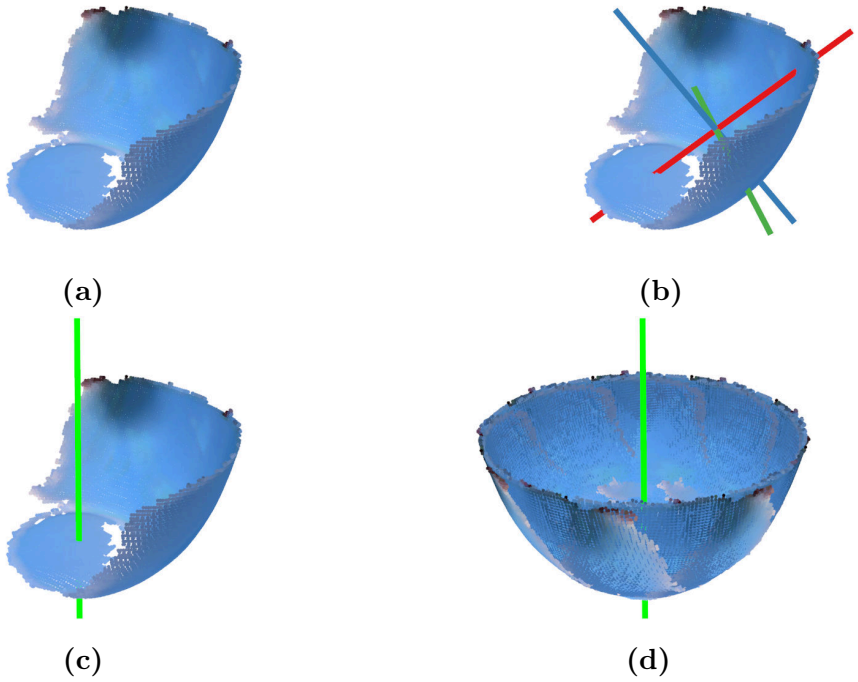


Figure 3.5: Rotational symmetry detection stages shown for the blue bowl segment in Figure 3.6. (a) Input pointcloud. (b) Initial symmetries. (c) Refined symmetry. (d) Cloud reconstructed by rotating the segment around the symmetry axis.

three axes are refined, the one attaining the minimum error is selected as the single final hypothesis. This is motivated by the fact that, except for a degenerate case of a sphere, any shape can have at most one rotational symmetry axis. Figure 3.5 shows the symmetry detected for a bowl. Note how our approach manages to accurately detect the symmetry from a partial pointcloud, which allows us to reconstruct the complete shape of the object.

In order to find symmetries for all of the rotational symmetric objects in a scene, we apply our detection approach to all of the segments returned by the smooth segmentation stage. While the resulting set oh hypotheses is likely to contain all of the true rotational symmetries, it will also contain many false positives, since the

scenes are expected to contain objects that have no rotational symmetries (e.g. a box). To filter out these cases we define a set of metrics that measure how well a rotational symmetry axis fits a given segment.

Symmetry score. Symmetry score measures how well a candidate symmetry axis fits the points of a segment. It is calculated as the total fitness error between the final symmetry axis and the segment, normalized to the $[0, 1]$ range:

$$Sym^{rot} = \frac{1}{N} \sum_{i} \frac{\min(\angle(S^{rot}, P_i), \alpha_{max})}{\alpha_{max}} \tag{3.5}$$

where $\theta_{max} = 60^{\circ}$ and N is the number of points in the segment. Lower symmetry score implies a better fit between a symmetry and a segment.

Perpendicularity score. Intuitively, there is no unique way of fitting a rotational symmetry to flat surface, such as a face of a box. In fact, given our definition of fitness between a symmetry axis and a point, all of the points of a flat segment achieve a fitness score close to zero for any symmetry axis that is perpendicular to the segment. This means that the symmetry score can not be used to filter out such segments. To handle this corner case we measure how perpendicular a segment is to a symmetry candidate. The perpendicularity score between a point and a symmetry axis is calculated as the angle between the point normal and the symmetry axis:

$$Perp^{rot}(P_i) = \frac{\angle(d^{rot}, n_i)}{90^{\circ}} \tag{3.6}$$

Perpendicularity between a segment and a axis is defined as the average of point perpendicularity scores.

Symmetries that satisfy all of the above filtering checks are accepted as final symmetry candidates. Specifically, a symmetry candidate $S^{rot}$is accepted as valid if satisfies the following conditions: $Sym^{rot} < Sym^{rot}_{max}$ and $Perp^{rot} < Perp^{rot}_{max}$. Symmetries that don't satisfy at least one of the measures are discarded.

3.5.3 Reflectional symmetry detection

Unlike the case of rotational symmetry, reflectional symmetry does not impose any constraints on the position or surface normal orientation of single points. Instead, relationships are established between pairs of symmetric points. We say that point P'_i is a symmetric correspondence of P_i under the reflectional symmetry S^{refl} if the two points "reflect" onto each other i.e. P'_i is similar to $S^{refl}(P_i)$ - the reflection of P_i by S^{refl}. This motivates our fitting approach that alternates between symmetric correspondence estimation and symmetry plane refinement.

We represent a symmetry plane $S^{refl} = \{p^{refl}, n^{refl}\}$by a point lying in the plane p^{refl} and the plane's normal n^{refl} (note that this is a different representation, from the one used in our curve-based approach). For each point P_i we compute its reflection $S^{refl}(P_i)$ and search for its nearest neighbor P'_i in the pointcloud. Since we are dealing with incomplete object scans, we do not expect every point to have a valid symmetric match. Thus we only establish a correspondence if the distance between the reflected point and its nearest neighbor is less than 1cm. Addition-

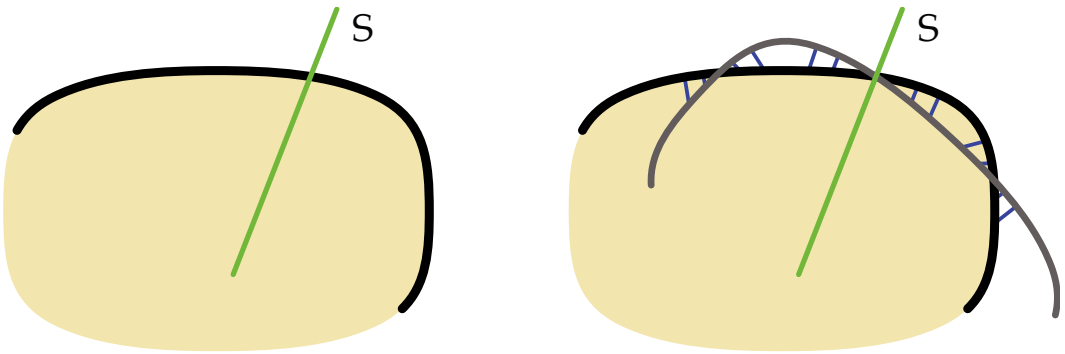


Figure 3.6: Visualization of the reflectional symmetry fitting approach (a) Input object pointcloud and candidate symmetry. Black curve denotes the observed points of the object. (b) Grey curve denotes the reflected pointcloud and blue lines show the estimated symmetric correspondences.

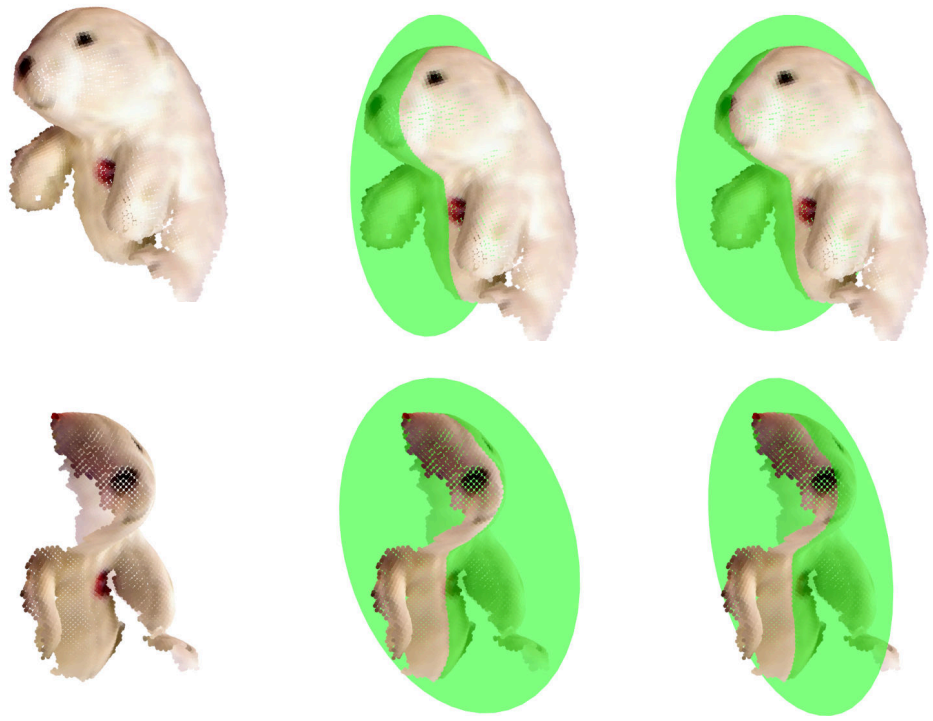
ally, we reject all correspondences for which the angle between the reflected normal $S^{refl}(n_i)$ and the corresponding normal n'_i is greater than 45°. After establishing the correspondences, we minimize the following function for the parameters of the symmetry plane:

$$\sum_{\{i\}} d_{p,pl}(S^{refl}(P_i), P'_i) \tag{3.7}$$

where $d_{p,pl}$ stands for point to plane distance and $\{i\}$ is the set of points that have a correspondence. This process is repeated until convergence or until a maximum number of 20 iterations is reached. Since correspondences are only established between points that are already approximately symmetric, this method results in very accurate alignment between a symmetry and a pointcloud. However, similar to ICP, it requires a good initial symmetry plane estimate [46].

