SURFACE-BASED 3D OBJECT MODELING, DETECTION, AND POSE ESTIMATION IN CLUTTERED ENVIRONMENTS FOR ROBOTIC MANIPULATION

by

Zhou Teng

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2016

Approved by:

____

Dr. Jing Xiao

____

Dr. Srinivas Akella

____

Dr. Jianping Fan

____

Dr. Min Shin

____

Dr. Andrew Willis

©2016
Zhou Teng
ALL RIGHTS RESERVED

ABSTRACT

ZHOU TENG. Surface-based 3D object modeling, detection, and pose estimation in cluttered environments for robotic manipulation. (Under the direction of DR. JING XIAO)

Automatic identification and pose estimation of a target object through robotic perception is necessary for many robotic tasks, such as object manipulation, scene reconstruction and view planning. However, general object recognition and pose estimation in cluttered 3D environments is still an unsolved and challenging problem, due to occlusions, complicated background, and great variation in object appearance caused by different illuminations or viewpoints.

In this dissertation, an appearance-based approach to general 3D object detection and pose estimation is introduced based on segmented 3D surfaces and their features, taking full advantage of RGB-D information. Our approach can detect and estimate the poses of occluded objects effectively, including occluded multiple instances of the same object, in cluttered environments. The results of the detected objects and their poses with respect to the current RGB-D camera frame can be used directly for robotic manipulation, and the reconstructed 3D scene can also be used directly for robot motion planning.

Leveraging the scene reconstruction results of our surface-based approach, a learning-based approach is further developed for evaluating scene recognizability of a cluttered scene with objects occluding one another from a single view and ranking views based on their scene recognizability. Our approach of evaluating scene recognizability provides a more accurate assessment of how good a view is for autonomous robotic tasks in cluttered environments than conventional evaluation based only on object visibility.

Last but not the least, this dissertation also explores interleaving RGB-D perception and robotic manipulation for automatic modeling and handling of unknown objects. Using a fixed RGB-D camera and starting from the first view of the object, our approach gradually builds and extends a partial model (based on what has been visible) into a complete object model. In the process, the partial model is also used to guide a robot manipulator to change the pose of the object to make more surfaces visible for continued model building. The alternation of perception-based model building and pose changing continues until a complete object model is built with all object surfaces covered. Our approach provides more flexibility to enable observing all object surfaces and building a complete object model, and can be further developed to facilitate manipulation of unknown objects in cluttered environments.

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor Prof. Jing Xiao for her continuous support of my Ph.D. study and related research, her patience, inspiration, enthusiasm, and immense knowledge in robotics science, which, taken together, make her a great mentor. Her guidance helped me through all the time of research and writing of this dissertation.

Besides my advisor, I would like to thank the rest of my Ph.D. dissertation committee: Prof. Srinivas Akella, Prof. Jianping Fan, Prof. Min Shin and Prof. Andrew Willis, for their insightful comments and encouragement, but also for the hard questions, which inspired me to broaden my research horizons.

My sincere thanks also go to Dr. Thomas Fuhlbrigge, Dr. Carlos Martinez, Dr. Remus Boca, Dr. Biao Zhang and Dr. Carlos Morato at ABB, for providing me the opportunity to join their group as a summer intern and offering me full support during my writing of this dissertation there.

I thank my fellow labmates at UNC Charlotte: Jinglin Li and Huitan Mao for excellent research collaboration, Sterling McLeod, Junjie Shan, Ning Zhou, and Zhiqiang Ma, for all the stimulating discussions, and everyone in the lab, for all the fun we had in the last five years. I also thank my friends: Rongcheng Lin, Qin Jie, Le Lu, Peng Wan, and others, for all the happy moments we shared together.

Last but not the least, I would like to thank my family: my mother Yunfeng Chen, my father Quanfang Teng, and my sister Juan Teng, for their love and support of me throughout this dissertation research.

TABLE OF CONTENTS

LIST OF FIGURES xi

LIST OF TABLES xvii

CHAPTER 1: INTRODUCTION 1

1.1. Robotic Perception 1

1.2. Existing Sensing Technologies 3

1.3. Computer Vision vs. Robotic Perception 4

1.4. View Planning for Robotic Perception 7

CHAPTER 2: RELATED LITERATURE 9

2.1. Object Visual Representation 9

2.2. Learning Algorithms 11

2.3. Keypoints Matching and Transformation Estimation 12

2.4. Challenges in Robotic Perception 13

2.4.1. Object Detection and Pose Estimation 14

2.4.2. View Evaluation 19

2.5. Combining Perception and Manipulation 20

2.5.1. Perception Incorporated with Manipulation 21

2.5.2. Automatic 3D Object Modeling 22

2.5.3. Grasp Strategies of Objects 25

CHAPTER 3: RESEARCH OBJECTIVES AND APPROACHES 26

3.1. Problem Description and Assumptions 26

3.2. Approaches 26

3.2.1. Surface-based Approach to Object Detection and Pose Estimation 27

3.2.2. Learning-based View Evaluation 31

3.2.3. Interleaving Perception and Robotic Manipulation 33

CHAPTER 4: SURFACE-BASED 3D OBJECT DETECTION AND POSE ESTIMATION 37

4.1. Object Representation and Model Building 37

4.2. Object Detection and Pose Estimation 39

4.3. Comparison of Two Types of Keypoints Matching 41

4.4. Detection of Multiple Instances of the Same Object 45

4.5. Experiments and Analyses 45

4.6. Conclusions of this Chapter 50

CHAPTER 5: EXTENSION TO SURFACE-BASED 3D OBJECT DETECTION AND POSE ESTIMATION 51

5.1. A Robust and Convenient Strategy for 3D Model Building 51

5.2. Surface-based Object Representation and Classification 55

5.3. Object Detection and Pose Estimation Using Graph-cut 57

5.3.1. Initial Detection and Estimation 57

5.3.2. Scene Reconstruction 61

5.4. More Experiments and Analyses 65

5.4.1. Training 66

5.4.2. Test Scenes and Images 66

5.4.3. Comparison of the Test Results: Surface Segments vs. Object Instances 68

5.4.4. Example Results of Scene Reconstruction and Discussion of Performance 71

5.4.5. Comparison to the MOPED Approach 74

5.5. Conclusions of this Chapter 80

CHAPTER 6: LEARNING-BASED VIEW EVALUATION 81

6.1. Evaluation Factors 81

6.1.1. Recognition Confidence 82

6.1.2. Fitness Cost 84

6.1.3. Recognizability of Object Instance 84

6.1.4. Recognition Rate and Uncertainty 91

6.1.5. Background Recognizability 91

6.2. View Evaluation 92

6.3. Experiments and Analyses 94

6.3.1. Training of View Evaluator 95

6.3.2. Test of View Evaluator 97

6.4. Conclusions of this Chapter 100

CHAPTER 7: OBJECT MODELING THROUGH PERCEPTION AND MANIPULATION 102

7.1. Aspect Graph of Object Poses 103

7.2. Image Registration in Perception 107

7.2.1. Pairwise Registration of Neighboring Object Point Clouds 107

7.2.2. Global Optimization of Pairwise Registration Results 109

7.2.3. Registration of Partial Object Models from Different Rotation Loops 114

7.3. Manipulation for Perception 114

7.3.1. Pushing 114

7.3.2. Change of Object Support Surfaces 115

7.4. Experiments and Analyses 124

7.4.1. Set Up of Our Experiments 125

7.4.2. Experiments 125

7.4.3. Comparison of the Registration Results: before and after Optimization 128

7.4.4. Analyses of Voxel Maps 133

7.5. Conclusions of this Chapter 134

CHAPTER 8: CONCLUSIONS AND FUTURE WORK 138

8.1. Object Detection and Pose Estimation 138

8.1.1. Contributions 138

8.1.2. Future Work 139

8.2. Learning-based View Evaluation 141

8.2.1. Contributions 141

8.2.2. Future Work 142

8.3. Object Modeling through Perception and Manipulation 142

8.3.1. Contributions 142

8.3.2. Future Work 143

REFERENCES 146

LIST OF FIGURES

FIGURE 1: A typical scene of an autonomous robotic manipulation problem. Coordinate systems (CSs) are introduced to describe the relations among the target objects, the robot, and the world. 2

FIGURE 2: An overview of different aspects in solving computer vision problems. The figure is revised based on similar figures in the surveys [118] and [30]. 5

FIGURE 3: An overview of different aspects in solving robotic perception problems. The figure is revised based on similar figures in the surveys [118] and [30]. 7

FIGURE 4: An overview of the procedure of generating high-level object visual representations from low-level image features. The figure is revised based on similar figures in the surveys [118] and [30]. 10

FIGURE 5: High-level flowchart of our automatic 3D object modeling system 36

FIGURE 6: A common cereal box and images of its smooth 3D surfaces from different views 38

FIGURE 7: Examples of reconstructed objects, from left to right: cereal box, cup noodle, milk box 1 and milk box 2 39

FIGURE 8: An example of handling inconsistent pose estimates: (a) original scene captured by a RGB-D camera, (b) intersecting projections of the model for the blue milk box at two inconsistent, initial object pose estimates, (c) object models at the best-fit pose estimates 41

FIGURE 9: Comparison of keypoints matching and pose estimation results: (a) the original images for matching, (b) the matched keypoints and the result of estimated object model based on matching the whole images, and (c) the matched keypoints and the result of the estimated object model based on matching the common side surface visible in both images 42

FIGURE 10: Comparison of keypoints matching and pose estimation results: (a) the original images for matching, (b) the matched keypoints based on the whole images and the result of estimated object model displayed in front and top views, (c) the matched keypoints based on matching the common top side surface visible in both images and the result of the estimated object model displayed in front and top views 44

FIGURE 11: Examples of images in the dataset: top two rows show some images of the four objects used for training; bottom row shows some background images for training 46

FIGURE 12: Examples of object detection and pose estimation results: each row starting with an original test image followed by four images from left, back, right, and top views that display the corresponding reconstructed scene 47

FIGURE 13: The precision-*k* curves for all the objects in the dataset 48

FIGURE 14: The recall-*k* curves for all the objects in the dataset 49

FIGURE 15: Two landmarks are placed by the side of a milk box to register different object views 52

FIGURE 16: The geometric shape of an object, no matter how complicated, can always be simplified as having up to 6 sides. 52

FIGURE 17: 6 different stable poses used for modelling a juice container 53

FIGURE 18: The whole space is roughly divided into 5 even spaces, and for each of them, an RGB-D image is captured for training. 53

FIGURE 19: 5 images from different views captured for a single pose of a cereal box 54

FIGURE 20: All the reconstructed object models in our current dataset, including objects with transparent or translucent surfaces (shown in the last column) 56

FIGURE 21: A common cereal box and images of its smooth 3D surfaces from different views of different poses 58

FIGURE 22: An example of selecting the best object model using a graph-cut based algorithm: in (b) and (c), both surface segments C and F predict the same object model of the blue milk box with very similar pose estimates, while surface segments D and E also predict the same object model of the red milk box but with inconsistent pose estimates. After applying the result of our graph-cut algorithm, we can see that the previous red milk box with inaccurate pose estimate in (c) is eliminated in (f). 62

FIGURE 23: Optimal accuracies achieved by the trained SVMs classifiers using different descriptions of visual signatures as the number of folds K changes in cross validation for different k values 67

FIGURE 24: Test images of different scenes 68

FIGURE 25: Test images from various views of the same scene 68

FIGURE 26: Precision-recall curves based on classifying all test surface segments to object labels in all test images and for all objects in our dataset 69

FIGURE 27: Examples of object detection and pose estimation results: each row starting with an original test image followed by four images from front, left, right, and top views that display the corresponding reconstructed scene 72

FIGURE 28: Keypoint matching results: keypoints matched to the training keypoints on the same object are displayed with the same color. 78

FIGURE 29: Incorrect pose estimation of *coffee can 2* caused by local keypoint mismatching in the MOPED approach using Projection clustering with Q-Score ranking: (a) the original scene, (b) the keypoint matching results, (c) the test results with the MOPED approach, and (d) the test results with our approach 79

FIGURE 30: Failure of detecting *coffee can 2* caused by local keypoint mismatching in the MOPED approach using Projection clustering with Q-Score ranking: (a) the original scene, (b) the keypoint matching results, (c) the test results with the MOPED approach, and (d) the test results with our approach 79

FIGURE 31: An example of object recognition and pose estimation by [107]: the left image is the original view captured by a Microsoft Kinect; the right image is the reconstructed scene with the projected 3D object models. 83

FIGURE 32: A monotonically decreasing exponential function is used to normalize the values of the fitness cost. 85

FIGURE 33: The distributions of the object instance recognizability values as the recognition confidence and fitness cost vary in their own value ranges respectively for different α values. Different colors in each figure indicate different object recognizability values. As α increases, the influence of fitness cost on recognizability increases. 86

FIGURE 34: A set of reconstructed object instances based on recognition and pose estimation from several views or scenes, with corresponding recognition confidence and fitness cost values. Each small circle represents a reconstructed object instance. Each red one indicates a correct result, while each blue one indicates an incorrect result. 90

FIGURE 35: An original RGB-D image captured by Microsoft Kinect (leftmost) and the 3D recognition and reconstruction result displayed from 3 different views 94

FIGURE 36: Optimal parameters are searched in the large scale first, then the search scale is narrowed down until solutions are found. Optimal accuracy = 86.43% can be achieved when the SVM parameters $c = 1$ and $\gamma = 2.38$. 95

FIGURE 37: An example scene for training: top row: 8 images from different views; bottom row: corresponding reconstructed scenes; the images are in the order ranked by a human observer from the best to the worst, and the ranks are 1, 1, 3, 4, 5, 6, 7, 7 as ground truth. 96

FIGURE 38: Testing results for **Scene 1**, **Scene 2** and **Scene 3**. Top row for each scene: 8 original images from different views; bottom row for each scene: corresponding reconstructed scenes. Rank of each view by the view evaluator is shown below the corresponding image of the reconstructed scene. 98

FIGURE 39: Recognized and reconstructed object instances of the eight different views $V_1 \sim V_8$ in **Scene 1** (left image), **Scene 2** (middle image), and **Scene 3** (right image) respectively. The displayed regions of high recognizability (red) and low recognizability (blue) is the result of optimizing the harmonic mean H. However, the view evaluator takes into account the effects of 10 different optimization criteria to create the ranking. 99

FIGURE 40: An example of the aspect graph of a tetrahedron: (a) shows the model of a tetrahedron; (b) shows the viewpoint space partition for the tetrahedron; and (c) shows the aspect graph for the tetrahedron. All the figures (a), (b) and (c) are the original figures from [80]. 104

FIGURE 41: An example of the oriented bounding box, generated from the point cloud of a partial model of the object *milk box blue* 105

FIGURE 42: A top view of the set up for our object modeling procedure, including a fixed camera and different poses of the target object after being rotated on the table step by step for about 360 degrees 106

FIGURE 43: Two images of neighbouring object point clouds and their keypoints matching results 108

FIGURE 44: The frames of all the object point clouds in the same 360° rotation loop and their relations, illustrated as a closed loop 110

FIGURE 45: A cross section of the voxel map built based on the point cloud of the current partial model of the object *milk box blue* blue - empty voxels; green - occupied voxels; red - occluded voxels; white - occplane voxels 120

FIGURE 46: The set up of our experiments: a Microsoft Kinect camera is used for the perception and a Barrett WAM arm is used for the object manipulation 124

FIGURE 47: A few snapshots of the object *milk box blue*, rotated by the Barrett WAM arm through pushing step by step in the 360° rotation loop for model modeling 126

FIGURE 48: A few snapshots of the change of the support surface, for the model building of the object *milk box blue* 127

FIGURE 49: The comparison of the registration results before and after global optimization, using all the point clouds of the object *milk box blue*, captured from a 360° rotation loop: (a) shows the partial model of the object *milk box blue*, built before global optimization, in which small distortion of the object surfaces (in the red circles) can be clearly observed; (b) shows the partial model of the object *milk box blue*, refined by the global optimization, in which all the distorted parts of the object surfaces are corrected 129

FIGURE 50: A statistic comparison of the qualities of the partial models of the object *milk box blue*, before and after global optimization, as shown in Fig. 49, based on the mean K-nearest neighbor distances of all the points in the point clouds of both models ($K = 100$) 131

FIGURE 51: A statistical comparison of the qualities of the partial models of the object *milk box blue*, before and after global optimization, as shown in Fig. 49, based on the numbers of the neighbors of all the points in the point clouds of both models, within the radius of 3 *mm* 132

FIGURE 52: The built complete model of the object *milk box blue*, visualized from different viewpoints 133

FIGURE 53: More built models of daily objects, with their corresponding OBBs: from left to right, they are *milk box red*, *spray* and *coffee box* 135

LIST OF TABLES

TABLE 1: Comparative evaluation of range sensing capabilities of several sensors 4

TABLE 2: Comparison of several existing approaches vs. our approach 29

TABLE 3: Detection and pose estimation (PE) results with k=10 49

TABLE 4: Combined object detection and pose estimation results 70

TABLE 5: Comparison of the test results: Mean Shift, P. Clust. (Q-Score), and our approach 74

TABLE 6: Comparison of the test results in precisions and recalls: Mean Shift, P. Clust. (Q-Score), and our approach 75

TABLE 7: The voxel map after each 360° rotation loop for *milk box blue* 134

CHAPTER 1: INTRODUCTION

A brief overview of the problem of robotic perception is first provided. Existing sensing technologies of different kinds are then introduced, and their advantages and disadvantages are discussed. Next, robotic perception is compared to classic computer vision, and major differences are discussed. The problem of view planning through robotic perception is also introduced.

1.1 Robotic Perception

Robotic perception is to perceive the real world and understand the environment through sensing for robotic tasks. A great variety of sensors can be used for sensing, such as digital cameras, RGB-D cameras, laser rangefinders and force sensors. The perceived information can provide useful cues for robotic tasks, for example, to guide the robots to develop better strategies for motion planning, or to provide real-time feedback of robotic operations to improve the operations and ensure success.

Compared to classic computer vision, which is mainly focused on understanding 2D images, robotic perception is more related to providing spacial information, such as 3D positions and orientations of objects needed in most robotic tasks. In a typical computer vision task, such as recognizing road signs, as long as all road signs are correctly detected, the goal is achieved. However, in a typical robotic per-

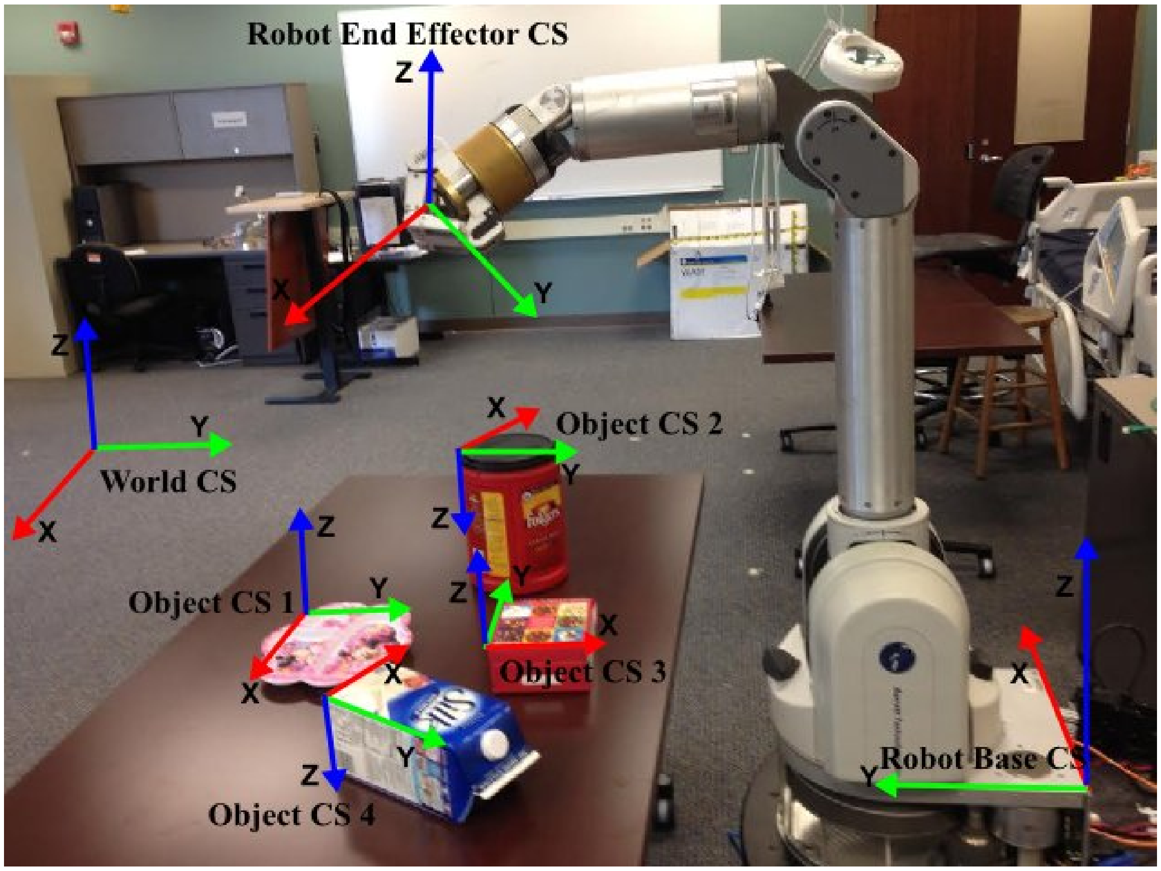


Figure 1: A typical scene of an autonomous robotic manipulation problem. Coordinate systems (CSs) are introduced to describe the relations among the target objects, the robot, and the world.

ception task, for example, to pick up a mug, not only should the mug be identified, its position and orientation have to be perceived as well.

Fig. 1 illustrates a scene of an autonomous robotic manipulation problem, where the objects shown have to be grasped. To successfully grasp a target object, it is necessary to not only recognize the object and but also obtain the exact position and orientation of the target object, which can be expressed in terms of a 4×4 homogeneous transformation matrix ${}^{W}T_{O}$ between the object coordinate system O

and the world coordinate system as follows:

$$^{W}T_{O} = \begin{pmatrix} X & Y & Z & P \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

where X, Y and Z are 3×1 unit vectors representing the directions of the x, y, and z axes of the object coordinate system described in the world coordinate system, and P is a 3×1 position vector of the origin of the object coordinate system in the world coordinate system.

The 6 degrees of freedom (DoF) position and orientation, i.e., the 6DoF *pose*, of the object with respect to the coordinate system of a robot hand can be obtained based on the perceived information of the object (i.e., its shape, size, colors, etc), and its estimated pose with respect to the world coordinate system. Such 6DoF object pose can be used to guide the robot manipulator to accomplish object manipulation.

In general, 3D object detection and 6DoF pose estimation is a necessary step to enable many robotic tasks.

1.2 Existing Sensing Technologies

There are a great variety of sensors that are commercially available. They are based on different sensing techniques and capture different information from the environment. Of particular interest to robotic perception are vision and range sensors. Some sensors, such as the Microsoft Kinect, combine visual and range sensing, called *RGB-D* sensors, because both visual (color) information and depth information of a 3D object can be sensed. A stereo vision sensor can also provide

Table 1: Comparative evaluation of range sensing capabilities of several sensors

	Bumblebee2	**SR4000**	**Microsoft Kinect**	**Sick LMS200**
Techniques	Stereo vision based on multiple digital cameras	Infrared light; Time-of-flight (ToF)	Infrared light; Structured light	Laser; Time-of-flight (ToF)
Accuracy	$\pm 1 \sim 2$ *mm*, up to ± 2 *m* within 20 *m*	± 10 *mm* (typ.)	± 3 *mm* within 2 *m*	± 15 *mm* (typ.)
Typical Range	20 *m*	$\pm 0.8 \sim 5.0$ *m*	$\pm 0.8 \sim 3.5$ *m*	10 *m*
Speed	Approx. 20 FPS	Up to 50 FPS	Approx. 30 FPS	Approx. 10 FPS
Environments	Indoor & Outdoor	Indoor	Indoor	Indoor & Outdoor
Advantages	No distance limit; Accurate	Fast; Less power	Fast; Accurate within small ranges; Affordable	Accurate; Large range
Disadvantages	Expensive; Affected by patterned textures; Requirements of good Illumination	Expensive; Background light; Objects' reflective properties; Multiple reflections; Interference of multiple cameras	Background light; Objects' reflective properties; Multiple reflections; Interference of multiple cameras	Expensive; Slow; Heavy weight

both visual and depth information. Range sensors can only provide depth information.

Table 1 compares depth sensing capabilities of several sensors, which adopt different sensing techniques and are commonly used in robotic perception. More detailed comparisons can be found in [101, 93, 50].

Other types of sensors can also be used in robotic perception, for example, force/torque sensors can measure the force/torque applied to the robot by objects during robotic manipulation. Sonar sensors can also be used to measure distances to the objects based on sound propagation. In this dissertation, the focus is on robotic perception through RGB-D sensing.

1.3 Computer Vision vs. Robotic Perception

Since vision-related techniques have been well studied in the computer vision research field in the past decades, we first present a broad overview of different aspects in solving computer vision problems and then explain the major differences

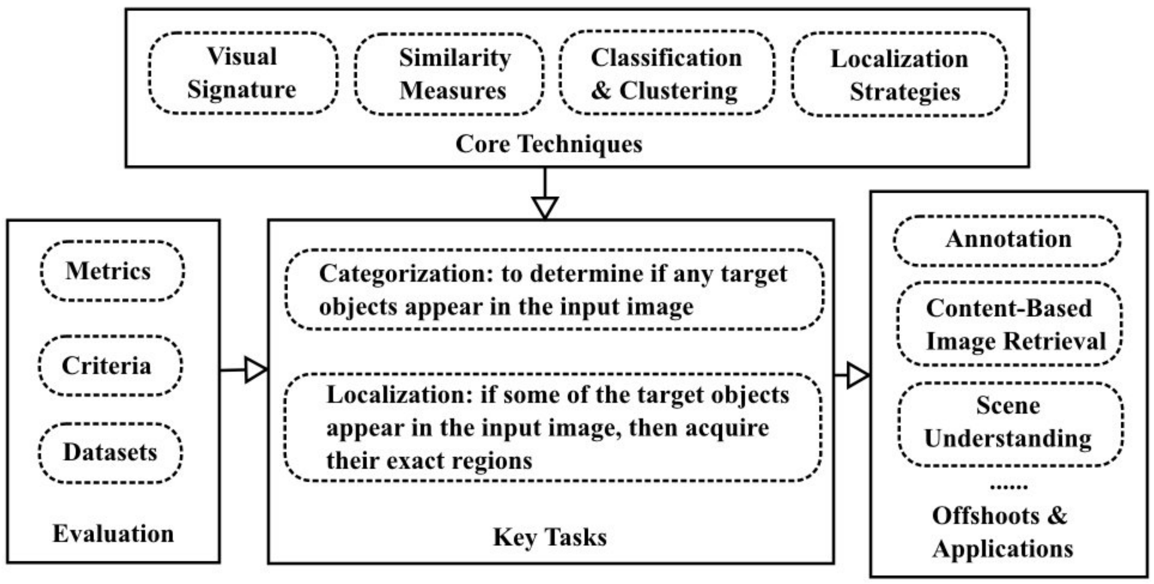

Visual Signature
Similarity Measures
Classification & Clustering
Localization Strategies
Core Techniques
Metrics
Criteria
Datasets
Evaluation
Categorization: to determine if any target objects appear in the input image
Localization: if some of the target objects appear in the input image, then acquire their exact regions
Key Tasks
Annotation
Content-Based Image Retrieval
Scene Understanding
......
Offshoots & Applications

Figure 2: An overview of different aspects in solving computer vision problems. The figure is revised based on similar figures in the surveys [118] and [30].

of techniques and purposes between computer vision and robotic perception tasks.

Several researchers [87, 30, 118] have conducted comprehensive surveys of existing computer vision techniques. As illustrated in Fig. 2, existing computer vision techniques focus on object categorization and object localization as key tasks. Object categorization is to determine if any target objects appear in the input image, while object localization requires to further acquire the exact regions of the target objects if some of them appear in the input image. On the basis of these key tasks, different kinds of offshoots and applications can be further developed, for example, to understand the scene based on the relations of all detected objects.

The core techniques used in solving computer vision problems are briefly described as follows: first, appropriate low-level image features and object appearance models need to be chosen for robust high-level object visual representation; next, efficient and effective similarity measures and classification or clustering algorithms need to be adopted both in the training and testing processes. To locate the objects in an input image, it is also required to choose suitable localization approaches, such as sliding-window and segmentation based approaches.

Robotic perception, on the other hand, requires additional core technologies. Fig. 3 illustrates an overview of different aspects in solving robotic perception problems. In robotic perception, researchers mainly focus on the key tasks of object detection and object pose estimation. Object detection is to determine if any target objects appear in the input image and acquire their exact regions if some of them appear. Object pose estimation is to further estimate the positions and orientations of the target objects with respect to the world coordinate system if some of them appear in the input image, which is related to but different from the problem of object localization in classic computer vision. On the basis of these key tasks, different kinds of offshoots and applications, such as object modeling, object grasping, and view planning can be further developed.

Besides the core techniques used in solving classic computer vision problems, to further estimate the poses of the detected target objects, it is also necessary to develop robust keypoints or features matching algorithms to directly find out the correct correspondences between the 2D or 3D characteristics of the target objects in the test process and the 2D or 3D characteristics of the target objects in the training process. Based on these correspondences, suitable transformation estimation algorithms are then chosen to eliminate the effect of noisy correspondences and obtain correct pose estimates of all detected target objects.

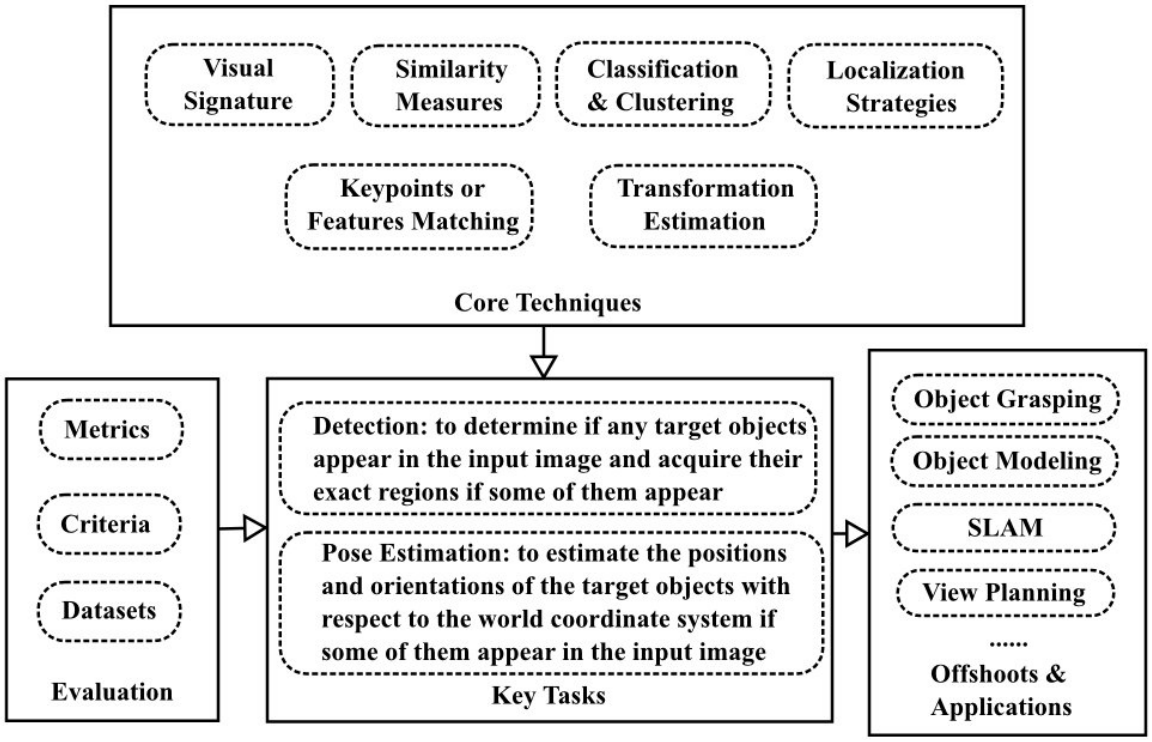

Visual Signature
Similarity Measures
Classification & Clustering
Localization Strategies
Keypoints or Features Matching
Transformation Estimation
Core Techniques
Metrics
Criteria
Datasets
Evaluation
Detection: to determine if any target objects appear in the input image and acquire their exact regions if some of them appear
Pose Estimation: to estimate the positions and orientations of the target objects with respect to the world coordinate system if some of them appear in the input image
Key Tasks
Object Grasping
Object Modeling
SLAM
View Planning
......
Offshoots & Applications

Figure 3: An overview of different aspects in solving robotic perception problems. The figure is revised based on similar figures in the surveys [118] and [30].

1.4 View Planning for Robotic Perception

In robotic perception, it is often necessary to recognize objects or understand scenes through multiple views. Thus, a key problem is how to find an optimal set of views as few as possible to most effectively and efficiently meet the objective of recognition or reconstruction. The study of autonomous views-selection approaches is usually called next-best-view (NBV) planning, or view planning problem (VPP).

Connolly [24] introduced the term next-best-view in 1985. NBV planning is always real-time, and some NBV planning approaches require training processes from which some prior knowledge can be learned, but the training process is not always necessary. Existing work on view planning has been focused on object modeling and object recognition.

3D object modeling, also called 3D object reconstruction, is usually performed as a preprocessing or learning step for subsequent 3D object detection and pose estimation tasks. Researchers from Stanford University have succeeded in accurately reconstructing some of the sculptures and architectures of Michelangelo in the Digital Michelangelo Project [1, 52], using laser rangerfinders and algorithms developed for combing multiple range and color images. However, the whole procedure is manual and considerably time-consuming and effort-consuming for human operators. Obviously, reliable autonomous approaches can greatly facilitate such a project.

Scott et al. [113] presented a comprehensive survey of most NBV algorithms proposed for autonomous 3D object reconstruction before 2002. More recent work can be found in [38, 56].

Real-time active perception for 3D object recognition through multiple views also requires good strategies of view planning. Most literature proposed for active recognition through view planning before 2002 has been well surveyed in [86]. More recent work can be found in [63, 61, 39, 16, 5].

CHAPTER 2: RELATED LITERATURE

In this chapter, we focus on the main challenges in robotic perception and discuss related state-of-the-art literature. First, we review the most relevant core techniques in visual representation, learning algorithms, keypoints matching and transformation estimation.

2.1 Object Visual Representation

To effectively and efficiently recognize objects, it is important to describe objects with appropriate visual signatures and structures, which are discriminative from other objects but also are robust to image variations, caused by different cameras, illuminations and viewpoints.

Fig. 4 illustrates the process of generating high-level object visual representations from low-level image features.

Existing literature in the past decades has proposed different kinds of low-level features to describe the characteristics of an image, for example, grayscale, color [69], shape [72, 10, 79, 8, 44], texture [69, 42, 31] and gradient [66, 116, 73, 9, 29]. A set of well-defined color and texture descriptors, which have been approved for the Final Committee Draft of the MPEG-7 standard, can be found in [69]. Development of local feature descriptors has made great progress in recent years. The scale invariant feature transform (SIFT) descriptor [66] is the most well-known lo-

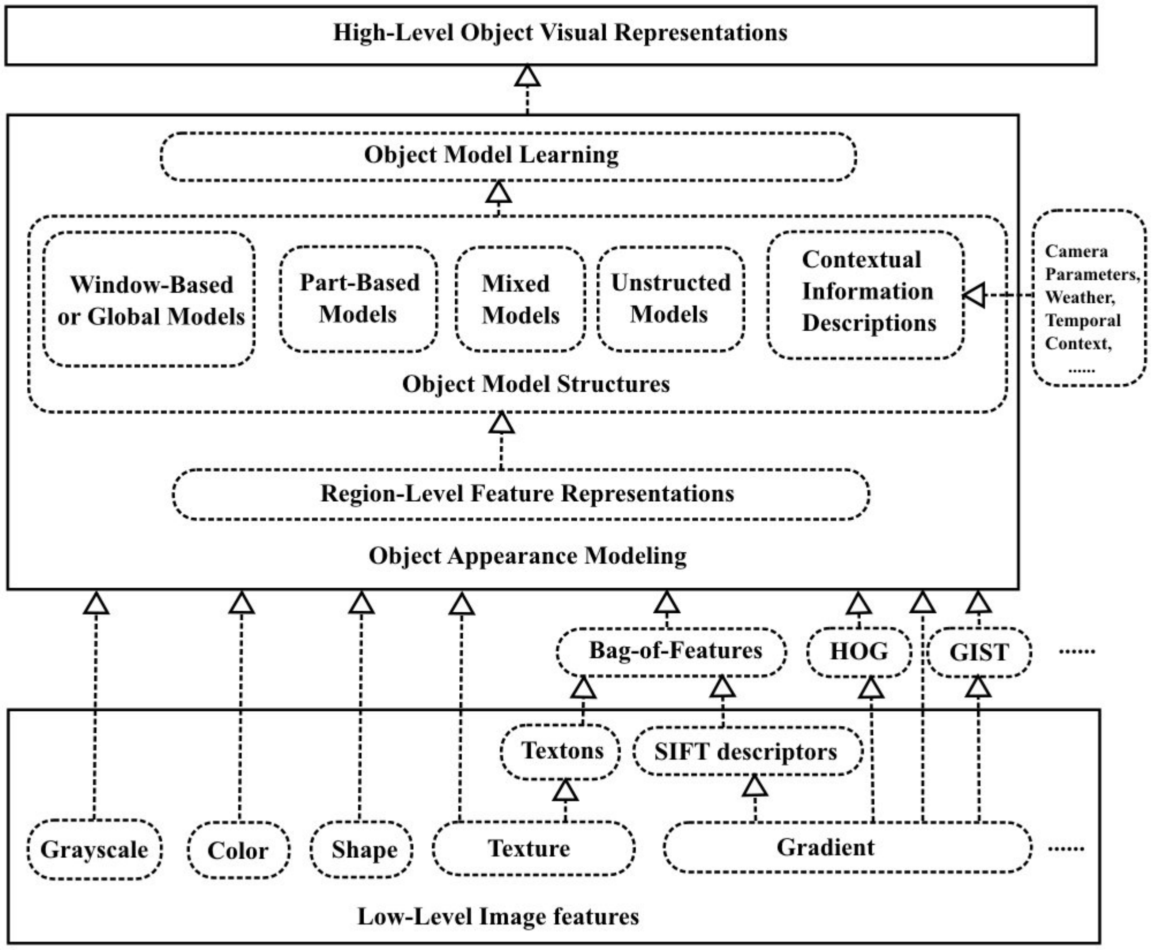

High-Level Object Visual Representations
Object Model Learning
Window-Based or Global Models
Part-Based Models
Mixed Models
Unstructed Models
Contextual Information Descriptions
Camera Parameters, Weather, Temporal Context, ...
Object Model Structures
Region-Level Feature Representations
Object Appearance Modeling
Bag-of-Features
HOG
GIST
......
Textons
SIFT descriptors
Grayscale
Color
Shape
Texture
Gradient
......
Low-Level Image features

Figure 4: An overview of the procedure of generating high-level object visual representations from low-level image features. The figure is revised based on similar figures in the surveys [118] and [30].

cal feature descriptor, which has been widely used in existing object recognition approaches. Several modifications have also been proposed based on the original SIFT, for example, PCA-SIFT [116] is proposed to make the feature description and matching more efficient, and ASIFT [117] is proposed to make the algorithm more robust to viewpoint changes. Histogram of oriented gradients (HOG) [29] has been demonstrated to be very powerful in detecting structured objects, especially in pedestrian detection.

