3D SCENE UNDERSTANDING: FROM SEGMENTS TO VOLUMES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Zhaoyin Jia

January 2014

3D SCENE UNDERSTANDING: FROM SEGMENTS TO VOLUMES

Zhaoyin Jia, Ph.D.

Cornell University 2014

Segmentation is one of the fundamental computer vision problems and has been investigated over years. In this thesis, we present algorithms for RGB-D image segmentation, and more importantly, the additional information that can be inferred from segmentations: depth ordering, 3D surfaces, occlusion boundaries and volumes of objects. All these clues lead to a more comprehensive 3D understanding of the scene as well as a higher level RGB-D interpretation. Also in return some of these clues can provide important feedbacks and improve the final scene segmentation performance.

We start by performing 3D depth interpretation from 2D color images only. We discover that the segment shapes enable us to learn the depth orderings of the objects. Specifically, from the initial segmentation we develop features to encode the information captured in boundaries and junctions. After a supervised learning procedure, our algorithm is able to produce a 3D depth ordering map from a single 2D color image.

Secondly, we proceed to 3D scene understanding using RGB-D images. The recent development of the depth sensors improves the performance of the traditional computer vision algorithms by a margin. Therefore, besides using one single image, we incorporate depth information along with it, and parse the scene based on 3D interpretation. We aim at the applications such as 3D point interpolation, boundary detection and scene segmentation. In detail, we propose algorithm for 3D surface segmentation, and show that combining this 3D

surface information with 2D color image achieves better performance for 3D interpolation. After that, we use both 2D color and 3D depth channels to find the occlusion and connected boundaries given a RGB-D scene. This serves as an extended 3D scene interpretation with a better understanding of occlusions between objects.

Finally we perform a 3D volumetric reasoning of the RGB-D image with support and stability. Objects occupy physical space and obey physical laws. To truly understand a scene, we must reason about the space that objects in it occupy, and how each objects is supported stably by each other. In other words, we seek to understand which objects would, if moved, cause other objects to fall. This 3D volumetric reasoning is important for many scene understanding tasks, ranging from segmentation of objects to perception of a rich 3D, physically well-founded, interpretations of the scene. In this thesis, we propose a new algorithm to parse RGB-D images with 3D block units while jointly reasoning about the segments, volumes, supporting relationships and object stability. Our algorithm is based on the intuition that a good 3D representation of the scene is one that fits the depth data well, and is a stable, self-supporting arrangement of objects (i.e., one that does not topple). We design an energy function for representing the quality of the block representation based on these properties. Our algorithm fits 3D blocks to the depth values corresponding to image segments, and iteratively optimizes the energy function. Our proposed algorithm is the first to consider stability of objects in complex arrangements for reasoning about the underlying structure of the scene. Experimental results show that our stability-reasoning framework improves RGB-D segmentation and scene volumetric representation.

THESIS COMMITTEE

Prof. Tsuhan Chen

School of Electrical and Computer Engineering,

Cornell University

Dr. Yao-Jen Chang

Siemens Corporation, Corporate Technology,

Princeton, NJ

Prof. Anthony P. Reeves

School of Electrical and Computer Engineering,

Cornell University

Prof. Ashutosh Saxena

Department of Computer Science,

Cornell University

Prof. Noah Snavely

Department of Computer Science,

Cornell University

Dedicated to Jing.

ACKNOWLEDGEMENTS

First and foremost, I thank my advisor, Prof. Tsuhan Chen, for his guidance over the past five years during my Ph.D. I am thankful for the abilities in academic research he taught me, but more importantly, the very little details and influences he provided every day: he showed me how to initiate a research idea, how to think critically over the literature, and how to present the story to the audience. He has always been very patient and supportive throughout my PhD, as well as provided me the freedom to explore. He provided me a perfect blend of big research pictures, small smart topics and practical guidance. I am extremely fortunate to have the opportunity to work with him, and proud of the accomplishments we have achieved.

I am also grateful to Dr. Andrew Gallagher, who I have closely worked with over years since 2007, when I was still an exchange undergraduate student in Carnegie Mellon University. He was the inspiration for directions, the support for difficulties, and a sincere friend in daily life. I am thankful for having the chance to work with him.

I thank Prof. Ashutosh Saxena, and Dr. Yao-Jen Chang, who taught and guided me in proceeding research projects. They have provided key insights into my research, and valuable suggestions that accomplish the work. Besides, they give me crucial supports in my PhD progress in every detail from coding to writing. I feel grateful to have the chance to collaborate with them.

I also thank my other committee members, Prof. Noah Snavely and Prof. Anthony Reeves, for giving me advises and suggestions for my research all over the years.

I owe very much to my other collaborators, friends, and all the AMP (Chen Lab) lab members in Cornell University, including Adarsh Kowdle, Amandi-

aneze Nwana, Amir Sadovnik, Congcong Li, Henry Shu, Kuan-chuan Peng, Ruogu Fang, Yimeng Zhang, and other friends. I enjoy the free and friendly environment in our Ward Lab, first floor. I will always miss the fun we have during every gathering, and the intense time before every deadline.

I thank my parents, Xiaoxia Zhao and Xiangdong Jia, for their many years of support, understanding and love.

Finally, I thank my wife, Jing Xia. She completes me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Figure 1.1: Exemplar input for image segmentation.

Image segmentation is one of the traditional computer vision problems, and many algorithms have been proposed for it. The input is usually one image, or a color and depth image pair, and the output is a set of pixel groups, where each group corresponds to one object in the scene. Segmentation on 2D color image involves computing color, texture or depth clue of each segment, and in some scenarios even combining algorithms such as object recognition and detection. Usually segmentation serves as the initial steps for a higher level scene understanding, and thus it is by all means an important problem and a core computer vision task to be solved. [2] [3] [4] [5] [6] [7] [8].

Many different criteria can be applied for image segmentation. One example is shown in Fig. 1.1. For the box placed in the center with "3D Vision" text, even for human beings, people will most likely have different proposals to segment

this particular object: one may label the frontal green surface and the top black surface as separate segments, because they have very different colors and surface normals. However, the others may group them as one segment, since these two surfaces are combined as a box, which supports the camera and the book in the image. Previous literatures also discussed that the image segmentation is generally an ill-posed problem, and many algorithms have been applied to meet the needs from different situations [3] [9].