One of the ways to generate initial hypotheses is by jointly sampling plane

positions from a grid defined around the pointcloud and plane orientations from a unit sphere. This is computationally prohibitive, since sampling from an $n \times n \times n$ grid is $\mathcal{O}(n^3)$ and densely sampling from a sphere incurs a large constant factor [32]. We avoid the costly position sampling by only sampling orientations and then refining plane positions based on the pointcloud structure. We start by sampling points from the surface of a unit sphere aligned to the principal axes of the segment pointcloud. Specifically, we rotate the unit vector corresponding to the largest principal axis by multiples of degrees 36° around the two remaining principal axes. In spherical coordinates, this corresponds to uniform angular sampling in polar and azimuth angles. If any two normals are opposites of each other, we discard one of them since they define the same symmetry plane. This results in 25 candidate orientations. For each orientation, we construct a symmetry plane S^{refl} that goes through the segment's center of mass. To refine the position of a plane we first find all points that could potentially form symmetric correspondences if the symmetry plane was shifted appropriately. Given a point P_i, its potential symmetric neighbors P_j are enclosed in a cylinder centered at P_i with an axis parallel to the plane normal n^{refl}. These "cylindrical" neighbors can be found efficiently by projecting the pointcloud onto the symmetry plane and performing a radius search. We filter out correspondences for which the angle between the normal n_i and the reflected neighbor normal n_j^{refl} is greater than 10°. For each of the remaining correspondences, we calculate the shift distance $d_{i,j}$ as the distance between the symmetry plane and the midpoint between P_i and P_j. We compute the median shift d_{median} from all correspondences and use it to shift the symmetry candidate's



(a) **(b)** **(c)**

Figure 3.7: Reflectional symmetry detection stages shown for a teddy bear segment in Figure 3.6. Top and bottom rows show different sides of the segment. (a) Input pointcloud. (b) One of the initial symmetries. (c) Refined initial symmetry.

point along its normal:

$$S^{refl} = \{p^{refl} + d_{median} * n^{refl}, n^{refl}\} \tag{3.8}$$

After generating the initial symmetries, we apply the global iterative refinement described above to obtain the final symmetry detections. Figure 3.7 shows the symmetry detected for a teddy bear.

Similarly to the case of rotational symmetry we implement several filtering measures.

Symmetry score. Symmetry fitness score is only defined for the points that have symmetric correspondences. For a point P with a symmetric correspondence P' the fitness score is calculated as the normalized angle between the reflected point normal and the corresponding point normal:

$$Sym^{refl}(P_i) = \frac{\min(\angle(S^{refl}(n_i), n_i'), \alpha_{max})}{\alpha_{max}} \tag{3.9}$$

with $\alpha_{max} = 20°$. The fitness between the symmetry plane and a segment is defined as the average symmetry score for all of the correspondences.

Inlier score. We want to reject symmetries which don't have sufficient support, i.e. that have too few symmetric correspondences. Inlier score is calculated as:

$$Inlier^{refl} = \frac{N_{corr}}{N} \tag{3.10}$$

where N_{corr} is the number of correspondences under a given symmetry and N is the number of points in the segment.

3.6 Experiments

To compare the proposed approaches we tested their performance on the Cluttered Tabletop Dataset. As discussed in Chapter 2 the scenes in the dataset contain a high amount of clutter which poses a significant challenge for precise symmetry detection. We evaluate our approaches in terms of their ability to predict individual

symmetries present in the scenes. In Section 3.6.1 we discuss the details of the experimental setup and the evaluation procedure used. We then presents the quantitative results and discuss the strengths and shortcomings of the proposed approaches in Section 3.6.2.

3.6.1 Evaluation procedure

Our evaluation procedure is inspired by the evaluation method used for 2D symmetry detection [47]. Similarity between a predicted symmetry S and the ground truth symmetry S_{GT} is measured based on the angular and positional differences between the two symmetries. The geometric definitions of the angle between two axes and the angle between two planes provide good measures of angular similarity between rotational and reflectional symmetries respectively:

$$\Delta_{angle}(S, S_{GT}) = \angle(S, S_{GT}) \tag{3.11}$$

Finding a measure of positional similarity between symmetries is more challenging. The distance between the lines/planes does not provide a meaningful measure of similarity between symmetries. For example, two rotational symmetry axes belonging to different objects in the scene may intersect outside the bounds of the scene. The situation is even more extreme in case of reflectional symmetries, since any two planes will always intersect, unless they are parallel. It is more revealing to measure the distance between the two symmetries in the spatial area defined by the object that exhibits the ground truth symmetry. Since our dataset contains object

segmentation masks associated with every ground truth symmetry, we use the centroids of the segments as the estimates of the locations of the object. Specifically, given a ground truth symmetry S_{GT} associated with a ground truth segment Seg_{GT} with centroid Ctr_{GT}, we define the distance to a predicted symmetry S as the Euclidean distance between the projections of the segment centroid onto the predicted and ground truth symmetries:

$$\Delta_{pos}(S, S_{GT}) = \|proj_{\mathbf{S_{GT}}}(Ctr_{GT}) - proj_{\mathbf{S}}(Ctr_{GT})\| \tag{3.12}$$

Given the above definitions we say that a ground truth symmetry S_{GT} in the scene is predicted correctly if there exists a predicted symmetry such that the angular difference $\Delta_{angle}(S, S_{GT})$ is less than $t_1 = 10^\circ$ and positional difference $\Delta_{pos}(S, S_{GT})$ is less than $t_2 = 10$ cm. If several of the predicted symmetries satisfy these criteria for the same ground truth symmetry, the predicted symmetry achieving the smallest distance difference is counted as a true positive detection, while the rest are considered false positive. Predicted symmetries that do not match to any of the ground truth symmetries are considered false positive, while ground truth symmetries that do not get matched to any predicted symmetries are considered false negative.

We use the evaluation procedure described above to perform the precision-recall analysis of our symmetry detection approaches. For each of the three methods we varied the parameters of the confidence measures used to filter the detected symmetries. The values for parameters used are summarized in Table 3.1. Each

approach was evaluated on the corresponding symmetry type i.e. curve-based and surface-based reflectional symmetry detection approaches were evaluated on reflectional ground truth symmetries, while surface-based rotational symmetry approach was evaluated on ground truth rotational symmetries.

Method	Parameter	min	max	step
Curve-based reflectional	M_S	1°	15°	1°
	n_{inlier}	0.1	0.5	0.1
Surface-based rotational	Sym^{rot}	0.004	0.03	0.002
	Per^{rot}	0.6	0.7	0.1
Curve-based reflectional	Sym^{refl}	0.7	0.95	0.05
	$Inlier^{refl}$	0.2	0.8	0.2

Table 3.1: Parameters used for evaluating symmetry detection. Parameters were varied from *min* value to *max* value in steps of size *step*.

3.6.2 Results and discussion

Figure 4.3 shows the PR curves and the maximum f-scores for the three algorithms. Curve-based reflectional symmetry detection achieves an f-score of 10%. While the algorithm achieves a high level of recall, it comes at a price of very low precision. This can be attributed to the fact that the output of the normal edge detection is very noisy. This results in many false positive symmetric matches between curves that belong to different objects or non-symmetric curves of the same object.