There can be different structures of visual representation of objects, such as global models [29, 112], part-based models [34, 40], and mixed models [35]. With global models, objects are described as single regions in training and testing; with part-based models, objects are described with a set of object parts and the geometrical relations between them; mixed models [35] combine both global models and part-based models together to better describe the characteristics of objects. Note that more complicated models usually bring in more parameters to tune in the training process.

2.2 Learning Algorithms

There exist a great variety of machine learning algorithms, which can be used for object classification. Popular machine learning algorithms used in computer vision can be categorized based on their similarities as: linear models, clustering methods, kernel methods, decision tree, artificial neural networks, Bayesian networks, boosting methods, and so on. More detailed and systematic analysis of existing machine learning algorithms can be found in [12]. Support vector machines (SVMs) [17], which are one of the kernel methods, have been one of the most widely used algorithms since 2000, because of the flexibility in the choice of kernel functions, the unique solution of a convex optimization problem, and the good out-of-sample generalization. Recently, deep learning [62, 97], based on neural networks of many layers, has become a hot topic in solving big data problems and achieved good performance in different kinds of applications.

2.3 Keypoints Matching and Transformation Estimation

Keypoints matching is one of the common ways to directly establish the correspondences between the 2D or 3D characteristics of the target objects in the test process and the 2D or 3D characteristics of the target objects in the training process. Features used for extraction and description of keypoints or keypoint pairs play a significant role in matching.

In 2D images, the SIFT [66] keypoint has been demonstrated to be very robust in dealing with scaling, rotation and illumination changes. However, it is only partially invariant to affine distortion caused by viewpoint changes, which is a common problem in 3D robotic perception. [117] extends it to Affine-SIFT (ASIFT) and improves its performance for matching under serious affine distortions.

Compared to features based on color, geometric features are more invariant to viewpoint changes. The simplest geometric features could be just 3D coordinates of points, which can be used by the iterative closest point (ICP) algorithm [11] to register two similar 3D point clouds. However, the ICP algorithm requires a good initial pose estimation, which is difficult or nearly impossible from images of real-world scenes with multiple occluded objects shot in random viewpoints. Surface normals and curvature estimates [90, 89], which can be computed fast and easily, are also commonly used geometric features. [89] proposed pose-invariant keypoints by combining geometric relations between the nearest k neighbors of these keypoints based on estimated surface normals.

However, individual geometric primitives, such as normals, curvatures, lines, and planes, are not very discriminative; as a result, researchers developed various oriented point pair features [32, 22, 21]. Pair features are more discriminative than individual primitives, which provide more informative characteristics. Based on keypoint pairs, an approach [22] combined both oriented surface points and boundary primitives, such as boundary points with directions and boundary line segments, to enhance the accuracy and efficiency of the algorithms for detection and pose estimation. The work was focused on parts for assembly with many geometric details.

As for calculating a transformation matrix based on matched keypoint pairs, a comprehensive survey [33] provides detailed comparision results for four major algorithms: singular value decomposition (SVD), orthonormal matrices (OM), unit quaternions (UQ) and dual quaternions (DQ). The random sample consensus (RANSAC) [37, 89] algorithm can also be utilized here to make the calculation more robust to noise. The ICP [11] algorithm can be adopted appropriately later to refine the results of localization and pose estimation after coarse estimation results are obtained from the approaches described above.

2.4 Challenges in Robotic Perception

Automatic identification and localization of a target object through robotic perception is necessary for autonomous robotic manipulation tasks. However, general object recognition in a cluttered 3D environment is still an unsolved and challenging problem in computer/robot vision, due to occlusions, complicated background, and great variation in object appearance caused by different illumination conditions or viewpoints. To further estimate the pose of an identified object, which consists of the position and orientation of the object in the 3D scene, is even more difficult. The emergence of advanced and inexpensive RGB-D cameras, such as the PrimeSense 3D sensor and Microsoft Kinect, which capture both vision and depth information, provide new opportunities for researchers to seek effective and efficient solutions to those problems.

2.4.1 Object Detection and Pose Estimation

Given an RGB-D image of a scene, existing approaches use different strategies to identify known objects and estimate their poses (which are either 3DoF or 6DoF). Object detection and pose estimation are processed either simultaneously or sequentially.

2.4.1.1 Keypoint-based Object Detection and Pose Estimation

With characteristics such as keypoints [90, 89] or keypoint pairs to describe known object models, some recent work [32, 23, 22, 21] proposes to extract, describe, and match keypoints or keypoint pairs directly between the point cloud of an entire test image and each trained 3D object model to find correspondences. In those approaches, objects are detected with their poses estimated simultaneously by either Hough voting [7, 32] or pose clustering in the pose parameter space [23, 21] based on recognized keypoints or keypoint pairs.

However, those approaches process the whole image as a single entity, which has several related drawbacks if the scene is cluttered and objects are partially occluded. One drawback is unnecessary and even incorrect processing of background or regions other than the target object, such as incorrect matches between some background regions and object models. Another drawback is likely insufficient treatment of the object regions, e.g., if the target object is heavily occluded and there are only small visible parts, there may not be sufficient number of keypoints or keypoint pairs for the correct object model to obtain enough votes to ensure correct pose estimation. Moreover, there are many parameters used in Hough voting or pose clustering, and the parameter values are empirically and manually decided, which depend on the objects and scenes considered. Hence, such approaches are not automatically adaptive to different kinds of objects and scenes and can have difficulties in dealing with the problems of noise and multiple instances of the same object.

Using the sliding window methods [29, 35, 59, 60, 58] as preprocessing can help reduce the candidate region of each target object, so that the keypoints or keypoint pairs can be extracted and matched subsequently to establish the correspondence between the candidate region and the target object model for more accurate pose estimation. Nevertheless, as the candidate region is usually represented as a bounding box, it often contains some parts of the background or other objects that occlude the target one in a cluttered scene. There can still be significant inaccuracy for keypoints at the boundaries of objects, especially in cluttered environments where objects are stacked together or physically attached. The sliding window methods are also not well suited for dealing with partially visible objects and multiple instances of the same object appearing side-by-side or occluding one another.

Approaches that solely rely on matching keypoints or keypoint pairs also suffer from several problems. First, the number of keypoints (pairs) increases considerably as the number of objects in the dataset increases. In a large object dataset, it can be very expensive to find one-to-one correspondences between the extracted keypoints (pairs) in each test image (averaged over 10,000 in [23], as an example) and all the keypoints (pairs) in the dataset based on keypoint (pair) descriptors. How to balance accuracy and speed properly is an unsolved problem. Second, by using only keypoint (pair) matching for object detection, such approaches cannot utilize many other useful object features and associated machine learning algorithms. Third, the more objects need detection and pose estimation, the more robust is required of the keypoint (pair) descriptors for distinctiveness and repeatability. However, [110] shows that finding robust 3D keypoints itself is also an unsolved and difficult problem. Therefore, approaches relying only on matching keypoints (pairs) are not effective enough to deal with complicated object recognition tasks, especially with a large dataset.

2.4.1.2 Segment-based Object Detection

The emergence of powerful and affordable RGB-D sensors inspires new approaches focusing on object detection from segmentation of RGB-D data [53, 115, 98, 83, 46]. With the exact region of each object instance completely separated from the test image point cloud, object pose estimation can be achieved based on matching between a separated object region and the target object model.

Recently, Richtsfeld *et al.* [84] introduced object segmentation following a hierarchical framework of four levels: signal level, primitive level, structural level, and assembly level. Through learning the relations of both neighboring and non-neighboring surfaces of the same object from Gestalt principles, they propose a graph-cut approach based on the probabilities returned by SVMs to group the test surface segments as belonging to the same object instance. A major limitation of their approach is the inability to split correctly pre-segmented surface patches if surface parts of different objects are aligned to the same plane. Stein *et al.* [100] used over-segmentation of a 3D image point cloud to obtain all the supervoxels first and then introduced a novel, strictly local criterion to classify all the edges between the supervoxels as convex or concave. Object regions are obtained by region growing of locally convex and connected supervoxels. However, if an object instance consists of more than one convex region or some concave regions, how to find the exact region of the object instance is still a problem. Schiebener *et al.* [96] integrated robotic manipulation and sensing capabilities to segment objects in an unknown and cluttered environment. The above approaches [84], [100] and [96] only address object detection but not 6DoF pose estimation.

Some existing work for semantic scene labeling [98, 83, 43, 115, 46] also relies on 3D surface segmentation, which, however, does not require 6DoF pose estimation. On the other hand, 6DoF pose estimation is necessary for object grasping and manipulation tasks.

Using the sliding window methods for preprocessing, Zhu *et al.* [120] presented an approach based on known silhouette shapes of each object for pose estimation. The approach over-segments the hypothesis bounding box obtained from an S-DPM classifier into superpixels at first, and then selects a set of superpixels that best match a known silhouette of the object model for detection. The object pose is recovered by aligning the object model projection with the foreground silhouette. However, this approach is only focused on 3D objects without complex textures and also requires high distinctiveness of silhouette shapes of different objects from different views.

The approaches in [102, 13] are focused on recognizing object point clouds but cannot be applied to cluttered scenes directly for object detection, because they assume that from the segmentation of the test image with some simple clues, the point cloud of each target object can be isolated, separate from those of the other objects.

2.4.1.3 Appearance-based 3D Object Modeling

In order to build 3D models of objects, existing work [21, 77, 99] has achieved impressive results but with very controlled environment set-ups. Object models are usually built from plenty of RGB-D images captured continuously. In [77], researchers move a Microsoft Kinect sensor around the target objects and utilize large overlaps between neighboring views to register different views. In [21, 99], objects are placed on a turntable with cameras around them, and a chessboard is used to register different views. In those approaches, objects are usually fixed, and the bottom surfaces are never observed. It is difficult to obtain complete object models even if a RGB-D sensor can capture all the surface points. If an object surface contains transparent regions, the current RGB-D sensors cannot capture them.

2.4.2 View Evaluation

In computer vision, 3D object recognition via perception through multiple views has also been well studied. Rosenfeld [85] proposed to represent an object with a series of 2D “characteristic views” or “aspects” for fast and robust recognition. In [54, 18], how to select a set of views to represent an object appropriately was addressed in different methods. Plantinga and Dyer [80] introduced an efficient aspect representation to represent the appearance of polyhedra from all viewpoints by an aspect graph in more general situations. Aspect-based recognition algorithms are especially suitable for objects with complicated geometric models. Many approaches [28, 65, 58, 35, 120] later were focused on how to take advantage of various visual features, geometrical features, or feature combinations, and create different kinds of object representation to improve recognition.

In real-time recognition and pose estimation of 3D objects, rather than choosing views randomly, researchers proposed “active perception” to recognize objects more efficiently. Bajcsy [6] first introduced active perception as a study of modelling and control strategies for purposefully changing a sensor’s state parameters for better perception. Wilkes and Tsotsos [114] applied this concept to the task of 3D object recognition. Roy *et al.* [86] presented a comprehensive survey of the research work before 2002 on active recognition through next-best-view planning.

More recently, many novel strategies of viewpoint selection for active object recognition have been introduced [63, 61, 39, 16, 5]. Laporte and Arbel [61] presented their viewpoint selection criterion for maximally distinguishing objects of various classes at different orientations by Bayesian classifiers. Forssen *et al.* [39] implemented their recognition system by combining active view planning with bag-of-features and geometric matching techniques for ranking potential objects. Browatzki *et al.* [16] proposed a perception-driven, multisensory exploration and recognition scheme to actively resolve ambiguities that emerge at certain viewpoints for a humanoid robot.

Most of the existing work is only focused on recognizing individual objects, but it is more important to evaluate how well a cluttered scene with many objects occluding and interacting with one another can be perceived through an image of a certain viewpoint. The challenge is to enable effective selection of good views that can facilitate automatic object recognition and scene reconstruction efficiently for cluttered scenes, which is important for many robotics applications.

2.5 Combining Perception and Manipulation

Industrial robots have been proven to be very effective and efficient in mass production, and they have been widely used in modern manufacturing. However, robot manipulation in those environments are repetitive without variation. It is still an open challenge to manipulate unknown objects autonomously in uncontrolled environments. The difficulties come from dynamic changes of the environment, real-time constraints, unknown object placement and pose, obstacles, and so on [49]. As a result, it is necessary to incorporate perception in manipulation.

2.5.1 Perception Incorporated with Manipulation

There already exist a great variety of approaches to combine visual perception and robotic manipulation, for the purposes of either improving the solutions of existing problems or providing solutions for new problems. In [74], a database which includes visual features, the CAD model, and a set of feasible grasps of each object is first created offline. Then, in real-time perception, once an object is recognized and localized, a grasp can be immediately selected from the database for execution taking into account the pose of the object. The approach requires good object models from training. In [15], an object of interest is placed in the hand of a humanoid robot and rotated to produce more views for perception, so that the object model can be trained and recognized fast and reliably. However, the approach starts with objects already grasped and located in the hand of the robot; in addition, objects are set to be always located approximately in the center of the observed image; hence object manipulation is relatively simple.

Without the requirements to build object models, [94] learns to identify good grasping locations of unknown objects by using different visual features and training on a large object dataset. By coupling manipulation and perception, [47] successfully extracts kinematic models of unknown tools, which can further be used to perform purposeful manipulation. Nevertheless, the approaches of both [94] and [47] are only tested in structured and uncluttered environments. [95] introduces a method for autonomously learning the visual appearance representation of an unknown object through integrating visual perception and simple object pushing by a robot hand. However, the proposed strategy for generation of pushing movements cannot be extended to unknown objects of arbitrary shapes and sizes. [20, 48, 78, 68] focus on some specific object manipulation tasks in unstructured and cluttered environments, such as arranging and sorting objects in piles, but still have not achieved satisfactory solutions for general applications.

2.5.2 Automatic 3D Object Modeling

Interacting perception with manipulation can also be introduced to help automatic 3D object modeling. How to build a 3D object model based on geometric and visual appearance of an object is an important problem, because many robotic manipulation tasks require the knowledge of 3D object models of the target objects.

3D object modeling has been studied for a long time with many methods introduced by researchers. Existing literature can be classified in two categories: *passive* approaches and *active* approaches.

Passive approaches are defined by fixing the cameras and viewpoints. A set of sensor views are usually pre-determined to collect the 2D or 3D information of a target object from different viewpoints. In [21, 99], objects are placed on a turntable so that they can be rotated. Multiple cameras mounted around the turntable, and a chessboard is used to register different views. Although the speed of rotation of the turntable and the positions of the cameras can be changed, the camera views are still limited in such a setting. Given a target object of arbitrary shape, Some parts of the object may not be visible by any of those fixed cameras even with the rotating turntable due to self-occlusion. Besides, the bottom part of the target object is always occluded by the turntable.

Active approaches for 3D object modeling refer to moving cameras carried by either human operators or robots. Human-guided object modeling can achieve accurate 3D reconstruction results in [52, 77]. Robot-guided object modeling is usually considered as a view planning problem (VPP), or next-best-view (NBV) [24] planning problem.

In a classic view planning problem for 3D object reconstruction, target objects are usually placed on a table, and a sensor is mounted on a robot manipulator. The research focus is on how to find a minimum set of views to enable reconstructing 3D object models reliably and accurately. Based on a comprehensive survey of most NBV algorithms proposed before 2002 [113], those algorithms can be classified as model-based or non-model-based. Model-based algorithms focus more on the space for sensor viewpoints and try to theoretically find an optimal set of sensor views which can achieve 100% object surface coverage, such as set theory methods [103] and graph theory methods [14]. Non-model-based algorithms focus more on obtaining object information and reducing uncertainty step by step and view by view, such as surface-based methods [71] and volumetric methods [70]. More recent non-model-based algorithms can be found in [38, 56, 111].

Most of the work on model-building often involves manual operations, which can be tedious and time-consuming for human operators. Some recent work uses certain special-purpose hardware, such as a turntable for rotating an object, to speed up the model-building process, but manual intervention is still needed in order to build a complete model that covers all surfaces of an object. Specifically, there is always some part of the object, such as the bottom surface on a table, that cannot be viewed and requires further pose changing before a complete object model cannot be built.

Some recent work explores 3D object model building by changing the pose of a target object with robotic manipulation [55, 67, 15]. However, the effectiveness of those approaches are limited by several factors. First, a robust real-time tracking algorithm is required to track the position and orientation of the target object and the robot end effector in [55, 67], but tracking can fail in some cases, such as tracking from certain oblique perception angles and tracking an object of uniform color and geometry. Second, the target object is assumed held by a robot manipulator before object model building [55], which is a strong assumption. The held object is always occluded by the robot end effector during perception [55, 67], especially if the object has a similar size as that of the end effector, which complicates object model building. Moreover, if some parts of the object are always occluded in all feasible grasps, a complete object model cannot be achieved. Finally, how to separate the target object from the end effector in perceived images is also an issue. A 3D model of the robot manipulator is required in [55], and [67] assumes some prior knowledge of the color of the robot end effector. In addition, in-hand manipulation itself is still a challenging problem [15].

Hence, automatic 3D model building for objects of arbitrary shapes is a challenging problem not yet fully solved.

2.5.3 Grasp Strategies of Objects

As for a successful grasp of an object, three main aspects should be considered [64]: contact models, grasp analysis, and grasp synthesis. Contact models determine how the forces and torques are exerted on the object by the robot end effector. Given a contact model, an object and a set of contacts which define a specific grasp, grasp analysis is used to evaluate if the grasp satisfies force-closure, form-closure or some other properties. Grasp synthesis is to study how to find an appropriate set of contacts for grasping to achieve those properties, given some constraints of the allowable contacts according to the object and the robot end effector. More detailed mathematical formulation can be found in [64, 45, 81, 26, 75]. Recently, [91] presented a survey on different algorithms for 3D object grasp synthesis before 2011.

CHAPTER 3: RESEARCH OBJECTIVES AND APPROACHES

This chapter starts with a problem description and assumptions, followed by an overview of our proposed approaches.

3.1 Problem Description and Assumptions

This dissertation is focused on the problem of detection and pose estimation of 3D object instances in cluttered environments, where objects are usually stacked together or physically attached. Thus, most target objects may occlude one another in all views. It is further assumed that all objects in a scene can be either characterized through built object models as known objects or classified as belonging to the background (i.e., are part of a trained object dataset or the trained background). For evaluating the scene recognizability of a view, the number of the target objects is also assumed known.

3.2 Approaches

The central theme of the dissertation is to develop appearance-based methods based on 3D object surfaces from RGB-D sensing for effective 3D object detection and pose estimation in cluttered environments to enable robotic manipulation of objects.

3.2.1 Surface-based Approach to Object Detection and Pose Estimation

The focus is on 3D object detection and pose estimation based on object appearance.

3.2.1.1 Overview

Our approach includes the following strategies.

- A robust and convenient strategy to establish object models based on segmented smooth surfaces of RGB-D images from different sensor views of different object poses

Such an object model consists of automatically segmented and registered 3D smooth surfaces from the RGB-D images, their visual features, and relative pose information to form a 3D reconstruction of the object. Our strategy works even for objects with disconnected surface patches and missing surface information in object images, such as due to transparent regions that cannot be captured by an RGB-D camera. It does not require precise information of the locations of the object and the camera where images were taken and thus facilitates autonomous object model building. As the object models are based on noisy appearance from RGB-D images, they are inherently robust to noise in object detection and pose estimation.

- A robust surface-based strategy for automatic appearance-based object detection and 6DoF pose estimation from a single image

Our strategy takes full advantage of our appearance-based object models composed of smooth 3D surfaces and powerful visual signatures of each surface to detect an object based on only its visible surface region and to estimate its pose. Thus, our strategy effectively detects partially visible objects due to either occlusion or surface transparency and estimates the 6 DoF poses of each entire object in a cluttered 3D scene. Our strategy also detects multiple instances of the same object in a cluttered scene with ease.

- An algorithm based on graph-cut to remove redundant and incorrect object recognition results

Our algorithm integrates the recognition and pose estimation results of all surfaces from the single test image in a graph, groups surface segments detected as belonging to the same object instance, and then selects the best-fit 6DoF pose estimate. As the result, even if certain surface segments of other objects are incorrectly recognized as belonging to an object instance A, as long as one surface segment of A is correctly recognized as belonging to A, A can be reconstructed with correct pose estimate from that single surface segment. That is, the combined accuracy of recognition and pose estimation of object instances by our overall system is much higher than that of object recognition alone based on surface segments.

3.2.1.2 Comparison to Existing Work

Table 2 compares several representative existing approaches to our approach. As shown in the table, existing approaches of [13, 84, 102] either do not consider cluttered scenes or do not perform 6DoF pose estimation for the objects in a scene.

Compared to existing approaches that solely rely on matching keypoints (pairs) for both object detection and 6DoF pose estimation [32, 21, 23], our surface-based

Table 2: Comparison of several existing approaches vs. our approach

Approach	Features and algorithms of object detection	6DoF pose	Keypoints of pose estimation	Time complexity of keypoints matching with brute force	Multiple instances of the same object	Cluttered scenes
Bo *et al.* [13]	Learn features from full RGB-D data: gray, RGB, depth, and surface normal channels; most existing features and learning algorithms can be utilized; as the number of objects in the dataset increases, the performance remains robust.	×	×	×	Hierarchical contour based segmentation; unable to deal with cluttered scenes with different objects or multiple instances of the same object stacked together or physically attached and occlude one another.	×
Richtsfeld *et al.* [84]	Use color, texture, depth and geometrical features; most existing features and learning algorithms can be utilized; as the number of objects in the dataset increases, the performance remains robust.	×	×	×	Graph-cut algorithm based on probabilities from SVMs, which learn the relations of both neighboring and non-neighboring object surfaces.	√
Tang *et al.* [102]	Use local SIFT features and hue histograms; most existing features and learning algorithms can be utilized; as the number of objects in the dataset increases, the performance remains robust.	√	SIFT keypoints	$O(nm')$, $m' < m$, for pose estimation	Segmentation with the assumption that objects are on a table or ground plane separately; unable to deal with cluttered scenes, where different objects or multiple instances of the same object are stacked together or physically attached and occlude one another.	×
Drost *et al.* [32]; Choi *et al.* [21]	Rely on keypoint pairs matching solely; require strong distinctiveness and repeatability for keypoint pair descriptors; the performance can be weakened sharply as the number of objects in the dataset increases.	√	Geometrical keypoint pairs [32] (with RGB [21])	$O(nmN)$, related to the number of objects in the dataset N	Voting and pose clustering; more manually decided thresholds to distinguish multiple instances from noise in pose clustering.	√
MOPED [23]	Rely on SIFT keypoints matching solely; require strong distinctiveness and repeatability for keypoint descriptors; the performance can be weakened sharply as the number of objects in the dataset increases.	√	SIFT keypoints	$O(nmN)$, related to the number of objects in the dataset N	Pose clustering; (a) a strong assumption: keypoints which are close enough belong to the same object instance, however, a threshold for closeness has to be chosen in clustering; (b) more manually decided thresholds to distinguish multiple instances from noise in pose clustering.	√
Our approach	Use HSV color histograms and ASIFT keypoint histograms; most existing features and learning algorithms can be utilized; as the number of objects in the dataset increases, the performance remains robust.	√	ASIFT keypoints	$O(nm')$, $m' < m$	Model-fitting based graph-cut; no extra manually decided thresholds to distinguish multiple instances from noise in solving pose ambiguities.	√

approach avoids some significant disadvantages as discussed in Section 2.4.1, and can still perform robustly in cluttered scenes. As outlined in Table 2, our approach has several advantages:

- Our approach for recognition of surface segments is flexible to use various kinds of existing mature object features and learning algorithms.

Whereas, the performance of object detection in [32, 21, 23] only relies on the distinctiveness and repeatability of the descriptors of keypoints or keypoint pairs, which can be weakened as the number of objects in the dataset increases.

- The time complexity of our approach is constant with respect to the number of objects in the dataset.

Let N is the number of objects in the dataset, n is the average number of training keypoints (pairs) of each object, and m is the average number of keypoints (pairs) in each test image. If brute-force search is considered, then the time complexities of the approaches [32, 21, 23] are $O(nmN)$, while the time complexity of our approach is $O(nm')$, where $m' < m$ is the number of the keypoints (pairs) from the recognized object regions, rather than from the whole test image.

- Our approach does not need manually setting many threshold values.

The approaches of [32, 21, 23] are algorithms based on voting or clustering. For those approaches, many thresholds have to be decided to distinguish multiple object instances from noise, which are usually object-related or scene-related. However, our approach does not require such additional thresholds to distinguish multiple object instances from the noise; as the result, our approach is more adaptive to different scenes.

Moreover, the approach of [23] assumes that keypoints that are close enough belong to the same object instance; however, a threshold for closeness has to be chosen and can affect the performance of the algorithm. Our approach does not have such an assumption.

Recently, an algorithm [76] was proposed to search for the best interpretation of the observed sensor data, by hypothesizing possible scenes with known 3D object models. However, they assume the number of objects in the scene is known and objects in the scene vary only in 3DoF (2DoF position and yaw).

3.2.2 Learning-based View Evaluation

In order to assess automatically how good a view is for machine vision to recognize objects that partially occlude one another in a scene, we also introduce a learning-based approach for evaluating scene recognizability from a single view. We are interested in developing an evaluation approach to compare images from different viewpoints for scene recognizability by machine vision. If an image has better scene recognizability than another image, its viewpoint is considered a better viewpoint for both recognition and reconstruction of the scene. The following factors are considered, formulated in terms of related parameters, in evaluating the recognizability of a cluttered scene with objects occluding one another from a single view:

1. the recognition confidence of each object from the view;
2. the fitness of each object, measuring the accuracy of pose estimation for the object in the scene (in terms of a 4x4 homogeneous transformation matrix with respect to the world coordinate system);
3. the recognition uncertainty, reflecting poorly recognized or wrongly recognized objects from the view;
4. the recognition confidence of the background from the view.

Higher confidence values and higher accuracy in pose estimation mean better recognition results; whereas lower uncertainty is better.

However, for two different views, if one is better in some factors but worse in some other factors, it can be difficult to decide which view is better overall. For example, if more objects can be recognized with high confidence from one view than the other view, but the objects recognized in the latter view have more accurate pose estimation, which view is better cannot be easily determined and may depend on the specific task environment. Hence, we propose a learning-based optimization method to best combine factors 1) and 2) in characterizing the recognizability of each object from a view. The optimization result separates objects of high-recognizability from those of low-recognizability. Different optimization criteria are considered. We further introduce a learning-based view evaluator to determine which view has a better overall scene recognizability by taking into account all the factors above. Our approach only requires intuitive human observation to aid the learning (i.e., training) process, but the trained view evaluator can provide more accurate view evaluation results than a human observer for the purpose of autonomous perception and action. Our learning-based approach also makes it easy for the view evaluator to adapt to the characteristics of specific types of task environments or scenes.

3.2.3 Interleaving Perception and Robotic Manipulation

The integration of RGB-D perception of an object and robotic manipulation of the object is also a focus topic to facilitate both object modeling and object manipulation. Using robotic manipulation can enable automatic object model building. RGB-D perception is applied to capture the information for appearance-based object modeling and also guiding robotic manipulation, and robot manipulation is introduced to move the target object for better RGB-D perception or to change the poses of the object to facilitate surface-based model building. Such techniques of interleaving perception and manipulation can also be extended to handling unknown objects in a scene, such as moving an unknown obstacle away in order to gain access to a target object, as well as modeling unknown objects encountered to expand the database of appearance-based object models for recognition and pose estimation.

Here, we address automatic building of a complete 3D model of a rigid object by covering all its surfaces. In our modeling set up, an RGB-D camera is placed at a fixed position with a fixed direction to view the target object, which is placed on a table. After the first image is taken, the object is rotated slightly by a robot manipulator to make new surfaces visible, and the following process of alternating perception and manipulation is conducted:

Repeat:

- take a new image;
- extend the partial object model based on adding the new image;
- plan and execute a manipulator motion based on the partial object model to change the object pose;

until images have been taken for all object surfaces.

In the above procedure, perception and manipulation are very much interdependent. Perception guides manipulation, which assists further perception.

Fig. 5 provides a high-level flowchart of our approach. Note that there are two loops in the flowchart to distinguish two kinds of manipulation to change object poses. One kind is to rotate the target object on table (in a consistent direction) gradually through pushing until after 360 degrees to cover side surfaces of the object; and the other kind is to change the support surface of the object, i.e., rotate the object to expose its previous bottom surface, which can be achieved by a greater push or by picking up the object, rotating it in the air, and placing it down (on a different surface), in order to cover unseen surfaces. We extend existing approaches for object model building based on registering images of overlapping views [11, 88] for our purpose and focus on the issue of how to adjust the pose of the target object through robotic manipulation to expose unseen surfaces based on the partial object model at the moment, especially for the kind of manipulation that changes the bottom (support) surface of the object.

We use a common volumetric representation [111, 92, 51, 57, 27] for our object model in order to facilitate the determination of the next object pose and how to perform object manipulation to achieve the desired next pose. We introduce an evaluation function of a candidate object pose by taking into account the image region of previously unseen surfaces vs. that overlapping with existing images, the stability of the pose, how easy to manipulate the object to change its current pose to the candidate pose considering manipulator constraints, and the effect on the quality of the (partial) object model. The volumetric representation also enables the detection of a complete model, if all surfaces of the object can be observed through manipulation.

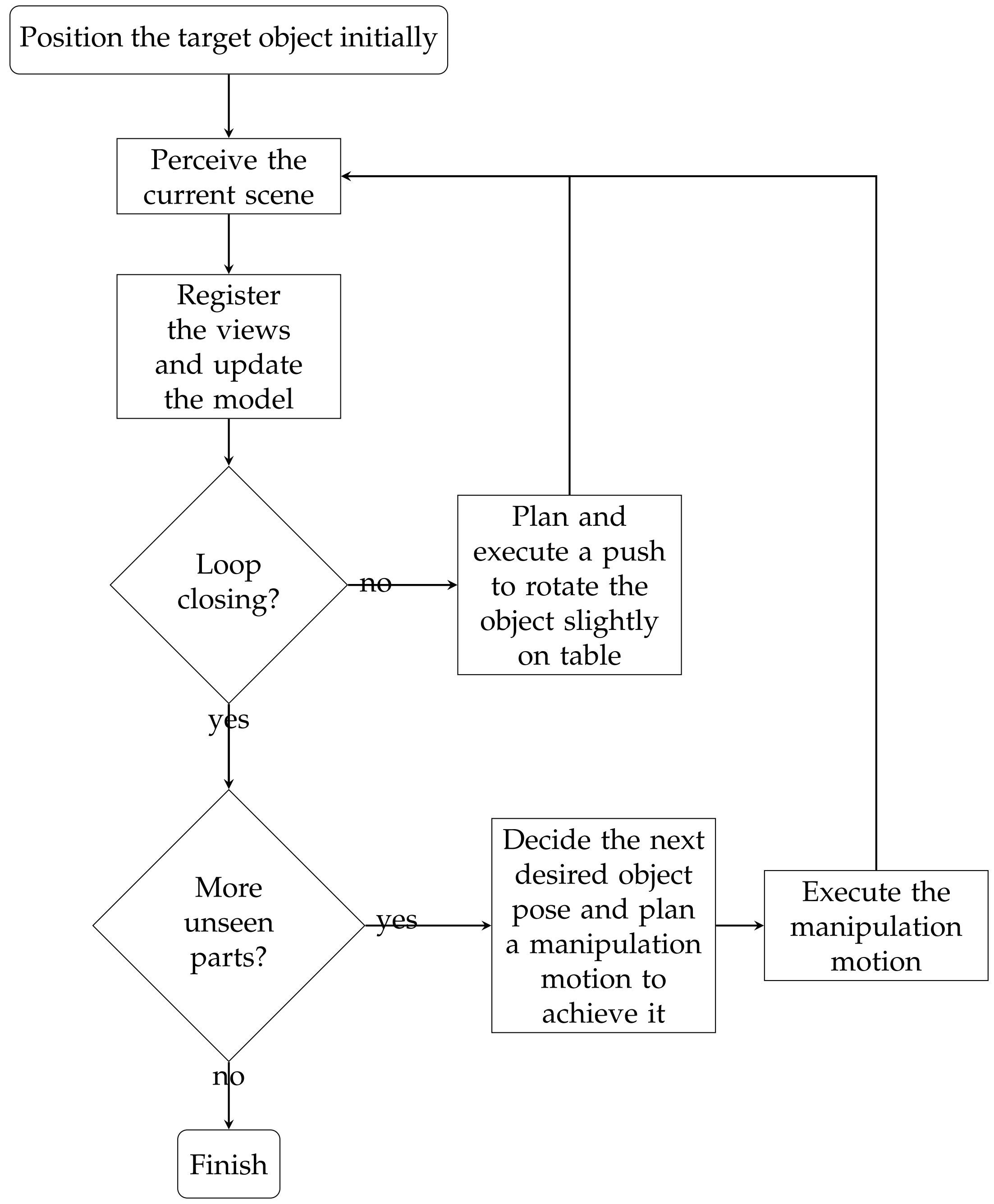

Position the target object initially
Perceive the current scene
Register the views and update the model
Loop closing?
no
Plan and execute a push to rotate the object slightly on table
yes
More unseen parts?
yes
Decide the next desired object pose and plan a manipulation motion to achieve it
Execute the manipulation motion
no
Finish

Figure 5: High-level flowchart of our automatic 3D object modeling system

CHAPTER 4: SURFACE-BASED 3D OBJECT DETECTION AND POSE ESTIMATION

In this chapter, a new appearance-based approach to general 3D object detection and pose estimation [107] is presented based on surface features in cluttered environments, taking full advantage of RGB-D information. Our approach identifies known object surfaces and subsequently both identifies the corresponding objects and their homogeneous transformation matrices w.r.t. the camera coordinate system in the 3D scene, even if the objects are partially occluded and may be partially transparent. Our approach can also effectively solve the problem of detection and pose estimation of multiple instances of the same object in a scene, which can be in occlusion of one another. The key characteristic of our approach is that we build object detection and pose estimation on the basis of smooth 3D surface segments and their visual signatures.

4.1 Object Representation and Model Building

Appearance-based object representation starts from a sufficient number of RGB-D images of different viewpoints of an object (captured by Microsoft Kinect). First, images are segmented based on smooth surfaces, where a smooth surface is defined by the continuity of depth and surface normal values, and its boundary is characterized by depth and surface normal discontinuity [106]. Each object is characterized in terms of surface segments from multiple views and the visual signatures of the surface segments. Both HSV color histogram and ASIFT keypoint histogram [117] are used as the visual signatures because of their robustness to viewpoint and illumination changes. As an example, Fig. 6 shows a common cereal box and its smooth 3D geometric surfaces in terms of the corresponding image segments from different views.