In this thesis, in addition to produce an image segmentation, we also aim to investigate the information generated from the segmentation. Traditional segmentation algorithm relies on the continuity in color, texture and depth domains. However, we extend these concepts with more semantic meanings. Specifically, we ask questions to get a more comprehensive 3D understanding, e.g., what are the depth information generated from the segmentation? Can we infer occlusions between segments? What are volumes and support relations between segments?

We propose algorithms to learn this information from boundaries and segments that generated by the segmentation algorithms (sometimes with the help of the depth image). Further, these extra clues can also be used to improve the segmentation results. For instance, once we reason the occlusion between objects and find the occlusion boundaries, enforcing the continuity in these boundaries will lead to better segments, and thus we can encourage loops in the boundary map to enclose a full object [10]; surface segmentation in 3D space can help 2D color segmentation performance [11] [12]; the support and stability of each individual segment also indicates the quality of the overall segmentation [13]. In the later chapters, we present our proposed algorithms for these tasks

in detail, and use some of the features as feedback for RGB-D segmentation.

1.1 Depth Ordering

First we examine the information in a single color image, and find the depth ordering from its segmentation. For example, given the image shown in Fig. 1.1, we as human have no difficulty in telling the depth ordering of each object, and the occlusion relations between them: we can identify that all the books occlude the wall in the behind, and the objects are all supported from bottom by the ground.

Human can achieve this 3D depth ordering understanding even without a semantic object representation, e.g. with pure abstract objects. One example is presented in Fig. 1.2. In this case, all the objects are presented as abstract shapes in rectangles or circles. However even in this situation, it is still quite easy for a human to tell the depth orderings, such as that segment B is on top of segment C, and all the segments are on top of the segment E, etc.

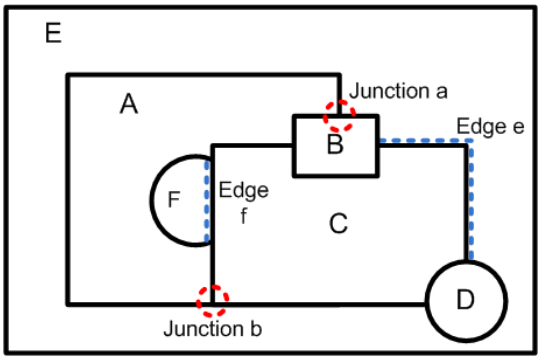


Figure 1.2: Human is able to understand the depth ordering of abstract shapes without any semantic meanings

This observation inspires us to design the features that can encode the depth

relation between the objects. We consider this serves as an extra 3D understanding of the scene and moves one step beyond segmentation. In the previous example with abstract objects, all the texture and semantic information becomes invalid, and the only clues left are the junctions and the boundaries between the segments. They are critical information for inferring the depth ordering in one image: for example, given a T-junction a in Fig. 1.2, it is more likely the parallel edges indicate the object in the front, i.e. segment B is in front of segment A and E.

However, simply applying this rule-based reasoning will fail in many cases. One counter example is presented at junction b. Applying the same junction rule to T-junction b gives an incorrect prediction: the segment E is beside the parallel edges, but appears to be the behind segment of A and C.

The similar situation also holds for the boundaries too. Usually the concaveness of a boundary indicates the depth ordering between the two segments: for example, edge e intrudes segment E from segment C, therefore it is more likely that segment C is on top of segment E. However this rule fails at edge f, where the boundary is a straight line. Thus more complicate procedure needs to be applied to find the correct depth ordering of segment C and F.

When dealing with real-world images, shown in Fig. 1.3 on the left, the scenario becomes more complicated. Given the segmentation, along with the boundaries (shown in red edges in Fig. 1.3, left) and the junctions (shown in blue circles in Fig. 1.3, left), we aim to identify the depth ordering of each object segment, shown in the right of Fig. 1.3. The boundary and junction features heavily rely on the quality of segmentation, therefore a reliable segmentation needs to be identified at the initial step in order to retrieve a more reasonable depth

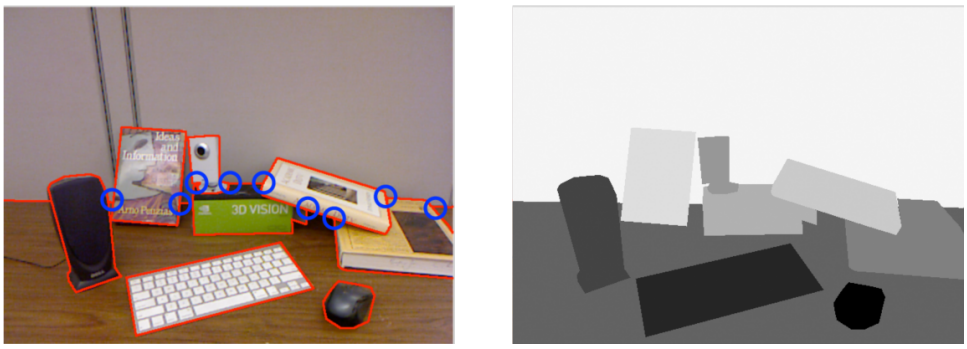


Figure 1.3: Depth ordering from a single image: from the boundary and junction features of the color image on the left, we want to infer the depth ordering of each segment on the right, which is color coded in a way that the more bright the segment is, the further way it is in depth.

ordering result. Besides, the images capturing the objects in the real world provides more complex junctions and boundaries, therefore it is almost impossible to apply simple rules to infer the depth ordering.

In this thesis, we propose a learning based approach: we extract features from junctions and boundaries, and supervisely learn a classification model to predict the depth ordering of each segment. As long as the features are descriptive and training data are enough, this learning based process can deal with the variant situations of irregular shapes in the boundaries and junctions, which are often produced from the automatic segmentation of real world images.