On the other hand, curve-based detection approaches achieve a comparable high recall, but have much higher precision. The dense geometric information avail-

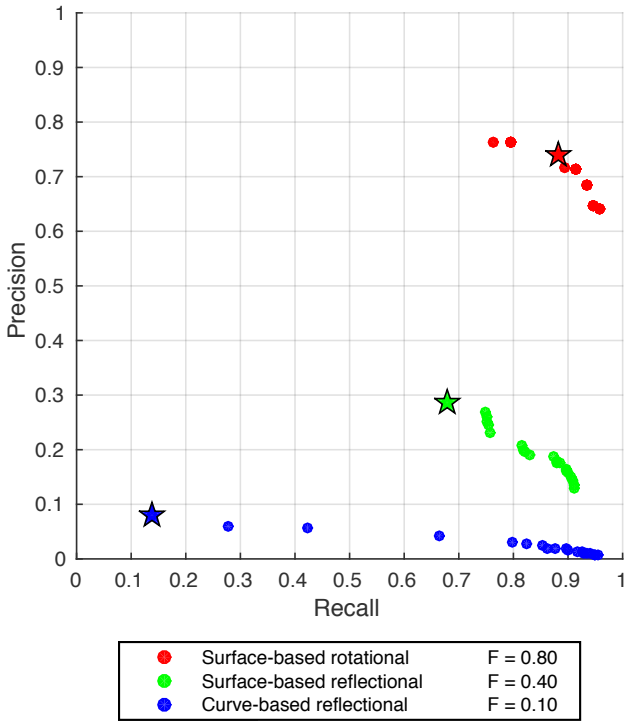


Figure 3.8: PR curves for symmetry detection evaluation. Stars mark the point of maximum F-measure.

able from the smooth segments allows these methods to detect the symmetries much more reliably, while ignoring many of the false positives. The reflectional symmetry detection approach achieves an f-score of 40%. Rotational symmetry detection has the highest f-score of 80%. This is due to the fact that rotational symmetry places a much stricter constraint on the shape of an object, compared to reflectional symmetry. In general, the main source of errors of the surface-based methods stem from the imperfections of the presegmentation process. Two kinds of errors can be distinguished. Firstly some of the segments can span multiple objects, which results in false negatives, since it becomes harder to extract true symmetries for both of the objects. Secondly, a segment that only spans a part of the object may possess

additional symmetries that are not present in the complete object, which results in false positive detections. Additionally, a large number of false positive reflectional symmetries come from the rotationally symmetric objects. This is due to the fact that for a rotational symmetric object, a plane passing through the symmetry axis defines a valid reflectional symmetry. As a result the filtering process can not distinguish these symmetries as invalid.

3.7 Conclusions

Symmetry detection is an important perceptual capability that serves as a preprocessing step for many visual tasks. In this chapter we presented two families of approaches for reflectional and rotational symmetry detection in pointclouds. An evaluation on a challenging dataset of cluttered scenes showed that the surface-based approaches deliver the highest quality results, especially for rotational symmetries. All of the methods achieve a very high recall ($> 90\%$) under relaxed filtering conditions. As expected, this comes at a cost of reduced precision. This observation suggests that a more sophisticated filtering approach may retain the same recall performance while increasing the precision by intelligently discarding false positives. In the next chapter we discuss how symmetry detection results can be used to aid object segmentation, and, how the segmentation step can be used to filter our false positive symmetry detections.

Chapter 4: Symmetry Segmentation

4.1 Introduction

Figure-ground organization, the process of grouping smaller percepts into a whole, is one of the core abilities required for visual perception. Symmetry is one of the main grouping laws proposed by Gestalt psychologists. In this chapter we propose a method for segmenting reflectional and rotational symmetric objects in cluttered scenes. Given a set of candidate symmetries we initialize multiple figure-ground segmentations. The goal of each segmentation is to find points in the scene that belong together according to the grouping principles of proximity and convexity, and at the same time are geometrically consistent with a given symmetry candidate. The key insight to our segmentation approach is that symmetry hypotheses that correspond to one of the objects in the scene are consistent with all of the observed points of that object, while incorrect symmetries are compatible with an unorganized set of points that can be easily discarded. Using the results from Chapter 3 we present a complete pipeline for detecting symmetries and using them to segment objects in a pointcloud. We evaluate our approach on a challenging dataset and demonstrate how a simplified version of our system can be used by a mobile robot to manipulate objects in a kitchen scenario.

4.2 Related Work

Image segmentation is one of the fundamental problems in Computer Vision and has been studied extensively. This problem is inspired by Gestalt theory of perceptual organization which states that there exist a number of bottom-up "grouping laws" which allow humans to perceive the visual stimuli in terms of coherent groups corresponding to meaningful objects and patterns [48]. The laws of proximity, similarity, continuity and closure are implicitly used in all modern segmentation algorithms. Most of the classical segmentation algorithms were developed for analyzing RGB or greyscale images and use texture, color and image contour features for grouping [49] [50] [51] [52]. Although suitable for certain tasks, they are not capable of reliably segmenting objects in cluttered environments. This is due to the fact that image features are too weak to be used in a purely bottom-up fashion. Discontinuities in intensity images are caused both by object boundaries as well as surface texture making the two extremely difficult to distinguish. Nevertheless, these algorithms serve as a foundation for more recent segmentation approaches which rely on 3D information to overcome many of the limitations of its predecessors.

One such approach is the active segmentation approach of Mishra et al. [53]. Depth and RGB cues are used to find an edge image of the scene, and objects are segmented by finding a closed contour in the edge map around a given fixation point. This method is able to segment compact objects in simple scenes. Additional segmentation cues can be obtained by projecting the depth image into three-dimensional space and analyzing the geometry of the resulting 3D pointcloud.

In [54] Richtsfeld et al. presegment the scene by fitting planes and NURBS to the pointcloud data and then merge the resulting segments into object hypotheses. The likelihood of two segments belonging to the same object is computed by training an SVM classifier on a number of appearance and geometrical features. The use of robust shape primitives in the presegmentation step allows this method to deal with compact objects in stacked configurations.

Several methods were proposed that explicitly model the assumption that objects tend to be convex in their shape. Stein et al. [55] use surface normals to oversegment the pointcloud into supervoxels and estimate their adjacency using the distance between their 3D centroids. Pairs of adjacent supervoxels are classified as either convex or concave based on the relative position and surface normal orientation difference of their centroids. Supervoxels are then merged to find components that are enclosed by convex edges. A favorable property of the algorithm is that non-convex objects tend to get divided into their convex parts. In another work Karpathy et al. find objects in 3D meshes of indoor scenes [56]. The input mesh is oversegmented using an adaptation of Felzenswalb's algorithm [49] producing a set of overlapping object hypotheses. Individual hypotheses are then classified as object or non-object based on a number of features including compactness, convexity, smoothness and symmetry. Zheng et al. proposed to reason about physical stability of objects in the scene [57]. Their method estimates the volumetric extent of segments extracted from pointcloud data and then combines them into configurations that are stable under the gravity constraint. This method is capable of correctly segmenting non-convex objects, but fails in situations when volumetric reconstruction

is unreliable.

Symmetry has been utilized as an attention mechanism to detect potential object locations in the scene. In [58] Koostra et al. generate 2D saliency maps by finding symmetric configurations of image gradients and show that these maps can be used to guide the segmentation process. Potapova et al. [59] showed that higher quality saliency maps can be obtained by extracting locally symmetric depth image patches. More recently Teo et al. [38] proposed to use symmetry both as an object detector and as a constraint in the segmentation process. Curved reflectional symmetries are detected and are used to setup a foreground segmentation that forces the segments to be symmetric with respect to the detected symmetry curves. The idea of using symmetry both as an attention and grouping mechanism is similar to our approach. However, unlike [38] which operates on single 2D images, our approach works on multiple view 3D pointclouds, which enables it to produce accurate results even in extremely cluttered scenes.

4.3 Approach

The goal of the segmentation step is to find object hypotheses i.e. subsets of points in the pointcloud that are consistent with the detected symmetries and at the same time respect the basic grouping law of convexity. To check if a point P_i of the pointcloud is consistent with a symmetry hypothesis S we analyze its reflection/rotation by the symmetry [41]. Consider the case of reflectional symmetry. We distinguish between three different options for the reflection of a point. If P is

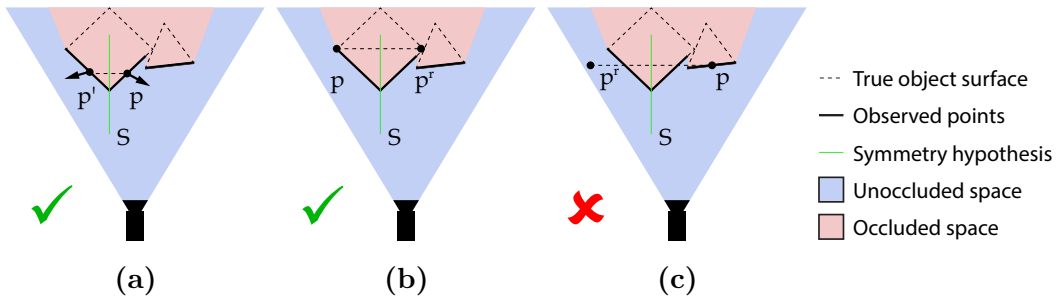


Figure 4.1: Symmetry consistency analysis. A square and a triangle are observed by the camera. p^r denotes the reflection of point p and p' denotes the symmetric neighbor of p. (a) p reflects to its symmetric neighbor. (b) p reflects to occluded space. (c) p reflects to unoccluded space. Note that all of the observed points belonging to the square and none of the points of the triangle are consistent with symmetry S.

reflected to another point of the pointcloud P', we call them symmetric neighbors and they are both considered to be supporting the symmetry S (Figure 4.1a). If on the other hand it reflects into occluded space it does not support the symmetry but is still compatible with it (Figure 4.1b). Finally if a point reflects to unoccluded space it violates the symmetry S (Figure 4.1c). Similar principles can be used for evaluating consistency with a rotational symmetry, with the difference that the point needs to be rotated around the symmetry axis instead of being reflected. To capture these properties, we define two pointwise measures: symmetry score $Sym(P)$ and occlusion score $Ocl(P)$. These scores are calculated differently for rotational and reflectional symmetry, but in both cases, the scores are normalized to lie in the $[0, 1]$ range and a lower score indicates better compatibility with a symmetry hypothesis.