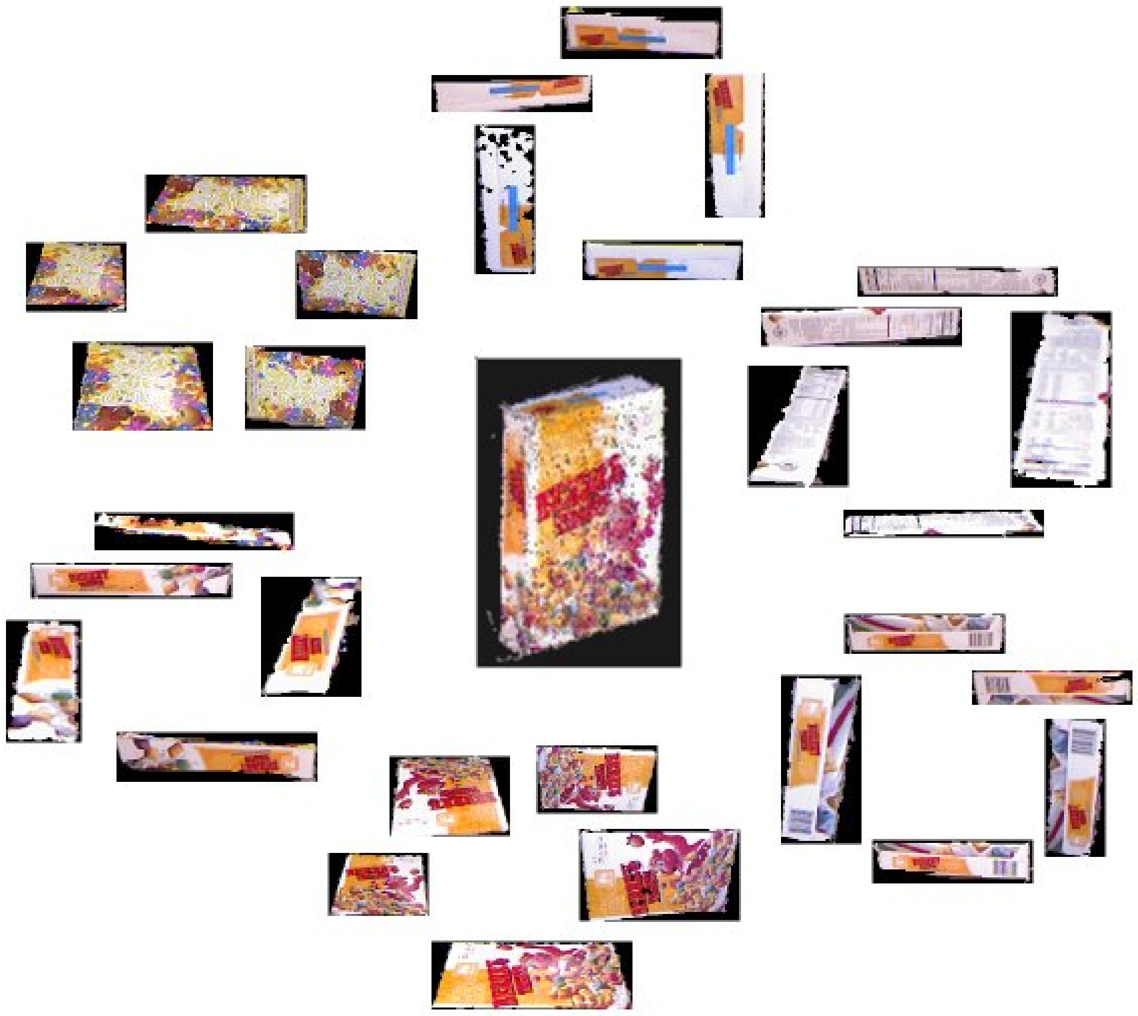


Figure 6: A common cereal box and images of its smooth 3D surfaces from different views

In the representation of an object, after surface segments are obtained from all images of different views, keypoint matching is then conducted to establish correspondence among surface segments of the same 3D smooth object surface in different images. A coarse 3D model of the entire object is next established based on the matching results, and an object frame (i.e., coordinate system) is specified[1]. This is done by applying SVD [33] and RANSAC algorithms. Fig. 7 shows some reconstruction results.

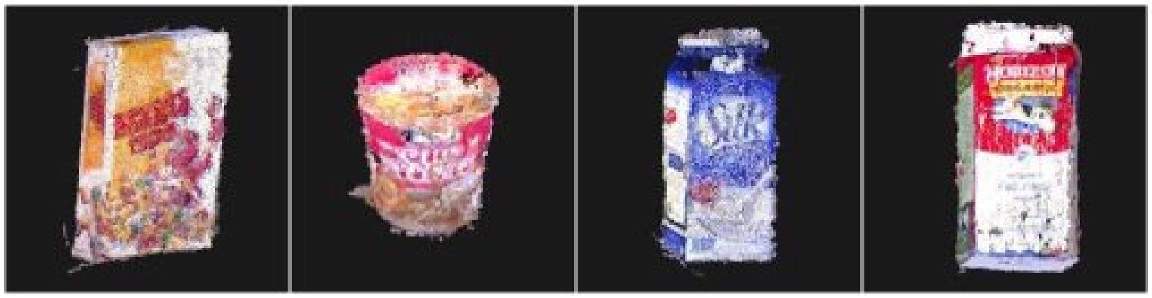


Figure 7: Examples of reconstructed objects, from left to right: cereal box, cup noodle, milk box 1 and milk box 2

4.2 Object Detection and Pose Estimation

To detect objects in a 3D test image, smooth geometric surfaces in the test image are first segmented, and their corresponding visual signatures are described. By applying the *k*-nearest neighbor algorithm [25], surface segments in the test image are identified and labeled by the objects they belong to respectively even if they are partially occluded.

Once object labels are provided to the surface segments in a test image, the pose of an object can be estimated in terms of a homogeneous transformation matrix with respect to the camera coordinate system. For each surface segment with the detected object label in the test image, our algorithm searches for the most similar image of the surface segment with the same object label in the object database. It

[1]Note that every surface segment has its own frame w.r.t. the object frame.

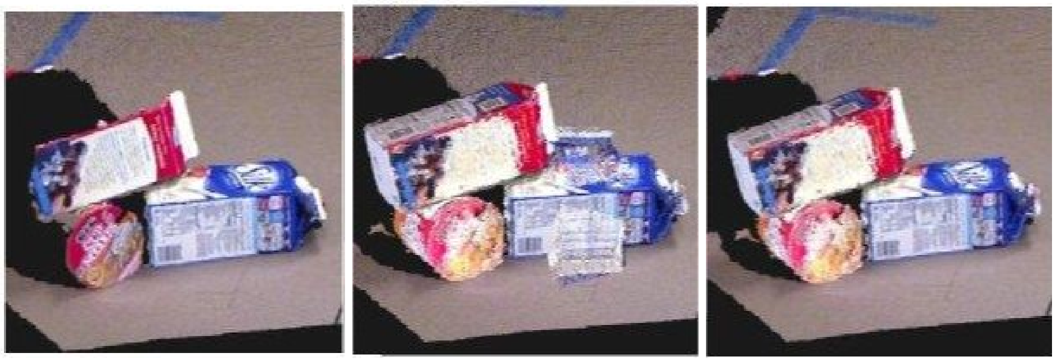
then matches the keypoints between the surface segments in the two images to estimate the pose of the corresponding object in the test scene. The ICP algorithm is further used to fit the predicted object model under the estimated pose to the corresponding actual point cloud in the test image to achieve a more accurate pose estimation.

To evaluate how well a predicted object model fits the surface segment in the test image, we introduce a *fitness cost* measure as a distance function between the point cloud of the surface segment s and the corresponding point cloud of the projected object model $M(s)$. For each point q_v in the point cloud of the surface segment s ($v = 1, \cdots, N_q$), we can find a matched nearest point $p(q_v)$ in the point cloud of the projected object model $M(s)$, and the fitness cost is computed as follows:

$$fc(s, M(s)) = \frac{1}{N_q} \sum_{i=1}^{N_q} ||q_v - p(q_v)||, \tag{2}$$

where N_q is the total number of the points in the surface segment s.

Multiple surface segments of the same object can provide different object pose estimates. If these estimates are quite similar, i.e., consistent, we can simply choose one of them as the object pose. However, with small segments, sometimes the pose estimates are quite different and inconsistent. In such a case, we use the fitness measure to choose the correct object pose estimate as the one with better fitness values from all segments. Fig. 8 shows an example, where two surface-based pose estimates for the blue milk box were inconsistent, and the best-fit predicted object model was chosen. Note that the viewpoint used to display the reconstructed scene is different from the viewpoint of the original test image, as evident from the red milk box – two sides of the red milk box not visible in the original test image were shown in the reconstructed images.



(a) (b) (c)

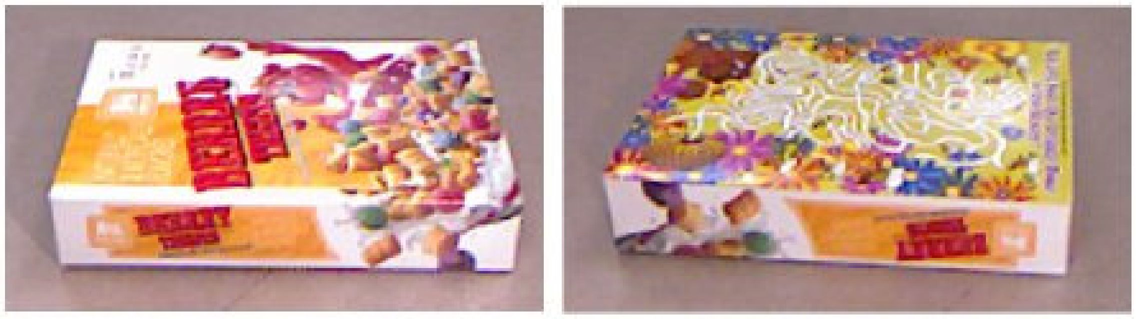
Figure 8: An example of handling inconsistent pose estimates: (a) original scene captured by a RGB-D camera, (b) intersecting projections of the model for the blue milk box at two inconsistent, initial object pose estimates, (c) object models at the best-fit pose estimates

4.3 Comparison of Two Types of Keypoints Matching

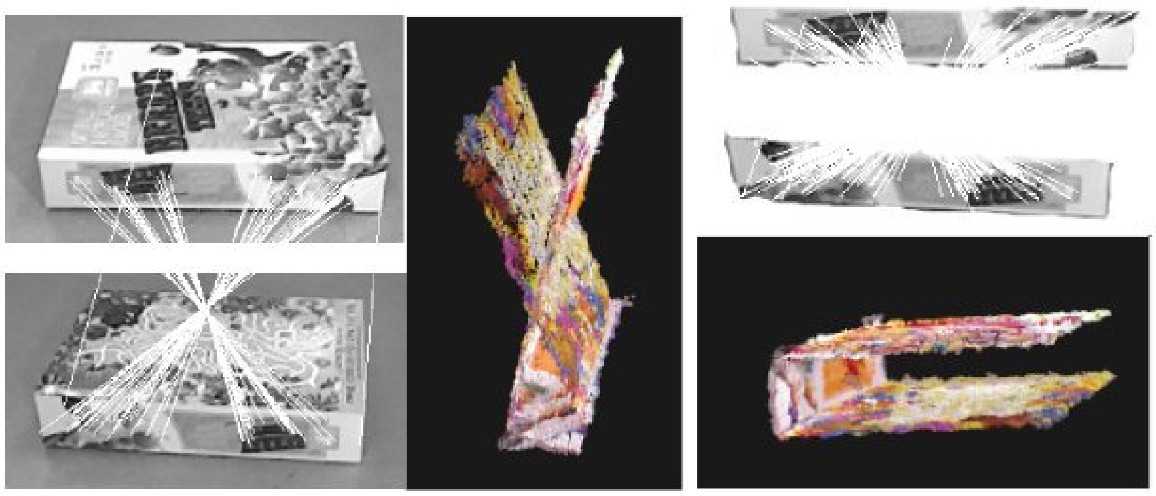
It is important to emphasize the major difference between our method for pose estimation and existing literature: we match keypoints between each pair of corresponding surface segments of two images respectively rather than between the two entire images directly. By providing corresponding surface segments as inputs to the matching algorithm, our method provides more accurate results of matching and subsequent object pose estimation.

First, for a partially occluded object, by focusing on its visible surfaces, we can provide a substantially greater number of valid matches of keypoints as input to the SVD algorithm to estimate the object. It is thus less likely to result in wrong registration or insufficient matched keypoint pairs for registration.

Fig. 9 shows an example: (a) shows the two images used to model the cereal box,



(a)



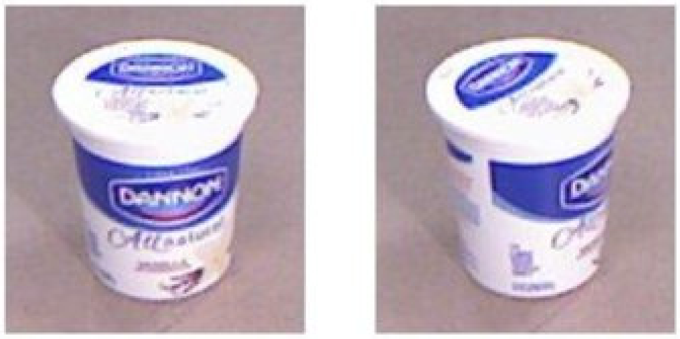
(b) (c)

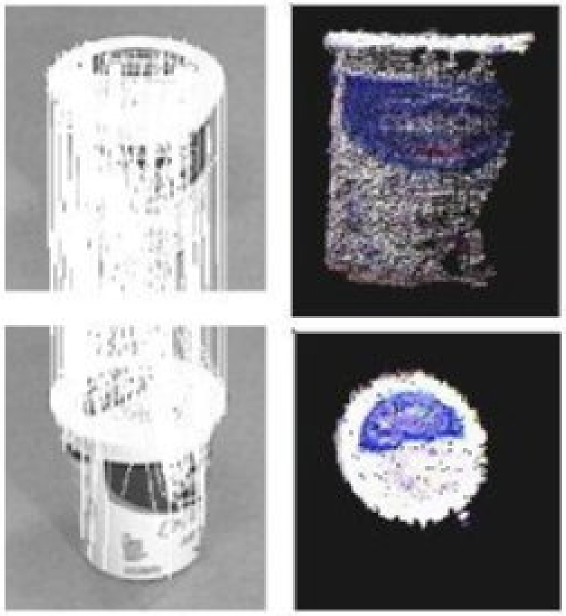
Figure 9: Comparison of keypoints matching and pose estimation results: (a) the original images for matching, (b) the matched keypoints and the result of estimated object model based on matching the whole images, and (c) the matched keypoints and the result of the estimated object model based on matching the common side surface visible in both images

(b) shows the keypoints matched based on the two whole images and the estimated object model, which suffers the singular problem, and (c) shows the keypoints matched based on the side surface detected in both the images and the estimated object model. Note that since only three surfaces of the cereal box can be seen in the two images, the result of estimation consists of just the three surfaces. Clearly, through focusing on surfaces, our algorithm obtained more correctly matched keypoints pairs and subsequently a much better estimation result. This is because both images are resized to the same size initially for keypoints extraction and matching. As the result, small surface segments obtain better resolutions after being resized when matching is done between only the surface segments. However, this does not work for matching based on the whole images.

Second, by extracting keypoints on the target object surfaces instead of an entire image, our method avoids the extraction of invalid keypoints, such as those on the boundaries between the target object and other objects or on the background in the image of a cluttered scene. For objects of regular and symmetric shapes with fewer discrimitive visual features on the boundaries, matching errors caused by invalid keypoints are very likely if an entire image is used for matching. Even if some sliding window is used to localize the target object, with objects occlude one another in a complex scene, matching errors can still occur. Our approach effectively avoids such errors by focusing on each surface segment itself, so that both the extraction and description of keypoints are not affected by anything outside the surface segments.

Fig. 10 shows an example: (a) shows two images used to model the yogurt cup,


(a)


(b)

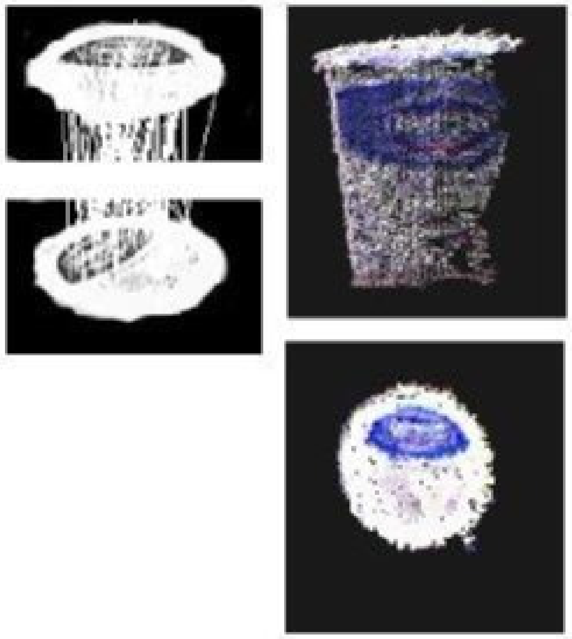

(c)

Figure 10: Comparison of keypoints matching and pose estimation results: (a) the original images for matching, (b) the matched keypoints based on the whole images and the result of estimated object model displayed in front and top views, (c) the matched keypoints based on matching the common top side surface visible in both images and the result of the estimated object model displayed in front and top views

(b) shows the keypoints matched based on the two whole images and the estimated object model displayed in both front and top views, and (c) shows the keypoints matched based on the top surface detected in both original images and the estimated object model displayed in both front and top views. From Fig. 10(b), we can see that the reconstructed model of the yogurt cup is quite distorted (from the front view image) and the blue spindle shape on the top surface is also clearly distorted (from the bottom view image). Whereas, the results in Fig. 10(c) are more accurate. By focusing on surfaces, our approach avoided a great number of incorrectly matched keypoint pairs and subsequently achieved a much better estimation result.

4.4 Detection of Multiple Instances of the Same Object

One important advantage of our approach is that it can directly detect multiple instances of the same object conveniently even if the object instances are occluded in different ways. In contrast, most existing approaches use the Hough voting or pose clustering schemes to determine object instances based on votes, which require setting some threshold values and can be subject to noise, especially for heavily occluded objects.

4.5 Experiments and Analyses

We have built an object database of four objects, including a cereal box, a cup noodle, and two different milk boxes, each from 24 images of different views. The four objects are shown in Fig. 11.

To verify the performance of our approach efficiently, we have also created 15

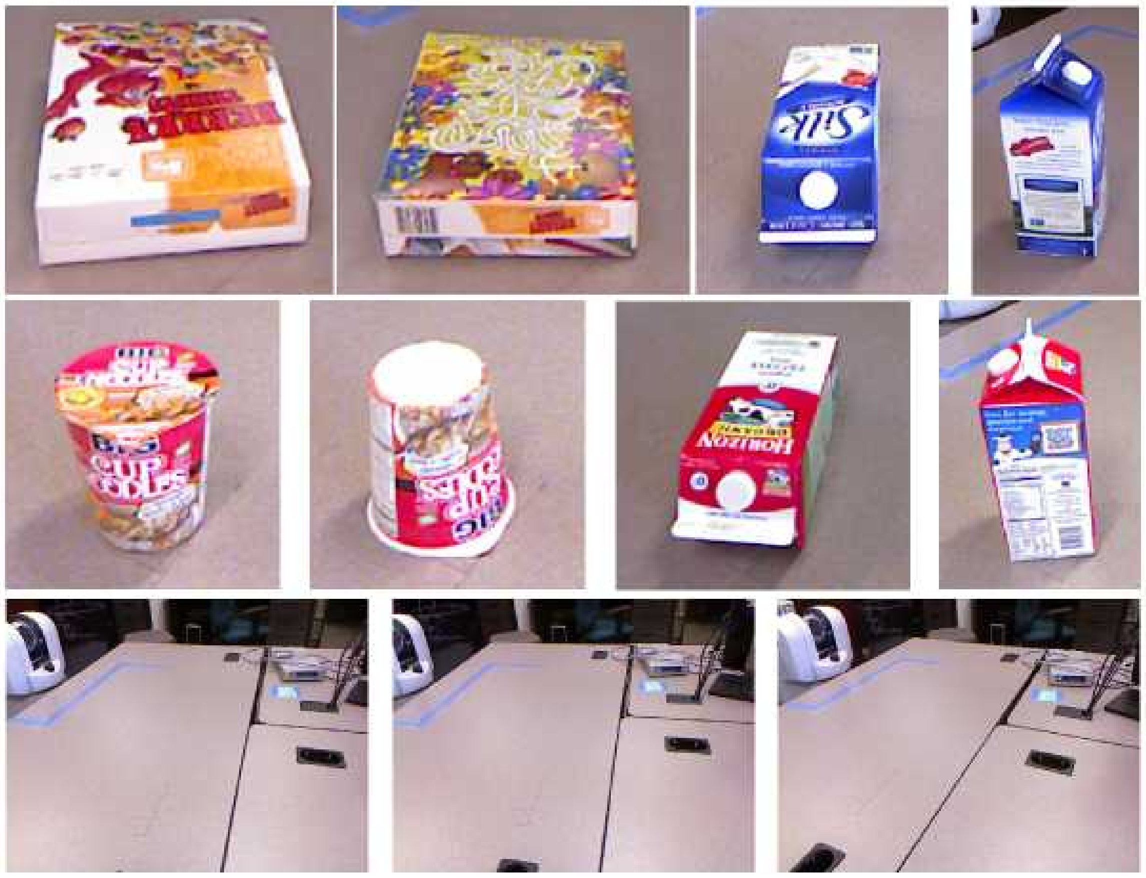


Figure 11: Examples of images in the dataset: top two rows show some images of the four objects used for training; bottom row shows some background images for training

different test images of cluttered environments, which include different objects or multiple instances of the same object. Each image includes 3–6 mutually occluded objects or object instances, as shown in the first column of Fig. 12.

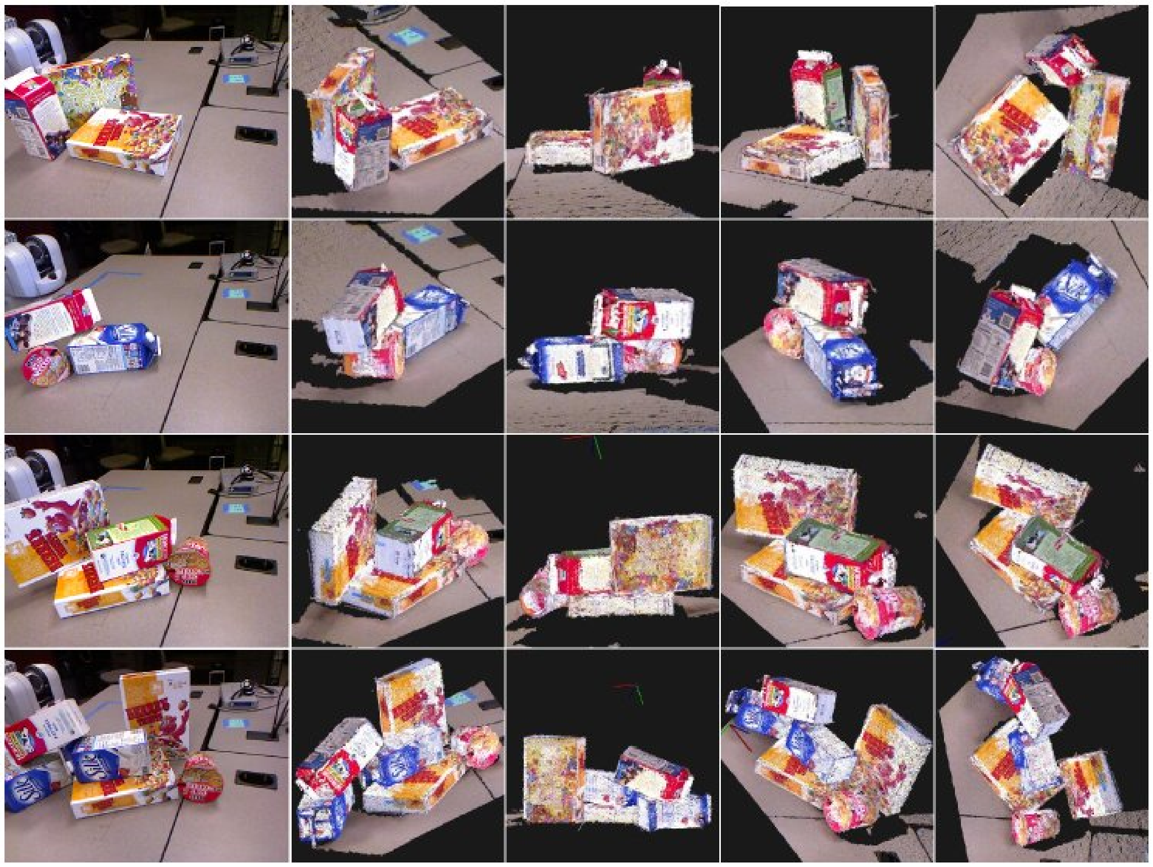


Figure 12: Examples of object detection and pose estimation results: each row starting with an original test image followed by four images from left, back, right, and top views that display the corresponding reconstructed scene

Fig. 12 shows the results of our approach for object detection and pose estimation for the example test images. The 1st column shows four test images captured by a RGB-D camera. The 2nd–4th columns display the corresponding object detection and pose estimation results in four different views: left, back, right, and top views. By localizing all the known objects in the current scene and estimating their poses correctly, our algorithm reconstructed the whole scene based on a test image from a single viewpoint, as shown in the images of the reconstruction results from multiple views.

As described above, we use the k-nearest neighbor algorithm to recognize all the detected surface segments in the testing scene and then estimate their poses. A surface segment is labeled by the most voted label among its most similar k surface segments in the training dataset. Euclidean distance is used here as our distance metric in the HSV color and the ASIFT feature spaces.

Selecting a proper value for k is essential for the performance of the recognition of the object surface segments. Fig. 13 and Fig. 14 show the precision-k curves and the recall-k curves for all the objects in the dataset. We can see that for the value of k in the range of 5–10, both precision and recall rates are quite high for all known objects in the dataset. In our experiments, we set $k = 10$.

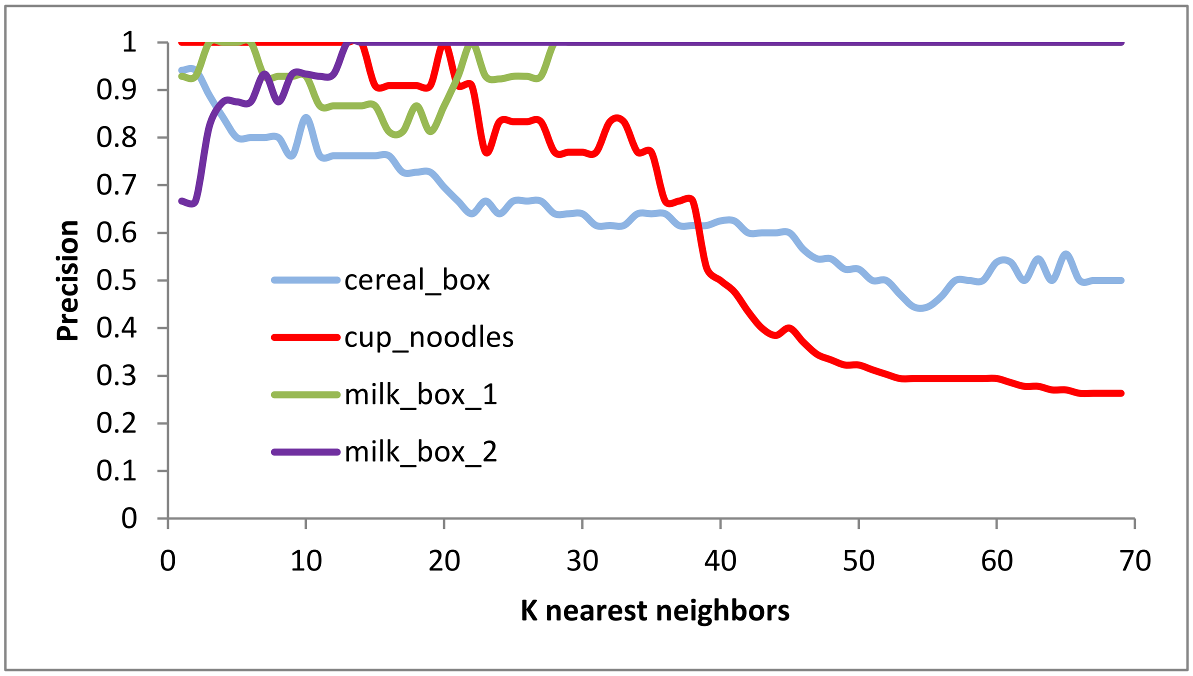


Figure 13: The precision-k curves for all the objects in the dataset

On the other hand, for very large k values, labels with most samples in the train-

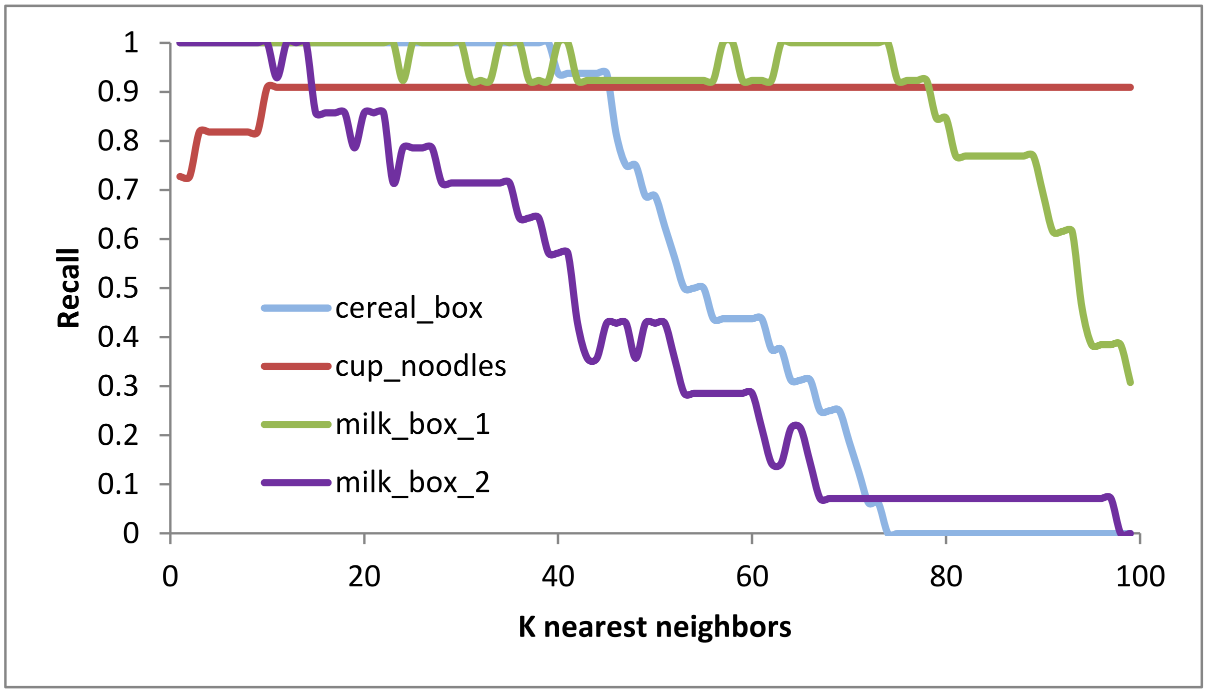


Figure 14: The recall-*k* curves for all the objects in the dataset

ing dataset tend to dominate the prediction of the labels for test samples. Since we had more background surface segments than object surface segments for each object used for training, with very large *k* values, the recall rates for all the objects in the dataset eventually approach zero, because segments of background tend to donimate the results of labeling. For *k* values set in the middle range of the charts, objects with more surface segments tend to affect the labeling of surfaces with similar features. For example, our experiments show that surfaces of milk box 2 tend to be misclassified as the cup noodle due to similar color features for $k > 10$.

Table 3: Detection and pose estimation (PE) results with *k*=10

	cereal box	cup noodles	milk box 1	milk box 2
#Instances	16	11	12	14
Correct PE	14	9	9	13
Wrong PE	1	1	0	0
Missing	1	1	3	1
Fitness(mm)	5	2.9	6.1	6.1

Table 3 shows the detection and pose estimation results for all the objects in our dataset with $k = 10$. Most of the object instances are detected with pose estimation correctly. Note that the fitness shown under each object is the average of fitness values associated with all tested instances of the same object, where for each object instance, the fitness value is the average of fitness values associated with all the surface segments that correctly detect the object. The accuracies of each average fitness is quite consistent with the accuracy of the raw data captured by Microsoft Kinect, which is about $\pm$3mm for objects within 1m from the camera. For the few missing and wrong detections, possible reasons include: wrong surface segments were detected based on visual features, or the detected surface segments were too small to provide sufficient information for generating reasonable poses.

4.6 Conclusions of this Chapter

To conclude, in this chapter, we have introduced a new appearance-based approach for general 3D object detection and pose estimation based on segmented 3D surfaces and their features, taking full advantage of RGB-D information. Our approach can detect and estimate the poses of occluded known objects, including occluded multiple instances of the same object, effectively in cluttered environments. In the following chapters, we will introduce more objects with more complicated shapes and surfaces to test and further improve and extend our approach; we will also integrate our work with robotic manipulation and motion planning approaches to enable automatic 3D object modeling based on sensing in real-time.

CHAPTER 5: EXTENSION TO SURFACE-BASED 3D OBJECT DETECTION AND POSE ESTIMATION

In this chapter, a significant extension to the work in Chapter 4, is presented [109]. With a new strategy for object model building, we expand the object dataset by including objects with transparent or semi-transparent regions. We also add the algorithm based on graph-cut to remove redundant or incorrect detection results. We further report new object detection and pose estimation results that were not reported in Chapter 4. Moreover, we provide a comparative study of the introduced approach vs. related existing approaches.

5.1 A Robust and Convenient Strategy for 3D Model Building

We now introduce a robust and convenient way to build 3D object models using keypoints from both object surfaces and the environment to register different object views. RGB-D images used for building object models are captured from different camera views of different object poses.

Although a chessboard is widely used to register different views of the same object, it still requires manual annotation to obtain the orientation of the chessboard when the camera view changes dramatically.

Instead, we use two images with rich visual information, called *landmarks*, as shown in Fig. 15, and put them by the side of target object as the reliable and main source of required keypoints in the environment to help registering different views

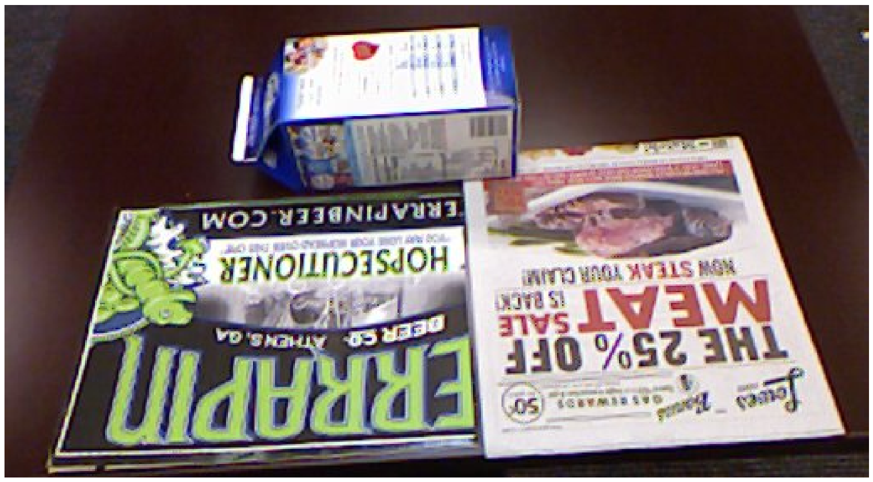


Figure 15: Two landmarks are placed by the side of a milk box to register different object views

of the same object.

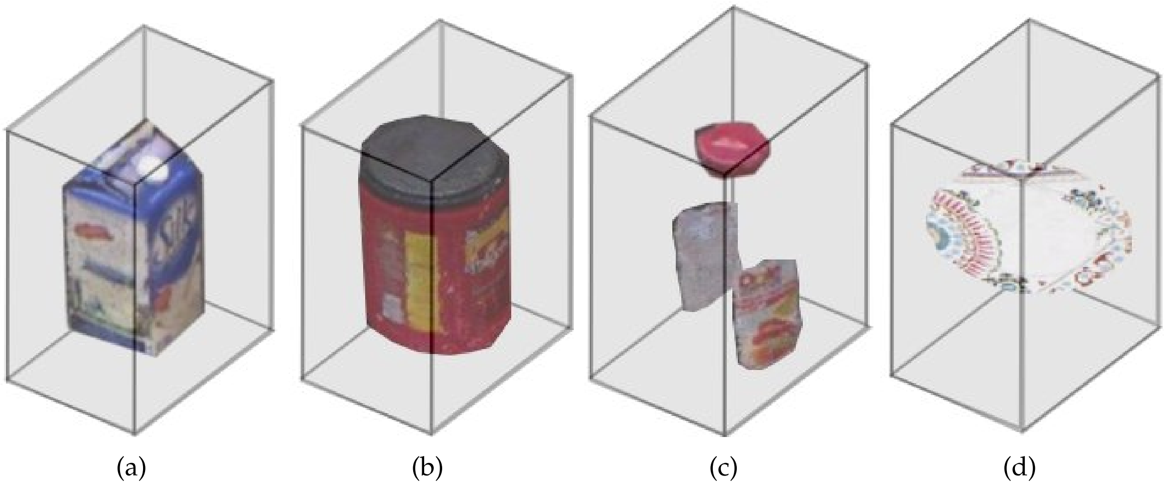


Figure 16: The geometric shape of an object, no matter how complicated, can always be simplified as having up to 6 sides.

For each object, we choose 6 different stable object poses for training, and each pose corresponds to having one of the six sides (see Fig. 16) roughly on top, as shown in Fig. 17.