Further, two additional aspects are also studied in this thesis: a) the potential loop in the depth ordering that are infeasible in real-world configuration: the loop in the depth ordering map is a special topic arise in this problem, and we propose a Markov Random Field based algorithm to encourage the solutions with no cycles. b) We improve of the segmentation quality by enforcing the closure of the boundaries. Segmentation results largely affect the depth ordering

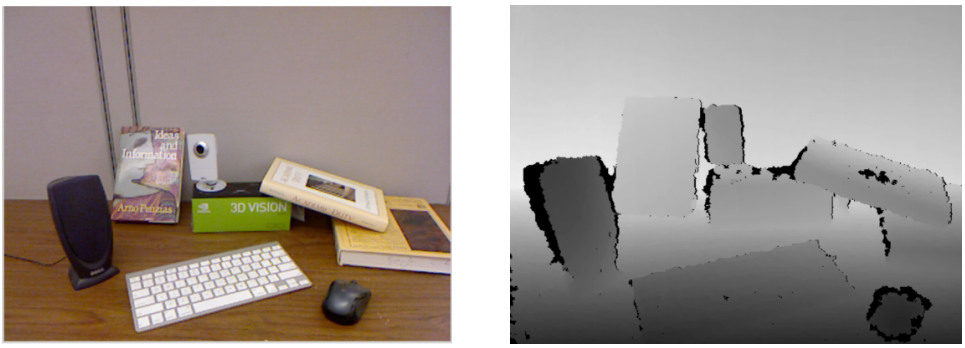


Figure 1.4: A exemplar pair of RGB-D images: the color image on the left and its corresponding depth information on the right.

performance. Therefore we also improve this step by incorporating the continuity and closeness of the boundaries. In the later chapters, we show that the features proposed for depth ordering can be also applied for other applications, e.g. finding the occlusion and connected boundaries of RGB-D images.

1.2 RGB-D Segmentation

With the recent development of the depth sensors, e.g. laser scanners or infrared depth sensors, computer vision algorithms are becoming to take advantage of this additional information. Many vision tasks, such as human pose estimation, object placement, 3D scene modeling and object recognition, have been benefited by using both color and depth channels [14] [15] [16] [17] [18] and the experiments show that incorporating this extra depth information usually helps these vision tasks improve a considerate amount.

In this thesis, we use the depth sensor as an another channel to perform the 3D scene understanding from the segmentation. One exemplar pair of images are presented in Fig. 1.4. In many ares the color channel has a large variance,

Figure 1.5: We analyze the depth data by finding the surfaces (shown on the left in different pixel colors), and occlusion (in green edges) and connected (in red edges) boundaries in the RGB-D image.

e.g. the texture of the books and the ground plane, but the depth channel gives clean and continuous pixel regions and thus is very useful for segmentation as well as 3D scene understanding.

However, directly using the depth information as another channel and applied the same algorithms in the color domain will not work very well. The depth image has its internal and different meaning other than a single gray scale image: it captures the 3D structure of the scene, usually from a single view. Therefore we propose features and semantically parse the RGB-D images with surfaces as well as occlusion/connected boundaries, presented in Fig.1.5.

Surfaces contribute one important element in analyzing depth data. Neighboring pixels and 3D points may belong to two different but close objects. The surface segmentation enable us to identify neighboring pixels and 3D points as different objects. In detail, we estimate the normal of each pixel and 3D point, and if there is abrupt change in this normal space, these points more likely belong to different objects. If points have similar normal directions, we group them into the same object region.

For example, in Fig. 1.5, the book covers have quite complex textures. Using the color channel only, any reasonable segmentation algorithm will separate these textures into many smaller segments, leading to an over-segmentation. However, from the depth image we observe that all the 3D points on the book share similar normal vectors, and they lies on one smooth surface. This provides us a very important clue for segmentation and RGB-D image understanding. On the left of Fig. 1.5, we present our surface segmentation result of this image: it shows that by using the depth only, we can reach reasonable segmentation and group the 3D points on the same surface into one segment.

In this thesis, we propose a surface segmentation algorithm based on the efficient graph cut [19] [11] [20]. Furthermore, we separate different types of 3D points, and only investigate the ones that affect the quality of the surface segmentation: for example, the normal vectors of the 3D points lie between the two surfaces are hard to estimate, and these points become noise when performing the surface segmentation. We propose a heuristic algorithm to filter out this 3D points, and reconstruct the plane and quadratic surface with only the reliable ones.

To combine the 3D surface segmentation with the 2D color information, we first propose an algorithm that merges two segmentation proposals through a Markov Random Field. This combination propagates the 3D surface labels to all the pixels in the color image, which usually has a higher resolution than the depth image. We apply this algorithm to some applications such as 3D interpolation, and experiments show that reasoning the 3D scene through the surfaces enables us to achieve better interpolation performance.

Furthermore, we extend the 3D scene understanding on RGB-D images into

occlusion and connected boundaries. Traditional object segmentation algorithms only provide the segment of each object as output. However, there are rich information between the objects that resides in the boundaries. One example is shown in Fig. 1.5 on the right. We group the object boundaries into two different categories: a) the occlusion boundary, which indicates that the two segments it lies in between has a drastic change in depth, and one object occludes the other in 3D space; b) the connected boundary, which indicates that two objects are connected with each other, and most likely one supports another. Occlusion and connected boundary inference is one important part for 3D scene understanding, and the building blocks for higher level object reasonings, e.g. object support and stability.

Although provided depth information, it is not trivial to estimate the occlusion and connected boundaries from RGB-D images. One major challenge is that the depth image does not usually has the same quality as its corresponding color image: it usually has a lower resolution, and becomes very shaky and noisy in the boundary area, where our task is focused. One example is shown in the Fig. 1.4, on the right. Therefore simply thresholding the depth image will produce poor result in finding boundaries, and thus not preferable.

We rely on a learning based approach to find the occlusion and connected boundaries. We incorporate features in the literature on color space, [12], and propose new features based on our surface segmentation. In addition, we consider the depth ordering as another hint for finding the segmentation as well as the occlusion and connected boundaries, and thus incorporate the features describing the junctions and boundaries into this learning framework. The overall algorithm is formed as a Conditional Random Field. The experiments show that

Figure 1.6: Incorrect segmentation leads to unstable boxes of the scene, e.g. the red ones.

our proposed algorithm reliably detects the occlusion and connected boundaries in different testing scenarios, and provides a better 3D understanding of the scene.

1.3 Block, Support and Stability

Depth ordering, surfaces, occlusion and connected boundaries are all very informative tools for 3D scene understanding, but there are still unsolved problems. For example, in the example shown in Fig. 1.1, in the middle there is a box with text " 3D vision" on it, which is composed of two surfaces with one green and one black in color. Also the two surfaces of the boxes are completely different with perpendicular normal vector directions, and therefore for any segmentation algorithm, either from color channel or the depth information, this box should be separated into two objects for their distinct features.