To segment all of the objects in the scene we setup multiple graph based foreground segmentation problems. A graph is constructed for each of the symmetry

hypotheses where nodes correspond to the points of the pointcloud and edges are established between adjacent points of the pointcloud. Let $\mathcal{L} = \{fg, bg\}$ be labels corresponding to object and background respectively. Object hypotheses are segmented by finding a labeling f that assigns a label $f_p \in \mathcal{L}$ to all points in the pointcloud, such that the following energy functional is minimized:

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{\{p_1, p_2\} \in \mathcal{N}} V_{p_1,p_2} \cdot \delta(f_{p_1} \neq f_{p_2}) \tag{4.1}$$

where $\delta(\cdot)$ denotes an indicator function. The first term is the unary term $D_p(f_p)$ that ensures that the points consistent with the current symmetry hypothesis are labeled as foreground. V_{p_1,p_2} is the binary term defined between neighbouring points of the pointcloud. It forces segment boundaries to lie along the surface normal edges and not across flat surfaces. λ determines the influence of the binary term. In our experiments we set both this weight to 2. Once the graphs are constructed they are optimized using the graph-cuts algorithm [60]. We will now discuss how the different terms are computed for rotational and reflectional symmetries.

4.3.1 Unary term

4.3.1.1 Rotational symmetry

As discussed in section 3.5.2, the fitness score between an oriented point and a rotational symmetry S is calculated as the normalized angle between the point

normal and the plane containing the symmetry axis and the point itself:

$$Sym^{rot}(P) = \sum_i \frac{\min(\angle(S^{rot}, P_i), \alpha_{max})}{\alpha_{max}} \tag{4.2}$$

For the occlusion score, we repeatedly rotate P around the symmetry axis by θ degrees and check the distance from the rotated point to the nearest occluded voxel. The distance lookup is done efficiently using the EDT computed during the reconstruction process (see section 2.1.2). Occlusion score for a single point is calculated as the maximum of these distances:

$$Occl^{rot}(P) = \frac{\max_i(EDT(S^{rot}(P, \theta * i)))}{d_{max}}, \tag{4.3}$$

where $\theta = 30°$ and $n \in \{1, ..., 12\}$ and $d_{max} = 3cm$ is the clamping distance used for the EDT.

The unary weights are set as:

$$D_p(fg) = (1 - Sym^{rot}(P)) * (1 - Ocl(P^{rot})) \tag{4.4}$$

$$D_p(bg) = Sym(P^{rot}) + Ocl(P^{rot}) \tag{4.5}$$

These terms encode the intuition that a point belonging to the surface of a rotationally symmetric object has to satisfy both the symmetry fitness and the occlusion constraints.

4.3.1.2 Reflectional symmetry

In the case of reflectional symmetry, the symmetry fitness score is only defined for the points that have symmetric correspondences. For a point P with a symmetric correspondence P' the fitness score is calculated as the normalized angle between the reflected point normal and the corresponding point normal:

$$Sym^{refl}(P) = \frac{\min(\angle(S^{refl}(n), n'), \alpha_{max})}{\alpha_{max}} \tag{4.6}$$

The occlusion score, which is defined for all of the points in the scene, is computed by reflecting the points by the symmetry plane and checking the distance to the nearest occluded voxel:

$$Ocl^{refl}(P) = \frac{EDT(P') - d_{max}}{d_{max}} \tag{4.7}$$

For a point that has a symmetric neighbor, the unary weights are computed similarly to the rotational symmetry case (equations 4.4 and 4.5). If a point has no symmetric correspondence, its foreground weight is set to 0, while the occlusion score is used for the background weight:

$$D_p(fg) = 0 \tag{4.8}$$

$$D_p(bg) = \frac{Ocl(P, S^{refl})}{d_{max}} \tag{4.9}$$

4.3.2 Binary term

Unlike the unary term, binary term does not depend on the symmetry type, and is computed in the same way for rotational and reflectional symmetries. Point adjacency in the pointcloud $\mathcal{N}$ is estimated by connecting every point to 9 of its nearest neighbors. Binary weights are set between every pair of neighboring points. One viable option is to set the edge weights according to the angle between the surface normals of neighboring points. A smaller angle indicates that points belong to the same surface while a large angle suggests the existence of a surface normal edge between the points. This approach can be improved by modulating the weights based on the convexity criterion. This modification captures the intuition that objects tend to be convex while boundaries between two objects on top of each other tend to be concave. Given two points P_1 and P_2 we define them to be in a convex arrangement if $n_1 \cdot (p_1 - p_2) > 0$. The binary weight between them is set as:

$$V_{p_1,p_2} = \begin{cases} \exp\left(-\dfrac{n_1 \cdot n_2}{\sigma_{convex}}\right) & \text{if } n_1 \cdot (p_1 - p_2) > 0 \\ \exp\left(-\dfrac{n_1 \cdot n_2}{\sigma_{concave}}\right) & \text{otherwise} \end{cases} \tag{4.10}$$

where $\sigma_{convex} = 2$ and $\sigma_{concave} = 0.15$.

4.3.3 Hypothesis rejection

For a given symmetry, the output of our segmentation approach is a segmentation mask, each associated with the symmetry hypothesis used to generate it.

In order to reject false positive segments we employ a filtering process based on a number of geometrical "objectness" measures. These measures are used to rate each segment hypothesis on how likely is it to correspond to an object in the scene.

Symmetry score. A valid segment predicted by our segmentation approach must be symmetric. Symmetry score of a segment is calculated as the average symmetry score for all of the points of the segment:

$$Sym(H) = \frac{1}{|H|} \sum_{P_i \in H} Sym(P_i) \tag{4.11}$$

where H is the predicted segment.

Occlusion score. As discussed previously, all of the points of the segment must lie in the occluded/occupied space of the scene when rotated/reflected by the symmetry. Segment occlusion score is the average of its point-wise occlusion scores:

$$Occl(H) = \frac{1}{|H|} \sum_{P_i \in H} Ocl(P_i) \tag{4.12}$$

Smoothness score. Object hypotheses should be smooth i.e. their boundaries should not cross flat surfaces. We measure the smoothness of a hypothesis H by counting the average number of smooth links between adjacent points of the pointcloud that are broken by the segmentation. To make sure that surface normal edges do not contribute to the score we only count links for which the angle between point normals is smaller than a threshold. Let $\mathcal{B}_H = \{P_i, P_j \mid P_i \in H, P_j \in \mathcal{P}/H, \{P_i, P_j\} \in \mathcal{N}\}$ be the set of links between neighboring points in the pointcloud

that were broken by the object hypothesis H. The smoothness score is calculated as:

$$Smooth(H) = \frac{1}{|H|} \sum_{\{P_i, P_j\} \in \mathcal{B}_H} \delta(\angle(n_i, n_j) < \alpha_{smooth}) \tag{4.13}$$

Object hypotheses that satisfy $Sym(H) < s_{sym}$, $Occl(H) < s_{occl}$ and $Smooth(H) < s_{smooth}$ are considered valid. The rest of the hypotheses are rejected. By changing these thresholds, our algorithm can be tuned to achieve a necessary precision-recall trade-off.