For each object pose, we capture 5 RGB-D images from different views, as shown in Fig. 18. A RGB-D sensor is always placed on the side of the object where the

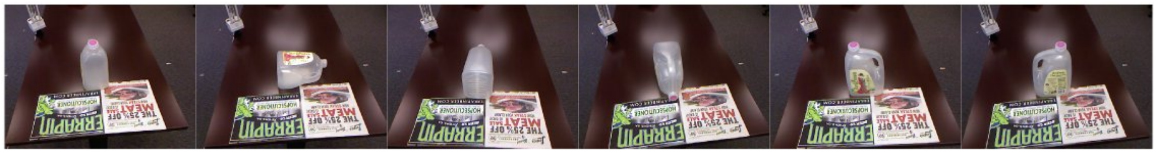


Figure 17: 6 different stable poses used for modelling a juice container

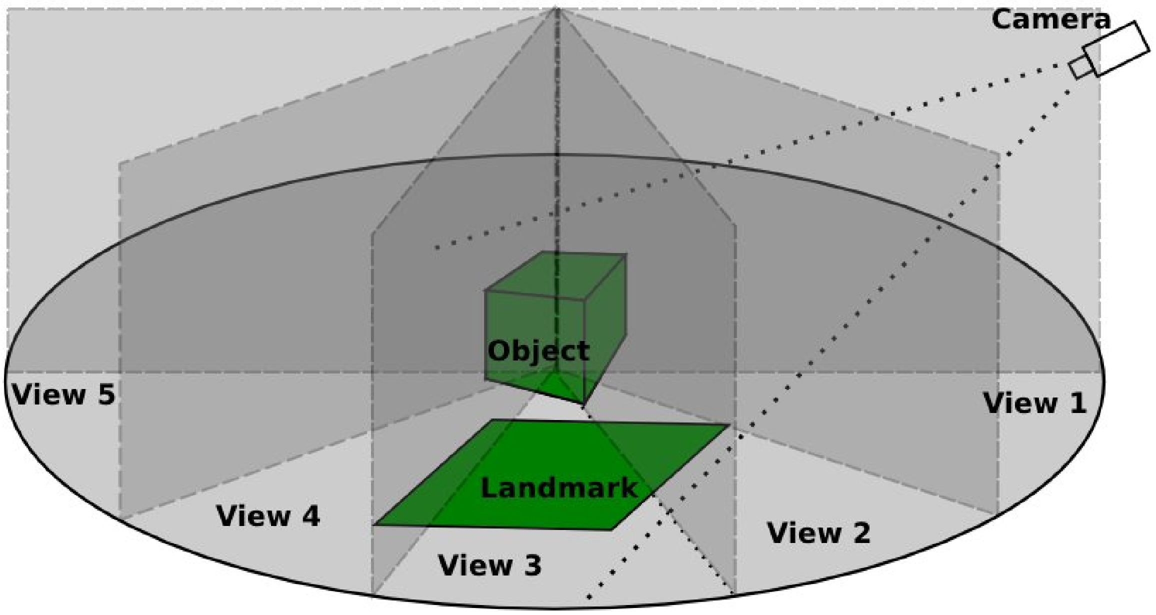


Figure 18: The whole space is roughly divided into 5 even spaces, and for each of them, an RGB-D image is captured for training.

landmarks are placed so that most of the landmark surfaces can be observed all the time. The whole space is roughly divided into 5 equal regions, and for each of them, the sensor is moved above the object and captures an RGB-D image for training. No accurate measurement or set up are required. Fig. 19 shows 5 images captured for modelling a cereal box from different views of a particular pose of the box.

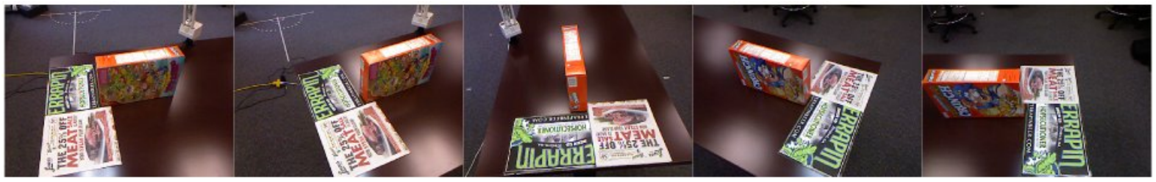


Figure 19: 5 images from different views captured for a single pose of a cereal box

We register different object views as follows for each object:

- For each set of RGB-D images from different camera views of the same object pose, we match keypoints from the *landmarks* in the environment to register different camera views.
- From the set of RGB-D images of different object poses, we choose a pair of images of different poses that have the most matched keypoint pairs on common object surfaces to register different object poses.

We use ASIFT [117] keypoints to register different RGB-D images of the same object. For each object O, a coarse point cloud model of the object M is next established based on the registration results, and an object frame (i.e., coordinate system) is specified for M. This is done by applying SVD [33] and RANSAC [89, 37] algorithms.

We also take images of the background without registering them.

Consider N objects plus the background in our training dataset, denoted as $\Omega^+ = \{O_0, O_1, O_2, \cdots, O_N\}$, where O_0 indicates the background.

Now, suppose N_I is the total number of RGB-D images in our training dataset, each RGB-D image I, $I \in \{I_1, I_2, \cdots, I_{N_I}\}$, is associated with an object (or background) label $L(I)$, $L(I) \in \Omega^+$, and a 4×4 transformation matrix $T(I)$ with respect to the corresponding object frame, so that each training RGB-D image can be characterized as:

$$\{I, L(I), T(I)\}. \tag{3}$$

Since the background does not have a point cloud model, $T(I)$ is assigned $NULL$ if I is an RGB-D image of the background.

Fig. 20 shows all the reconstructed object models in our current dataset. As object models are built under the most common in-door lighting conditions and based on noisy appearance from RGB-D images, our approach is more robust in detecting objects under similar noisy conditions.

Our approach does not require any precise information of object poses and camera poses, and a complete coarse object model can be achieved easily (from the point clouds that an RGB-D camera can capture). It thus facilitates autonomous object model building.

5.2 Surface-based Object Representation and Classification

On the basis of all RGB-D images captured from different views of different poses for each object, we next establish an appearance-based representation of the

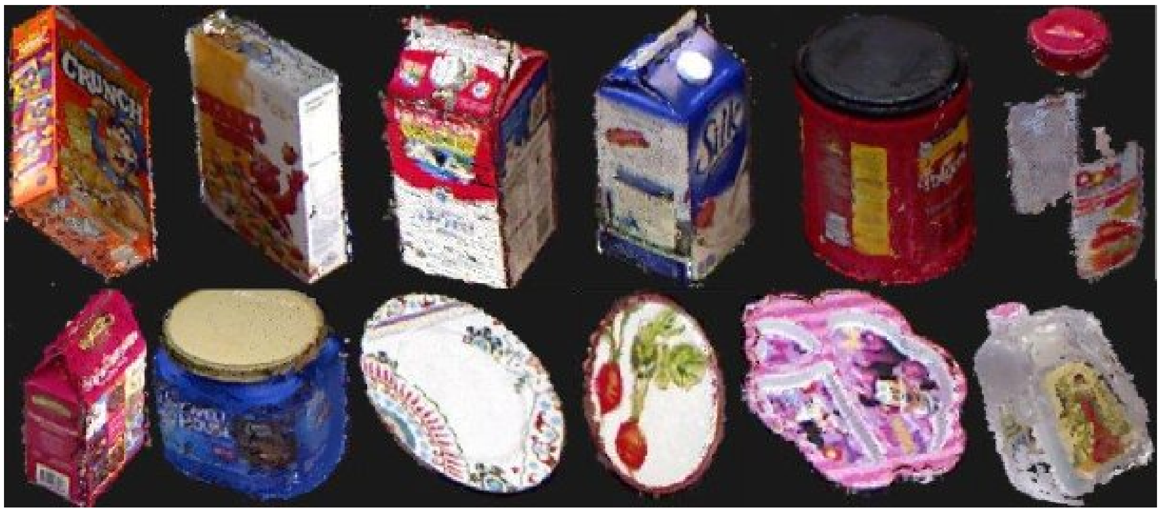


Figure 20: All the reconstructed object models in our current dataset, including objects with transparent or translucent surfaces (shown in the last column)

object. First, the target training object is segmented out from the background in each object RGB-D image. Next, the target object is further segmented based on smooth surfaces, where a smooth surface is defined by the continuity of depth and surface normal values, and its boundary is characterized by depth and surface normal discontinuity [106]. Each RGB-D image of the background is directly segmented into smooth surface patches.

Then, each object (or background) is described in terms of surface segments from different views of different poses and the visual signatures of the surface segments. We use both HSV color histogram and ASIFT keypoint histogram [117] as the visual signatures because of their robustness to viewpoint and illumination changes.

Let N_S be the total number of surface segments from all images in the training set, where each surface segment S, $S \in \{S_1, S_2, \cdots, S_{N_S}\}$, belongs to a unique training RGB-D image $I(S)$, $I(S) \in \{I_1, I_2, \cdots, I_{N_I}\}$. Then, each training surface segment can be characterized by:

$$\{S, L(I(S)), T(I(S)), h(S), a(S)\}, \tag{4}$$

where $h(S)$ is the visual feature in terms of the extracted HSV color histogram of S, and $a(S)$ is the visual feature in terms of the extracted ASIFT keypoint histogram of S.

As an example, Fig. 21 shows a common cereal box and its smooth 3D geometric surfaces in terms of the corresponding image segments from different views of different poses.

Note that our object representation only uses a segmentation of geometric surfaces obtained automatically, which makes it flexible to model daily objects with irregular shapes other than those of special shapes, such as cuboid or cylinder, and to tolerate small bulges on a surface.

Using all surface segments $S_1, \cdots, S_{N_S}$ and all the corresponding features, we train a multi-class classifier based on support vector machines (SVMs) [19], which can be used to classify the surface segments of each test scene to different objects.

5.3 Object Detection and Pose Estimation Using Graph-cut

In this section, we explain our surface-based approach for 3D object detection and pose estimation in detail.

5.3.1 Initial Detection and Estimation

To detect objects from an RGB-D test image, smooth geometric surfaces in the test image are first segmented [34], and their corresponding visual signatures are

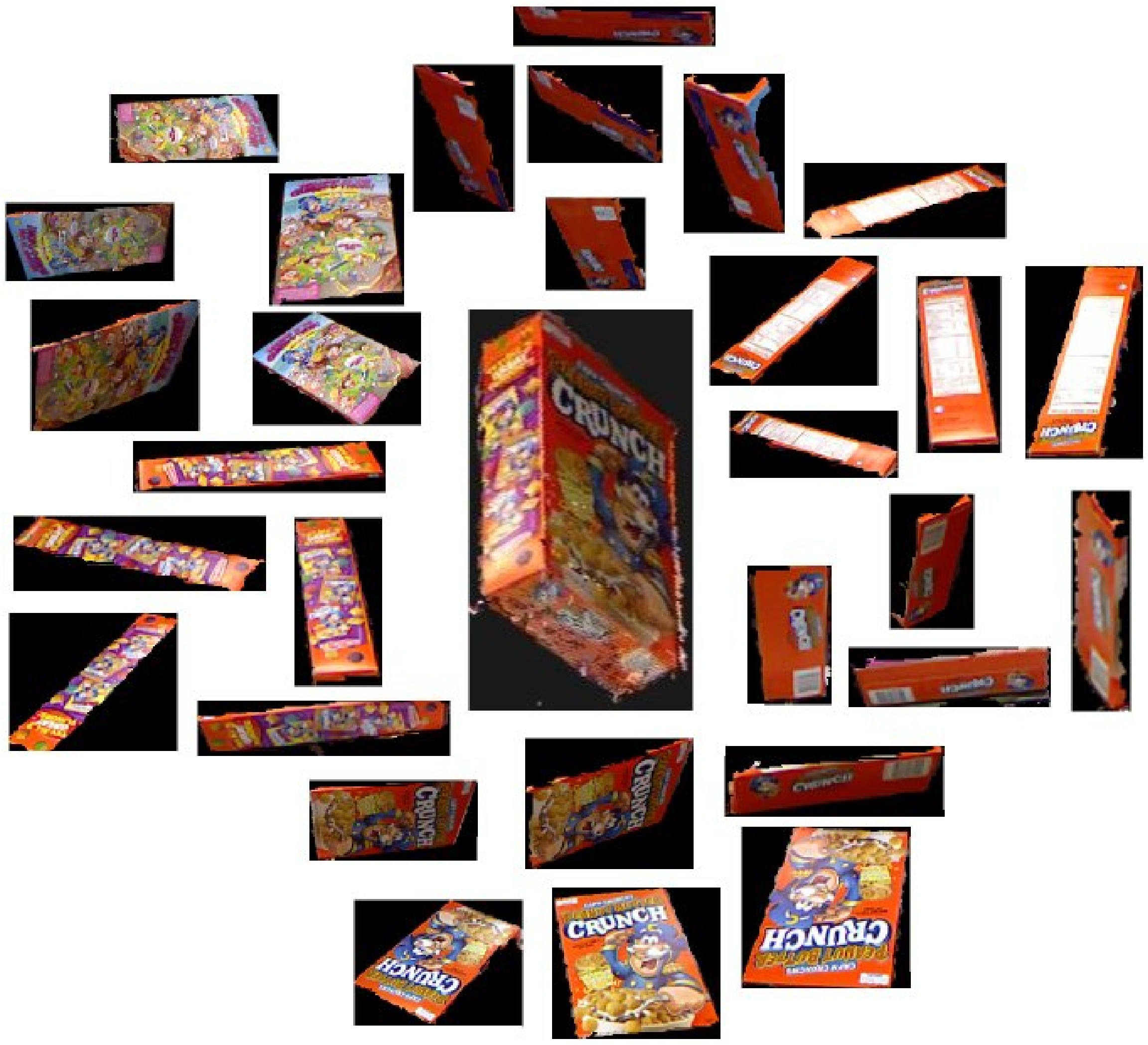


Figure 21: A common cereal box and images of its smooth 3D surfaces from different views of different poses

described. Next, surface segments in the test image are recognized by the trained classifiers based on their visual signatures and labelled by the corresponding objects respectively.

Suppose s is one of the segmented smooth surfaces from the test image I', with its visual signature described as $h(s)$ and $a(s)$, the normalized recognition probabilities for each object (or background) O, $O \in \Omega^+$, returned by the trained multi-class SVMs classifier is denoted as $P(O|h(s), a(s))$, and then the surface segment s is labelled as belonging to the object (or background) label $O(s)$ corresponding to the greatest probability:

$$\begin{aligned} O(s) &:= \arg\max_{O} \; P(O|h(s), a(s)), \\ s.t. &\qquad O \in \Omega^+, \end{aligned} \tag{5}$$

Once the surface segments of a test image are assigned object labels, the 6DoF poses of the corresponding predicted objects can be estimated in terms of homogeneous transformation matrices with respect to the camera coordinate system in the following steps.

First, suppose the surface segment s is recognized as belonging to one of the training objects, $O(s) \in \{O_1, O_2, \cdots, O_N\}$, then the most similar surface segment $S(s)$ with the same object label in our training dataset in terms of ASIFT keypoint histogram can be obtained by:

$$\begin{aligned} S(s) &:= \arg\min_{S} \; Dist(a(s), a(S)), \\ s.t. \quad & L(I(S)) = O(s), \\ & S \in \{S_1, S_2, \cdots, S_{N_S}\}, \end{aligned} \tag{6}$$

where $Dist(a(s), a(S))$ is the Euclidean distance between the ASIFT keypoint histograms $fs(s)$ and $fs(S)$.

It then matches the ASIFT keypoints between the surface segments s and $S(s)$. If a sufficient number of matched keypoint pairs are found, our algorithm predicts a candidate object model $M(s)$ with the object label as $O(s)$, and its pose $H(M(s))$ is estimated by combining two 4×4 transformation matrices: (1) $T(I(S(s)))$, which indicates the transformation from the camera frame of the surface segment $S(s)$ to the basic frame of the object model $M(s)$, and (2) a transformation matrix $\Theta(s, S(s))$, obtained based on the matched keypoint pairs by using SVD [33] and RANSAC [89, 37] algorithms, which indicates the transformation from the camera frame of the surface segment s to the camera frame of the surface segment $S(s)$, so that

$$H(M(s)) = \Theta(s, S(s))^{-1} T(I(S(s)))^{-1}. \tag{7}$$

Finally, the pose $H(M(s))$ indicates the transformation from the basic frame of object model $M(s)$ to the current camera frame of the surface segment s in the test image.

If necessary, the ICP algorithm [11] can be further utilized to fit the point cloud object model under the estimated pose $H(M(s))$ to the corresponding actual point cloud in the test image I' to achieve a more accurate pose estimation $H(M(s))'$. As the result, a candidate object model $M(s)$ from the surface segment s in the test image I' can be finally described as:

$$\{s, O(s), H(M(s))'\}. \tag{8}$$

5.3.2 Scene Reconstruction

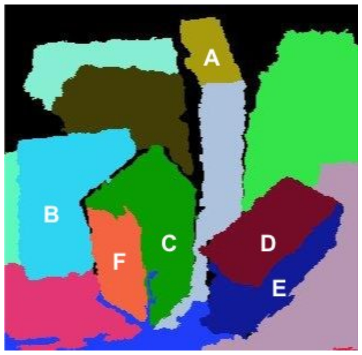
For all surface segments in a test image, which are identified as belonging to some trained objects and successfully assigned corresponding candidate object models as described in Section 5.3.1, we next reconstruct the scene of the test image to validate all the recognized objects and their estimated poses as follows.

To reconstruct the scene, we project all the predicted object models back to the original test image. This can be done very easily since the pose of each predicted object model indicates the transformation from its own object coordinate system to the camera coordinate system of the test image.

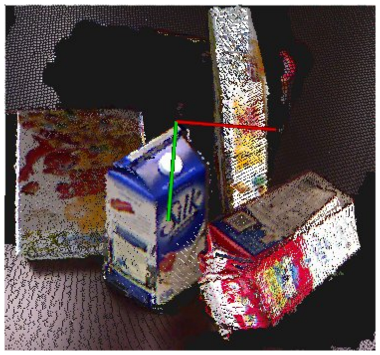
Then, we check if each predicted object model fits its own surface segment well. We introduce a rough threshold value $\delta = 10\ mm$ for the fitness cost, as defined in Section 4.2. If the fitness cost of a predicted object model is greater than δ, then it is ruled out as incorrect estimation and is abandoned. Our experimental results show that the threshold value δ is mainly related to the accuracy of the sensor (e.g., Microsoft Kinect) and the SVD algorithm.

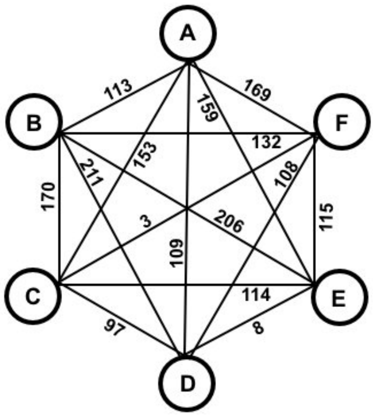
Fig. 22 (a) $\sim$ (c) illustrate the step-by-step procedures of the initial scene reconstruction described above: (a) shows the original test image; (b) shows the coarse segmentation result based on smooth surfaces; as shown in the figure, surface seg-

(a) The original test image

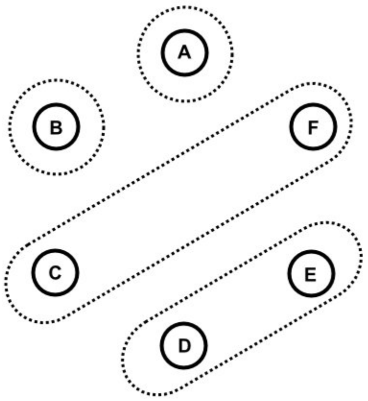


(b) The coarse segmentation result


(c) The initial reconstruction result



(d) The complete graph



(e) The graph cut result

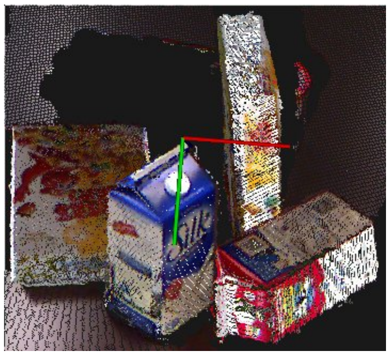

(f) The refined reconstruction result

Figure 22: An example of selecting the best object model using a graph-cut based algorithm: in (b) and (c), both surface segments *C* and *F* predict the same object model of the blue milk box with very similar pose estimates, while surface segments *D* and *E* also predict the same object model of the red milk box but with inconsistent pose estimates. After applying the result of our graph-cut algorithm, we can see that the previous red milk box with inaccurate pose estimate in (c) is eliminated in (f).

ments $A \sim F$ successfully predict various object models, and the fitness costs of those object models to their corresponding surface segments are within the threshold value δ according to their initial pose estimates; (c) shows the scene reconstruction result after the initial object detection and pose estimation.

Note that an object instance usually has multiple surface segments visible in a test image so that some redundant object models of the same object instance are in the result. For example, in Fig. 22 (b) and (c), both surface segments C and F predict the same object model of the blue milk box with very similar pose estimates so that their projections back to the original test image coincide and look like just one object. However, although both surface segments D and E also predict the same object model of a red milk box, the pose estimates for the model based on D and E are not very consistent, and there are obvious distortions on the top side from two predicted red milk boxes after we project both of them back to the original test image.

Hence, we propose a general graph-cut based algorithm to remove redundant object models, no matter if they provide similar or inconsistent pose estimates.

First, our method builds a graph based on all the predicted object models and their corresponding surface segments after the fitness cost of each predicted object model is validated. As shown in the example in Fig. 22(d), each surface segment represents a node in the graph; every node is connected to every other node to form a complete graph. The complete graph enables our algorithm to apply also to disconnected and partial surface segments, which are usually caused by either occlusion (so that only disconnected surface patches are visible) or transparency (so that the transparent region of a surface is not visible).

The weight of each edge that connects two nodes is assigned as follows. Suppose there are two nodes A and B, which represents their corresponding surface segment s_a and s_b, respectively, and their corresponding predicted object models are denoted as $M(s_a)$ and $M(s_b)$. Now, the weight $w(A,B)$ of the edge that connects nodes A and B is defined as

$$fc(s_a, s_b, M(s_a)) = \max\left(fc(s_a, M(s_a)), fc(s_b, M(s_a))\right), \tag{9}$$

$$fc(s_a, s_b, M(s_b)) = \max\left(fc(s_a, M(s_b)), fc(s_b, M(s_b))\right), \tag{10}$$

$$w(A,B) = \min\left(fc(s_a, s_b, M(s_a)), fc(s_a, s_b, M(s_b))\right), \tag{11}$$

where $fc(s_a, s_b, M(s_a))$ and $fc(s_a, s_b, M(s_b))$ are the fitness costs of using the object models $M(s_a)$ and $M(s_b)$ to fit both surface segments s_a and s_b, respectively. Therefore, $w(A,B)$ indicates the best fitness (minimal fitness cost) if both surface segments s_a and s_b share the same predicted object model.

Once the graph is built, we use the graph cut algorithm [36] to group the surface segments that belong to the same object instance. The only required threshold function τ in [36] is set as:

$$\tau = \frac{\delta}{|\Phi|}, \tag{12}$$

where δ is the threshold for fitness cost introduced above, and $|\Phi|$ denotes the number of the nodes in the group Φ.

Fig. 22(e) shows the grouping results of surface segments using the graph built in Fig. 22(d). Based on the grouping results, for all surface segments that are grouped to indicate the same object instance, only the object model predicted from one surface segment that has the minimum fitness cost of fitting into all these surface segments in the group is kept, and the predicted models from the rest of the surface segments are discarded.

To summarize, for all predicted object models of the same object instance, only the one with the best fitness cost is chosen and the rest are discarded.

Fig. 22(f) shows the final refined result of object detection and pose estimation from Fig. 22(c). After applying the graph-cut algorithm, we can see that the pose inconsistency of the previous red milk box is resolved with the incorrect pose eliminated.

5.4 More Experiments and Analyses

In this section, we first provide more detailed description of the training process for establishing our object models. Then, we briefly introduce our test scenes and test images. We next compare the recognition results of all test surface segments from all test images with the combined detection and pose estimation results of all the object instances and discuss why the latter has much higher accuracy. We further show example results of scene reconstruction and discuss time efficiency of our approach. Finally, we compare the performance of our approach to the MOPED approach [23] using two different clustering algorithms: one is Mean Shift clustering, and the other is Projection clustering with Q-Score ranking, which is the overall best-performer in [23], and discuss the advantages and disadvantages of each approach.

5.4.1 Training

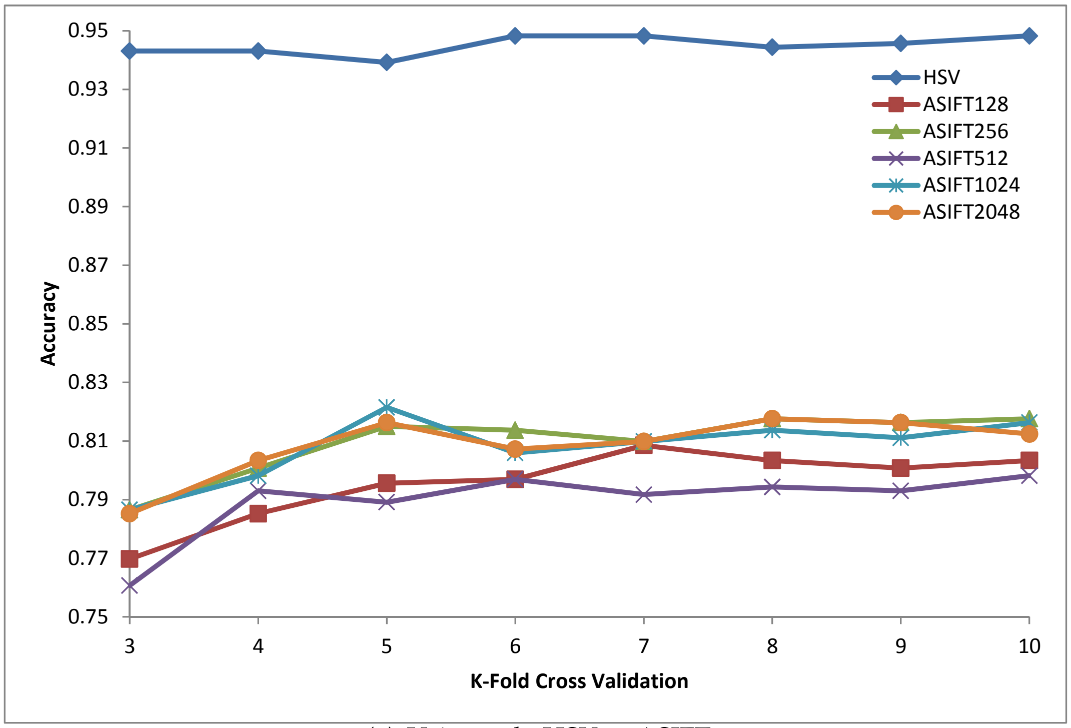
As described in Section 5.1, we capture 5×6 RGB-D images for each object and 12 RGB-D images for the background for training. Coarse point cloud object models and surface-based object representations are generated subsequently.

We use a k-dimensional bag-of-words ASIFT histogram as one of the visual signatures for each object surface segment. First, we cluster all training ASIFT descriptors using the k-means algorithm, where k is the number of clustering centroids (or codewords). In order to select a proper value of k for good recognition performance, we tested five different values of k: 128, 256, 512, 1024, and 2048.

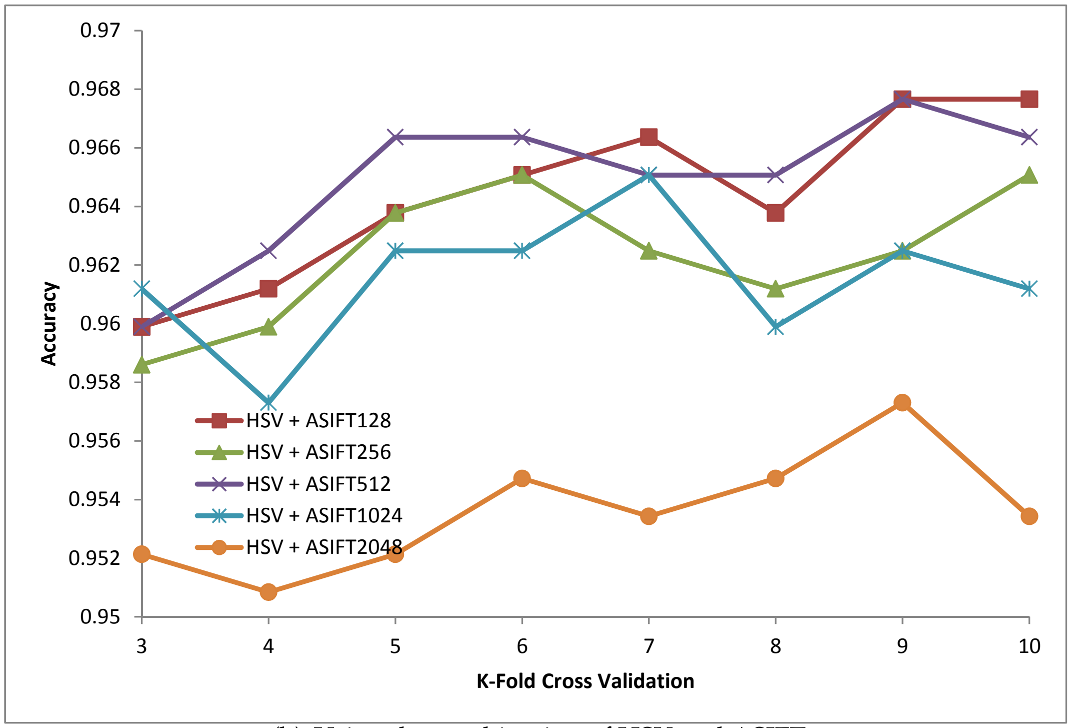
We use the SVMs and apply the kernel based on the radial basis function to them as the classification model for the recognition of the object instance from a surface segment. We also use K-fold nested cross validation in the training of the SVMs. Fig. 23(a) shows optimal accuracies achieved by the trained SVMs classifiers using only HSV or ASIFT visual signature as the number of folds K changes in cross validation, while Fig. 23(b) shows optimal accuracies achieved by the trained SVMs classifiers as K changes using the combination of HSV and ASIFT visual signatures for classification. Based on the performances, $k = 512$ is chosen as the dimension of the ASIFT histogram; $C = 6.73$ and $\gamma = 0.04$ are chosen as the SVMs training parameters.

5.4.2 Test Scenes and Images

We have created 17 test sets [105] including a total of 92 RGB-D images, which were captured by Microsoft Kinect from different views of different cluttered scenes.



(a) Using only HSV or ASIFT



(b) Using the combination of HSV and ASIFT

Figure 23: Optimal accuracies achieved by the trained SVMs classifiers using different descriptions of visual signatures as the number of folds K changes in cross validation for different k values

Each test image includes 6 $\sim$ 14 occluded objects or object instances of the same object. Fig. 24 shows examples of test images of different scenes, and Fig. 25 shows examples of test images from different views of the same scene. As the view changes, we can observe the obvious changes of both the illumination and occlusion for each object.

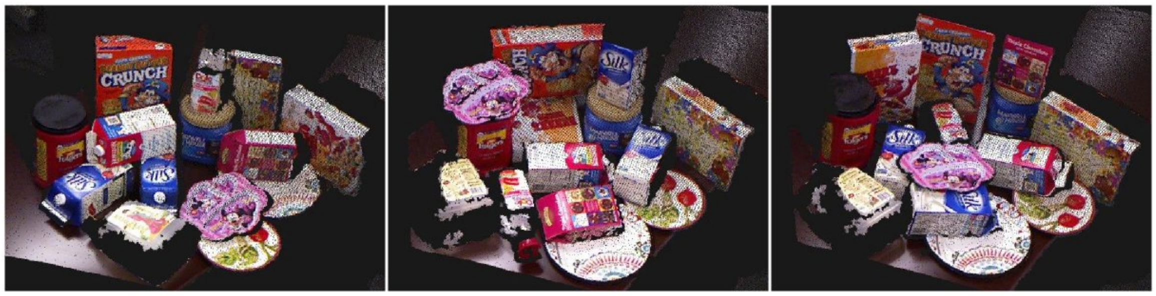


Figure 24: Test images of different scenes

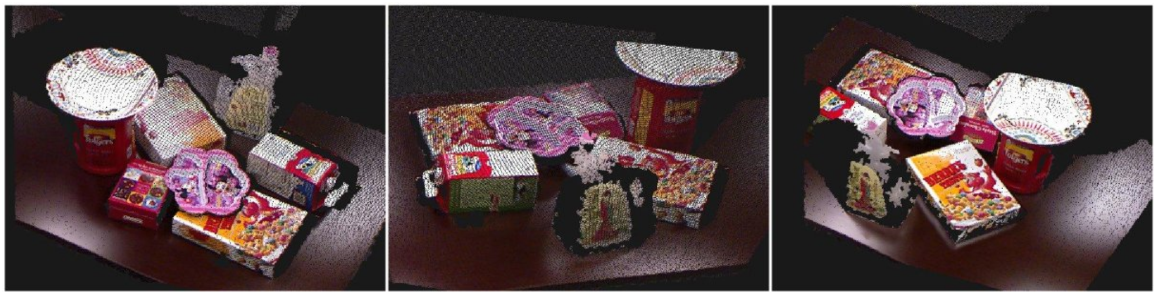


Figure 25: Test images from various views of the same scene

5.4.3 Comparison of the Test Results: Surface Segments vs. Object Instances

For all test images, we always use the same set of fixed parameters and values for smooth surface segmentation. Fig. 26 shows the precision-recall curves based on classifying all test surface segments to object labels in all test images and for all objects in our dataset. As described in Section 5.3.1, for a surface segment from a test image, we always use the object label which corresponds the greatest probability returned by the trained SVMs classifiers as the recognized object label of the test surface segment. This strategy is equivalent to choosing a very small recognition confidence threshold, and the recognition results usually achieve the highest recalls for all objects but with low precisions.

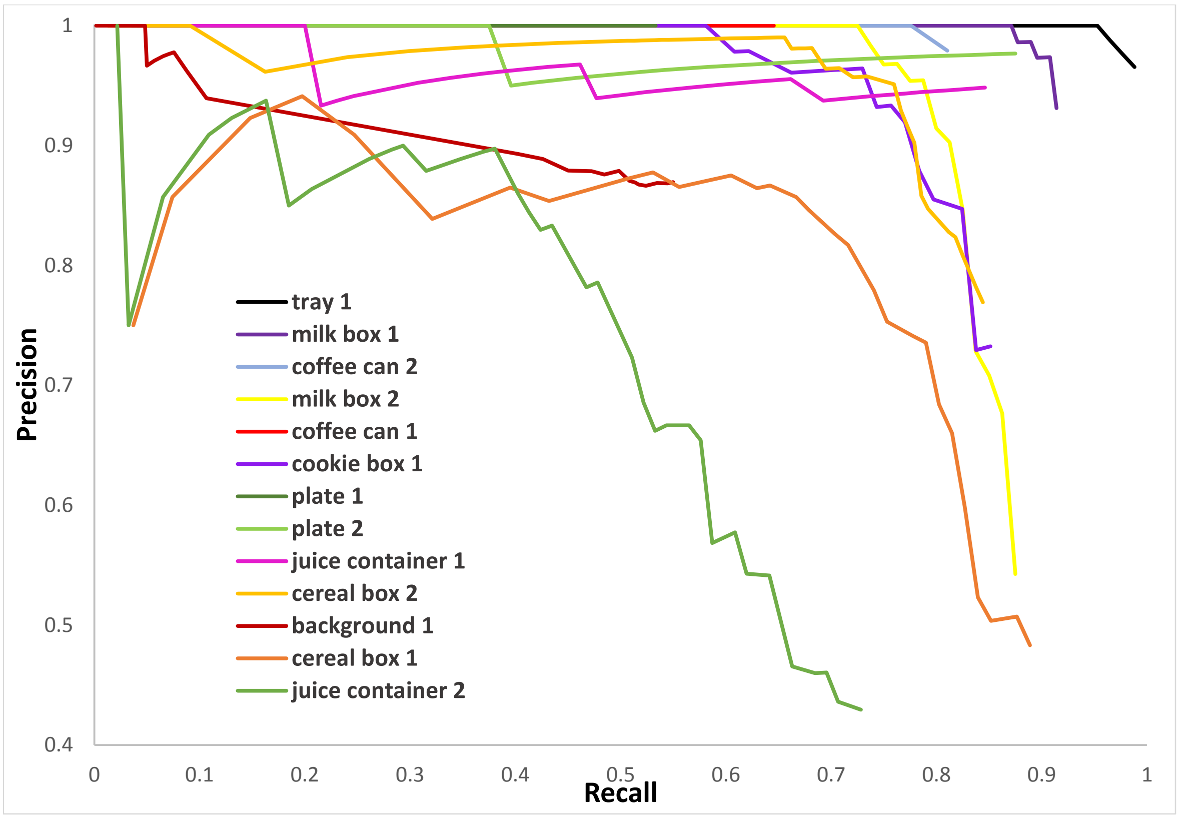


Figure 26: Precision-recall curves based on classifying all test surface segments to object labels in all test images and for all objects in our dataset

Table 4 shows the combined detection and pose estimation results for all the objects in the test images. The 1st column shows all the objects in our dataset; for each object, the 2nd column shows the ground truth of the total number of instances of each object in all the test images; the 3rd column "CRCP" shows the number of correctly detected object instances with correct pose estimation; the 4th column "WR" shows the number of wrongly recognized object instances; the 5th column "CRWP" shows the number of detected object instances which are correctly recognized but either redundant or with incorrect pose estimation; the 6th column shows the average fitness cost values of all the detected instances with correct pose estimation.