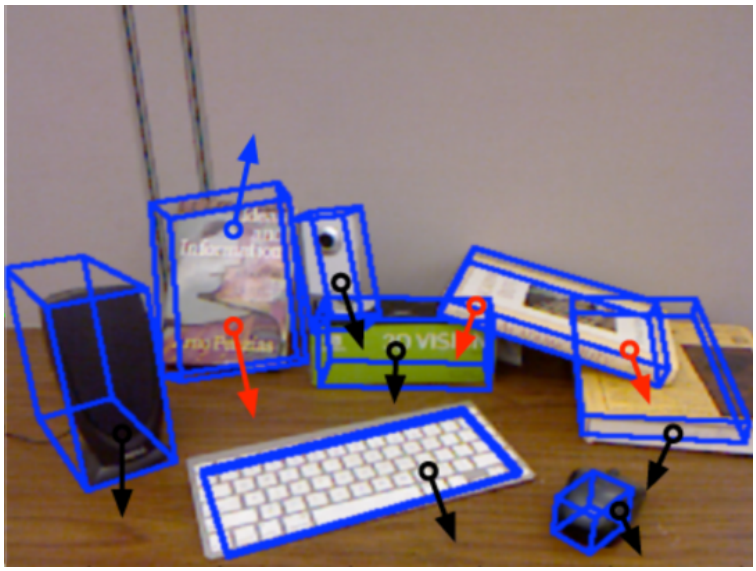


Figure 1.7: A volumetric representation of the object segments, and the support relations between the objects

However, we human can still reason that this is one box and the two surfaces of them are hinged together. We are interested in why this happens. One clue would be, if the surface on the top is a separate object, the whole scene would not be stable, and many objects would topple.

The exemplar segmentation is shown in Fig. 1.6, which describes the stability of the 3D scene. We represent each segment as a 3D oriented bounding box. Given the segmentation, many 3D boxes would not be stable, and can not support the objects on top. These boxes are presented in red in Fig. 1.6. Therefore, we consider this stability information of each segment an extra and important clue for 3D scene understanding.

We propose to reason the 3D scene through 3D oriented blocks. Based on this block representation, we find the support relation between the objects, and infer the stability of each block. One example is shown in Fig. 1.7.

The RGB-D image enables us to estimate the block representation of the segments using the depth channel. The block world provides us important volumetric information: we can estimate the space each object possesses, and where are the free spaces in the 3D scene.

However, estimating the volume of each object is not trivial: the depth images have lower resolutions and are usually noisy. More importantly, it is only a single shot of the scene from one view, and therefore the objects are usually only partially observed. Thus we propose a novel 3D block fitting algorithm to overcome this limitation in the depth channel, and it leads to a better volumetric representation of the object orientation in 3D space.

Given this volumetric representation in blocks, we estimate the support relations and the stabilities of objects. Further, we incorporate the block properties, support relations and stability of the scene into a learning framework, and use these additional features as feedback to improve the RGB-D segmentation.

Finally, the potential segmentation candidates of one RGB-D image images are huge in number. It is intractable to explicitly explore the whole segmentation space and reason through the block, support and stability of each segmentation configuration. Therefore we propose a sampling algorithm to reach a more reasonable segmentation by combining the clues from the volumetric representation. We supervisely learn a potential function incorporating the volumetric features on both individual box and pairwise boxes, and minimize this energy function through a random sampling process. Experiment results show that our proposed learning algorithm and sampling methods improve the RGB-D segmentation, and achieve a better 3D scene understanding.

1.4 Organization of this Thesis

The rest of this thesis is organized as follows: in Chapter 2, we first describe our proposed 2D features on boundaries and junctions in color image, and propose a novel algorithm for inferring the depth on it. Then in Chapter 3, we introduce our 3D surface segmentation algorithm, and the 3D interpolation application based on it. Chapter 4 describes our algorithm for detecting occlusion and connected boundaries in RGB-D images. Chapter 5 presents our final combined algorithm on RGB-D reasoning with volumetric blocks, support and stability features based on them. Finally, we conclude this thesis in Chapter 6.

1.5 First Published Appearances of Described Contributions

Most of the contributions presented in this thesis have appeared as publications as follows:

- Chapter 2: Z. Jia, A. Gallagher, Y. Chang and T. Chen [10].
- Chapter 3: Z. Jia, Y. Chang, T. Lin and T. Chen [20], and Z. Jia, Y. Chang, T. Lin and T. Chen [11].
- Chapter 4: Z. Jia, A. Gallagher and T. Chen [12].
- Chapter 5: Z. Jia, A. Gallagher, A. Saxena and T. Chen [13], and Z. Jia, A. Gallagher, A. Saxena and T. Chen [21].

Other contributions are not discussed in this thesis because of the scope, including the following publications: Z. Jia, A. Gallagher and T. Chen [22], Z. Jia, A. Saxena and T. Chen [23], Z. Jia, A. Saxena and T. Chen [24], Y. Zhang, Z. Jia and T. Chen [25], Z. Jia, Y. Chang and T. Chen [26], and Z. Jia and Y. Chang and T. Chen [27].

CHAPTER 2

2D FEATURES: DEPTH ORDERING

2.1 Overview

Depth estimation is instrumental for a variety of vision tasks, such as segmentation [5] [28], object recognition [7] [29], and scene understanding [8] [30] [1]. For some purposes, instead of estimating the exact depth value, it may suffice to derive the relative depth ordering of the objects in an image. Humans are adept at this task: in Fig. 2.1 (a), we may not exactly know how far these objects are, but we can understand the depth ordering of the objects: the mouse is on the top, then the book, and the laptop is deeper in the pile, supported by the table. The depth ordering not only gives us a coarse interpretation of the 3D geometry of the objects, but also enables us to interact further with the scene, e.g. we need to remove the mouse and the stapler in order to manipulate the book.

Humans have no trouble inferring the depth order even when the image is extremely abstract with only line drawings [31], such as Fig. 2.1 (b). We still understand that segment B is in front of segment A and C, segment D is in front of segment C, C is in front of F and so on. If we use "$\rightarrow$" to indicate the "in front of" relation, then we have $D \rightarrow C$; $B \rightarrow C \rightarrow F \rightarrow A \rightarrow E$. Early works from Barrow et al. [32] and Waltz et al. [33] present rule-based algorithms to understand 3D geometry in abstract images.