4.4 Pipeline

In this section we present an approach to combining the symmetry detection methods described in Chapter 3 with the segmentation method described above, into a complete pipeline for segmenting rotationally and reflectional symmetrical objects in 3D reconstructions of cluttered scenes. Figure 4.2 shows the flowchart of the system. It consists of two sequential streams: one for rotational and the other for reflectional symmetries. Given an input pointcloud and an EDT of the scene, rotational objects are detected first. Next the points belonging to the detected rotational object segments are removed from the input pointcloud, before passing it to the reflectional symmetry stage. By doing this, we avoid the previously discussed problem of detecting reflectional symmetries on rotationally symmetric surfaces (Section 3.6.2). It has an added benefit of speeding up the reflectional symmetry stage of the pipeline, since it is operating on the smaller pointcloud. After

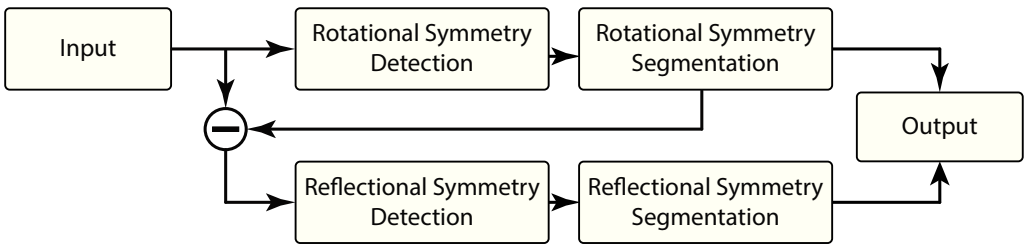


Figure 4.2: System flowchart.

segmenting both rotational and reflectonal objects we combine the outputs from both stages. Since some of the objects in the scene may have multiple symmetries (e.g. a box), the segments generated for any of these symmetries should be identical. To handle these cases, we merge the segments for which the intersection over union is greater than 95% and concatenate the symmetries used to generate them. This allows our pipeline to return segments associated with multiple symmetries.

4.5 Experiments

We evaluate our approach on the scenes from the challenging Cluttered Tabletop Dataset. We investigate its ability to predict accurate segmentation masks as well as to detect object symmetries. We evaluate three versions of our approach. The first version uses the complete pipeline with surface-based approaches used for rotational and reflectional symmetry detection (*Rot + Refl (surface)*). To investigate the importance of modeling rotational symmetries, we test two versions of our pipeline that only use reflectional symmetry to segment the scenes. The two versions use surface-based (*Refl (surface)*) and curve-based (*Refl (curve)*) approaches for the symmetry detection stage.

The performance of our pipeline directly depends on the quality of the detected symmetries. If none of the symmetries of an object are detected, the segmentation stage will not be able to segment it. On the other hand, if an incorrect symmetry is supplied to the segmentation stage, there is a good chance it will be rejected by the segment hypothesis filtering process. Thus we use a set relaxed filtering parameters for the symmetry detection stages, that maximize recall of the detected symmetries at the cost of precision. The values of the parameters used are shown in table 4.1.

Method	Parameter	Value
Curve-based reflectional	M_S	7°
	n_{inlier}	0.3
Surface-based rotational	Sym^{rot}	0.02
	Per^{rot}	0.6
Curve-based reflectional	Sym^{refl}	0.8
	$Inlier^{refl}$	0.3

Table 4.1: Filtering parameters used for the symmetry detection stages in the evaluation of our pipeline.

4.5.1 Segmentation evaluation

Unlike the majority of modern segmentation algorithms which assign a single object label to every point in the pointcloud, our method returns a set of object segments that can be spatially overlapping. Moreover, some points might not get assigned to any segment at all. We believe that this property does not reflect negatively on our approach. We argue that the utility of a segmentation process for a robotic system should be measured not by the number of points in the scene

that were assigned a correct object label, but by the number of objects in the scene for which sufficiently accurate segmentation masks were returned. This principle motivates our choice of the evaluation metric. An object in the scene is considered to be segmented correctly if there exists a predicted segment that overlaps sufficiently with its ground truth segmentation mask (intersection over union metric is used to compute the overlap). Recall is calculated as the fraction of the objects in the dataset that were segmented correctly and precision as the number of correctly segmented objects normalized by the total number of returned segments. Denoting by $\mathbf{H}^T$ the set of ground truth masks for all objects in all of the scenes of a dataset and by $\mathbf{H}^P$ the set of predicted object hypotheses:

$$P = \frac{1}{|\mathbf{H}^P|} \sum_{H_t \in \mathbf{H}^T} \delta(\max_{H_p \in \mathbf{H}^P} O(H_p, H_t) > \sigma_o) \tag{4.14}$$

$$R = \frac{1}{|\mathbf{H}^T|} \sum_{H_t \in \mathbf{H}^T} \delta(\max_{H_p \in \mathbf{H}^P} O(H_p, H_t) > \sigma_o) \tag{4.15}$$

where

$$O(H_1, H_2) = \frac{H_1 \cap H_2}{H_1 \cup H_2} \tag{4.16}$$

and σ_o is the desired overlap score. In our evaluation we use the value of $\sigma_o = 0.9$. We choose such a high overlap requirement since small errors in segmentation can lead to significant errors in later stages of the processing pipeline of a robot (i.e. grasp selection and path planning).

In addition to the three versions of our pipeline we evaluate two bottom-up

segmentation approaches that rely on convexity as the main grouping principle. The first one is the popular *Felzenswalb* graph segmentation algorithm adapted to point-cloud data [56]. The second is the *LCCP* [55] that segments the scene by merging supervoxels found in the scene. The meta-parameters used for these algorithms as well as the segmentation stages in out pipeline are described in Table 4.2.

Method	Parameter	min	max	step
Reflectional segmentation	Sym^{refl} Ocl^{refl}	0.1 0.005	0.6 0.045	0.1 0.01
Rotational segmentation	Sym^{rot} Ocl^{rot}	0.01 0.01	0.03 0.015	0.01 0.005
Felzenswalb	k	0.5	7.0	0.5
LCCP	β_{thresh}	10°	70°	5°

Table 4.2: Parameters used for evaluating symmetry detection. Parameters were varied from *min* value to *max* value in steps of size *step*.

The PR curves for the evaluated algorithms are shown on figure Figure 4.3. *LCCP* and *Felzenswalb* achieve a similar performance with maximum F-measures of 31% and 33% respectively. Both methods are significantly outperformed by the symmetry-based methods. This shows that local convexity alone is not sufficient for accurate object segmentation in highly cluttered environments. Looking at results in Figure 4.4, we can identify two distinct types of object configurations where *Felzenswalb* and *LCCP* methods fail:

1. touching objects where the transition between objects is not concave are merged together. In row 1 transition between the speaker and the coffee box is non-concave.

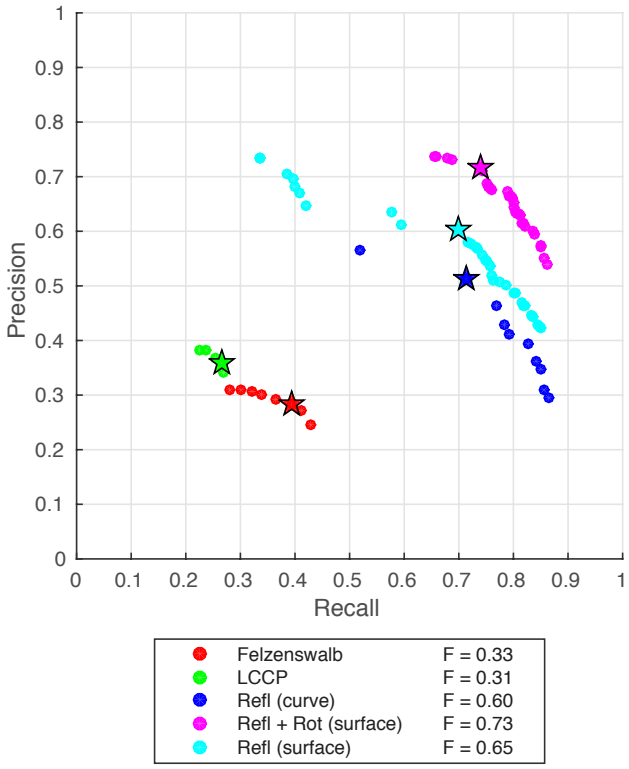


Figure 4.3: PR curves for segmentation evaluation. Stars mark the point of maximum F-measure.

2. objects separated by an occlusion boundary are oversegmented. In row 2 the box is occluded by the milk carton.

In both of these cases convexity criterion is not sufficient to correctly group the points into objects. On the other hand symmetry provides a global object-level grouping principle, which allows our method to correctly segment these scenes.