Table 4: Combined object detection and pose estimation results

Objects	Total	CRCP	WR	CRWP	Fitness cost (mm)
cereal box 1	52	47	1	0	3.39 ± 1.34
cereal box 2	104	83	1	0	3.60 ± 1.32
cookie box 1	52	46	0	1	2.34 ± 0.85
juice container 1	52	50	0	0	1.77 ± 0.97
juice container 2	52	36	1	2	2.13 ± 0.78
milk box 1	104	83	0	0	3.33 ± 1.62
milk box 2	52	43	0	1	3.23 ± 1.49
coffee can 1	52	32	0	3	2.43 ± 1.42
coffee can 2	52	39	0	5	2.67 ± 1.09
plate 1	52	19	0	0	3.38 ± 1.76
plate 2	52	41	0	1	1.78 ± 0.61
tray 1	52	37	0	0	2.54 ± 1.58

In the total 572 detected object instances in our final test results, only 3 of them are wrongly recognized, even though the precisions of classifying test surface segments to object labels alone in Fig. 26 are relatively low when the highest recalls for most objects are achieved. This result shows the unique strength of our approach: if multiple surface segments match an object, even if some of the matches may be wrong, as long as there is one correct match, the wrong results can be eliminated because of the following reason. If a wrong object is recognized from a test surface segment, there can be two cases: (i) there will not be sufficient number of matched keypoint pairs for correct pose estimation when the test surface segment is next matched to a training surface segment of the wrong object so that the wrong object is not predicted, or (ii) a high fitness cost will likely result when the wrong object model is predicted and projected to the test surface; in this case, our graph-cut based algorithm can effectively remove such incorrect detection results.

Table 4 also shows that, for the 569 correctly recognized object instances, most of them are detected with pose estimation correctly. Note that the fitness cost shown under each object is the average of fitness cost values associated with all tested instances of the same object, where for each object instance, the fitness cost value is the average of fitness cost values associated with all the surface segments that belongs to the object. The accuracies of each average fitness cost is quite consistent with the accuracy of the raw data captured by a Microsoft Kinect, which is about ± 3 *mm* for objects within 1 *m* from the camera.

However, Table 4 shows that a large number of object instances of *plate 1* are not detected (i.e., missing). This happened sometimes when the flat and shallow *plate 1* was directly put on the background table and the surface segmentation could not separate the plate well from the table. Introducing more clues for better surface segmentation could resolve this problem. Possible reasons for the other few cases of missing and wrong results include: wrong surface segments were detected based on visual features, or the detected surface segments were too small to provide sufficient information for generating reasonable poses.

5.4.4 Example Results of Scene Reconstruction and Discussion of Performance

Fig. 27 shows the results of our approach to object detection and pose estimation for some example test images. The 1st column shows five test images captured by a RGB-D camera. The 2nd $\sim$ 4th columns display the corresponding object detection and pose estimation results in four different views: front, left, right, and top views. By localizing all the known objects in the current scene and estimating their poses correctly, our algorithm reconstructed the whole scene based on a single test image (i.e., from a single viewpoint), as shown in the images of the reconstruction results from multiple views.

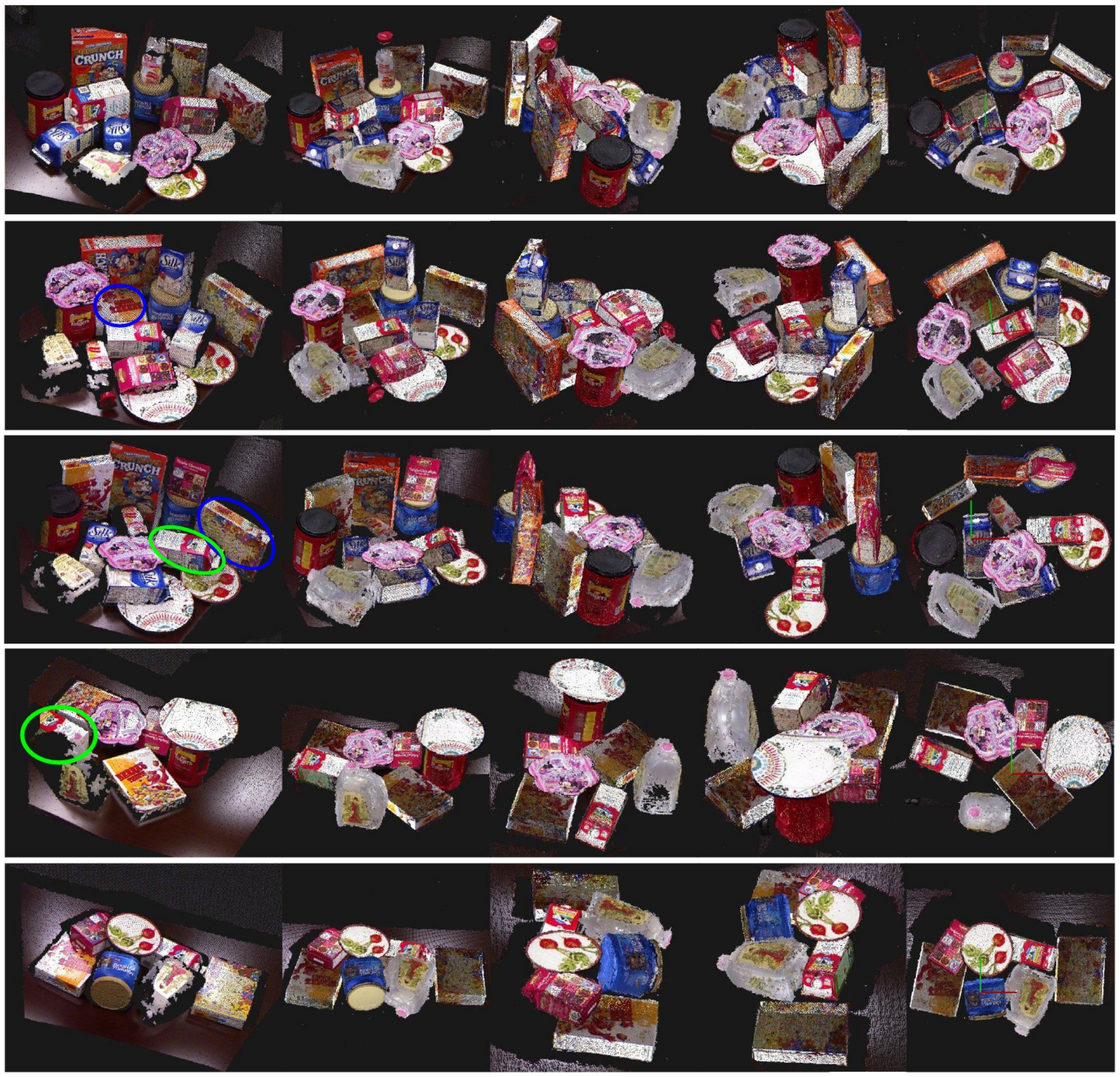


Figure 27: Examples of object detection and pose estimation results: each row starting with an original test image followed by four images from front, left, right, and top views that display the corresponding reconstructed scene

Note that in Fig. 27, the red milk box in the 3rd row (in a green circle) is occluded much less than the one in the 4th row (also in a green circle); however, our approach obtained better pose estimation results for the red milk box in the 4th row than the one in the 3rd row. For another example, the cereal box in the 2nd row (in a blue circle) is obviously occluded more than the one in the 3rd row (also in a blue circle); however, our approach successfully detects the cereal box in the 2nd row with pose estimation correctly but fails to detect the one in the 3rd row. Thus, the extent of occlusion to an object does not seem to affect the results of object detection as much as how distinctive the unoccluded regions of the object are.

From our test results, it is interesting to see the effects of different kinds of surfaces: some surfaces are indistinctive and cannot be used effectively for object recognition, but some distinctive surfaces for object recognition are ill suited for object pose estimation, such as object surfaces with constant colors; surfaces with rich graphics or text information are effective for both object recognition and pose estimation with our approach.

All our algorithms are currently implemented in single threads. It takes about 20 $\sim$ 30 seconds to process each surface segment for the recognition and pose estimation in a test image. Most of the time is spent in computing and matching ASIFT keypoints.

More data and results from our experiments can be found in [105].

5.4.5 Comparison to the MOPED Approach

In order to compare our approach to the MOPED approach [23] more quantitatively, we implemented the MOPED approach and applied it to our testing dataset.

Table 5 compares the detection and pose estimation results for all the objects in all the test images of our approach vs. those of the MOPED approach using Mean Shift clustering and the MOPED approach using Projection clustering with Q-Score ranking.

Table 5: Comparison of the test results: Mean Shift, P. Clust. (Q-Score), and our approach

Objects	Total	Mean Shift			P. Clust. (Q-Score)			Our Approach		
		CR CP	WR	CR WP	CR CP	WR	CR WP	CR CP	WR	CR WP
cereal box 1	52	52	0	22	52	0	1	47	1	0
cereal box 2	104	100	0	87	100	0	4	83	1	0
cookie box 1	52	51	0	40	50	0	1	46	0	1
juice container 1	52	50	0	23	49	0	1	50	0	0
juice container 2	52	37	0	43	38	0	0	36	1	2
milk box 1	104	101	0	119	101	0	3	83	0	0
milk box 2	52	50	0	32	49	0	1	43	0	1
coffee can 1	52	46	0	12	46	0	0	32	0	3
coffee can 2	52	15	0	10	19	0	7	39	0	5
plate 1	52	51	0	42	51	0	1	19	0	0
plate 2	52	52	0	17	51	0	1	41	0	1
tray 1	52	50	0	22	50	0	2	37	0	0

Table 6 compares the precisions and recalls of our approach vs. those of the MOPED approach using Mean Shift clustering and the MOPED approach using Projection clustering with Q-Score ranking. To compute the precisions, we consider a detected object instance to be false positive if it is redundant or wrongly recognized or if its pose is wrongly estimated. In the test results of the MOPED approaches, all the false positive results come from the redundant object instances or object instances with incorrect pose estimation.

Table 6: Comparison of the test results in precisions and recalls: Mean Shift, P. Clust. (Q-Score), and our approach

Objects	Mean Shift		P. Clust. (Q-Score)		Our Approach	
	Precision	Recall	Precision	Recall	Precision	Recall
cereal box 1	0.7027	1.0	0.9811	1.0	0.9792	0.9038
cereal box 2	0.5348	0.9615	0.9615	0.9615	0.9881	0.7981
cookie box 1	0.5604	0.9808	0.9804	0.9615	0.9583	0.8846
juice container 1	0.6849	0.9615	0.98	0.9423	1.0	0.9615
juice container 2	0.4625	0.7115	1.0	0.7307	0.9474	0.6923
milk box 1	0.4591	0.9711	0.9711	0.9711	1.0	0.7981
milk box 2	0.6097	0.9615	0.98	0.9423	0.9773	0.8269
coffee can 1	0.7931	0.8846	1.0	0.8846	0.9143	0.6153
coffee can 2	0.6	0.2884	0.7308	0.3654	0.8864	0.75
plate 1	0.5484	0.9808	0.9808	0.9808	1.0	0.3654
plate 2	0.7536	1.0	0.9808	0.9808	0.9762	0.7885
tray 1	0.6944	0.9615	0.9615	0.9615	1.0	0.7115

In the MOPED approach, the pose clustering results using Mean Shift clustering are used to initialize the input of Projection clustering with Q-Score ranking to remove noisy pose hypotheses. In essence, Projection clustering with Q-Score ranking is a voting procedure, which assigns each matched keypoint pair to the best fit pose hypothesis and only keep those pose hypotheses which are voted by matched keypoint pairs exceeding a threshold number as the final results.

As shown in Table 6, the MOPED approach using Mean Shift clustering achieves high recalls but pretty low precisions for all objects. This is because the MOPED approach uses loose thresholds with a large number of RANSAC iterations to generate as many object pose hypotheses as possible at the beginning. However, the pose hypotheses that are false positive are often too distant from any correct hypothesis; as the result, they cannot be merged into the clusters of correct pose hypotheses and are considered as independent clusters by regular clustering algorithms, such as Mean Shift.

By further introducing the Projection clustering with Q-Score ranking, the MOPED approach does greatly improve the precisions and can achieve comparable precisions as our approach does. However, our approach has advantages in several aspects as explained below.

The MOPED approach using Projection clustering with Q-Score ranking requires far more manually decided thresholds than our approach. Besides the thresholds used in clustering keypoints, pose hypothesis generation with RANSAC and pose hypothesis validation, this approach also introduces many thresholds in clustering and voting to remove redundant pose hypotheses and eliminate wrong pose hypotheses. For example, there are a radius threshold for similarity searching in clustering pose hypotheses with Mean Shift, a minimum size threshold of the cluster after clustering to remove noisy pose hypothesis clusters, a minimum number threshold of votes to remove the noisy or wrong pose hypotheses, and a pose similarity threshold to merge the final similar pose hypotheses. The values of those thresholds were empirically and manually determined in our implementation in order to achieve good performance. There is no strategy for automatically adjusting those threshold values [23].

In contrast, our approach does not introduce manually decided thresholds to solve pose ambiguities. As the result, our approach is more adaptive and robust to different objects and scenes.

As pointed out in Section 2.4.1.1 and Table 2, the performance of object detection of the MOPED approach strongly relies on the distinctiveness and repeatability of the descriptors of the SIFT keypoints. However, the SIFT keypoint descriptors, while robust for image registration, can be ill-suited for object recognition due to keypoint mismatches, especially for a dataset with a large number of objects. As shown in Table 6, keypoint mismatching has caused both the low precision and the extremely low recall in the test results of *coffee can 2*. From the experimental results, we can find two different kinds of keypoint mismatching here: (1) global keypoint mismatching between *coffee can 2* and other objects, and (2) local keypoint mismatching between different parts of *coffee can 2*.

Fig. 28 shows the global keypoint mismatching between *coffee can 2* (in a green circle) and other objects, where the test keypoints matched to the same object are displayed in the same color. Hence, many different colors of the test keypoints on *coffee can 2* indicate that many of them are wrongly matched to other objects. Global keypoint mismatching usually leads to failed detection of the target object instance or incorrect detection of other object instances with similar shapes. Also note that, there are many mismatched keypoints on the boundaries between objects or between objects and the background, as discussed in Section 4.3.

Local keypoint mismatching usually causes incorrect pose estimation, as shown in Fig. 29, and even the failure to detect the target object instance, as shown in Fig. 30. In both Fig. 29 and Fig. 30, there are a lot of test keypoints matched to the correct object *coffee can 2*, as indicated by that many keypoints on *coffee can 2* share the same color, but they are not matched to the correct training keypoints on *coffee*

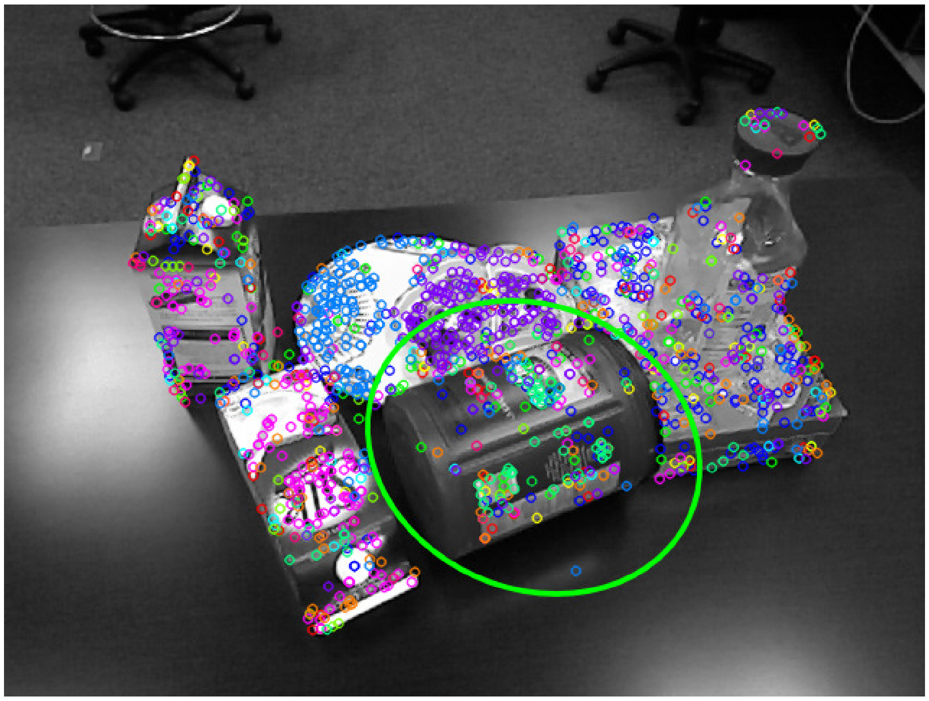


Figure 28: Keypoint matching results: keypoints matched to the training keypoints on the same object are displayed with the same color.

can 2.

In contrast to the MOPED approach, the performance of object detection in our approach depends on combining different kinds of useful features and also the learning algorithm. Keypoint matching in our approach is only used between two similar images for pose estimation in its classic way and not for object detection. Compared to using keypoint matching for object detection, our approach is more robust for detection involving a large number of objects. The correct detection results by our approach in Fig. 29 and Fig. 30 show the advantage of using visual features of surface segments to detect objects in our approach.

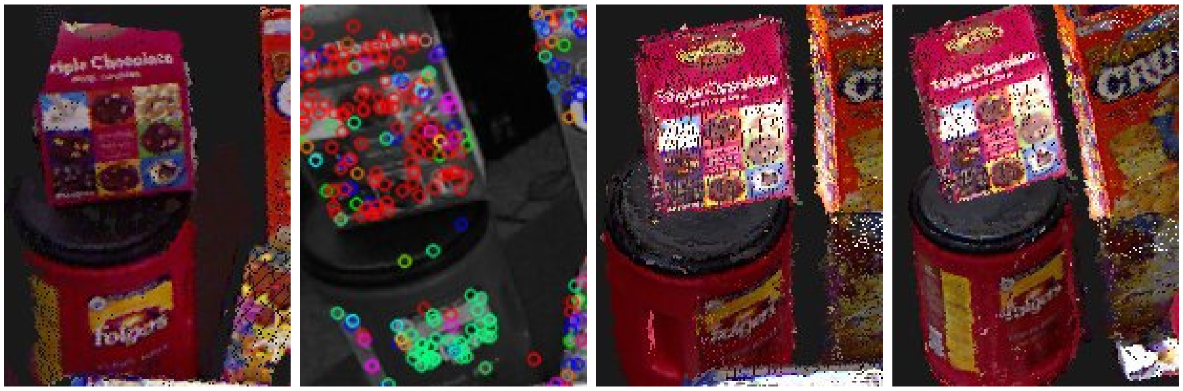


Figure 29: Incorrect pose estimation of *coffee can 2* caused by local keypoint mismatching in the MOPED approach using Projection clustering with Q-Score ranking: (a) the original scene, (b) the keypoint matching results, (c) the test results with the MOPED approach, and (d) the test results with our approach

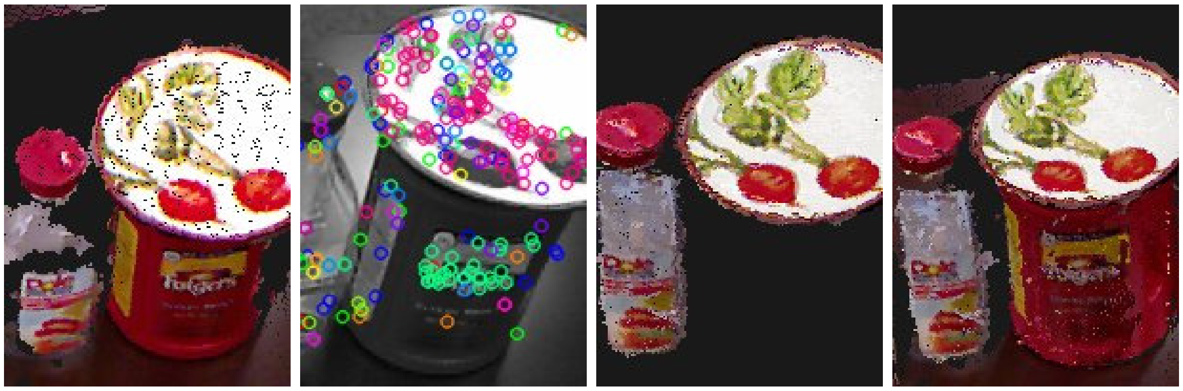


Figure 30: Failure of detecting *coffee can 2* caused by local keypoint mismatching in the MOPED approach using Projection clustering with Q-Score ranking: (a) the original scene, (b) the keypoint matching results, (c) the test results with the MOPED approach, and (d) the test results with our approach

5.5 Conclusions of this Chapter

To conclude, in this chapter, we have significantly extended the work in Chapter 4, by providing a new strategy for object model building and proposing an algorithm based on graph-cut to remove redundant or incorrect detection results. We have expanded the object dataset by including transparent or semi-transparent objects, added more results based on the new dataset, and presented a comparative study of our approach vs. related existing approaches. Our approach is shown to have advantageous over related existing approaches. The results of the detected objects and their poses with respect to the current RGB-D camera frame can be used directly for robotic manipulation, with the pose of the camera frame to the robot base frame known, and the reconstructed 3D scene can also be used directly for robot motion planning.

CHAPTER 6: LEARNING-BASED VIEW EVALUATION

Leveraging the scene reconstruction results of our surface-based approach, this chapter presents a learning-based approach [108] for evaluating scene recognizability from a single view, taking into account the quantity and quality of recognition of object instances in the scene, the recognition uncertainty, and the recognizability of the background. The introduced approach can be used to assess automatically how well a scene can be understood from an image without having a complete view of everything. It can be used to guide view planning in a cluttered environment for tasks requiring automatic recognition and reconstruction of objects, where there is no complete visibility of anything from any view. Even in case certain objects can be seen entirely through a sequence of views, using the introduced evaluation approach of scene recognizability could improve the efficiency of view planning because a shorter sequence of views or even a single view could be sufficient to recognize and reconstruct the objects correctly.

6.1 Evaluation Factors

We evaluate the scene recognizability of an image taking into account the aforementioned factors that affect both the quantity and quality of object instances recognized in a scene and the quality of background recognition. We describe those factors in detail below.

6.1.1 Recognition Confidence

In general, each known object instance has a model or classifier that can be used to identify the object, and the classifier often returns some measure of confidence in terms of the probability that a target object is a particular known object instance. We use such a confidence measure in our evaluation of recognizability of an object from a view.

For example, an appearance-based object model is used to characterize object instances based on visual features of their surface segments in [107], and support vector machines (SVMs) are used to classify a target object with a probability output [19] for each known object instance to indicate the confidence of classifying the target object to that object instance.

Suppose there are M objects plus the background in the object model dataset, denoted as $c_0, c_1, c_2, ..., c_M$, where c_0 indicates the background. Given n_s surface segments in the current image, denoted as $S_1, S_2, ..., S_{n_s}$. Suppose n object instances in the scene are recognized and reconstructed (excluding the background) [107], denoted as $O_1, O_2, ..., O_n$.

As we can see from Fig. 31, an object instance is usually enclosed by several surface segments. Let Ψ_i denote the set of surface segments that belong to the ith object instance, as follows:

$$\Psi_i = \{S_{i1}, S_{i2}, ..., S_{in_{si}}\} \subseteq \{S_1, S_2, ..., S_{n_s}\}, \tag{13}$$

n_{si} is the number of surface segments that belong to the ith object instance, or the

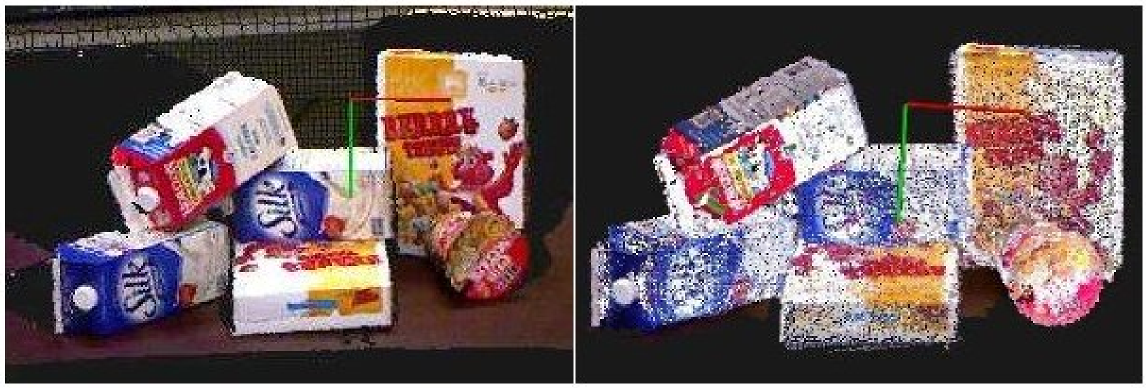


Figure 31: An example of object recognition and pose estimation by [107]: the left image is the original view captured by a Microsoft Kinect; the right image is the reconstructed scene with the projected 3D object models.

cardinality of the set Ψ_i.

The *recognition confidence* of the *i*th object instance can be defined as

$$cf(O_i) = \max_{\forall S_{ij} \in \Psi_i} P_{ij}(L_i|S_{ij}), \ j = 1, 2, ..., n_{si}, \tag{14}$$

where L_i is the recognized object label for the *i*th object instance, i.e., $L_i \in \{c_1, c_2, ..., c_M\}$, $P_{ij}(L_i|S_{ij})$ is the probability that the surface segment S_{ij} belongs to the object category L_i returned by the SVM classifiers, $0 \leq P_{ij}(L_i|S_{ij}) \leq 1$. Here we use the greatest probability as the recognition confidence of the object instance because it corresponds to the surface segment that captures the most representative and discriminating characteristics of the object instance.

Note that the recognition confidence introduced above only captures the reliability of the recognition based on the characteristics of the target object's visual appearance, regardless of its pose.

6.1.2 Fitness Cost

It is also important to measure how well the identified target object in a scene matches the projection of the object model to the scene, i.e., how fit the pose estimation of the identified object is.

We introduce the *fitness cost* of the recognized ith object instance as a distance function between the point cloud Λ_i of the projected ith object model, $i = 1, 2, ..., n$, and point cloud Π_i of all the surface segments that are recognized as belonging to the ith object. Let T_i be the number of points in Π_i.

For each point $q_k \in \Pi_i, k = 1, 2, ..., T_i$, we can find a matched nearest point in the point cloud Λ_i, denoted as $p(q_k) \in \Lambda_i$, then the fitness cost $ft(O_i)$ of the ith object instance can be defined as the average Euclidean distance between all the points in the point cloud Π_i and their matched nearest points in the point cloud Λ_i:

$$ft(O_i) = \frac{1}{T_i} \sum_{k=1}^{T_i} ||q_k - p(q_k)||. \tag{15}$$

6.1.3 Recognizability of Object Instance

We introduce the harmonic mean function combining the recognition confidence and fitness cost to characterize the recognizability of each object instance.

6.1.3.1 Definition of Recognizability

First, we normalize the values of the fitness cost by using an exponential function of the negative fitness cost $e^{-\alpha ft(O_i)}$ to make the values fall between 0 and 1, as shown in Fig. 32, where α is a parameter. The smaller the fitness cost value, the better the predicted object model fits its corresponding surface segments in a scene.

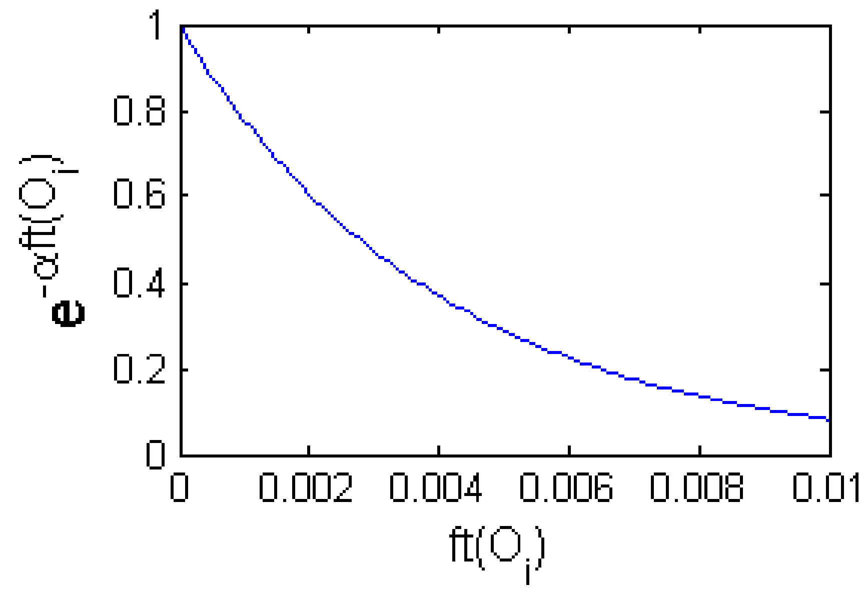


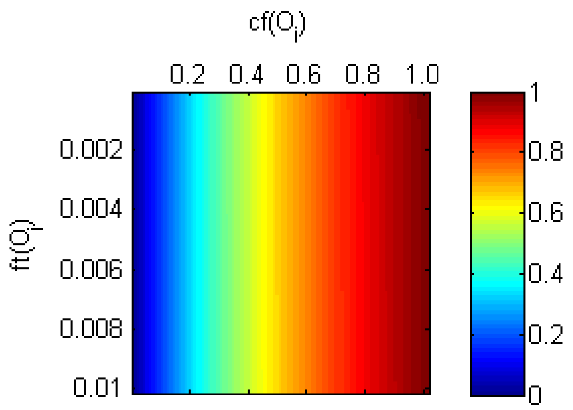
Figure 32: A monotonically decreasing exponential function is used to normalize the values of the fitness cost.

Now, we define the *recognizability* of the *i*th object instance as the harmonic mean [3] of the confidence and fitness cost:

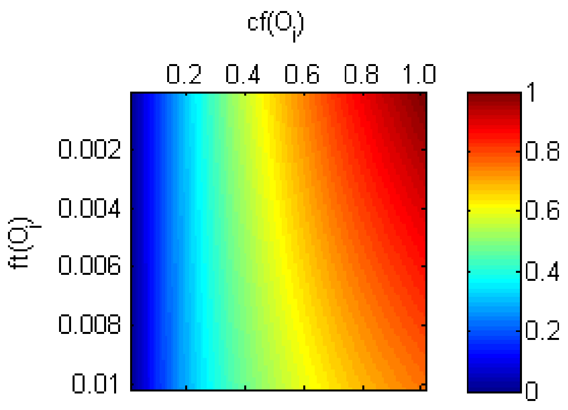
$$rg(i) = \frac{2cf(O_i)\mathrm{e}^{-\alpha ft(O_i)}}{cf(O_i) + \mathrm{e}^{-\alpha ft(O_i)}}. \tag{16}$$

Typically, the harmonic mean is appropriate for situations when the average of rates is desired. In our problem, both $cf(O_i)$ and $\mathrm{e}^{-\alpha ft(O_i)}$ are in the range of $(0, 1)$; therefore, a high recognizability value from this definition requires both high confidence and low fitness cost.

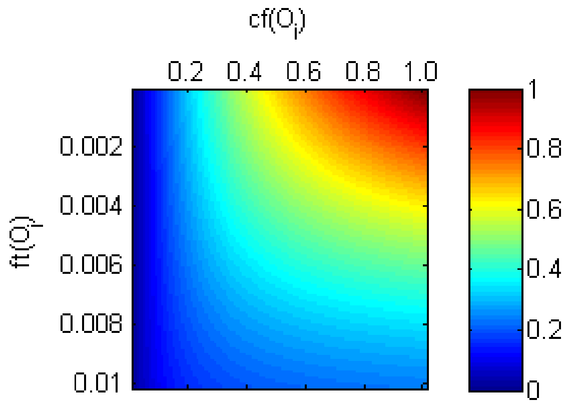
Fig. 33 shows the distribution of the object recognizability values as the recognition confidence and fitness cost vary in their own value ranges respectively for different values of the parameter α. For each α value, a color contour in the corresponding distribution represents a single value of object recognizability, which



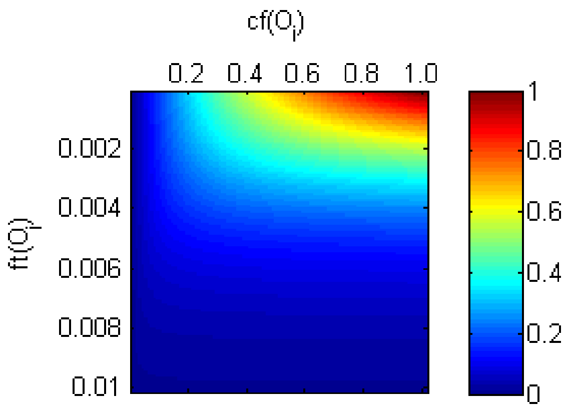
(a) $\alpha = 1$



(b) $\alpha = 50$



(c) $\alpha = 200$



(d) $\alpha = 500$

Figure 33: The distributions of the object instance recognizability values as the recognition confidence and fitness cost vary in their own value ranges respectively for different α values. Different colors in each figure indicate different object recognizability values. As α increases, the influence of fitness cost on recognizability increases.

comprises different pairs of values of the recognition confidence and fitness cost. Clearly, as α increases, the influence of fitness cost on recognizability increases. Each color contour also divides the distribution into two regions. We are interested in finding an optimal threshold value ξ together with an optimized value of α to achieve the best separation of the distribution into a high-recognizability region (with values greater than ξ) and a low-recognizability region.

6.1.3.2 Optimization of α and ξ

The choices of α and ξ values determine the standard for high object recognizability. However, the values of these parameters can be quite dependent on specific types of task environments or task scenes. Therefore, for a certain type of task scenes, our approach is to learn α and ξ values from a sufficient number of examples of recognition and reconstruction results of different views and scenes. For each recognized and reconstructed object instance, we manually label it as either “correct” or “incorrect” as the ground truth, based on whether the object is correctly identified and its pose correctly estimated (i.e., with sufficient accuracy) from comparing the reconstructed 3D object model (which can be viewed from all viewpoints) against the target object in the original scene. Also included in the data for each recognized and reconstructed object instance is the corresponding recognition confidence and fitness cost values. We aim at finding a pair of values for α and ξ that can effectively distinguish the correctly recognized and accurately estimated object instances, which indicate high recognizability, from the rest of the object instances, i.e., finding a binary classifier.

Since a given dataset normally consists of more correct recognition results than incorrect ones as the results of a reasonable object identifier (i.e., one with more than 50% correct rate of recognition), the dataset is not balanced for learning, and the examples of incorrect recognition are important for determining the decision contour of our intended binary classifier of recognizability. Therefore, we use the classification criteria called the sensitivity, or true positive rate (TPR), and the specificity, or true negative rate (TNR).

From each pair of α and ξ in our problem, the sensitivity (TPR) is defined as the ratio of the number of object instances in the "correct" or high-recognizability region (with recognizability values $> \xi$) to the number of object instances labelled "correct" in the ground truth; the specificity (TNR) is defined as the ratio of the number of object instances in the "incorrect" or low-recognizability region to the object instances labelled "incorrect" in the ground truth.

We propose two types of optimization criteria to obtain suitable α and ξ as follows:

1. Maximize the harmonic mean of sensitivity and specificity:

$$H = \max \frac{2 \cdot TPR \cdot TNR}{TPR + TNR}. \tag{17}$$

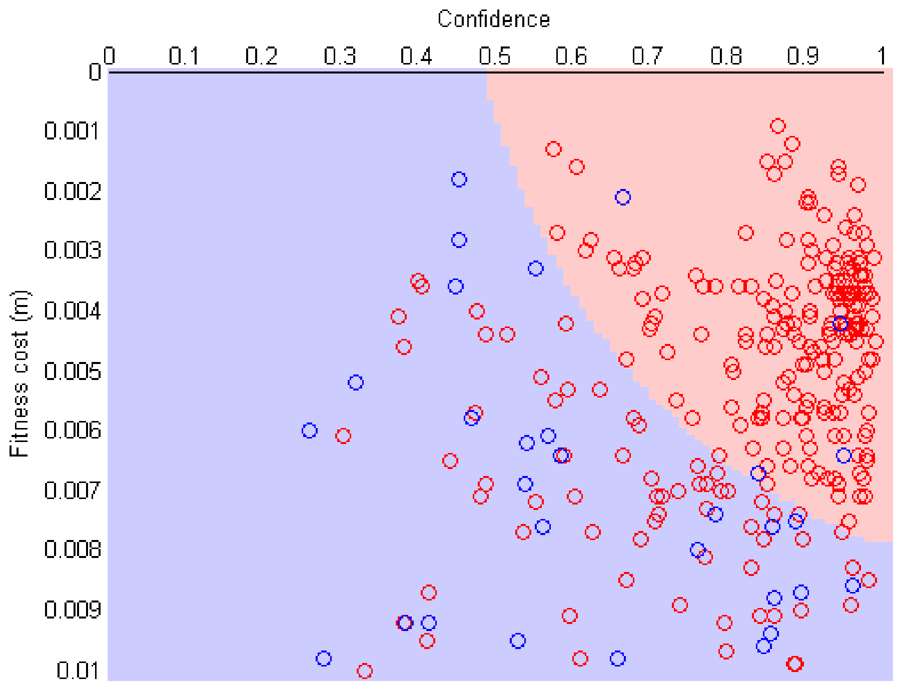
   The harmonic mean function H requires both high TPR and TNR to achieve a high value.