These examples inspire us to investigate the features that determine how we perceive the image depth ordering. Line drawings take out all the color, texture, and semantic high-level interpretation of the image. Clearly in this situation,

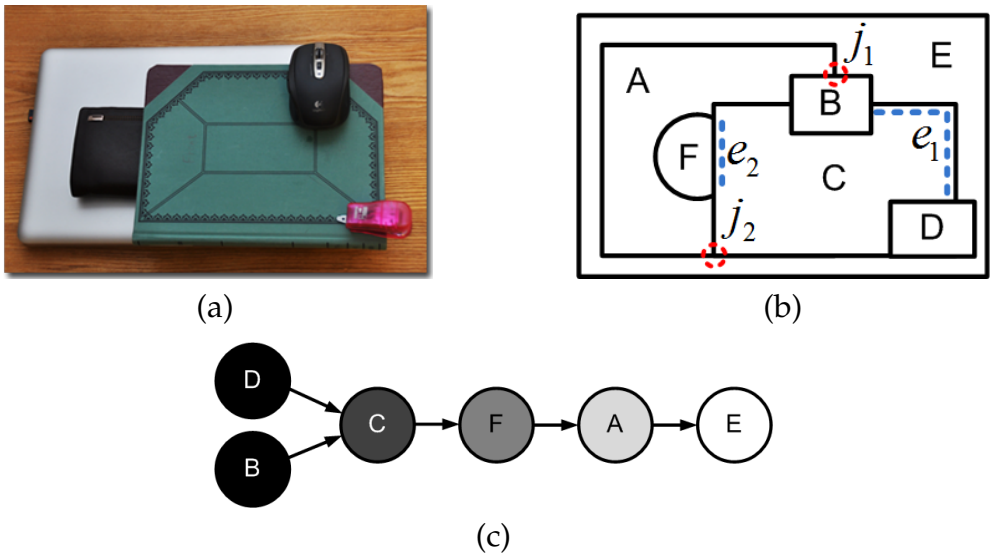
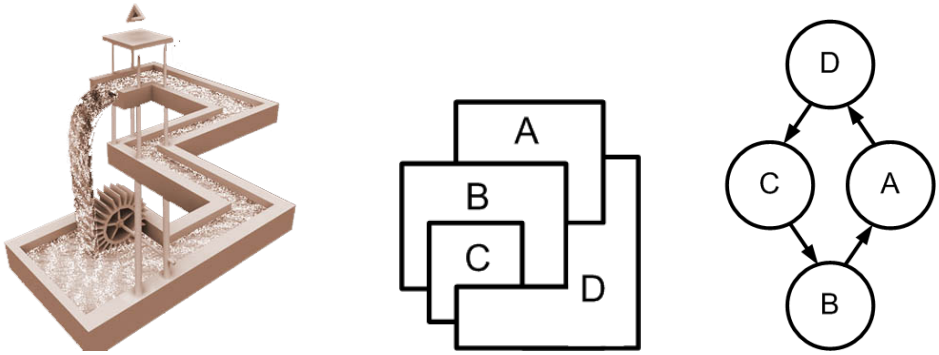


Figure 2.1: (a) Given one image, humans can infer the depth ordering of each object, and even with (b) very abstract line-drawing segments. Motivated by how humans reason about the depth ordering from junctions and boundaries, we develop an algorithm to do that. Our algorithm produces the depth ordering that represented in the form of a graph as in (c), where each node corresponds to one segment, and the directed edge means one segment is in front of another. The depth is colored in a way that the closer an object is, the darker it appears.

only two types of information are available, i.e., boundaries and junctions, such as e_1, e_2, j_1, j_2 in Fig. 2.1 (b). However, depth ordering based on this information is not easily captured by hand-crafted rules, particularly in complex scenarios. Therefore, we adopt a data-driven approach to handle its complexity. We design new features on boundaries and junctions, and use them as the basis to learn depth ordering.

Inferring the depth order from junction or boundary individually has some natural flaws, however. For example in Fig. 2.1 (b), junction j_1 and j_2 have the same T-shape, but imply inverse depth orders. Boundary e_2 is a straight line and provides little information by itself. Therefore, we must combine these different



(a) (b) (c)

Figure 2.2: (a) The Escher Waterfall shows that local reasoning cannot ensure the global consistency. (b) The same is true for depth ordering: although we can determine the pairwise relation between any two segments, it is difficult to decide the global depth order, and the corresponding depth order graph (c) forms a loop.

features to form a better feature set.

Furthermore, having inferred local depth orders from the combined feature sets, we need to ensure the global consistency across the segments. Simply aggregating the local decisions can lead to an invalid understanding of the scene, as the famous Escher Waterfall in Fig. 2.2 (a) vividly illustrates for height perception. This point carries over to depth ordering, and Fig. 2.2 (b) gives one similar example: locally, we can easily determine the relative depth order between any two segments, such as $D \rightarrow C$, $C \rightarrow B$, $B \rightarrow A$ and $A \rightarrow D$. However, when aggregated, it is not a valid depth ordering, i.e. it forms a depth order graph with a loop, as shown in Fig. 2.2 (c). Therefore, to ensure global consistency in the depth ordering, we propose a Markov Random Field based algorithm to infer a likely depth ordering and penalize an invalid ordering of segments. With this algorithm, global consistency is encouraged through message passing, which in turn enables better performance.

In addition, a reliable segmentation is an essential preparation for depth ordering. For natural images, we follow [34] to detect occlusion boundaries and generate object segments. We discover that, in many scenarios, the occlusion boundaries are not only locally continuous, but also form a closed loop to enclose the object. At the same time, the edges connected to and inside of this loop are less likely to be actual occlusion boundaries. Enforcing this constraint, which is a more global enforcement than local continuity, leads to a better object segmentation for depth ordering.

We collected a new depth order dataset with over a thousand images displaying different arrangements of various objects. Each image is manually segmented and includes depth information from Kinect. We tested different algorithms on this and two other datasets: one synthetic dataset and one with natural images [34]. Experiments proved the effectiveness of our proposed new features, and show that our proposed algorithm reliably outperforms the baselines.

To summarize, our major contributions are:

1. New features (on junctions and boundaries) and a learning-based framework for the depth ordering task.
2. A novel approach to globally encourage the depth order consistency through a graphical model.
3. A new depth ordering dataset including more than 1000 images with human segmentation and depth information.
4. A new approach that favors closed loops for occlusion boundary detection.