Among the symmetry-based methods *Refl (surface)*, our complete pipeline with rotational and reflectional symmetry stages, achieves the highest f-score of 73%. Approaches relying solely on the reflectional symmetry achieve similar recall, but have a lower precision. This can be attributed to two reasons. Approaches that do

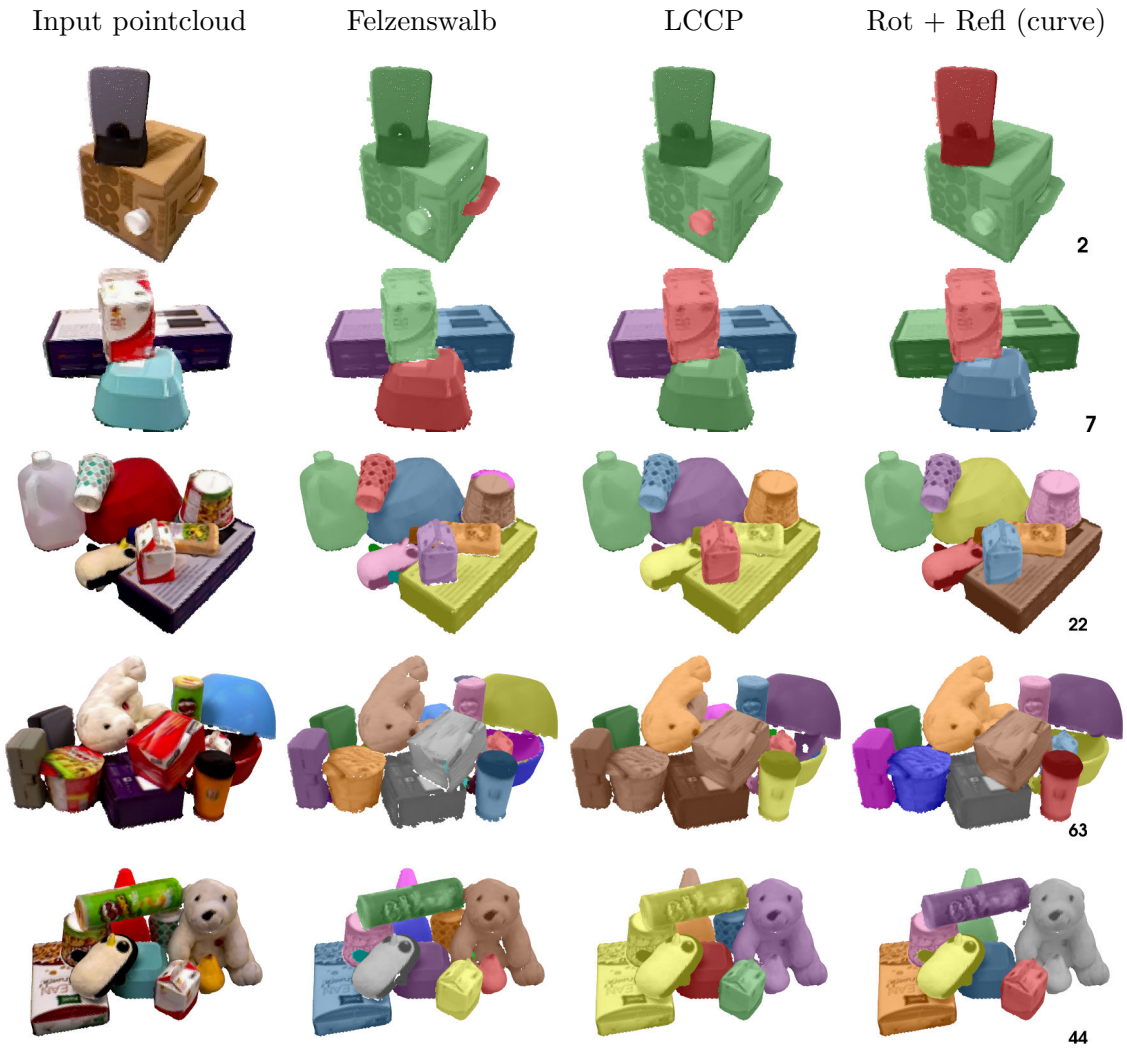


Figure 4.4: Comparison of segmentation results on individual scenes. For each method best segments were automatically selected by choosing the predicted segments that have the highest overlap with ground truth masks.

not explicitly model rotational symmetry can only segment rotational objects if they happen to detect a reflectional symmetry that goes through the rotational symmetry axis. Even if such symmetry is detected, reflectional segmentation imposes fewer constraints on the segment, compared to a rotational method. As a result it is less likely to segment it correctly. Additional segmentation errors arise if the axis of a rotationally symmetric objects lies in the symmetry plane of a different objects. The two will be merged together by a reflectional segmentation approach. Finally, *Refl (surface)* achieves an f-score of 65% outperforming *Refl (curve)* with 60%. The lower performance can be explained by the fact that *Refl (curve)* uses an inferior symmetry detection stage.

4.5.2 Symmetry evaluation

We investigate the symmetry detection performance of our pipeline *Refl + Rot (surface)*. For each of the scenes, symmetries associated with the returned segments are concatenated together and are treated as outputs of a symmetry detection approach. The procedure described in section 3.6.1 is used for evaluation. Figure 4.5 shows the PR curves for the reflectional and rotational symmetry detection results after the segmentation process. The results for surface-based symmetry detection approaches (without the segmentation step) are shown shown for comparison.

Rotational symmetries are detected very reliably with an f-score of 89%, while reflectional symmetries get a more modest score of 63%. For both symmetry types, our pipeline significantly outperforms the raw surface-based approaches. This con-

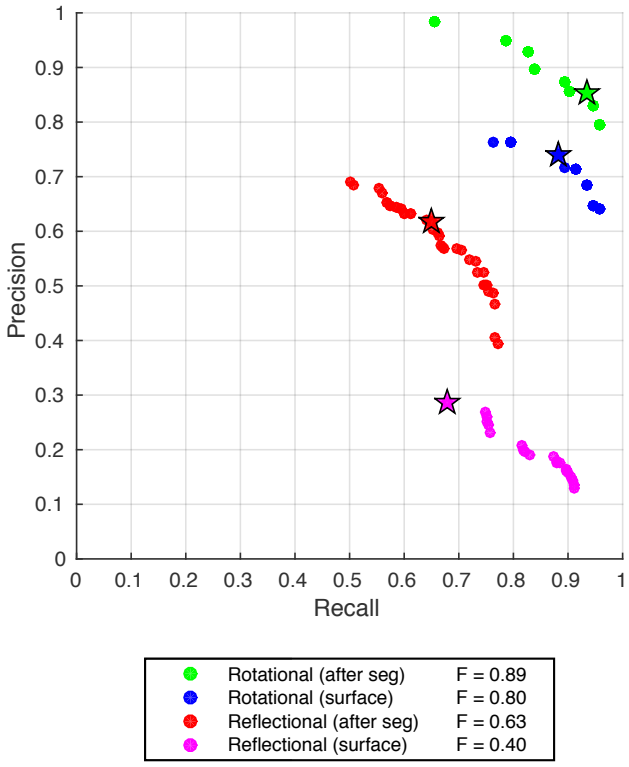


Figure 4.5: PR curves for symmetry detection results for the complete pipeline. Stars mark the point of maximum F-measure.

firms our hypothesis, that segmentation can be viewed as a reliable filtering stage for the detected symmetries. The performance gap between pre and post segmentation results is particularly notable for reflectional symmetries - an increase of 23%. This is due to the fact that our pipeline removes rotationally symmetric parts of the scene before detecting reflectional symmetries. This has an effect of "decluttering" the scene, making further processing easier.