2. Given a minimum tolerance σ on the specificity (TNR), maximize the sensitivity (TPR). Different values of σ provide different standards for qual-

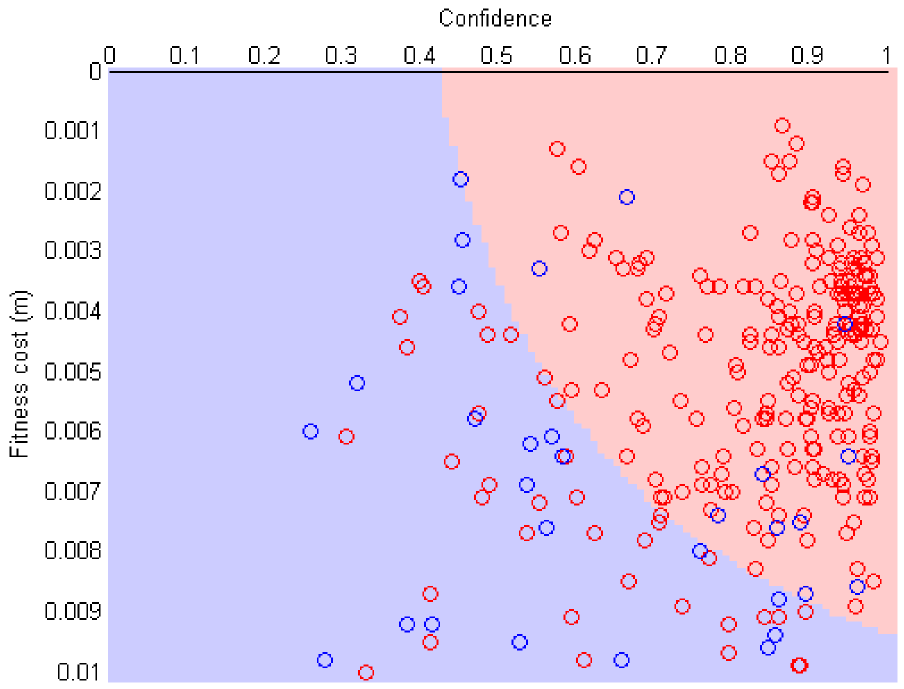
ity. A low σ value means tolerating more "incorrect" object instances in the high-recognizability region (i.e., with recognizability values $> \xi$), whereas a high σ value means tolerating fewer "incorrect" object instances in the high-recognizability region.

Fig. 34 shows the optimized results of classification using two optimization criteria, a) maximizing the harmonic mean H and b) maximizing TPR given $\sigma = 0.7$, respectively. In each subfigure, a set of recognized and reconstructed object instance examples are divided into two regions, the high-recognizability red region and the low-recognizability blue region, as determined by the optimized α and ξ values. A red or blue circle indicates a given recognized and reconstructed object instance labelled "correct" or "incorrect" respectively. The different effects of the two optimization criteria on the results are quite apparent: the result of using criterion a) includes fewer incorrect samples in the high-recognizability (red) region at the expense of including fewer correct samples in the red region; whereas, the result of criterion b) includes more correct samples and also more incorrect samples in the red region. Depending on the task objectives, different types of optimization criteria can be used, and for the type 2), different σ values can be selected.

Note that some blue circles either have high confidence and high fitness cost or vice versa; this is because an incorrect result may have correct object recognition but wrong pose estimation or vice versa. Therefore, it is important that our binary classifier distinguishes high-quality results of both recognition and estimation from low-quality ones instead of simply distinguishing correct vs. incorrect



(a) $\alpha = 91, \xi = 0.66$ as the result of maximizing the harmonic mean H



(b) $\alpha = 92, \xi = 0.60$ as the result of maximizing TPR given the tolerance $\sigma = 0.7$ of TNR

Figure 34: A set of reconstructed object instances based on recognition and pose estimation from several views or scenes, with corresponding recognition confidence and fitness cost values. Each small circle represents a reconstructed object instance. Each red one indicates a correct result, while each blue one indicates an incorrect result.

recognition results based on the confidence measure alone.

6.1.4 Recognition Rate and Uncertainty

Given the results of binary classification of object instances into high-recognizability region and low-recognizability region, we can compute the *recognition rate* Θ as the ratio of the number of object instances in the high-recognizability region, denoted as n_{hr}, to the total number m of object instances in a scene:

$$\Theta = \frac{n_{hr}}{m}. \tag{18}$$

We can also compute the *recognition uncertainty* Φ as

$$\Phi = \frac{n - n_{hr}}{m}. \tag{19}$$

where n is the total number of object instances recognized and reconstructed. The recognition uncertainty reflects the number of object instances in the low-recognizability region. We use the term "uncertainty" to capture the fact that object instances of low recognizability based on one view may become ones with high recognizability from a different view, and thus, they provide clues for next-view exploration to further identify and estimate objects in a scene.

6.1.5 Background Recognizability

Background recognizability is used to measure the quality of scene recognition from an image (or the corresponding view) from the perspective of the background.

For each surface segment recognized as the background, there does not exist a specific object model to align with. Hence, we only consider the recognition confidence based on the characteristics of visual appearance.

Given n_{bk} surface segments recognized as the background in the current image, denoted as $SB_1, SB_2, ..., SB_{n_{\text{bk}}}$, $SB_1, SB_2, ..., SB_{n_{\text{bk}}} \in \{S_1, S_2, ..., S_{n_s}\}$, and their sizes, $W_1, W_2, ..., W_{n_{\text{bk}}}$, where W_i $(1 \leq i \leq n_{\text{bk}})$ is defined as the number of the pixels in a 2D image, the *background recognizability* can be defined as the weighted average of the recognition confidences of all background surface segments:

$$\Omega = \sum_{i=1}^{n_{\text{bk}}} \omega_i P_i(c_0|SB_i), \tag{20}$$

$$\omega_i = \frac{W_i}{W_1 + W_2 + ... + W_{n_{\text{bk}}}}, \; i = 1, 2, ..., n_{\text{bk}}, \tag{21}$$

where $P_i(c_0|SB_i)$ is the probability that the surface segment SB_i belongs to the object category c_0 (the background) returned by the SVM classifiers, $0 \leq P_i(c_0|SB_i) \leq 1$.

6.2 View Evaluation

Now we can consider how to compare two views and evaluate which view has a better overall *scene recognizability* based on the recognition rate, recognition uncertainty, and background recognizability of each view. Since it is not obvious how those factors interact to affect the overall scene recognizability of a view as that may depend on different task environments, it is difficult to come up with an analytical formula and numerical value for scene recognizability. Therefore, we again propose a learning approach to determine which view has a better scene recognizability.

Given an image (corresponding to a view), we characterize its scene recognizability v by combining its background recognizability Ω with a set of pairs:

$$v = \left\{\Omega, (\Theta_1, \Phi_1), (\Theta_2, \Phi_2), ..., (\Theta_J, \Phi_J)\right\} \tag{22}$$

where each pair shows the result of recognition rate and recognition uncertainty based on a different optimization criterion or a different quality standard (i.e., a different σ value of criterion type 2), and J is the number of different optimization criteria or quality standards considered.

Next we use a set of K images corresponding to different views for a scene to produce samples for learning. All these images are first ranked by a human observer from the reconstruction results. Human observation considers the following three factors in the order of priority: the greater the number of correctly recognized and reconstructed object instances the better, the smaller the number of incorrect object instances the better, and the greater background recognizability the better. Views can have the same rank if neither is clearly better than the other. Based on the ranking results, for each pair of images A and B among the K images, if A is ranked better than B, a positive sample $(v_A - v_B, 1)$ is generated; otherwise, a negative sample $(v_A - v_B, -1)$ is generated. For K images, a maximum of $K(K-1)$ samples can be generated automatically.

We then train a classifier using the samples. The resulting classifier, which we call the *view evaluator*, can be used to rank the scene recognizability of different views (i.e., corresponding images) of a scene, based on learned high-recognizability and low-recognizability regions, which are optimized by taking into account recognition rate, recognition uncertainty, and background recognizability with respect to different optimization criteria. For a specific task environment, such a learning process also reveals which optimization criteria are most effective for evaluating scene recognizability.

6.3 Experiments and Analyses

In order to validate our approach for evaluating scene recognizability of a view, we use 15 different scenes cluttered with object instances of four known objects: a cereal box, a cup noodle, and two different milk boxes. The object dataset includes appearance-based models of the four objects and a background model. Each object model was built from 24 images of different views [107].

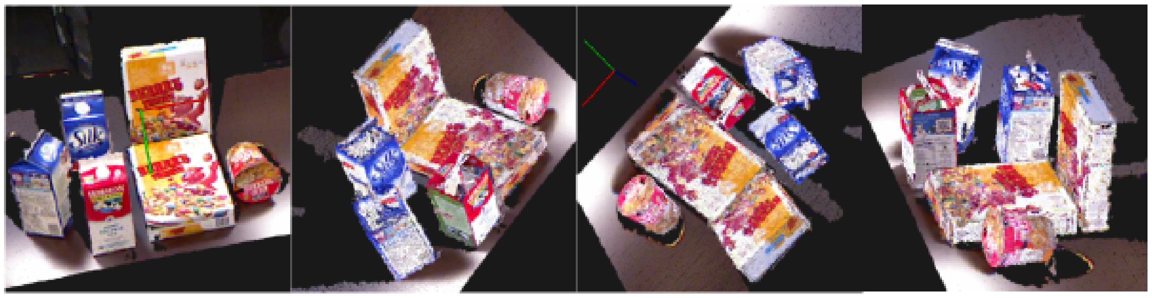


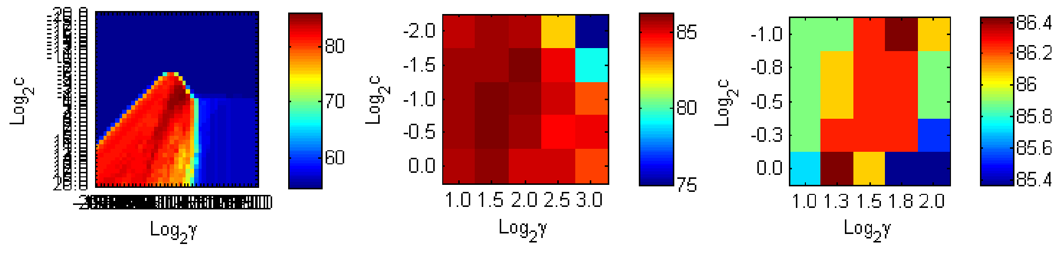
Figure 35: An original RGB-D image captured by Microsoft Kinect (leftmost) and the 3D recognition and reconstruction result displayed from 3 different views

For each scene, 8 images from distinct views are captured, and object instances in each image are recognized and reconstructed based on the approach in [107]. The 3D recognition and reconstruction (i.e., pose estimation) result can be displayed in our system from any viewpoint so that one can easily check by eyeballing if each object instance is correctly recognized and reconstructed. Fig. 35 shows an example.

6.3.1 Training of View Evaluator

We use the images and the corresponding recognition and reconstruction results from 10 scenes to form a dataset for the training required in our approach (for parameter optimization – see Section 6.1.3 and for evaluating scene recognizability – see Section 6.2). We used $J = 10$ different optimization criteria to generate 10 pairs of optimized α and ξ values: one of the criteria is maximizing the harmonic mean H and the rest are corresponding to 9 different σ values $\sigma = 0.1, 0.2, ..., 0.9$ respectively (see Section 6.1). Next the 10 binary classifiers are used to compute the corresponding pairs of recognition rate and uncertainty for each image so that we can form samples to train a view evaluator (Section 6.2).

We use the support vector machines (SVMs) and apply the kernel based on the radial basis function to it for classification. We also use 5-fold nested cross validation in the SVM training, as shown in Fig. 36. First, the optimal parameters are search in the large scale, then the search scale is narrowed down until the optimal values are found.



(a) large scale (b) medium scale (c) small scale

Figure 36: Optimal parameters are searched in the large scale first, then the search scale is narrowed down until solutions are found. Optimal accuracy = 86.43% can be achieved when the SVM parameters $c = 1$ and $\gamma = 2.38$.

Fig. 37 shows an example scene for training with 8 images from different views.

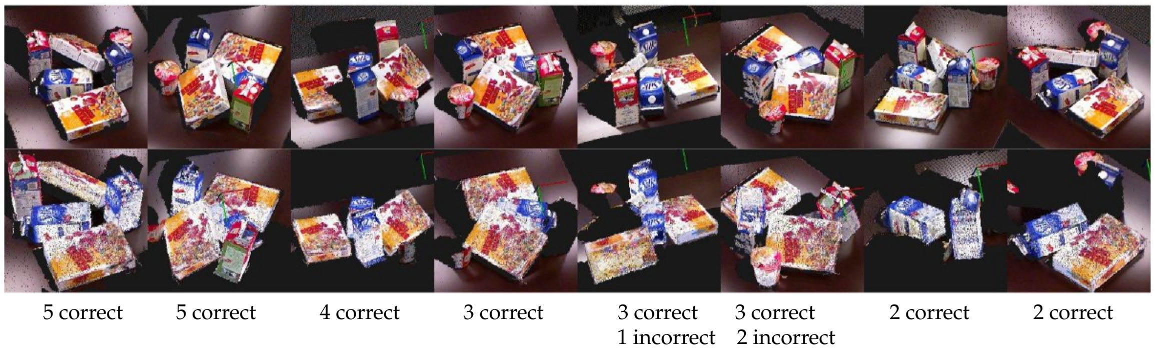


Figure 37: An example scene for training: top row: 8 images from different views; bottom row: corresponding reconstructed scenes; the images are in the order ranked by a human observer from the best to the worst, and the ranks are 1, 1, 3, 4, 5, 6, 7, 7 as ground truth.

The scene consists of $m = 6$ object instances: two instances of the same cereal box, two instances of the same blue milk box, one different red milk box, and one cup noodle. The top row shows the 8 images, and the bottom row shows the corresponding results of scene recognition and reconstruction in the order from the best to the worst ranked by human observation as ground truth. From left to right, the ranks are 1, 1, 3, 4, 5, 6, 7, 7.

Note that in the scene of Fig. 37 no view can produce perfect recognition result in terms of correct recognition of all the objects and background because of occlusion, varied illumination effects (such as caused by shadowing), and weak visual features in certain objects. Sometimes an object is recognized correctly but its pose estimation is incorrect due to insufficient information provided by the visible parts of the object. For example, from the $6th$ view, both the blue and the red milk boxes are recognized, but their pose estimations are incorrect. Sometimes an object or background surface is incorrectly recognized as a surface of another object, see, for example, the recognized cereal box in the front from the $5th$ view. Sometimes an object surface is incorrectly recognized as a part of the background (shown as small surface pieces in the reconstruction results), which lowers the score of background recognizability. Therefore, it is justified to take into account the effects of all those factors in evaluating scene recognizability of a view.

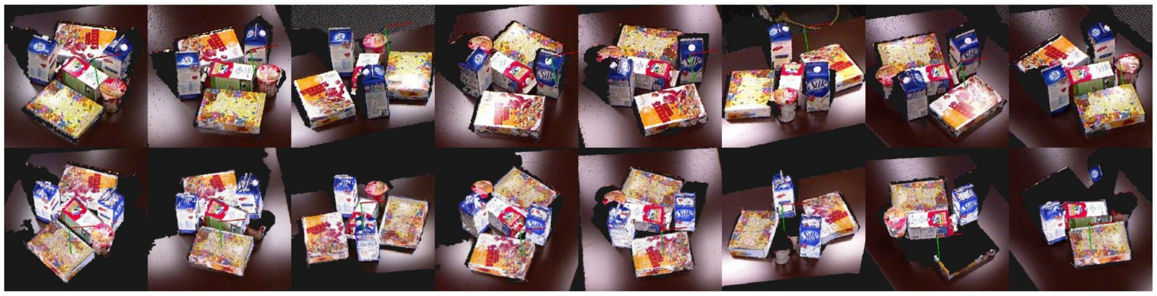
6.3.2 Test of View Evaluator

We use the images and the corresponding recognition and reconstruction results of the remaining 5 scenes for testing our approach. The testing scenes must have a similar number of objects as in the training scene. For each pair of views from each testing scene, we apply the trained view evaluator to detect which one has a better recognizability and to rank all the views from the results of pair-wise comparison. If the ranking order is circular for some views, these views are assigned the same rank. Fig. 38 shows the ranking results for views in 3 different testing scenes.

Note that the ranking results from the trained view evaluator may not exactly match those based on human observation. This is because the view evaluator has more precise information about object instance recognizability (Section 6.1.3) and background recognizability than a human observer, who can only roughly judge if an object instance is correctly recognized with its pose correctly estimated by eyeballing. Sometimes a human observer has difficulties in distinguishing which view has better scene recognizability between two views.

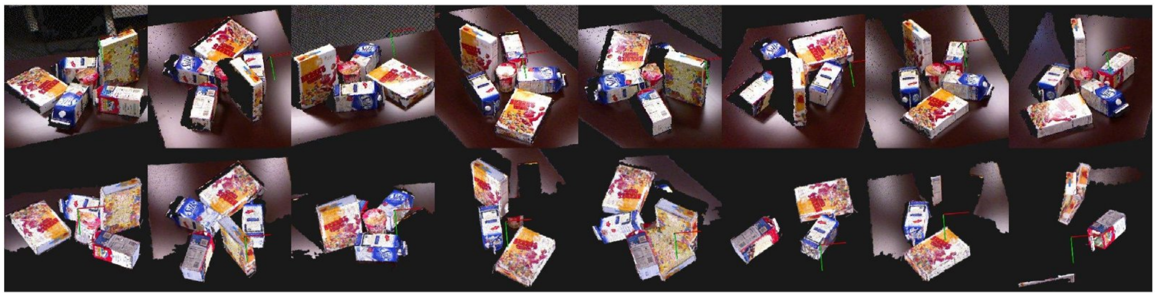
With training samples of sufficient quantity and diversity, our learning-based approach for view evaluation can provide better view evaluation results for the purpose of autonomous perception and action than a human observer. Fig. 39

Scene 1



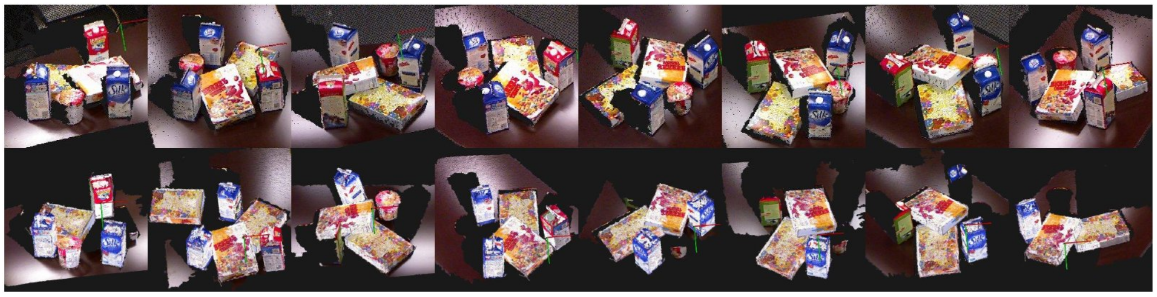
rank 1 rank 2 rank 2 rank 2 rank 2 rank 6 rank 6 rank 8

Scene 2



rank 1 rank 1 rank 3 rank 4 rank 4 rank 6 rank 7 rank 8

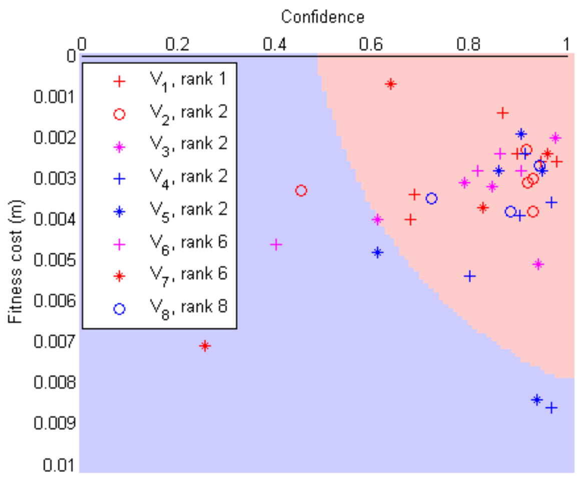
Scene 3



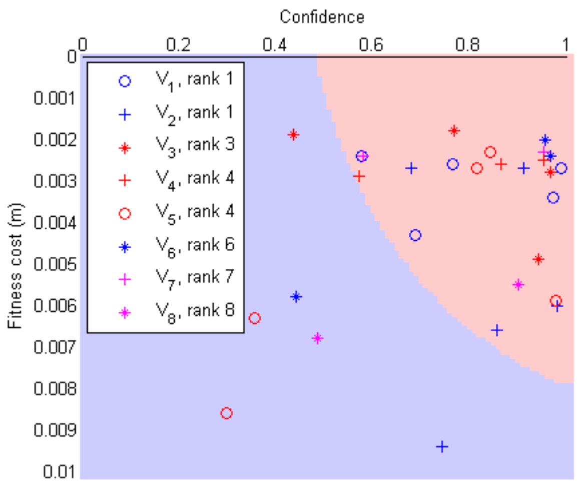
rank 1 rank 1 rank 3 rank 3 rank 5 rank 5 rank 5 rank 8

Figure 38: Testing results for **Scene 1**, **Scene 2** and **Scene 3**. Top row for each scene: 8 original images from different views; bottom row for each scene: corresponding reconstructed scenes. Rank of each view by the view evaluator is shown below the corresponding image of the reconstructed scene.

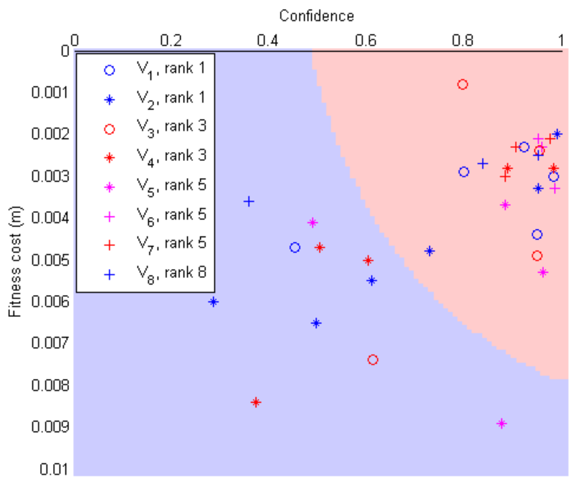
further indicates object instances of ranked eight views for each scene in Fig. 38 respectively.



(a) **Scene 1**



(b) **Scene 2**



(c) **Scene 3**

Figure 39: Recognized and reconstructed object instances of the eight different views $V_1 \sim V_8$ in **Scene 1** (left image), **Scene 2** (middle image), and **Scene 3** (right image) respectively. The displayed regions of high recognizability (red) and low recognizability (blue) is the result of optimizing the harmonic mean H. However, the view evaluator takes into account the effects of 10 different optimization criteria to create the ranking.

The experimental results provide strong evidence that increasing visibility does not necessarily lead to better recognizability and pose estimation of objects, due to occlusion, illumination effect (including shadowing), and weak visual features of objects. For example, in Scene 2 of Fig. 38, the blue milk box at the back of the 3rd view is more visible in the 8th view, but it can be recognized and reconstructed from the 3rd view and not from the 8th view, due to different occlusion effect. In Scene 1, even though the front cereal box on the table is more visible in the 7th view than in the 6th view, it is still not recognizable from the 7th view because part of its top surface is turned white due to the illumination effect. Also, it is more challenging to recognize the cup noodles based on its side appearance than its top appearance because of weak visual features.

The view evaluator is trained offline. After the testing scenes are reconstructed from different views, it takes less than 1 second for our trained view evaluator to rank all the views based on their overall scene recognizabilities.

6.4 Conclusions of this Chapter

To summarize, in this chapter, we have proposed a novel learning-based approach for evaluating scene recognizability of a cluttered scene with objects occluding one another from a single view and ranking views based on their scene recognizability. Scene recognizability of a view is evaluated by taking into account the quality and quantity of recognized and reconstructed objects and the background from the view rather than mere visibility through learning. The approach is validated by experimental results, which also provide evidence that better visibility does not necessarily lead to better recognizability of objects because of the effects of occlusion, illumination, and weak visual features. Those effects are captured and reflected in our learning-based approach for view evaluation. Our approach of evaluating scene recognizability points to a way for more accurate assessment of how good a view is for autonomous robotic tasks that require scene understanding for object recognition and manipulation in cluttered environments. As the next step, this approach can be further incorporated in viewing planning algorithms for robotic manipulation tasks.

CHAPTER 7: OBJECT MODELING THROUGH PERCEPTION AND MANIPULATION

As introduced in Section 2.5, perception is useful to guide object manipulation, and robotic manipulation can also assist object modeling. The interaction of perception and manipulation is necessary for manipulating unknown objects, especially in unstructured and cluttered environments. However, integrating perception and manipulation is a relatively new problem, and existing research is limited.

In this chapter, we introduce a method to build complete 3D object models for unknown objects from RGB-D images automatically by interleaving perception with robotic manipulation [104]. We assume that objects are small and light enough to be grasped by a robot end effector, and the appearance and geometrical characteristics of objects will not be changed by robotic manipulation.

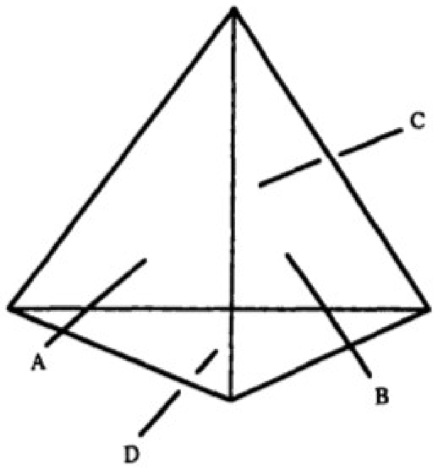
Here the key idea is that perception and manipulation depends on each other: perception will provide partial information of the object necessary for the robot manipulator to change the pose of the object, and manipulation will, in turn, enable the perception of more object information. By alternating perception and manipulation, a complete object model can be built. The framework of our approach is outlined in Section 3.2.3. In the following, the related algorithms are introduced in detail.

7.1 Aspect Graph of Object Poses

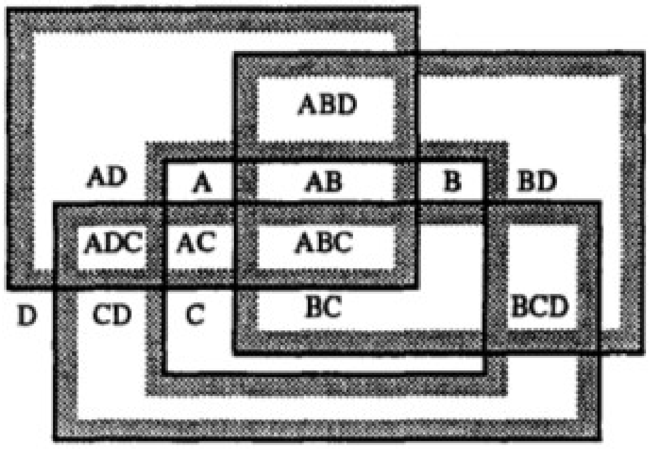
First of all, to build a complete model of a target object, all the surfaces of the target object should be observed at least once from the perspective of the fixed camera during the object modeling procedure. This is similar to the full view coverage problem in the research topic of view planning, and an aspect graph [80] is usually introduced to capture all different aspects/views, as illustrated in Fig. 40, which are discrete, to solve the problem.

In our problem of modeling an unknown object, to capture all the aspects/views of the target object, we generate an oriented bounding box (OBB) [2, 41] based on some rough, initial information about the target object and also update the OBB along the way as new information of the object is sensed, as shown in Fig. 41. The six sides of the OBB partition the whole view space of the object into six main *viewing regions*, V_1, ..., V_6, and the viewing region V_i is a Voronoi region of the i-th side that captures all lines of sights to that side of the OBB. Depending on the actual shape of the object covered by the i-th side of the OBB, multiple (discrete) views can be needed to obtain all aspects of V_i.

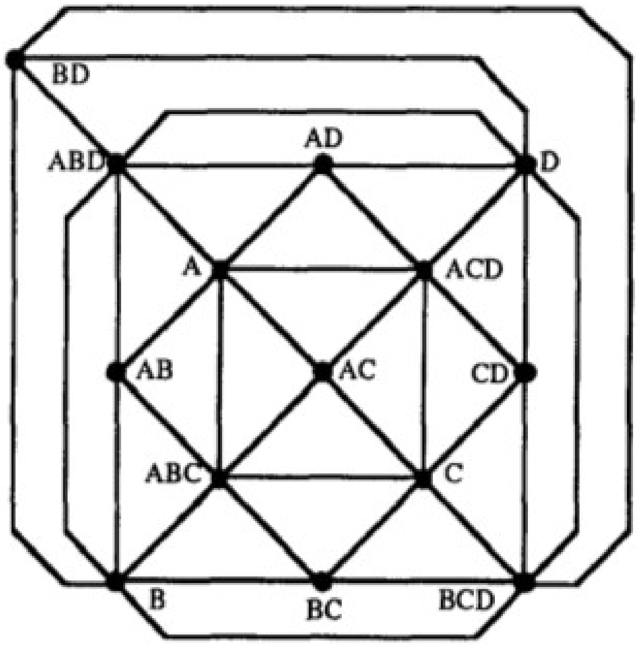
Our strategy is to fix the RGB-D camera and manipulate the target object to expose each viewing region V_i to the camera and all the different aspects in the region. Starting from a stable pose of the object on a table, where one viewing region V_s contains the support surface of the object and is occluded by the table, the robot manipulator is used to rotate the object on the table, one small step at a time, 360 degrees to obtain every aspect in the viewing region on top of the object



(a)



(b)



(c)

Figure 40: An example of the aspect graph of a tetrahedron: (a) shows the model of a tetrahedron; (b) shows the viewpoint space partition for the tetrahedron; and (c) shows the aspect graph for the tetrahedron. All the figures (a), (b) and (c) are the original figures from [80].

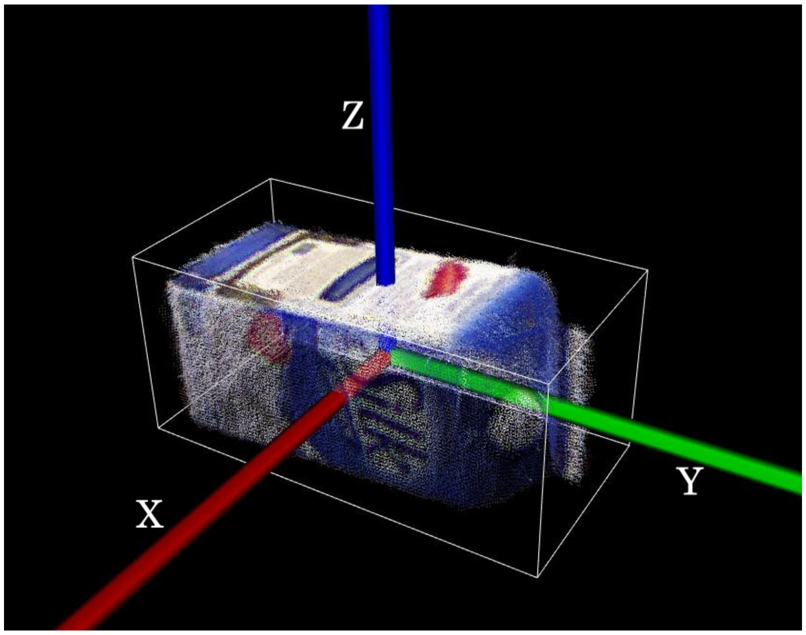
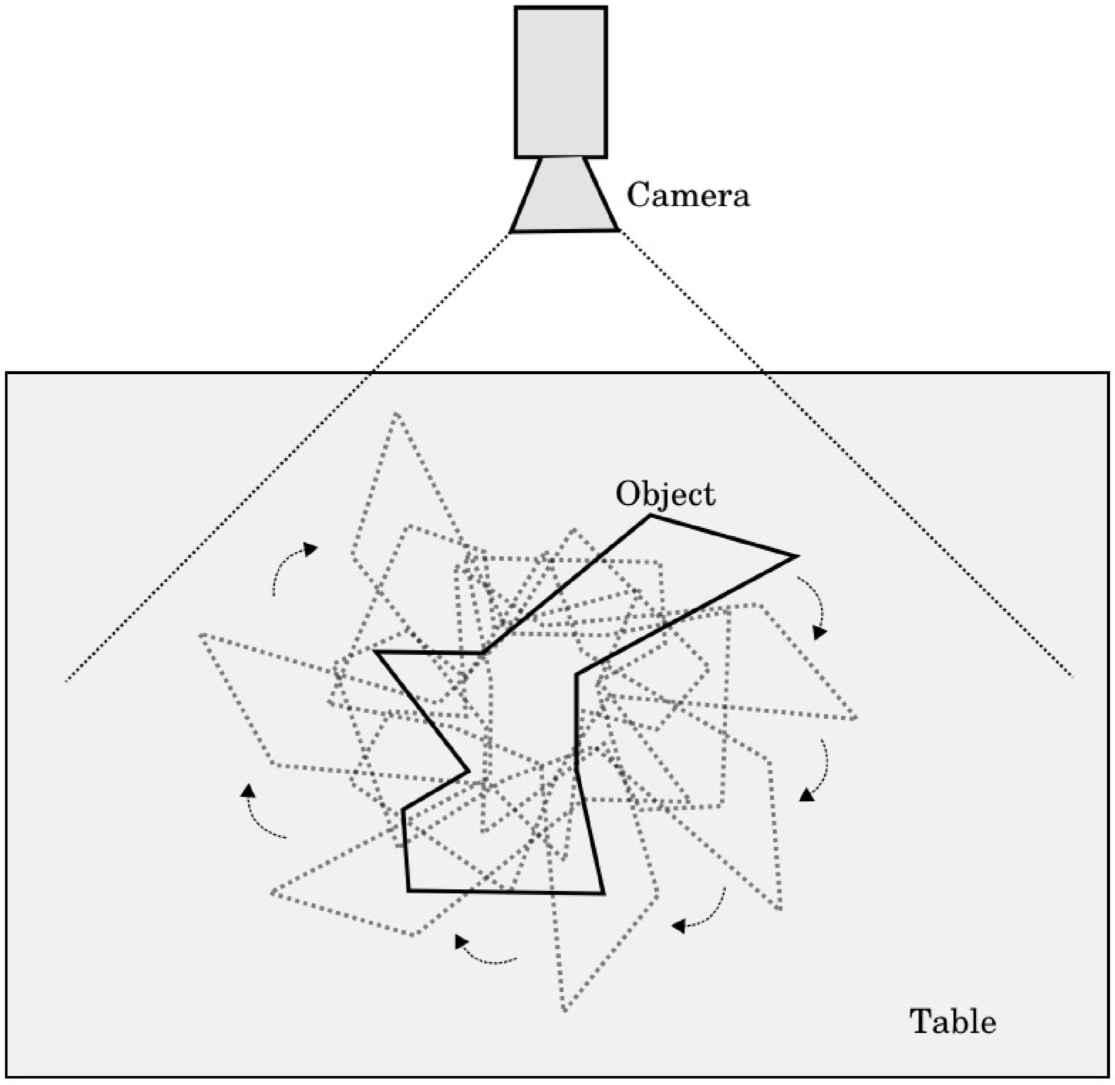


Figure 41: An example of the oriented bounding box, generated from the point cloud of a partial model of the object *milk box blue*

and also the four viewing regions that are perpendicular to V_s, see Fig. 42. Once images of the object are captured for all those aspects, the robot manipulator is used to change the pose of the object by changing the support surface of the object. Next, it rotates the object step by step again 360 degrees to capture the images from the new set of aspects, and so on. Once all viewing regions and their aspects are exposed to the camera so that their corresponding images are captured, the target object is considered completely seen.

The above strategy works well for many daily objects. However, for objects with many concavities and extremely complicated self-occlusion, such as a sculpture with many small details, the approach may not be able to expose every small surface of the object to the viewing range of the fixed camera. For such cases, making the robot hold the object in certain poses rather than putting it on a table can



(a)

Figure 42: A top view of the set up for our object modeling procedure, including a fixed camera and different poses of the target object after being rotated on the table step by step for about 360 degrees

help complete the coverage, as discussed in Chapter 8.

7.2 Image Registration in Perception

Image registration serves two main purposes in our object modeling process: one is to merge the currently sensed object point cloud (through the RGB-D camera) into the gradually built point cloud of the (partial) object model, in order to obtain a complete object model eventually; the other is to obtain the accurate pose (i.e., position and orientation) of the target object in the scene to enable object manipulation.

Image registration is conducted in three ways in our approach:

- pairwise registration between the object point cloud of one image and the point cloud of the subsequent image in the step by step rotation of the object on table for a total of 360 degrees;
- global registration of all object point clouds of the images captured from the same 360° rotation loop;
- registration between the point clouds of two partial object models.

The detailed algorithms for the above three ways of registration are described in the following sections.

7.2.1 Pairwise Registration of Neighboring Object Point Clouds

We use the following strategy to register neighbouring object point clouds obtained from RGB-D images taken at two adjacent steps from the same 360° degree rotation loop:

- Extract and match ASIFT [117] keypoints between the two RGB-D images of the objects, as shown in Fig. 43; if the matched keypoint pairs are sufficient, compute a transformation matrix based on their 3D coordinates, using the RANSAC [37, 89] and SVD [33] algorithms;

- Apply the ICP algorithm [11]; if a transformation matrix is computed from the previous step, use it as the initial input to the ICP algorithm; otherwise, use an identity matrix as the initial estimate. Note that as each rotation step is very small, our approach avoids large jumps in coordinates from one image to the other, which can cause the ICP algorithm to fail. Each rotation step can always be further reduced to guarantee robust registration if necessary.

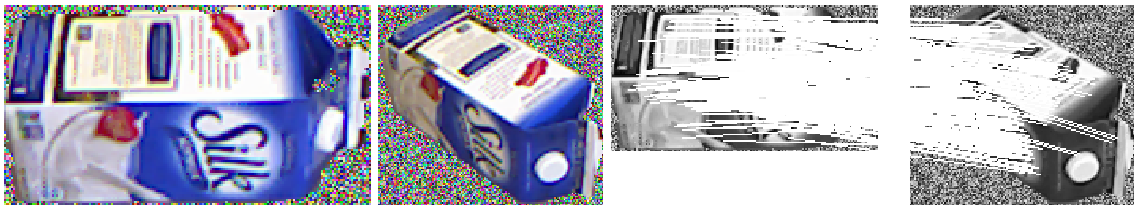


Figure 43: Two images of neighbouring object point clouds and their keypoints matching results

With an initial transformation estimate from the keypoints matching, the ICP algorithm converges much faster and also achieves registration results in better quality.