2.2 Related work

Reasoning about the 3D structure from a single image has been studied since Barrow et al. [32] and Waltz et al.[33]. They present the work of understanding line drawings and converting them into 2.5D images. These works show the first attempts to solve the depth reasoning with rule-based algorithms, and demonstrate the ability of 3D understanding from low level features in abstract images. [35] learns the depth information by T junction from video. These works demonstrate the ability of 3D understanding from the low level features in both real and abstract image.

Our work assumes that the scene is composed of objects in distinct depth order, and is closely related to the works from Dimiccoli et al. [36] and Palou et al. [37], which infer the depth ordering from an elaborate set of rules on T-junctions. Our work differs and improves upon previous works in the following aspects: a) in past works, the rules of inferences are designed without any learning process. They work in certain settings, but may not adapt to new environments. On the contrary, our approach is a learning-based framework and is data-driven. b) Their algorithms focus only on the angles in T-junctions, while we show that combining boundary features with junctions is necessary and achieves better results. c) When aggregating local decisions to produce a global ordering, these works handle contradictions by dropping orders with the lowest predicted beliefs. We formulate this task as a graph inference problem, which achieves global consistency more accurately with the help of graphical model optimization.

Depth ordering is related to the boundary ownership or the figure and

ground assignment problem [34] [38] [39] [40]. However we believe that these tasks are non-trivially different and produce different results. Figure and ground assignment is usually based on each edge as presented by Ren et al. [39], while depth ordering is based on segments. As a result, their work places more focus on features from edges, while we use a complementary feature set of junctions and boundaries. Depth ordering also introduces new problems, such as global consistency in depth, that may not exist for the figure/ground assignment problem. In addition, depth ordering requires reliable segmentation, and we propose a new approach for occlusion boundary detection in order to generate object segments for depth ordering.

Another approach is to infer depth based on high-level understanding of the scene, as in Hoiem et al. [34] and Liu et al. [30]. They parse an image into different semantic labels, such as "ground", "sky", etc., upon which they infer the depth mainly based on the connecting edge between the object and the ground plane. In their works, usually there is no need for encouraging the global consistency. The semantic labels can largely solve this problem, e.g., "ground" always supports "vertical surfaces", and these are placed before "sky". However, these geometric contexts may not always be applicable, such as shown in Fig. 2.1. In particular, these algorithms excel in natural scenes but fall short with micro objects or plan views, or may have difficulty in estimating the depth when "ground" falls outside of the image. Our algorithm complements this shortage well and aims to achieve reliable depth ordering from low-level features without specific context.

When this geometric context labels becomes unavailable, then the problem of contradictory in the depth ordering will appear, and we enforce the global

consistency through our graph-based approach.

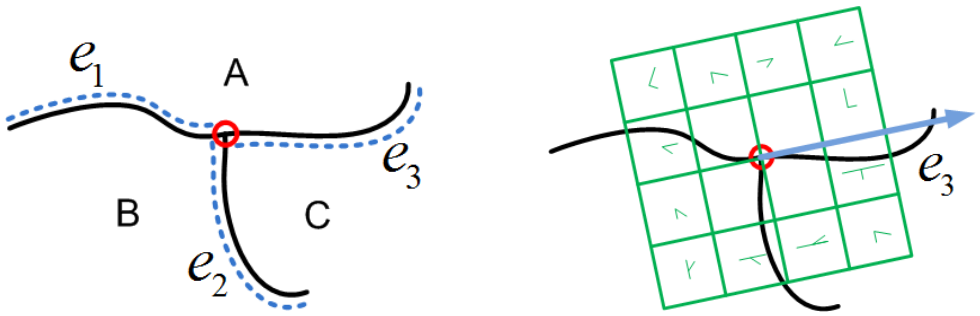
Saxena et al. [1] propose a regression for depth based on super-pixel features, and produce a continuous depth estimation. In contrast, our problem is based on occluded segments. The tasks and the approaches are significantly different. We believe we are able to achieve more meaningful depth relation between objects from reasoning about segment occlusions.

2.3 Local depth ordering

We first detect the occlusion boundaries in one image, and based on them we transform this image into segments. Then we compute features for depth ordering, build the depth order graph and assign a discrete depth value to each segment. We mainly rely on two sets of features for depth ordering: features on the T-junction (**pJF**) and on the boundary (**pBF**).

2.3.1 Junction feature

A T-junction is where three boundaries and three segments meet, illustrated in Fig. 2.3 (a), and we aim to identify which segment is in front of the other two. Note that classifying which segment is in front is identical to classifying which one out of the three boundaries is occluded by the foreground segment, because the segments that are attached to this "behind boundary" are also behind (see Fig. 2.3 (a)). We will first classify this behind boundary, and then convert the result to the segment depth ordering.



(a) (b)

Figure 2.3: (a) One T-junction includes three segments (A, B, C) and three boundaries $(e_1, e_2, e_3$, in dashed blue line). One segment is in front of the other two (A is in front of B and C), and correspondingly, one edge is behind the other two (e_2 is behind e_1 and e_3). (b) A vector $\vec{v}(e_3)$ pointing outwards is fit to the boundary e_3. Then an oriented-SIFT descriptor is computed in align with $\vec{v}(e_3)$.

Angle: An ideal T-junction will include one 180° angle between two boundaries, indicating the segment within is in front, and two 90° angles, indicating the segments are behind. We include these angles as our features. First, for each boundary e inside a junction, we fit a boundary vector $\vec{v}(e)$ to calculate its direction, shown in Fig. 2.3 (b), and calculate the angles from $\vec{v}(e)$ to the other two boundary vectors: $\theta_1, \theta_2 \in [0, \pi]$. We record them as a two-dimension feature $f_a(e)$ for boundary e within in this junction.

Texture: Junctions have different appearances in natural images, and thus using angles alone can be unreliable, so we also capture the texture information of a junction using an oriented SIFT descriptor [41]. SIFT descriptors can record the edge distributions within a junction, while tolerating some appearance variation by using histograms. The SIFT descriptor is centered at the junction, and aligned with every boundary vector $\vec{v}(e)$ pointing outwards, as shown in Fig. 2.3 (b). The size of the descriptor is determined with respect to the boundary length and limited to 40 pixels.

In order to learn the intrinsic appearance of a junction, we use two types of images for this feature: the original image $f_o(e)$ and the binary edge image $f_b(e)$. The binary edge image is a blank image with only the occlusion boundaries labeled in white. While $f_o(e)$ can capture a junction's appearance in the natural image, $f_b(e)$ excludes all the luminance and texture information from the environment, focusing on the boundary distribution within a junction.