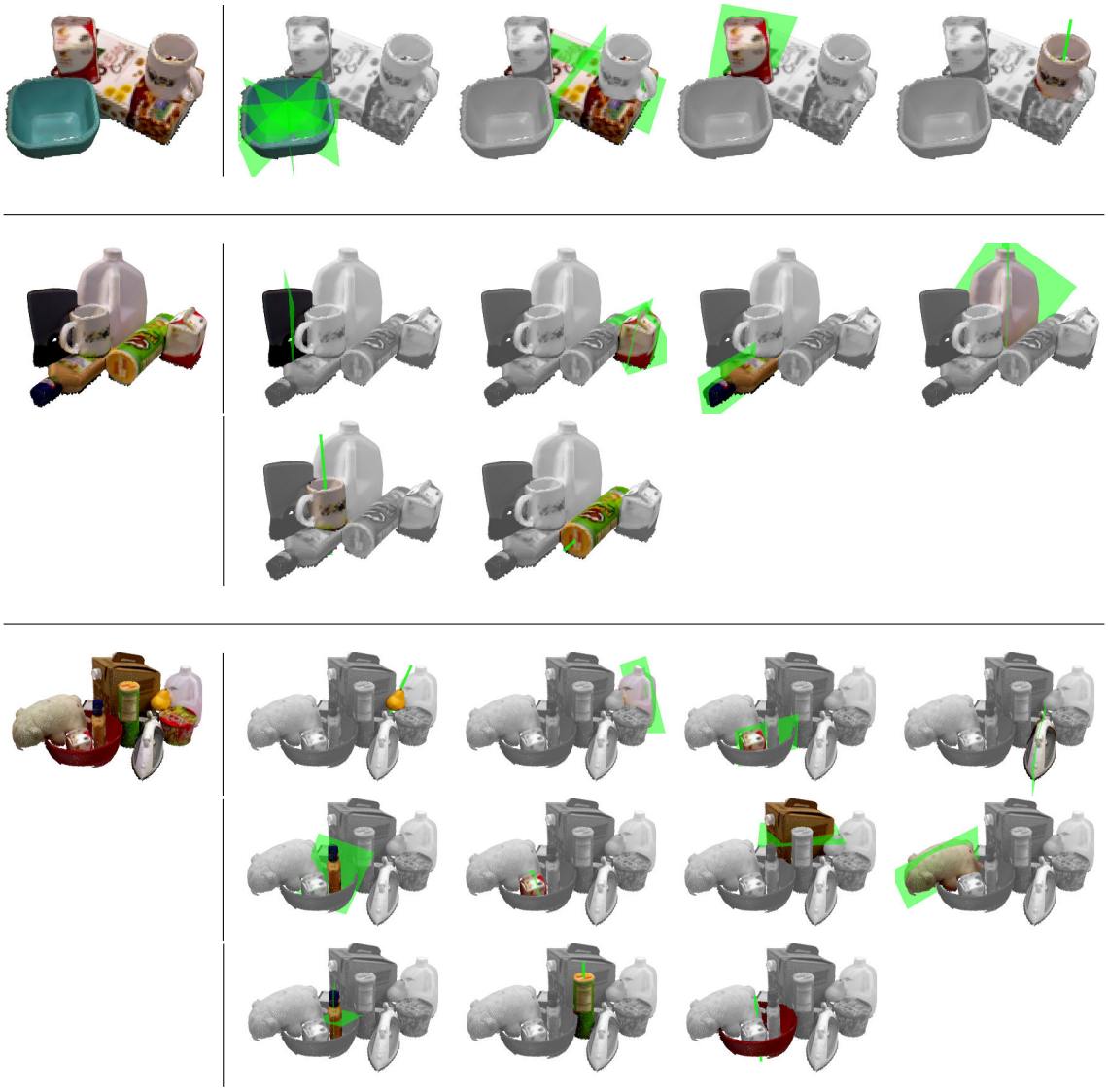


Figure 4.6: Output of our complete pipeline. First image in a row shows the input pointcloud. Following images show the detected objects and their symmetries.

4.5.3 Runtime analysis

The runtime of our complete pipeline (Section 4.4) depends on the number and of the objects present in the scene, the complexity of their individual shapes as well as on their arrangement. The computational complexity of the symmetry detection (both rotational and reflectional) for a single segment is bounded by the performance of the Levenberg-Marquardt optimization, which has the worst-case computational complexity of $\mathcal{O}(n^3)$ where n is the number of points in a pointcloud. Symmetry detection algorithms have to be run on every segment returned by the smooth surface segmentation process. The symmetry segmentation steps have the same complexity of $\mathcal{O}(n^3)$ due to the use graph-cuts algorithm. A graph-cuts algorithm has to be run for every symmetry detected during the symmetry detection stage.

Table 4.3 shows the average computational times for the different stages of our pipeline. Performance were measured on a modern Intel i7 CPU. Scenes consisting of 4 or less objects required 1.5 seconds, while more complex scenes took 8.4 seconds to process.

Pipeline stage	4 or less objects	more than 10 objects
Rotational stage	0.3 sec	0.9 sec
Reflectional stage	1.2 sec	7.5 sec
Total	1.5 sec	8.4 sec

Table 4.3: Average runtimes for different stages of our pipeline for the scenes from the Cluttered Tabletop Dataset.

4.6 Object grasping by a mobile robot

In this section we describe an application of a simplified version of our perception pipeline in the context of robot manipulation. A mobile robot was tasked with picking dishes such as bowl mugs and plates, from a table in a kitchen scenario. The shape of the graspable objects was modeled as cylinders. This model is generalizes well to a large amount of dishes and can be used by the robot to pick up previously unseen objects.

4.6.1 Robot platform

Our robot consisted of a dual arm Baxter collaborative robot mounted on an omnidirectional mobile base. The right arm was equipped with a ReFlex 3-fingered gripper and was used for object grasping. An Asus Xtio Pro RGB-D sensor was used as the main vision sensor. It was mounted on the left arm, such that it's position could be controlled to get a better view of the scene. The pose of the camera relative to the arm was estimated by running stereo calibration on the RGB images from the depth sensor and the RGB camera installed in the wrist of robot's arm. The Robot Operating System (ROS) [61] was used to program and control the robot.

4.6.2 Perception pipeline

The perception pipeline consisted of two main stages: reconstruction and detection. To reconstruct the scene multiple RGB-D frames were captured by moving the depth sensor in a sequence of predefined arm poses. The poses were chosen such

that the table and its objects would be clearly visible to the camera. Due to the low precision of the motor encoders of the Baxter robot as well errors in camera to robot calibration, the camera pose available through the ROS transform tree was too noisy to be used for reconstruction purposes. Instead, a purely computer vision approach was used. SIFT feature correspondences were extracted from the RGB-D frames an SolvePNP was used to find camera poses. The pointclouds from each of the frames were then stitched together using ICP.

Next, object pointcloud was acquired, by detecting the table plane and extracting points lying above it. The resulting pointcloud was then segmented using Euclidean Clustering. A symmetry axis was then fitted to every segment using the approach described in Section 3.5.2. Finally, the parameters of the bounding cylinders were estimated for every segment for which a valid rotational symmetry was detected. Given a segment and a symmetry axis, the radius of the cylinder was found as the maximum distance between the detected symmetry axis and any point of the segment. Cylinder bounding planes were found by projecting segment points onto the axis and finding the bounding points.

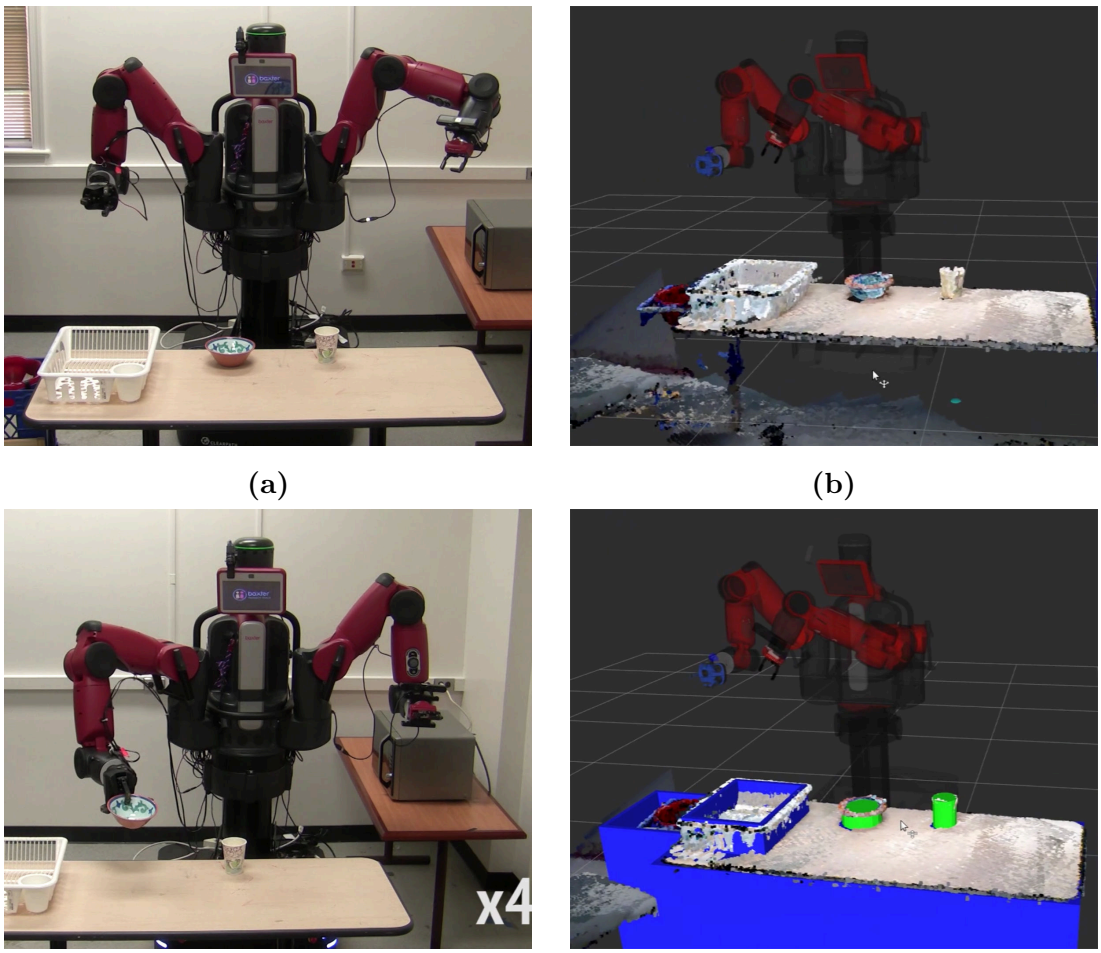
4.6.3 Grasping

Given the parameters of a bounding cylinder a set of candidate grasps were proposed. The grasps were generated such that the end effector would envelop the object around the symmetry axis while respecting the size of the cylinder. The final grasp was chosen based on the reachability analysis. Once the desired pose of the

end effector was selected, arm motion was planned using the RRT planner.