To make the registration more robust, we consider the following factors in evaluating the quality of matching of point pairs in the ICP algorithm:

- the distance between two points;

- the color similarity of two points based on RGB color values;

- the normal difference of two points;
- the distance from one point to the plane where the other point lies in;

A weighted sum of the above factors forms the evaluation function.

7.2.2 Global Optimization of Pairwise Registration Results

In order to reduce the accumulated registration error from pairwise registration, our approach further register all object point clouds obtained in each step of the rotation from the same 360° rotation loop using a global optimization algorithm, developed based on the virtual mate approach [82].

Suppose there are m RGB-D images captured from the same 360° rotation loop, and their corresponding object point clouds are also segmented out, denoted as C_1, C_2, ..., C_m. The frame (i.e., coordinate system) of point cloud C_i, $i = 1, ...m$, is denoted as O_i.

Using the strategies described in the section 7.2.1, the transformation from the frame of each object point cloud to the frame of its next neighbouring object point cloud is computed, denoted as ${}^{O_2}\hat{T}_{O_1}$, ${}^{O_3}\hat{T}_{O_2}$, ..., ${}^{O_m}\hat{T}_{O_{m-1}}$. Here, each ${}^{O_{i+1}}\hat{T}_{O_i}$, $i = 1, 2, ..., m-1$, is the locally optimized estimate of the transformation ${}^{O_{i+1}}T_{O_i}$ from O_i to O_{i+1} by registering the point clouds C_i and C_{i+1} directly, without considering globally the other object point clouds in the same 360° rotation loop.

Note that after 360° rotation of the object (on the table), the object point cloud C_m should be very similar to the object point cloud C_1, and the pairwise transformation from O_m to O_1 can also be estimated locally by registering C_m and C_1 directly, denoted as ${}^{O_1}\hat{T}_{O_m}$. Fig. 44 shows the frames of all the object point clouds obtained

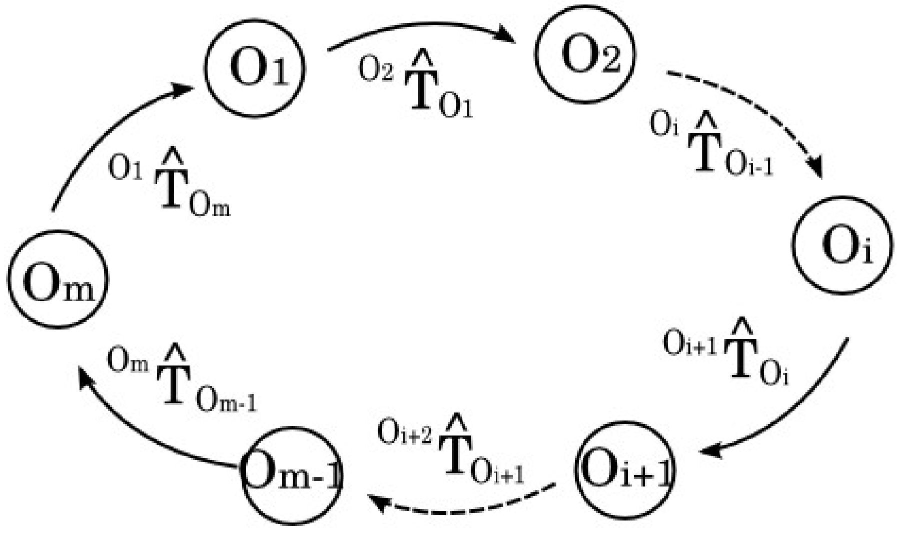


Figure 44: The frames of all the object point clouds in the same 360° rotation loop and their relations, illustrated as a closed loop

in the same 360° rotation loop and their relations.

The next step is to merge all the point clouds together. The issue is that, for each pair of the neighboring object point cloud frames O_i and O_{i+1}, the inverse ${}^{O_i}\hat{T}_{O_{i+1}}$ of their locally optimized transformation estimate ${}^{O_{i+1}}\hat{T}_{O_i}$ from the pairwise registration of the object point clouds C_i and C_{i+1} is not always exactly equal to the transformation transition result ${}^{O_i}\hat{T}_{O_{i-1}}...{}^{O_2}\hat{T}_{O_1}{}^{O_1}\hat{T}_{O_m}...{}^{O_{i+3}}\hat{T}_{O_{i+2}}{}^{O_{i+2}}\hat{T}_{O_{i+1}}$, which is estimated based on the relations among all the other object point cloud frames, on the path $O_{i+1} -> O_{i+2} -> ... -> O_m -> O_1 -> ... -> O_i$ from the 360° rotation loop. Take the frames of the object point clouds O_m and O_1 as an example, ideally, the inverse of the local transformation estimate ${}^{O_1}\hat{T}_{O_m}$ from the pairwise registration of the point clouds C_1 and C_m should be exactly equal to the transformation transition result ${}^{O_m}\hat{T}_{O_{m-1}}{}^{O_{m-1}}\hat{T}_{O_{m-2}}...{}^{O_3}\hat{T}_{O_2}{}^{O_2}\hat{T}_{O_1}$. However, due to the accumulated registration error in the transformation transition, they could be quite different from each other.

To build a partial object model out of those point clouds, we first create a basic frame of the object model, denoted as D, which initially has the same origin and axes as the frame of the last object point cloud O_m. Then, the pose of each object point cloud frame O_i, $i = 1, ..., m$, with respect to the object model frame D, denoted as ${}^{D}T_{O_i}$, is estimated from the transformation transition as follows:

$$ {}^{D}\hat{T}_{O_m} = I, \tag{23}$$

$$ {}^{D}\hat{T}_{O_i} = {}^{D}\hat{T}_{O_{i+1}} {}^{O_{i+1}}\hat{T}_{O_i}, \quad i = m-1, m-2, ..., 1. \tag{24}$$

In order to reduce the accumulated registration error embedded in the transformation transition result ${}^{D}\hat{T}_{O_i}$, $i = 1$, ..., m, our approach is to further refine ${}^{D}\hat{T}_{O_i}$ by registering the point cloud C_i to both its neighbours C_{i-1} and C_{i+1} based on virtually generated matched point pairs [82].

The virtually generated matched points purely based on the transformation matrix are also called *virtual mates*. To generate virtual mates between C_i and its neighbours C_{i-1} and C_{i+1}, we first sample points from C_i in the overlap regions between C_i and C_{i-1} and between C_i and C_{i+1} respectively.

Suppose p is a sampled point from C_i, with its position in the object point cloud frame O_i denoted as $\mathbf{p}$, and let q be the virtually generated mate of p:

- if p is sampled from the overlap region between C_i and C_{i+1}, then the position of its virtual mate ${}^{i+1}\mathbf{q}$ is obtained by transforming $\mathbf{p}$ from the frame O_i to the frame O_{i+1} based on ${}^{O_{i+1}}\hat{T}_{O_i}$;

- if p is sampled from the overlap region between C_i and C_{i-1}, then the position of its virtual mate ${}^{i-1}\mathbf{q}$ is obtained by transforming $\mathbf{p}$ from the frame O_i to the frame O_{i-1} based on ${}^{O_{i-1}}\hat{T}_{O_i}$, and ${}^{O_{i-1}}\hat{T}_{O_i}$ is the inverse matrix of ${}^{O_i}\hat{T}_{O_{i-1}}$.

Notice that virtual mates do not really exist in the point clouds C_{i-1} or C_{i+1}. They are introduced as the constraints of both ${}^{O_i}\hat{T}_{O_{i-1}}$ and ${}^{O_{i+1}}\hat{T}_{O_i}$ in the optimization to refine the pose ${}^{D}\hat{T}_{O_i}$. ${}^{O_i}\hat{T}_{O_{i-1}}$ and ${}^{O_{i+1}}\hat{T}_{O_i}$ cannot be directly used as constraints, because it is difficult to directly measure the change of the registration result from the change of the rigid 3D transformation matrix in the optimization.

Finally, we further transform all the positions of the generated virtual mates ${}^{i+1}\mathbf{q}$ and ${}^{i-1}\mathbf{q}$ to the object model frame D, based on the current poses of their corresponding object point cloud frames ${}^{D}\hat{T}_{O_{i+1}}$ and ${}^{D}\hat{T}_{O_{i-1}}$, respectively. The refined pose ${}^{D}\hat{T}_{O_i}$ is computed based on all of those virtually generated point pair positions $\mathbf{p}$ and $\mathbf{q}$ with respect to the object model frame D.

Our algorithm starts from the refinement of ${}^{D}\hat{T}_{O_m}$. If the difference of the pose ${}^{D}\hat{T}_{O_i}$ (in terms of the sum of the absolute differences of all matrix elements between two transformation matrices) before and after the optimization is greater than a tolerance threshold, then ${}^{D}\hat{T}_{O_{i-1}}$ and ${}^{D}\hat{T}_{O_{i+1}}$ also need to be refined; each pose ${}^{D}\hat{T}_{O_i}$ may be refined more than once, depending on C_{i-1} and C_{i+1}. Our algorithm ends when no more object pose needs to be refined or a maximum number of iteration is reached, and the globally optimized estimate ${}^{D}T_{O_i}$ for each object point cloud frame O_i is obtained. The algorithm can be implemented using a double-ended queue.

The pseudo code of our algorithm is given as follows:

Procedure: Global Optimization

```
SET ${}^{D}\hat{T}_{O_m} = I$
FOR $i = (m-1)$ to 1
    ${}^{D}\hat{T}_{O_i} = {}^{D}\hat{T}_{O_{i+1}}\,{}^{O_{i+1}}\hat{T}_{O_i}$
END FOR
SET Queue $= \emptyset$
Queue $\leftarrow {}^{D}\hat{T}_{O_m}$
WHILE Queue $\neq \emptyset$ and $num\_iterations < maximum\_num\_iterations$
    ${}^{D}\hat{T}_{O_i} = \text{Queue}.pop\_front()$
    ${}^{D}\hat{T}_{O_i} = \text{Align}(C_i, C_{i-1}, C_{i+1})$, subject to ${}^{D}\hat{T}_{O_{i-1}}$, ${}^{D}\hat{T}_{O_{i+1}}$, ${}^{O_i}\hat{T}_{O_{i-1}}$, and ${}^{O_{i+1}}\hat{T}_{O_i}$
    IF the change of ${}^{D}\hat{T}_{O_i} > tolerance\_threshold$
        Queue.$push\_back({}^{D}\hat{T}_{O_{i-1}}, {}^{D}\hat{T}_{O_{i+1}})$
    END IF
END WHILE
```

7.2.3 Registration of Partial Object Models from Different Rotation Loops

Once an optimized partial object model is obtained from each 360° object rotation loop, the next task is to register such partial models from different rotation loops. We still apply the ICP algorithm. Since changing rotation loops means changing the object pose by robotic manipulation, the transformation matrix for changing the object pose is used as the initial estimate of the transformation for the ICP algorithm. Notice that there usually exists a large overlap between two optimized partial object models, since a 360° rotation loop usually covers most of the side and top surfaces of the target object, which makes the registration more robust. A similar global optimization algorithm, such as [82], can also be introduced for the registration of all partial object models, if necessary.

7.3 Manipulation for Perception

As mentioned in Section 3.2.3, two kinds of manipulation are used to change object poses in our approach: one is rotating the target object in small steps by pushing during each 360° rotation loop, and the other is changing the object support surface by grasping, rotating, and pushing the object down at a different surface after each 360° rotation loop. In the following sections, we describe the details of the algorithms used for the two kinds of manipulation.

7.3.1 Pushing

Given that the target object sits stably on a table (or some other support structure), after the first RGB-D image of the object is taken, our strategy is to detect the leftmost edge from the object point cloud C_1 and the corresponding leftmost surface segment and choose a point with position $\mathbf{u_1}$ on the surface segment that is close to the middle of the leftmost edge as the point on the object for pushing. The direction for pushing $\mathbf{v_1}$ is oppose to the normal of the surface segment. Both $\mathbf{u}_1$ and $\mathbf{v}_1$ are with respect to the frame O_1. After the push, the camera can view more of the target object, and the next point cloud C_2 is obtained.

In general, given a point cloud C_{i+1}, i=1, ...,m, the pushing position and direction $\mathbf{u_{i+1}}$ and $\mathbf{v_{i+1}}$ with respect to frame O_{i+1} can be computed as:

$$\begin{bmatrix} \mathbf{u_{i+1}} \\ 1 \end{bmatrix} = {}^{O_{i+1}}\hat{T}_{O_i} \begin{bmatrix} \mathbf{u_i} \\ 1 \end{bmatrix}, \tag{25}$$

$$\begin{bmatrix} \mathbf{v_{i+1}} \\ 0 \end{bmatrix} = {}^{O_{i+1}}\hat{T}_{O_i} \begin{bmatrix} \mathbf{v_i} \\ 0 \end{bmatrix}, \tag{26}$$

where ${}^{O_{i+1}}\hat{T}_{O_i}$ is the transformation from O_i to O_{i+1}, as explained in Section 7.2.2.

The position and direction of each (ith) push can then be transformed to be with respect to the robot base frame, the corresponding robot hand frame to realize the pushing can be computed as the starting hand configuration, and the ending hand configuration can next be computed based on the pre-determined push distance. Subsequently, an automatic planner [119] is used to plan the robot arm motion to execute the push.

7.3.2 Change of Object Support Surfaces

After each 360° rotation loop, a decision has to be made, either to terminate the current object modeling procedure or to continue the model building using a different 360° rotation loop with a new support surface.

First, we update the oriented bounding box (OBB) of the current (partial) model of the target object, which consists of the object surface information from all the previous perception steps, as shown in Fig. 41. Based on the aspect graph of object poses (see Section 7.1), our approach simultaneously determines:

1. if new object surfaces can be observed by changing the support surface of the object, and how to change the support surface;
2. if no new object surface can be found so that the model building process can be terminated.

The basic idea of our approach is to form a finite set of candidate tasks for changing object support surfaces based on considering all possible sides of the current OBB and evaluate each candidate task based on a set of constraints/criteria to either (1) find the best task to execute or (2) to discover that the termination condition for model building is satisfied.

7.3.2.1 Candidate Tasks for Changing Object Support Surface

With the target object sitting on a table so that one side of the object OBB corresponds to the support surface of the object, there are five available OBB sides above the table that can be considered for the selection of a new support surface. Without losing generality, as shown in Fig. 41, we denote each OBB side based on their position in the OBB frame as the $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$ OBB sides, where the first five sides are the available ones above the table, and the corresponding object surfaces are already modeled (during the 360° rotation loop).

A robotic task to change the current object support surface s to a new support surface s' can be described at a high level in terms of two available sides of the object OBB involved: one side S_1 defines the approach vector of the robot hand/gripper by its normal pointing into the object, and the other side S_2 is the side for the new support surface. Thus, we denote a high-level candidate task simply as S_1S_2, and there are a total of 17 high-level candidate tasks: $+X-X$, $+X+Z$, $-X+X$, $-X+Z$, $+Y-Y$, $+Y+Z$, $-Y+Y$, $-Y+Z$, $+Z+X$, $+Z-X$, $+Z+Y$, $+Z-Y$, $+X+X$, $-X-X$, $+Y+Y$, $-Y-Y$, and $+Z+Z$. These candidate tasks will be evaluated so that the best one can be chosen. Once a candidate task is chosen and the actual change of the support surface is complete, the object can be pushed again in a new 360° loop to continue model building.

For each of the first 12 candidate tasks, the corresponding low-level robot motion can be described as the following: let the hand/gripper move along the inward-pointing normal of S_1, and squeeze two adjacent sides of S_1 that are not the side of the current support surface to grasp the object (note that there is already corresponding surface model of the object for both adjacent sides of S_1 to facilitate grasping); lift the object, and rotate it so that the side S_2 of the new support surface faces the table; put the object on the table by moving it downward along the normal of the new support surface, which is relatively flat surface corresponding to S_2. How to choose the new support surface given S_2 is described later. We call such a robot motion *one-step* motion to change the support surface of the object.

Note that each of the last 5 candidate tasks uses the same OBB side both for the approach vector of the hand/gripper and for deciding the new support surface. Hence, the corresponding low-level robot action actually consists of two one-step motions, with the first one-step motion landing the object on an intermediate support surface, and the second one-step motion setting the object on a support surface defined by the task. One may wonder why we even consider the last 5 candidate tasks instead of viewing each as a combination of two candidate tasks from the first 12 tasks. This is because not all support surfaces facilitate pushing well due to kinematic constraints of the robot manipulator and the relative arrangement between the target object and the robot manipulator, and an intermediate support surface could be one that does not facilitate pushing as well as the final support surface defined by one of the last 5 candidate tasks.

To either choose one of the above candidate tasks as the best one for execution or to discover that the modeling process can terminate, our algorithm evaluates both the requirements of object modeling and the manipulation constraints for task execution. We describe them in detail below.

7.3.2.2 Requirements of Object Modeling

Given a candidate task, which describes a new support surface side of the OBB, we evaluate the following factors:

- if there is enough overlap between the new visual images and the previous ones used for building the current partial model of the object to ensure robust registration,
- if the new object surfaces that can be observed are all the remaining surfaces for completing the object model, and

- if the quality of the object model is improved.

Note that there are only 5 new candidate OBB sides for a new support surface, and the observation of the target object under each new support OBB side is simulated in the following way: instead of actually rotating the object step by step in a 360° rotation loop under the new OBB side for support surface, our algorithm virtually move the camera around the object to capture the simulated view of the object after each rotation step via the ray tracing algorithm [51].

We further use a voxel map [111, 92, 51, 57, 27] to organize the existing observed information of the object within its current OBB and to facilitate evaluation of the simulated observation of the object for each candidate task. The voxel map discretizes the OBB into a grid of voxels, where each voxel can be in one of five states depending on whether and how it is observed: *unlabeled* to indicate that the status of the voxel is not known, *empty* to indicate that it is observed as not occupied by the object, *occupied* to indicate that it is observed as occupied by the object, *occluded* to indicate that it is observed as occluded by occupied voxels, and *occplane* to indicate that it is observed as occluded but with neighboring empty voxels or it is on the boundary of the voxel map.

The voxel map is built in the following way. It is first initialized with all voxels unlabeled. Next, occupied voxels are labeled based on the point cloud of the current (partial) object surface model, and then, from each previous camera view (i.e., each previous RGB-D image), empty and occluded voxels are labeled using the ray tracing algorithm [51]. Finally, occluded voxels adjacent to empty voxels are labeled as occplane voxels.

Note that both occupied and occplane voxels are the voxels on the surface of the target object, which can be observed during the modeling procedure. The main difference between them is that: occupied voxels have already been observed on the object surface from the previous perception, while occplane voxels are the hypothetical voxels on the object surface, which need to be validated from the planned perception. With new observation, occplane voxels could become empty or occupied voxels. If occplane voxels are found to be empty, then new occplane voxels could be generated based on the new empty voxels. Occluded voxels are always occluded by occupied or occplane voxels, and they are inside the hypothetical model of the target object. Fig. 45 shows a cross section of the voxel map built based on the partial object model.

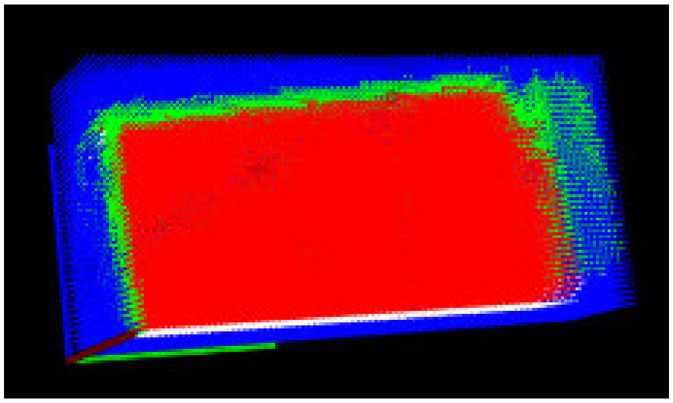


Figure 45: A cross section of the voxel map built based on the point cloud of the current partial model of the object *milk box blue* blue - empty voxels; green - occupied voxels; red - occluded voxels; white - occplane voxels

Based on the voxel map, for a given candidate task, we can further detect both new and overlap surfaces for evaluation: the occupied voxels observed in simulated camera views (from the candidate task) indicate overlap surfaces, while the occplane voxels observed indicate new surfaces. We compute the area factor f_{area} proposed in [111] that combines both the ratios of occupied and occplane voxels for evaluation purpose.

If, after evaluating all candidate tasks, the following two conditions are satisfied, our object modeling procedure can be terminated:

1. No more occplane voxel can be observed, which means the model of the target object cannot be further improved, and the object model could be complete[2].

2. Occplane voxels do not exist any more, which means the object model is complete.

Note that when either one of the above two termination conditions is achieved, f_{area} becomes zero.

The model quality f_{quality} is used to estimate the quality of further model building, given a candidate task for changing the object support surface. Suppose there are n_{occupied} occupied voxels from n_{camera} simulated views with the new support surface, the model quality is computed as:

$$f_{\text{quality}} = \frac{1}{n_{\text{occupied}}} \sum_{j=1}^{n_{\text{occupied}}} \max_{l=1}^{n_{\text{camera}}} n_{lj} \cdot n_j, \tag{27}$$

where n_{lj} is the normalized camera orientation vector from the occupied voxel to the (simulated) camera origin, and n_j is the object surface normal of the occupied voxel. Similar algorithms have been used in [4, 111].

[2]However, it could still be incomplete, if the surface of the target object is very complicated with many small concavities, and in such cases, the modeling procedure would require more viewing directions than the current procedure considers, as discussed in Chapter 8.

7.3.2.3 Manipulation Constraints

Given a candidate task, we evaluate the following factors to assess how easy the task can be executed:

- if the target object can be grasped from the given OBB side for the approach direction;
- if an approximately flat surface s' exists for the given OBB side for support, and if the target object can stand stably on the surface s';
- if the target object can be pushed effectively with the new support surface;
- if the manipulation utility is sufficient.

Theoretically, to make an object stable, two conditions must be satisfied: the support surface should define a surface region (i.e., at least consists of three points) that has a sufficiently large area, and the projection of the center of gravity of the object on the surface of the support region falls inside the support region. In our approach, we use the center of the OBB as the estimated gravity center and look for a nearly flat surface based on the support surface side of the OBB, in which the projection of the center of the OBB falls.

We use a Boolean function $f_{\text{stability}}$ to measure the stand stability of the object. If the above conditions are satisfied, we simply assign the value of $f_{\text{stability}}$ as 1; otherwise, it is assigned the value 0.

We also use a Boolean function f_{push} to measure if the target object can be pushed effectively during the new 360° rotation loop for the considered candidate task. In our current experiment setting, if the target object is too narrow compared to the size of the gripper, it is difficult to be pushed in a consistent way. Therefore, we introduce a threshold as the minimum object width, beyond which the object can be rotated by pushing effectively. f_{push} is assigned as 1 or 0, depending on if the requirement of the minimum object width is satisfied.

We further use a Boolean function f_{grasp} to measure if the target object can be successfully grasped from the considered approach vector. We use a simple gripper model with two fingers, and check if two approximately parallel surfaces (based on their surface normals) on the two contacted OBB sides can be found for grip; we also check if the approached OBB side is too wide to fit into the gripper. f_{grasp} is assigned a value 1 or 0, depending on if both the above grasp constraints can be satisfied.

The manipulation utility f_{pathcost} is used to evaluate the path cost of the robotic motion for changing the object support surface, given a candidate task. We use a mathematical function [111] that is monotonically decreasing with respect to the path cost for the evaluation. The greater the path cost, the smaller the numerical value of the manipulation utility f_{pathcost}.

7.3.2.4 Evaluation of Candidate Tasks

To evaluate all the candidate tasks for the change of the current object support surface, we use a combination strategy similar to [92, 111] to combine all the factors described above as follows:

$$\Theta = f_{\text{area}} f_{\text{stability}} f_{\text{push}} f_{\text{grasp}} (1 + f_{\text{pathcost}} + f_{\text{quality}}). \tag{28}$$

If the Θ of one candidate task equals zero, then it is an invalid task. Based on all the valid candidate tasks, we choose the one with the greatest Θ for execution.

Notice that when Θ becomes zero for all candidate tasks, it indicates that one of the two termination conditions of our object modeling procedure has been achieved, as described in Section 7.3.2.2.

7.4 Experiments and Analyses

In this section, our experiments are described, and detailed experiment analyses is also presented.

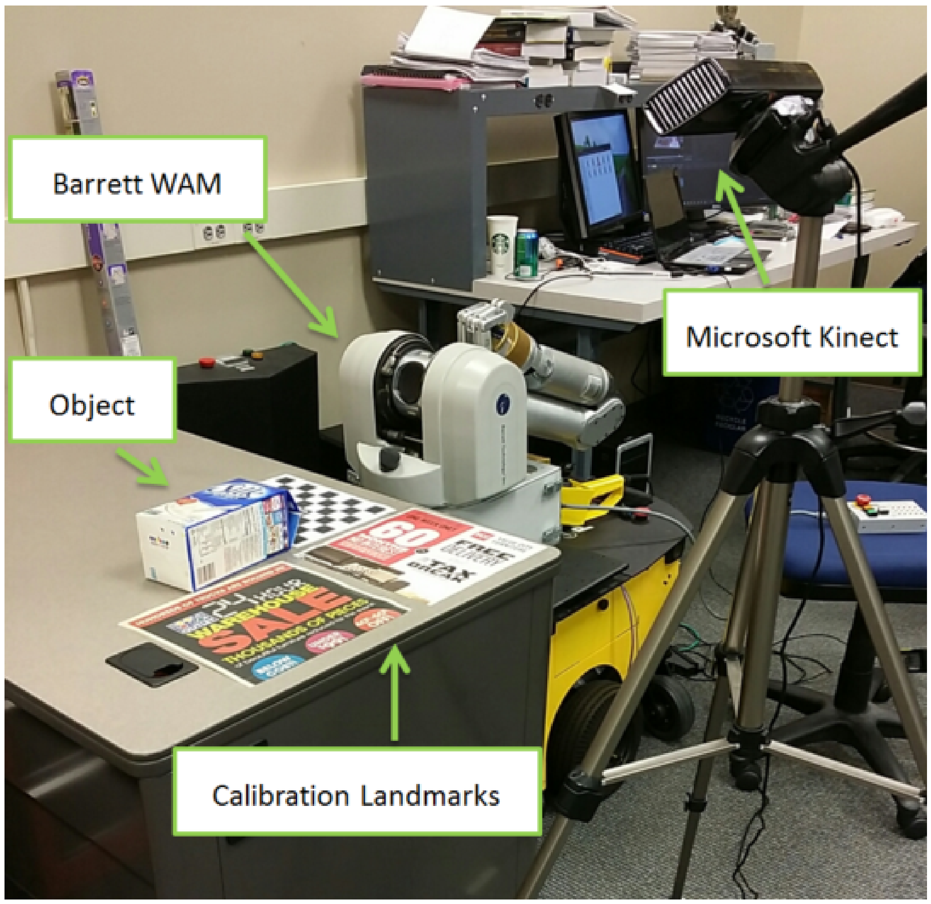


Figure 46: The set up of our experiments: a Microsoft Kinect camera is used for the perception and a Barrett WAM arm is used for the object manipulation

7.4.1 Set Up of Our Experiments

Fig. 46 shows the set up of our experiments for automated 3D object modeling. The target modeling object is positioned on a table, in the center of the field of view of the Microsoft Kinect camera. Colorful images on the table are used as landmarks for the calibration between the robot and camera coordinate systems. A Barrett WAM arm with 6 degrees of freedom is used for object manipulation.

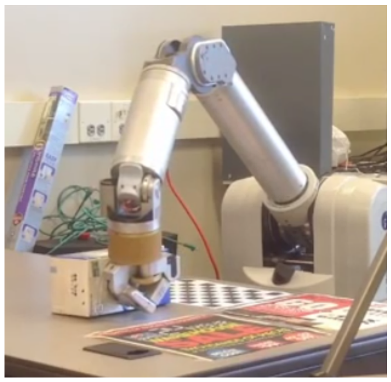
7.4.2 Experiments

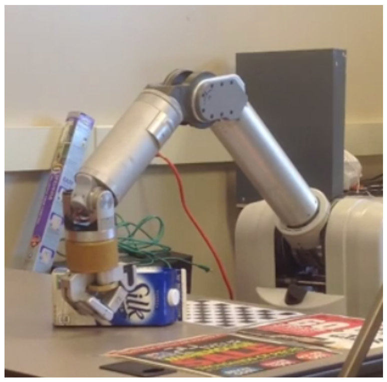
The two important procedures in our automatic model-building process, the 360° rotation loop and the change of object support surfaces, are illustrated as follows, using the object *milk box blue* as an example.

Fig. 47 provides a few snapshots of the object *milk box blue*, rotated by the Barrett WAM arm through pushing step by step in the 360° rotation loop for model modeling.

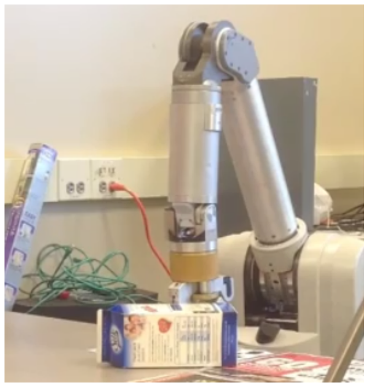
Fig. 48 provides a few snapshots of the change of the support surface, for the model building of the object *milk box blue*. The task executed was chosen from the candidate tasks that use the same OBB side for both the approach vector and the new support surface, and thus, it was completed in two consecutive one-step motions. Notice that the top side of the object *milk box blue* is used as both the approached side for grasping and the new support surface side in this example.

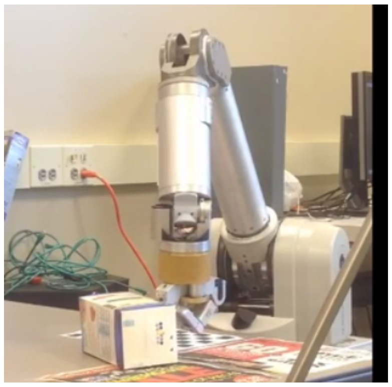
All our algorithms are currently implemented in single threads. To ensure safety, robotic manipulation is executed at a low speed. The time cost mainly comes from 360° rotation loops. For each step in the 360° rotation loop, it takes about 1 minute


(a)


(b)

(c)


(d)


(e)

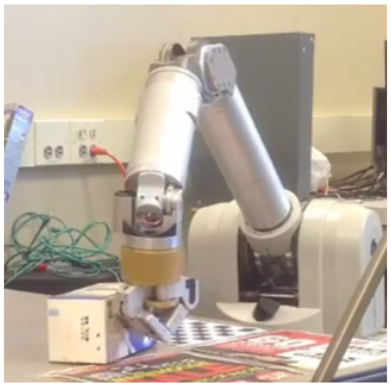

(f)

Figure 47: A few snapshots of the object *milk box blue*, rotated by the Barrett WAM arm through pushing step by step in the 360° rotation loop for model modeling

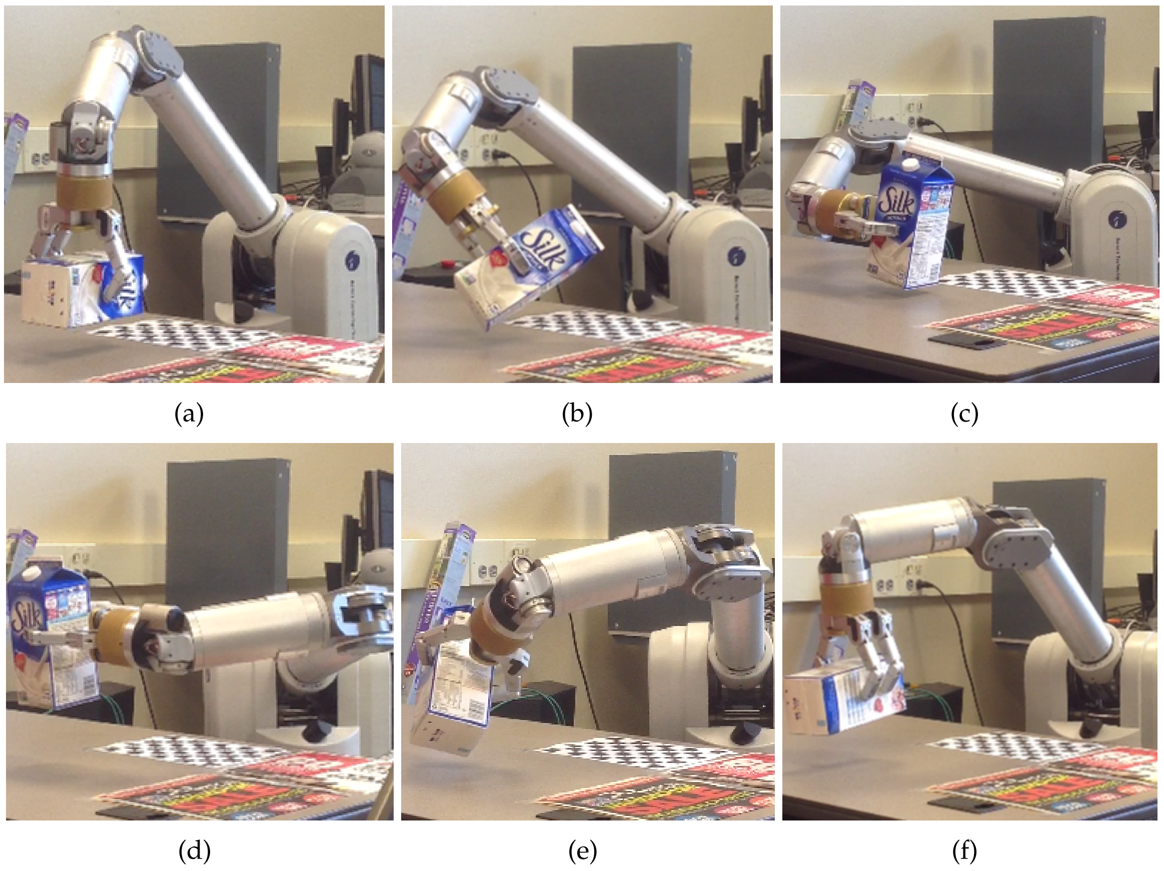


Figure 48: A few snapshots of the change of the support surface, for the model building of the object *milk box blue*

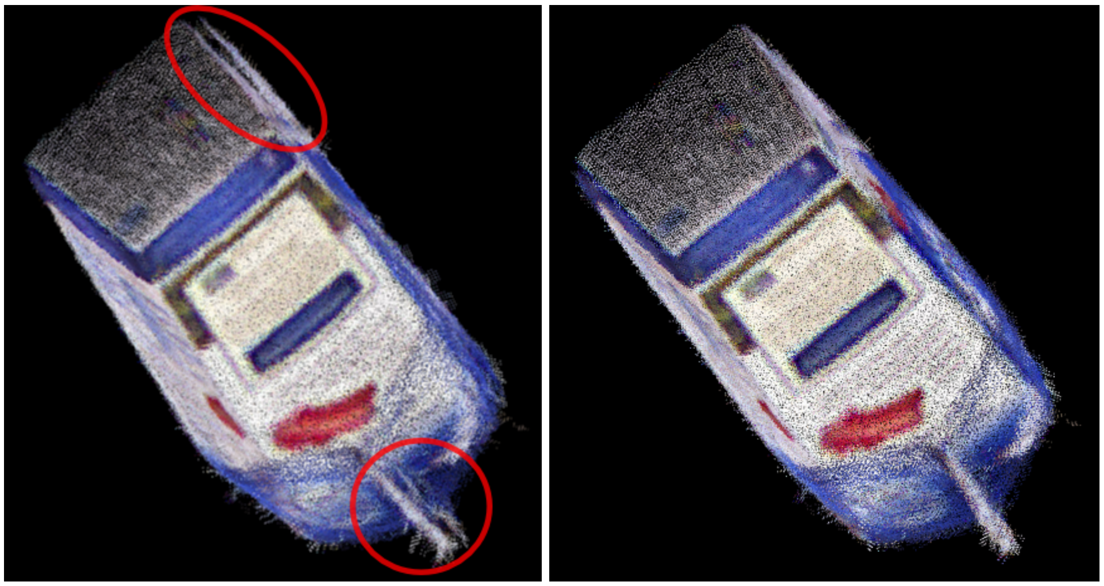
for the registration of neighbouring point clouds and $2 \sim 3$ minutes for the execution of pushing by the robot end effector. Each 360° rotation loop consists of about $10 \sim 15$ steps. Our whole object modeling procedure usually requires more than two 360° rotation loops, depending on the surfaces of the modeled object. However, it is important that the whole object modeling procedure is automated by our algorithms without the time and effort of a human operator.

7.4.3 Comparison of the Registration Results: before and after Optimization

Fig. 49 compares the registration results before and after global optimization, using all the point clouds of the object *milk box blue*, captured from a 360° rotation loop: (a) shows the partial model of the object *milk box blue*, built before global optimization, in which small distortion of the object surfaces (in the red circles) can be clearly observed; (b) shows the partial model of the object *milk box blue*, refined by the global optimization, in which all the distorted parts of the object surfaces are corrected.

We now provide detailed statistical analysis about how the quality of the partial model of the object *milk box blue* is improved by the global optimization in two different ways, based on the partial models of the object *milk box blue*, shown in Fig. 49.