We concatenate the above three sets of features as the final junction feature set: $f_j(e) = [f_a(e), f_o(e), f_b(e)]$. Within one junction, the boundaries in front are labeled as $y = 1$ and the boundary behind is labeled as $y = -1$. Then a SVM classifier h_j is trained. During testing, as there is one and only one behind boundary in a valid junction, we enforce this constraint by choosing the behind boundary as the one with the smallest predicted cost.

2.3.2 Boundary feature

In addition to junctions, boundaries are also important for depth ordering. Hoiem et al. [34] proposes local features $f_d(e)$ to encode many edge attributes, and we include them as a subset of our boundary features [1].

Additionally, we consider the boundary convexity an informative clue. Take Fig. 2.4 (a) as one example, the convexity of boundary e implies that segment A occludes segment B, and thus determines the depth ordering.

Therefore, we design features to explicitly capture the boundary convexity. First, we connect the starting point p_s and the ending point p_e of a boundary,

[1] To follow the convention in this thesis, we exclude the high-level geometric context features, which are not applicable for the settings.

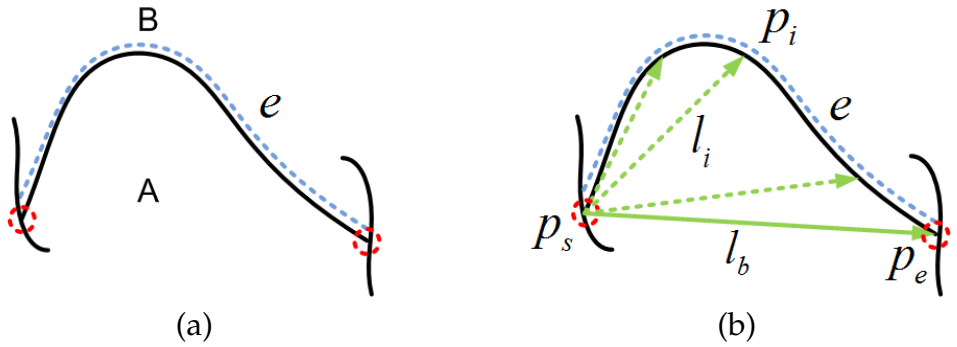


Figure 2.4: The boundary convexity feature: (a) one occlusion boundary (e.g., e) lies in between two segments (e.g., A, B). Boundary e bends towards segment B, indicating that more likely A is in front of B. (b) A base vector l_b can be set by connecting the two ends p_s and p_e. For each point p_i on the boundary, we link p_s and p_i to create a new vector l_i, and record the angle between l_b and l_i. We histogram these angles as new features for e.

and form the base vector l_b. The distribution of each point p_i on the boundary with respect to l_b provides the convexity information. We connect every point p_i along the boundary to p_s, and form a new vector l_i. We record the angle between l_i and l_b: $\theta_i = \arccos(l_i \cdot l_{base}) \in [-\pi, \pi]$, as shown in Fig. 2.4 (b). After getting $\{\theta_i\}$ for all $\{p_i\}$, we quantize $[-\pi, \pi]$ into 36 bins and histogram $\{\theta_i\}$, and append this histogram as the new feature $f_c(e)$ in addition to $f_d(e)$: $f_b = [f_d, f_c]$. Since now the boundary is directed from p_s to p_e, for training we label the boundary $y = 1$ if its left segment is in front of its right segment, and $y = -1$ otherwise. Following the same rule, we retrieve the depth ordering of segments during testing.

2.3.3 Combined features

Junction and boundary features alone have their own strengths and weaknesses, and we combine them together to complement each other. Since the features in each junction $f_j(e)$ are already computed on the basis of the boundary within it,

we can append $f_b(e)$ to $f_j(e)$ to form the combined feature $f_c(e) = [f_j(e), f_b(e)]$.

Accordingly, the learning process on the junction now becomes a ranking problem on the three boundaries/segments. We use a structured SVM [42] $h_c(f_c(e))$ to solve it. For example, in Fig. 2.3, we can first associate each boundary with the segment on its left. Suppose the ground truth depth order is $A \rightarrow C \rightarrow B$. Then for boundaries: $e_3 \rightarrow e_2 \rightarrow e_1$. During training, the constrains become $h_c(A) > h_c(C)$ and $h_c(C) > h_c(B)$, i.e. $h_c(e_3) > h_c(e_2)$ and $h_c(e_2) > h_c(e_1)$. (We omit f_c for brevity, and in the following we use segment instead of boundary to indicate the depth order, since they are identical.) During testing, $x_i = \{f_c(A), f_c(B), f_c(C)\}$ is the combined feature on junction i , and y_i^{ABC} indicates the segment order $A \rightarrow B \rightarrow C$. We define the likelihood *li* of assigning the depth order y_i^{ABC} from the SVM margin:

$$li(y_i^{ABC}|x_i) = \sum_{(M,N)} h_c(f_c(M)) - h_c(f_c(N)), \tag{2.1}$$

where $(M, N) \in \{(A, B), (B, C), (A, C)\}$.

2.4 Towards global depth reasoning

D-order graph: After the local inference for depth ordering, a depth order graph is built (**d-order** graph), shown in Fig. 2.5 (b), and we assign the depth order for each segment according to this graph. One node in the d-order graph represents one segment in the image. The directed edge indicates one segment is in front of another. With the combined feature f_c, each junction will order its three segments in depth. For example, junction α in Fig. 2.5 (a) may infer the depth ordering $A \rightarrow B \rightarrow D$, and produce three directed edges in the d-order graph: $A \rightarrow B$, $A \rightarrow D$ and $B \rightarrow D$.