4.6.4 Results

Figure 4.7 shows the multiple stages of the perception and grasping processes. In our experiments the robot was able to reliably pick objects placed in different poses on the table. The perception pipeline was found to be very robust and general enough to handle multiple scenarios that it was not specifically programmed for.



(a) **(b)**

(c) **(d)**

Figure 4.7: Robot grasping application. (a) Robot reconstructs the scene by moving the depth sensor mounted on the left arm. (b) The reconstructed view of the table. (c) Robot grasping the bowl (d) Cylinders fitted to the reconstruction shown in green.

4.7 Final Conclusions and Outlook

In this thesis we have investigated the use of rotational and reflectional symmetries for delivering perceptual constructs required for unconstrained robot object manipulation. We have presented a complete pipeline for segmenting symmetric objects in 3D reconstructions. Our pipeline uses symmetry first as an attention operator, that provides seeds for the segmentation process and then as a grouping principle to segment individual objects. This tandem use of symmetry borrows inspiration from the studies of human visual system. In our experiments we have demonstrated the ability of our approach to deliver high quality segmentations and symmetry detections of objects in extremely cluttered scenes. This confirms our hypothesis that symmetry imposes a strong constraint on the three dimensional shape of the objects, which can and should be exploited by perceptual systems. Finally we have demonstrated how a system similar to ours can be deployed on a mobile robot.

There are several avenues for improvement of our work. We have observed that rotational symmetry, while more specialized, imposes a stronger constraint on object shape. Modeling additional "combinatorial" symmetry classes that correspond to a more limited, yet common set of object, classes could improve the performance of our system even further. An example is a set of 3 perpendicular planes which correspond to box-like objects. It would be interesting to see if the sound geometrical principles used in our pipeline can be encoded in a more advanced inference framework such as a Deep Convolutional Neural Network. This has the potential of automating some

of the parameter learning steps required to fine-tune out system. Additionally it can significantly reduce the computational load of the system making it applicable to real-time robotics. Finally, to make our system truly autonomous we would need to automate the data collection process. Symmetry could be used to guide this process by providing cues for selecting the next viewpoint used to observe the scene.

While fully unconstrained manipulation remains an elusive task, we believe that the approaches demonstrated in this thesis bring us closer to that goal.

Bibliography

[1] Matthias Sebastian Treder. Behind the looking-glass: A review on human symmetry perception. *Symmetry*, 2(3):1510–1543, 2010.

[2] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[3] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3936–3943. IEEE, 2014.

[4] Aitor Aldoma, Federico Tombari, Johann Prankl, Andreas Richtsfeld, Luigi Di Stefano, and Markus Vincze. Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2104–2111. IEEE, 2013.

[5] Federico Tombari and Luigi Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*, pages 349–355. IEEE, 2010.

[6] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012*, pages 548–562. Springer, 2013.

[7] Karl Pauwels and Danica Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1300–1307. IEEE, 2015.

[8] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1–6. IEEE, 2009.

[9] Lucian Cosmin Goron, Zoltan-Csaba Marton, Gheorghe Lazea, and Michael Beetz. Robustly segmenting cylindrical and box-like objects in cluttered scenes using depth cameras. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6. VDE, 2012.

[10] Kester Duncan, Sudeep Sarkar, Redwan Alqasemi, and Rajiv Dubey. Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4238–4243. IEEE, 2013.

[11] Ana Huamán Quispe, Benoît Milville, Marco A Gutiérrez, Can Erdogan, Mike Stilman, Henrik Christensen, and Heni Ben Amor. Exploiting symmetries and extrusions for grasping household objects. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3702–3708. IEEE, 2015.

[12] Franc Solina and Ruzena 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.

[13] Juan D Delius and Brigitte Nowak. Visual symmetry recognition by pigeons. *Psychological research*, 44(3):199–212, 1982.

[14] Martin Giurfa, Birgit Eichmann, and Randolf Menzel. Symmetry perception in an insect. *Nature*, 382(6590):458, 1996.

[15] Jon Driver, Gordon C Baylis, and Robert D Rafal. Preserved figure-ground segregation and symmetry perception in visual neglect. *Nature*, 360(6399):73–75, 1992.

[16] Marco Bertamini, Jay D Friedenberg, and Michael Kubovy. Detection of symmetry and perceptual organization: The way a lock-and-key process works. *Acta psychologica*, 95(2):119–140, 1997.

[17] Gert Kootstra and Lambert RB Schomaker. Prediction of human eye fixations using symmetry. In *The 31st Annual Conference of the Cognitive Science Society (CogSci09)*, pages 56–61. Cognitive Science Society, 2009.

[18] Jeannette Bohg, Matthew Johnson-Roberson, Beatriz León, Javier Felip, Xavi Gratal, N Bergstrom, Danica Kragic, and Antonio Morales. Mind the gap-robotic grasping under incomplete observation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 686–693. IEEE, 2011.

[19] Yiannis Aloimonos. *Active perception*. Psychology Press, 2013.

[20] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[21] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[22] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[23] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[24] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[25] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[26] Cuda helper library for vision. https://github.com/arpg/Kangaroo, (accessed July 5, 2017).

[27] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1154–1160. IEEE, 2011.

[28] Alex Teichman, Stephen Miller, and Sebastian Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and Systems*, 2013.

[29] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[30] An efficient probabilistic 3d mapping framework based on octrees. https://github.com/OctoMap/octomap, (accessed July 5, 2017).

[31] Ehud Barnea and Ohad Ben-Shahar. Depth based object detection from partial pose estimation of symmetric objects. In *European Conference on Computer Vision*, pages 377–390. Springer, 2014.

[32] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(3):549–559, 2006.

[33] Pablo Speciale, Martin R Oswald, Andrea Cohen, and Marc Pollefeys. A symmetry prior for convex variational 3d reconstruction. In *European Conference on Computer Vision*, pages 313–328. Springer, 2016.

[34] Daniel Reisfeld, Haim Wolfson, and Yehezkel Yeshurun. Context-free attentional operators: the generalized symmetry transform. *International Journal of Computer Vision*, 14(2):119–130, 1995.

[35] Gareth Loy and Jan-Olof Eklundh. Detecting symmetry and symmetric constellations of features. In *Computer Vision–ECCV 2006*, pages 508–521. Springer, 2006.

[36] H. Fu, X. Cao, Z. Tu, and D. Lin. Symmetry constraint for foreground extraction. *IEEE Trans. on Systems, Man and Cybernetics*, 44(5), 2014.

[37] Tammy Riklin-Raviv, Nahum Kiryati, and Nir Sochen. Segmentation by level sets and symmetry. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1015–1022. IEEE, 2006.

[38] Ching L Teo, Cornelia Fermuller, and Yiannis Aloimonos. Detection and segmentation of 2d curved reflection symmetric structures. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1644–1652, 2015.

[39] Changming Sun and Jamie Sherrah. 3d symmetry detection using the extended gaussian image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):164–168, 1997.

[40] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[41] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1824–1831. IEEE, 2005.

[42] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, 2007.

[43] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2027–2034. IEEE, 2013.

[44] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[45] Frank van den Heuvel Rabbani, Tahir and G. Vosselmann. Segmentation of point clouds using smoothness constraint. pages 248–253, 2006.

[46] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[47] Jingchen Liu, George Slota, Gang Zheng, Zhaohui Wu, Minwoo Park, Seungkyu Lee, Ingmar Rauschert, and Yanxi Liu. Symmetry detection from realworld images competition 2013: Summary and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013.

[48] Max Wertheimer. Laws of organization in perceptual forms. 1938.

[49] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[50] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[51] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[52] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[53] Ajay Mishra, Yiannis Aloimonos, and Cornelia Fermuller. Active segmentation for robotics. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3133–3139. IEEE, 2009.

[54] Andreas Richtsfeld, Thomas Morwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796. IEEE, 2012.

[55] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 304–311. IEEE, 2014.

[56] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2088–2095. IEEE, 2013.

[57] Bo Zheng, Yibiao Zhao, Joey C Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3127–3134. IEEE, 2013.

[58] Gert Kootstra, N Bergstrom, and Danica Kragic. Fast and automatic detection and segmentation of unknown objects. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 442–447. IEEE, 2010.

[59] Ekaterina Potapova, Karthik Mahesh Varadarajan, Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Attention-driven object detection and segmentation of cluttered table scenes using 2.5 d symmetry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4946–4952. IEEE, 2014.

[60] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[61] Morgan Quigley, Josh Faust, Tully Foote, and Jeremy Leibs. Ros: an open-source robot operating system.