A statistical comparison of the qualities of the partial models of the object *milk box blue*, built before and after global optimization is presented in Fig. 50, based on the mean K-nearest neighbour distances of all the points in the point clouds of both models ($K = 100$). As we can see from the histogram in Fig. 50, the



(a) Before Optimization (b) After Optimization

Figure 49: The comparison of the registration results before and after global optimization, using all the point clouds of the object *milk box blue*, captured from a 360° rotation loop: (a) shows the partial model of the object *milk box blue*, built before global optimization, in which small distortion of the object surfaces (in the red circles) can be clearly observed; (b) shows the partial model of the object *milk box blue*, refined by the global optimization, in which all the distorted parts of the object surfaces are corrected

object point cloud after global optimization has a greater number of points with their mean *K*-nearest neighbor distances in the ranges 1.4 *mm* $\sim$ 1.7 *mm* and 2.5 *mm* $\sim$ 3.4 *mm*, while the object point cloud before global optimization has a greater number of points with their mean *K*-nearest neighbor distances in the ranges 1.8 *mm* $\sim$ 2.4 *mm* and 3.5 *mm* $\sim$ 5.4 *mm*. Therefore, the histogram shows two main distribution changes of the mean *K*-nearest neighbor distances of all points in the object point cloud before and after global optimization: many points have their mean *K*-nearest neighbor distances reduced from the range 1.8 *mm* $\sim$ 2.4 *mm* to the range 1.4 *mm* $\sim$ 1.7 *mm*; and also many points have their mean *K*-nearest neighbor distances reduced from the range 3.5 *mm* $\sim$ 5.4 *mm* to the range 2.5 *mm* $\sim$ 3.4 *mm*. As the result, Fig. 50 indicates that the points in the point cloud of the partial model are distributed more closely after global optimization.

Fig. 51 shows a statistical comparison of the qualities of the partial models of the object *milk box blue*, before and after global optimization, based on the numbers of the neighbors of all the points in the point clouds of both models, within the radius of 3 *mm*. As we can see from the histogram in Fig. 51, the object point cloud after global optimization has a greater number of points with their numbers of neighbors within the radius of 3 *mm* in the ranges 30 $\sim$ 60 and 155 $\sim$ 215, while the object point cloud before global optimization has a greater number of points with their numbers of neighbors within the radius of 3 *mm* in the ranges of 5 $\sim$ 25 and 65 $\sim$ 150. Therefore, the histogram also shows two main distribution changes of the numbers of neighbors within the radius of 3 *mm* of all points in the object point cloud before and after global optimization: many points have their numbers

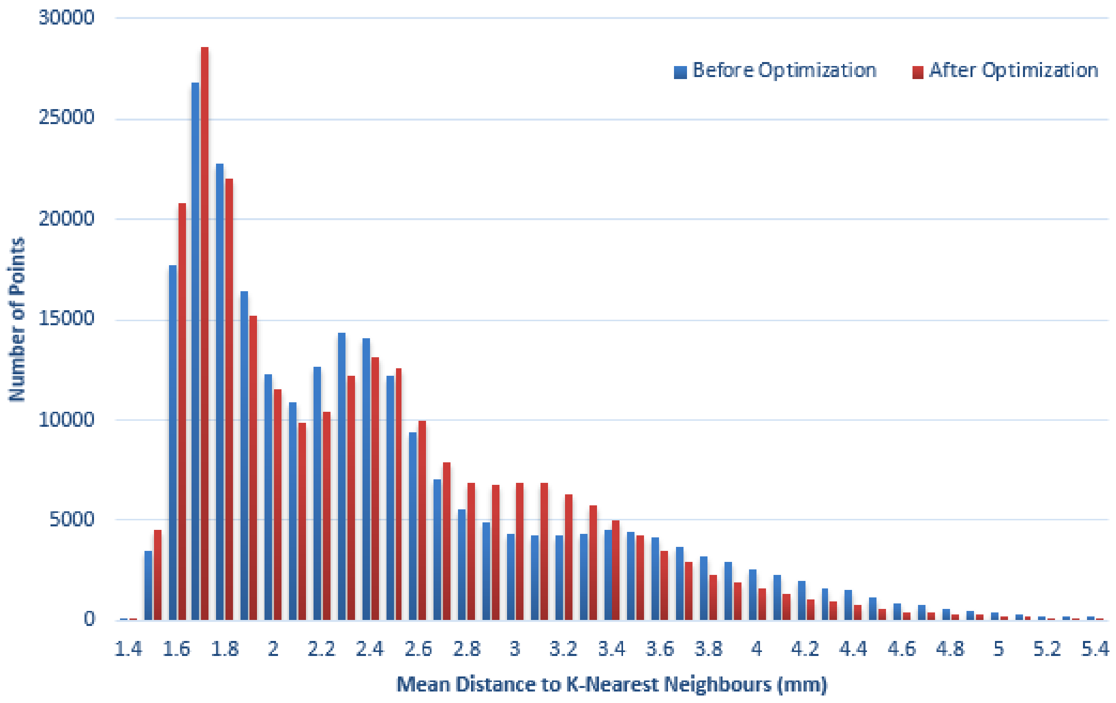


Figure 50: A statistic comparison of the qualities of the partial models of the object *milk box blue*, before and after global optimization, as shown in Fig. 49, based on the mean K-nearest neighbor distances of all the points in the point clouds of both models ($K = 100$)

of neighbors within the radius of 3 *mm* increased from the range 5 $\sim$ 25 to the range 30 $\sim$ 60; and also many points have their numbers of neighbors within the radius of 3 *mm* increased from the range 65 $\sim$ 150 to the range 155 $\sim$ 215. Hence, Fig. 51 also indicates that the points in the point cloud of the partial model are distributed more closely together after global optimization.

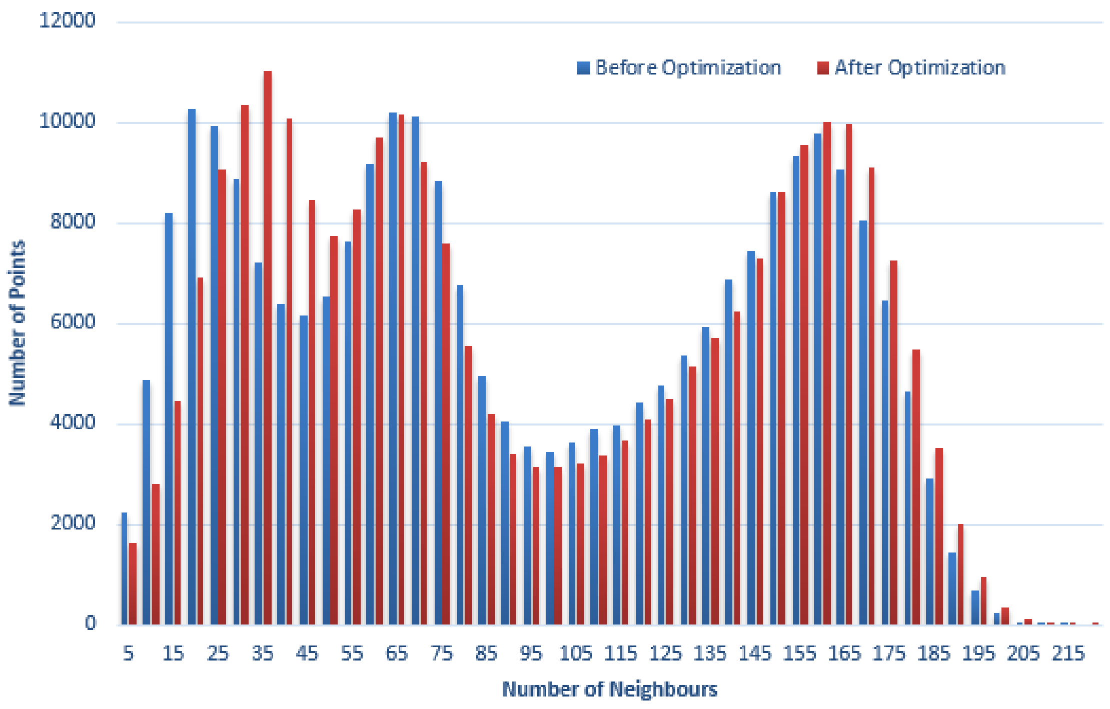
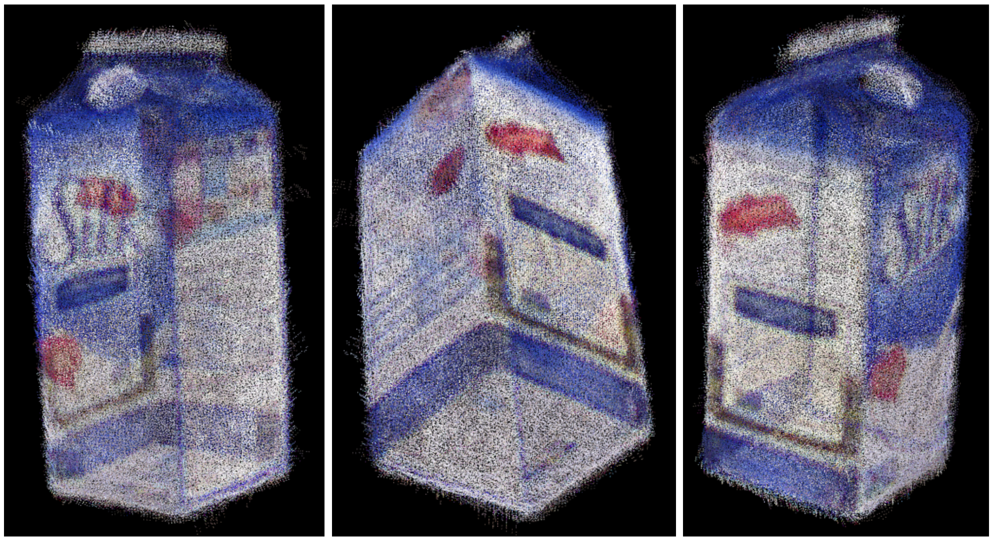


Figure 51: A statistical comparison of the qualities of the partial models of the object *milk box blue*, before and after global optimization, as shown in Fig. 49, based on the numbers of the neighbors of all the points in the point clouds of both models, within the radius of 3 *mm*

Both the smaller mean *K*-nearest neighbor distance and the greater number of neighbors within the same radius demonstrate that all point clouds of the object *milk box blue* used for the current model building are aligned better together after global optimization.

7.4.4 Analyses of Voxel Maps

As described in Section 7.3.2, a voxel map is built based on the oriented bounding box of the point cloud of the object model, which is updated in real time as more information of the target object is sensed. In our experiments, the voxel size is set as $3\ mm \times 3\ mm \times 3\ mm$.



(a) (b) (c)

Figure 52: The built complete model of the object *milk box blue*, visualized from different viewpoints

The complete model of the object *milk box blue*, as shown in Fig. 52, is built using two 360° rotation loops. All the information of the voxel map after each 360° rotation loop is given in Table 7.

As we can see from Table 7, after two 360° rotation loops, there are only a few occplane voxels left in the voxel map, and the model of the object *milk box blue* is almost complete. From the 1st loop to the 2nd loop, occplane voxels are greatly reduced, indicating the great reduction of the unobserved region after the 2nd

Table 7: The voxel map after each 360° rotation loop for *milk box blue*

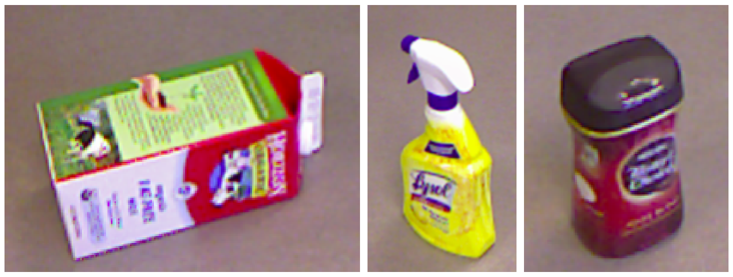
Loop no.	Size of OBB (mm^3)	Size of voxel map	#Empty	#Occupied	#Occluded	#Occplane
1	128 × 257 × 119	45 × 88 × 42	72299	21275	70237	2509
2	128 × 261 × 123	45 × 90 × 44	86826	30902	60448	24

loop. This is also reflected in the significantly increased number of occupied voxels. However, the number of increase in occupied voxels is much greater than the number of reduction in occplane voxels; this is because the occupied voxels might come from both the previously occluded and occplane voxels. Besides, due to the limited depth accuracy of the Microsoft Kinect camera and the sensing noise, after image registration from multiple views, the object surface is usually thicker than one voxel, meaning that more occupied voxels are generated. The few remain occplane voxels might be caused by the sensing inaccuracy and noise, or tiny self-occluded surfaces which cannot be observed by more 360° rotation loops.

Fig. 53 shows more built models of daily objects, with their corresponding OBBs. Notice that, even though many daily objects do not have rectangular shapes similar to their OBBs, for example, the object *spray* shown in Fig. 53, there are still smooth surfaces corresponding to some OBB sides that can be used in our approach.

7.5 Conclusions of this Chapter

To conclude, in this chapter, we have introduced a method to build 3D object models from RGB-D images automatically by interleaving model building with


(a) (b) (c)

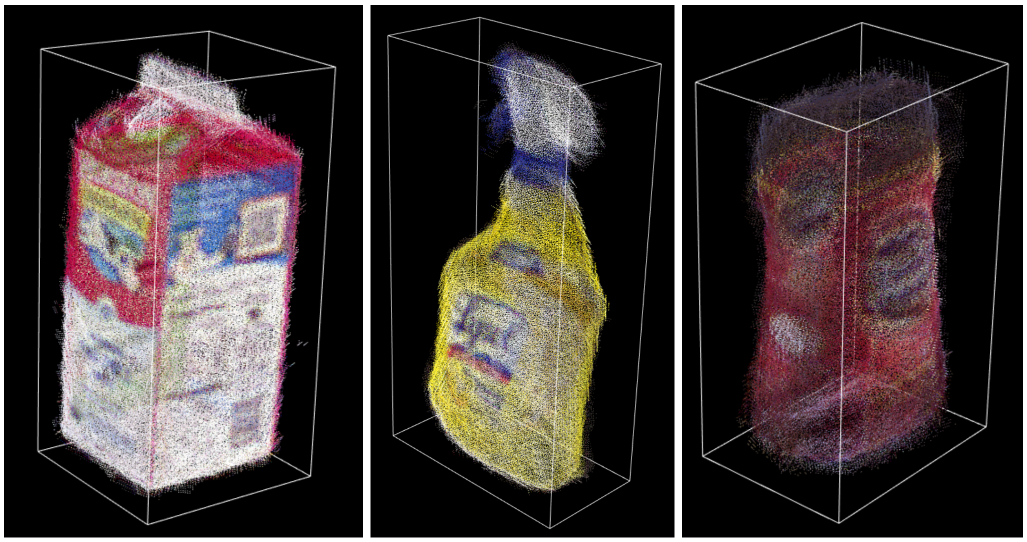

(d) (e) (f)

Figure 53: More built models of daily objects, with their corresponding OBBs: from left to right, they are *milk box red*, *spray* and *coffee box*

robotic manipulation. Using a fixed RGB-D camera and starting from the first view of the object, our approach gradually builds a complete object model by using a partially built model (based on what has been visible) to guide the pose change of the object by a robotic manipulator to make more surfaces visible for continued model building. The expanded partial model further guides the robotic pose change, and so on. The alternation of perception-based model building and pose changing continues until a complete object model is built with all object surfaces covered.

As shown in Fig. 47, by introducing an automatic push procedure, our approach does not assume that the robot can achieve a first grasp of the object somehow [55] without the knowledge of the object model, i.e., we do not assume the need of human guidance to achieve an initial grasp. Our approach only assumes that the object is initially placed stably on a support structure, such as a table, so that it can be pushed, which is usually feasible for most daily objects with otherwise diverse shapes. After observing many side surfaces of the object through the pushing loop, automatic grasping of the object becomes possible, as shown in our experiments.

The experimental results also show that our approach tolerates motion uncertainty in pushing the object quite well: the motion does not have to be a precise rotation, some translation is fine, as long as an adjacent, previously unseen surface of the object can be seen after the push motion, and the amount of push is small so that a new image has overlapping regions with the previous image. Our approach does not require a real-time tracking algorithm. It does not involve the robotic manipulator in any image of the target object so that there is no need to deal with separation of the manipulator (end-effector) from the object in an image.

A global optimization algorithm, especially for the registration of all object point clouds obtained in each step of the rotation from the same 360° rotation loop, is developed based on the virtual mate approach [82]. Experiment results show that our algorithm can effectively improve the quality of the built object model.

The oriented bounding box, updated along the way as new information of the object is sensed, is introduced to help capture all the aspects/views of the target object, and also to reduce the difficulty of the real robotic manipulation planning.

Our approach provides a promising solution for automatic 3D rigid object modeling in situations where a camera cannot be moved and re-positioned freely, for example, in cluttered environments, and where objects cannot be pre-positioned on any special hardware (such as a turntable) or pre-grasped. Moreover, in many situations, because of some constraints, such as background occlusion and object self-occlusion, it is usually difficult to view all surfaces of an object by altering only the camera viewpoint. Our approach of interleaving perception with manipulation provides more flexibility to enable observing all object surfaces and building a complete object model.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the contributions of this dissertation and discuss possible future work.

8.1 Object Detection and Pose Estimation

In Chapters 4 and 5, an appearance-based approach to general 3D object detection and pose estimation is introduced based on segmented 3D surfaces and their features, taking full advantage of RGB-D information.

8.1.1 Contributions

Our approach is characterized by the following novel aspects:

- representation of object models, based on segmented smooth surfaces of RGB-D images from different sensor views of different object poses, which allows modeling of objects with disconnected surface patches and missing surface information in object images and does not require precise information of the locations of the object and the camera where images were taken;
- a surface-based framework for object detection and pose estimation, capable of detecting partially visible objects due to either occlusion or surface transparency and multiple instances of the same object as well as estimating the 6 DoF poses of each object in a cluttered 3D scene by taking advantage of the surface-based object models;

- a model-fitting based global optimization algorithm, which integrates the recognition and pose estimation results of all surfaces from the single test image and effectively solves the ambiguities caused by redundant or incorrect results.

Compared to existing approaches that solely rely on matching keypoints (pairs) for both object detection and 6DoF pose estimation, our approach for recognition of surface segments is more flexible to incorporate various kinds of existing mature object features and learning algorithms; also, the time complexity of our approach in keypoints matching is constant with respect to the number of objects in the dataset.

With our approach, the results of detected objects and their poses with respect to the current RGB-D camera frame can be used directly for robotic manipulation, with the pose of the camera frame to the robot base frame known. The reconstructed 3D scene can also be used directly for robot motion planning.

8.1.2 Future Work

To make 3D object detection in our approach more general and robust, other different kinds of features can be introduced, such as textures, shapes and other geometrical features. As for pose estimation, keypoints or keypoint pairs based on geometrical features, which are more invariant to viewpoint changes, can be further integrated; other geometric primitives, such as normals, curvatures, lines and planes, can be also used to make pose estimation more accurate. More efficient pose estimation algorithms can help our approach run in real time. However, the detection and pose estimation of non-rigid objects is still an open issue.

Semantic segmentation can greatly improve our approach, even though it is still an unsolved challenge. First, our current approach cannot deal with some extreme situations when objects are placed side by side very closely and all their surfaces are separated as one large smooth surface segment. In such cases, other kinds of features, domain knowledge or knowledge obtained from learning should be utilized to help the correct segmentation of the input image. Second, in our approach, objects are represented with all its smooth 3D surfaces, however, with more advanced semantic segmentation algorithms, objects can be represented with several semantic surface parts, which are not necessarily smooth but also have a lot of geometrical details. For example, a bicycle can be roughly represented with the wheels, the frame, the saddle area and the front set. Given target objects with their semantic surface parts, our approach can start from semantic segmentation, then apply 3D object detection and pose estimation based on semantic segments, and end with graph-cut to merge redundant detected objects and also remove incorrectly detected objects. Our approach can be applied in more general situations.

Our approach only provides all the detected objects and their poses in the target scene, however, more useful information should be further analysed for many daily tasks, for example, the structure of all the objects in the target scene and how they are physically related to each other; such kind of information is necessary to ensure robotic manipulation is successful and safe.

To improve the time efficiency of our approach, many procedures in our program, such as normal estimation, feature extraction, and keypoints matching of each surface segment, can be readily implemented in parallel. Besides, the time efficiency of our approach can be further improved by choosing more efficient but also effective keypoints or features for pose estimation.

8.2 Learning-based View Evaluation

In Chapter 6, a learning-based approach is introduced for evaluating scene recognizability of a cluttered scene with objects occluding one another from a single view and ranking views based on their scene recognizability.

8.2.1 Contributions

To evaluate scene recognizability of a view, we take into account the quality and quantity of recognized and reconstructed objects and the background from the view rather than mere visibility through learning. Our approach is validated by experimental results, which also provide evidence that better visibility does not necessarily lead to better recognizability of objects because of the effects of occlusion, illumination, and weak visual features. Those effects are captured and reflected in our learning-based approach for view evaluation.

Compared to most of the existing work which is only focused on how to better recognize individual objects, our approach of evaluating scene recognizability points provides a more accurate assessment of how good a view is for autonomous robotic tasks that require scene understanding for object recognition and manipulation in cluttered environments. Our learning-based approach also makes it easy for the view evaluator to adapt to the characteristics of specific types of task environments or scenes.

8.2.2 Future Work

The evaluation result of a view can be even more informative for tasks of different purposes. Besides scene recognizability, for example, if the purpose of the task is for object manipulation, a good view should indicate enough space to avoid obstacles and to reach target objects, and also easier object stacking structures. As another example, if the purpose of the task is only concerned with several specific objects, these target objects should play more important roles than all the other objects in view evaluation. All those task-related factors should be taken into account in the training of view evaluators for different tasks.

8.3 Object Modeling through Perception and Manipulation

In Chapter 7, a method is introduced to build complete 3D object models for unknown objects from RGB-D images automatically by interleaving perception with robotic manipulation.

8.3.1 Contributions

We provide a novel and systematic framework that effectively integrates RGB-D perception of an object and robotic manipulation of the object to facilitate both object modeling and object manipulation. Our approach enables the building of a complete object model (if all surfaces of the object can be observed through manipulation), which cannot be achieved by most of existing work.

By introducing an automatic push procedure, our approach does not assume that the robot can achieve a first grasp of the object somehow without the knowledge of the object model, i.e., we do not assume the need of human guidance to achieve an initial grasp. Our approach only assumes that the object is initially placed stably on a support structure, such as a table, so that it can be pushed, which is usually feasible for most daily objects with otherwise diverse shapes. After observing many side surfaces of the object through the pushing loop, automatic grasping of the object becomes possible.

Compared to existing work, our approach does not require a real-time tracking algorithm, and it tolerates motion uncertainty in pushing the object quite well. Our approach also does not involve the robotic manipulator in any image of the target object so that there is no need to deal with separation of the manipulator (end-effector) from the object in an image.

By using the real-time updated oriented bounding box of the partial model of the object, our approach nicely balances the requirement of the full view coverage for model building and robotic manipulation planning. We have introduced more factors of manipulation constraints in our modeling procedure, which are not considered by most of existing work.

In general, our approach provides the flexibility to enable observing all object surfaces and building a complete object model.

8.3.2 Future Work

Our work of automatic object modeling through perception and manipulation can be further developed to facilitate manipulation of unknown objects in cluttered environments. For instance, in order to reach a known object occluded by unknown objects in a cluttered environment, the interleaving perception and manipulation can guide the robot manipulator to gradually move the unknown objects away if possible, either for the need of creating models of the unknown objects or for the need of having just partial models of the unknown objects sufficient to serve the purpose of guiding the robot to securely grasp them and remove them.

How much perceived information of an object, which results in a partial object model, is necessary to guide manipulation depends on the feedback from manipulation, e.g., if the robot can successfully push or grasp the object. If the first try of grasping is not successful, more perception is needed so that the robot can explore a different way of grasping or pushing, etc. This process of interleaving perception and manipulation has more to do with (partially) modeling the physical shape, pose, and weight of the object so that robotic manipulation can be accomplished, regardless of the reason of manipulation.

To build the models of objects with more complicated shapes through our proposed approach, it is possible to introduce a polyhedron as a more refined bounding volume, which still balances the requirement of the full view coverage for model building and the difficulty of the robotic manipulation planning well. To build the models of objects of the same shape and size, the model information of the first object can help speed up the registration of the captured point clouds of other objects.

Our approach introduces many factors, such as the stand stability, the push effectiveness and the grasp validity, for 3D object modeling through perception and manipulation. Most of them are still open challenges for objects with arbitrary and complicated shapes, and better solutions for each of them can also help improve the robustness and flexibility of our approach.

As for the issues mentioned in Section 7.1 and Section 7.3.2.2, more views of the target object with complicated surfaces for modeling can be further provided in two ways: one is to use the robot manipulator to move the camera around the target object [71, 70, 113, 38, 56, 111], which requires the algorithms of view planning for guidance, and the other is to use the robot manipulator to hold the target object in different poses [55, 67, 15], which requires the algorithms for both the tracking of the target object and the segmentation of the target object from the background (including the robot manipulator). By systematically integrating both strategies discussed above into our approach, complete models of most objects can be successfully achieved.

REFERENCES

[1] The digital Michelangelo project. Website: graphics.stanford.edu/projects/mich. Accessed: 2015-04-20.

[2] Geometric tools. Website: geometrictools.com. Accessed: 2016-07-01.

[3] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables, 9th printing*. Dover Publications, 1972.

[4] M. T. L. Albalate, M. Devy, J. Miguel, and S. Marti. Perception planning for an exploration task of a 3D environment. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 704–707, 2002.

[5] N. Atanasov, B. Sankaran, J. L. Ny, T. Koletschka, G. J. Pappas, and K. Daniilidis. Hypothesis testing framework for active object detection. In *IEEE International Conference on Robotics and Automation, (ICRA)*, 2013.

[6] R. Bajcsy. Active perception. In *Proceedings of the IEEE, Special issue on Computer Vision*, volume 76, pages 966–1005, 1988.

[7] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[8] I. Bartolini, P. Ciaccia, and M. Patella. Warp: Accurate retrieval of shapes using phase of fourier descriptors and time warping distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):142–147, Jan 2005.

[9] H. Baya, A. Essa, T. Tuytelaarsb, and L. V. Goola. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, Jun 2008.

[10] S. Berretti, A. D. Bimbo, and P. Pala. Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Mutimedia*, 2(4):225–239, 2000.

[11] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[12] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

[13] L. Bo, X. Ren, and D. Fox. Learning hierarchical sparse features for RGB-(D) object recognition. *The International Journal of Robotics Research (IJRR)*, 33:581–599, 2014.

[14] K. Bowyer and C. Dyer. Aspect graphs: An introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2:315–328, 1990.

[15] B. Browatzki, V. Tikhanoff, G. Metta, H. Bulthoff, and C. Wallraven. Active in-hand object recognition on a humanoid robot. *IEEE Transactions on Robotics*, 30(5):1260–1269, Oct 2014.

[16] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bulthoff, and C. Wallraven. Active object recognition on a humanoid robot. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 2021–2028, 2012.

[17] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[18] J. B. Burns and L. Kitchen. Recognition of 2D images of 3D objects from large model bases using prediction hierarchies. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI'87)*, volume 2, pages 763–766, 1987.

[19] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

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

[21] C. Choi and H. I. Christensen. 3D pose estimation of daily objects using an RGB-D camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3342–3349, 2012.

[22] C. Choi, Y. Taguchi, O. Tuzel, M. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3D sensor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1724–1731, 2012.

[23] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research (IJRR)*, 30:1284–1306, 2011.

[24] C. Connolly. The determination of next best views. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 432–435, 1985.

[25] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[26] J. J. Craig. *Introduction to robotics: mechanics and control (3rd Edition)*. Prentice Hall, 2004.

[27] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM SIGGRAPH 1996*, pages 303–312, 1996.

[28] C. M. Cyr and B. B. Kimia. 3D object recognition using shape similiarity-based aspect graph. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 254–261, 2001.

[29] N. Dalal and B. Triggs. Histograms of orientaed gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[30] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(5):1–60, 2008.

[31] M. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158, Feb 2002.

[32] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005, 2010.

[33] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-D rigid body transformations: A comparison of four major algorithms. *Machine Vision and Application*, 9:272–290, 1997.

[34] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 990–997, 2010.

[35] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32:1627–1645, 2010.

[36] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[37] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[38] T. Foissotte, O. Stasse, A. Escande, P. Wieber, and A. Kheddar. A two-steps next-best-view algorithm for autonomous 3D object modeling by a humanoid robot. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 1159–1164, 2009.

[39] P. Forssen, D. Meger, K. Lai, S. Helmer, J. J. Little, and D. G. Lowe. Informed visual search: combining attention and object recognition. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 935–942, 2008.

[40] R. B. Girshick, P. F. Felzenszwalb, and D. A. McAllester. Object detection with grammar models. In *Advances in Neural Information Processing Systems 24*, pages 442–450, 2011.

[41] S. Gottschalk. *Collision queries using oriented bounding boxes*. PhD thesis, The University of North Carolina at Chapel Hill, 2000.

[42] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 14–19, Nov 1990.

[43] Y. Jiang, H. Koppula, and A. Saxena. Hallucinated humans as the hidden context for labeling 3D scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2993–3000, 2013.

[44] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449, May 1999.

[45] I. Kao, K. Lynch, and J. W. Burdick. Contact modeling and manipulation. In *Handbook of robotics*, chapter 27, pages 647–669. Springer, 2008.

[46] A. Karpathy, S. Miller, and L. Fei-Fei. Object discovery in 3D scenes via shape analysis. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095, 2013.

[47] D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 272–277, May 2008.

[48] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Auton Robot*, 37:369–382, 2014.

[49] C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *Robotics Automation Magazine, IEEE*, 14(1):20–29, March 2007.

[50] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12:1437–1454, 2012.

[51] M. Klingensmith, M. Hermann, and S. Srinivasa. Object modeling and recognition from sparse, noisy data via voxel depth carving. In *International Symposium on Experimental Robotics (ISER)*, 2014.

[52] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno. Protected interactive 3D graphics via remote rendering. *ACM Transactions on Graphics (TOG)*, 23:695–703, 2004.

[53] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *Advances in Neural Information Processing Systems 24*, pages 244–252. Curran Associates, Inc., 2011.

[54] M. R. Korn and C. R. Dyer. 3D multiview object representations for model-based object recognition. *Pattern Recognition*, 20:91–103, 1987.

[55] M. Krainin, B. Curless, and D. Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5031–5037, May 2011.

[56] T. Kriegel, T. Bodenmuller, M. Suppa, and G. Hirzinger. A surface-based next-best-view approach for automated 3D model completion of unknown objects. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 4869–4874, 2011.

[57] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 2000.

[58] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3D scene labelings. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 3050–3057, 2014.

[59] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824, 2011.

[60] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3D scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1330–1337, 2012.

[61] C. Laporte and T. Arbel. Efficient discriminant viewpoint selection for active bayesian recognition. *International Journal of Computer Vision*, 68:267–287, 2006.

[62] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, May 2015.

[63] S. Lee, S. Lee, J. Lee, D. Moon, E. Kim, and J. Seo. Robust recognition and pose estimation of 3D objects based on evidence fusion in a sequence of images. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 3773–3779, 2007.

[64] B. León, A. Morales, and J. Sancho-Bru. *From robot to human grasping simulation*. Springer, 2014.

[65] D. G. Lowe. Local feature view clustering for 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 682–688, 2001.

[66] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[67] L. Ma, M. Ghafarianzadeh, D. Coleman, N. Correll, and G. Sibley. Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1344–1351, 2015.

[68] N. Macias and J. Wen. Vision guided robotic block stacking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 779–784, Sept 2014.

[69] B. S. Manjunath, J. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11:703–715, 2001.

[70] N. Massios and R. Fisher. A best next view selection algorithm incorporating a quality criterion. In *British Machine Vision Conference*, pages 780–789, 1998.

[71] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:417–433, 1993.

[72] R. Mehrotra and J. E. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 28:57–62, 1995.

[73] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct 2005.

[74] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5663–5668, Oct 2006.

[75] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[76] V. Narayanan and M. Likhachev. PERCH: perception via search for multi-object recognition and localization. *Computing Research Repository (CoRR)*, abs/1510.05613, 2015.

[77] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.

[78] J. Pajarinen and V. Kyrki. Robotic manipulation in object composition space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6, Sept 2014.

[79] E. G. M. Petrakis, A. Diplaros, and E. Milios. Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1501–1516, Nov 2002.

[80] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision (IJCV)*, 5:137–160, 1990.

[81] D. Prattichizzo and J. C. Trinkle. Grasping. In *Handbook of robotics*, chapter 28, pages 671–700. Springer, 2008.

[82] K. Pulli. Multiview registration for large data sets. In *1999. Proceedings. Second International Conferences on 3-D Digital Imaging and Modeling*, pages 160–168, 1999.

[83] X. Ren, L. Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2759–2766, 2012.

[84] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Learning of perceptual grouping for object segmentation on RGB-D data. *Journal of Visual Communication and Image Representation*, 25:64–73, 2014.

[85] A. Rosenfeld. Recognizing unexpected objects: A proposed approach. *International Journal of Pattern Recognition and Artificial Intelligence*, 1:71–84, 1987.

[86] S. D. Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: A survey. *Pattern Recognition*, 37:429–446, 2004.

[87] Y. Rui, T. S. Huang, and S. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10:39–62, 1999.

[88] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[89] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[90] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3384–3391, 2008.

[91] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, March 2012.

[92] J. Sanchiz and R. Fisher. A next-best-view algorithm for 3D scene recovery with 5 degrees of freedom. In *British Machine Vision Conference*, pages 163–172, 1999.

[93] G. Sansoni, M. Trebeschi, and F. Docchio. State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, 9:568–601, 2009.

[94] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2):91–110, February 2008.

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

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

[97] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[98] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 601–608, 2011.

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

[100] S. Stein, F. Worgotter, M. Schoeler, J. Papon, and T. Kulvicius. Convexity based object partitioning for robot applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3213–3220, 2014.

[101] T. Stoyanov, R. Mojtahedzadeh, H. Andreasson, and A. J. Lilienthal. Comparative evaluation of range sensor accuracy for indoor mobile robotics and automated logistics applications. *Robotics and Autonomous Systems*, 61:1094–1105, 2013.

[102] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3467–3474, May 2012.

[103] G. Tarbox and S. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61:84–111, 1995.

[104] Z. Teng, H. Mao, and J. Xiao. Automatic object modeling through integrating perception and robotic manipulation. In *to appear in the International Symposium on Experimental Robotics (ISER)*, October 2016.

[105] Z. Teng and J. Xiao. Surface-based 3D object detection and pose estimation. Website: imilab.uncc.edu/research/3d-object-detection-and-pose-estimation. Accessed: 2015-04-20.

[106] Z. Teng and J. Xiao. 3D object detection based on geometrical segmentation. In *2013 International Conference on Computer and Robot Vision (CRV)*, pages 67–74, 2013.

[107] Z. Teng and J. Xiao. Surface-based general 3D object detection and pose estimation. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 5473–5479, 2014.

[108] Z. Teng and J. Xiao. A learning-based approach for evaluating scene recognizability of a view. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4265–4272, 2015.

[109] Z. Teng and J. Xiao. Surface-based detection and 6DoF pose estimation of 3D objects in cluttered scenes. *Accepted by the IEEE Transactions on Robotics (T-RO)*, 2016.

[110] F. Tombari, S. Salti, and L. D. Stefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102:198–220, 2013.

[111] J. Vasquez-Gomez, L. Sucar, R. Murrieta-Cid, and E. Lopez-Damian. Volumetric next-best-view planning for 3D object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11:1–13, 2014.

[112] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518 vol.1, 2001.

[113] G. R. W. R. Scott and J. F. Rivest. View planning for automated 3D object reconstruction inspection. *ACM Computing Surveys (CSUR)*, 35:64–96, 2003.

[114] D. Wilkes and J. K. Tsotsos. Active object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 136–141, 1992.

[115] C. Wu, I. Lenz, and A. Saxena. Hierarchical semantic labeling for task-relevant RGB-D perception. In *Robotics: Science and Systems (RSS)*, 2014.

[116] K. Yan and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–506–II–513 Vol.2, June 2004.

[117] G. Yu and J. M. Morel. ASIFT: An algorithm for fully affine invariant comparison. *Image Processing On Line*, 2011.

[118] X. Zhang, Y. Yang, Z. Han, H. Wang, and C. Gao. Object class detection: A survey. *ACM Computing Surveys (CSUR)*, 46(10):1–53, 2013.

[119] L. Zhu, H. Mao, X. Luo, and J. Xiao. Determining null-space motion to satisfy both task constraints and obstacle avoidance. In *IEEE International Symposium on Assembly and Manufacturing*, Fort Worth, Texas, USA, August 2016.

[120] M. Zhu, K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3D object detection and pose estimation for grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, 2014.

PUBLICATIONS

[1] Z. Teng and J. Xiao. Surface-based detection and 6DoF pose estimation of 3D objects in cluttered scenes. Accepted by the IEEE Transactions on Robotics (T-RO), 2016.

[2] Z. Teng, H. Mao, and J. Xiao. Automatic object modeling through integrating perception and robotic manipulation. To appear in the International Symposium on Experimental Robotics (ISER), October 2016.

[3] Z. Teng and J. Xiao. A learning-based approach for evaluating scene recognizability of a view. In IEEE International Conference on Robotics and Automation (ICRA), pages 4265–4272, 2015.

[4] Z. Teng and J. Xiao. Surface-based general 3D object detection and pose estimation. In IEEE International Conference on Robotics and Automation (ICRA), pages 5473–5479, 2014.

[5] Z. Teng and J. Xiao. 3D object detection based on geometrical segmentation. In 2013 International Conference on Computer and Robot Vision (CRV), pages 67–74, 2013.

[6] J.L. Li, Z. Teng, J. Xiao, A. Kapadia, A. Bartow and I. Walker. Autonomous Continuum Grasping. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4569–4576, Tokyo, Japan, November 2013.

[7] J.L. Li, Z. Teng and J. Xiao. Can a continuum manipulator fetch an object in an unknown cluttered space?. IEEE Robotics and Automation Letters, 2(1):2–9, 2016.