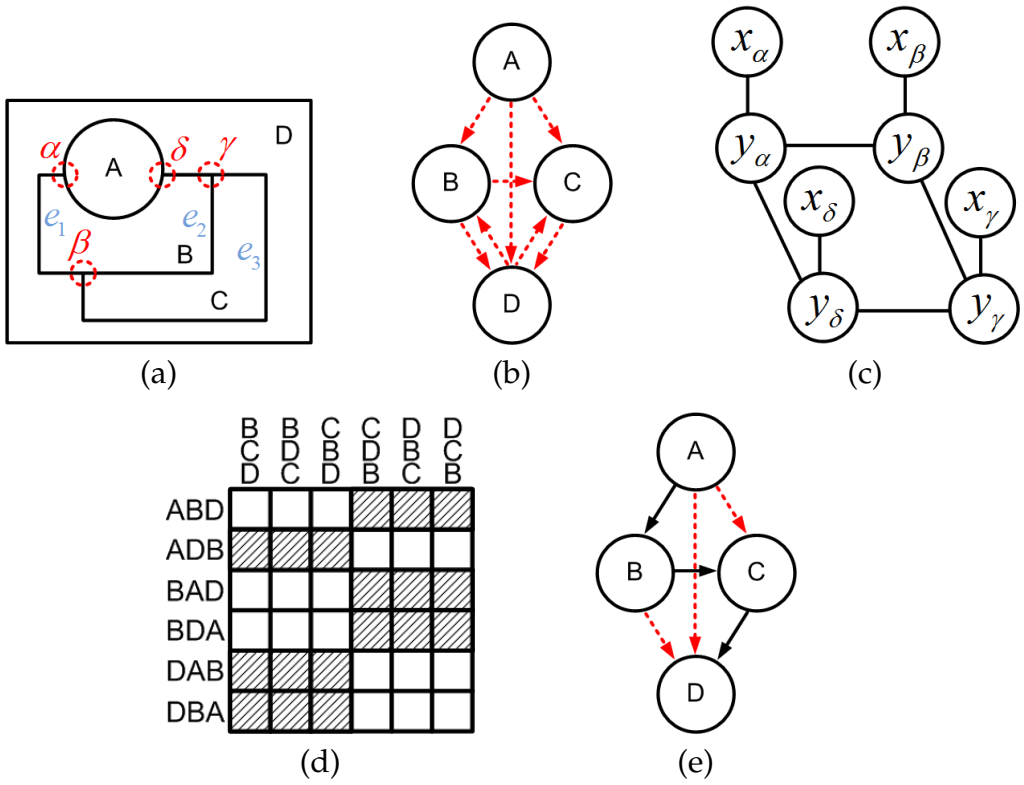


Figure 2.5: (a) Global depth reasoning example. (b) Each junction produces three directed edges in the depth order graph, e.g. junction α produces the directed edges $A \rightarrow B$,$B \rightarrow D$, and $A \rightarrow D$. (c) We use MRF to encourage the global consistency. Each node corresponds to one junction, and is connected with its neighbors. (d) The edge potential in our MRF gives high penalties (solid) if the segments' orders contradict between two nodes. (e) The depth ordering is assigned by the longest path in the final depth order graph (shown in solid arrow), from which we retrieve the depth ordering, such as $A \rightarrow B \rightarrow C \rightarrow D$.

However, relying solely on local decisions can lead to invalid configuration of d-order graph. Take Fig. 2.5 (a) as one example: if junction γ incorrectly predicts the order as $D \rightarrow B \rightarrow C$, while the others have the correct classification, a contradiction is introduced. This results in a loop of nodes B, C, D in the depth order graph, and makes it impossible to determine the depth order. To solve this problem, we propose a new approach using a Markov Random Field to encourage a more global consistency.

Global: We treat each junction in the image as one node in our MRF graph, shown in Fig. 2.5 (c). The label space for each node y_i is the possible order permutation of the segments, e.g. for junction α, its y_α will have 6 possible labels of the segment orders: *ABD*,*ADB*,...,*DBA*. The node potential $\phi(y_i|x_i)$ is calculated by taking the negative of Eq.2.1. The edge in our MRF is defined by the boundary. We link two junctions if they are connected by a boundary in the image. Also, if two junctions are connected by a boundary, they must share at least the two segments that this boundary separates. Therefore, the edge potential $\psi(y_i, y_j)$ is defined as the consistency between the segments' orders.

For instance, in Fig. 2.5 (a), junction α and β are linked by boundary e_1 (in light blue), and thus α and β share segment B and D that e_1 separates. Accordingly, the segment order on both junctions must be consistent, e.g. the order $A \to B \to D$ on junction α is consistent with the order $B \to C \to D$ on junction β, but the same order for α is inconsistent with the order $C \to D \to B$ on β, because the relative orders of B and D contradict. We build the edge potential $\psi(y_i, y_j)$ following this intuition: we assign zero penalties for the consistent orders, and high penalties for the inconsistent ones. Fig. 2.5 (d) gives an example of the potential matrix on the edge between node α and β in the MRF, with solid squares representing high penalties.

We use Tree Reweighted Decomposition (TRW) to minimize the total energy function $E = \sum_i \phi(y_i|x_i) + \sum_{i,j} \psi(y_i, y_j)$ for this MRF. Because of the penalties for the inconsistent orders, this optimization process encourages the consistent orders in a more globally optimized manner. Beliefs from other segments are passed through messages to help local decisions. In practice the inference process usually produces a consistent depth ordering, which enables us to trim the loop in

the depth order graph more safely. After that, we find the longest path in the depth order graph (now acyclic), and use this path as the skeleton for depth ordering, as shown in Fig. 2.5 (e). All the other nodes that are not in this skeleton path are assigned with depth values according to this path.

2.5 Occlusion boundary with closed loops

Segmentation is a necessary preparation for the depth ordering task, and we rely on the occlusion boundary detection to generate it: first a dense segmentation using watershed is performed to extract all the possible edges. Then each edge is classified as an occlusion boundary or not. After that the object segmentation is achieved by merging the regions between non-occlusion boundaries. Our detailed approach is presented as follows:

BoW features: In addition to [34], we propose new features based on bag-of-words [29] for occlusion boundary detection, for they effectively capture the texture information. Each edge from the initial segmentation lies in between two segments. We compute the dense SIFT words within these segments, and histogram them as the new features. Besides, the edge appearance itself provides rich information. If the edge is shaky or non-smooth, it is unlikely to be an occlusion boundary. Therefore, we also histogram the dense SIFT words along each edge. Together, these histograms form the new features for the occlusion boundary detection.

Enforcing the closed loop: Furthermore, occlusion boundaries are not independent. They usually enclose one object and form a closed loop, even when the object is occluded by others. For example, in Fig. 2.5 (a) segment C is enclosed

by edge e_2 and e_3, which together form a closed loop, even though e_2 belongs to segment B. Also the edges inside a loop are less likely to be actual occlusion boundaries.

(a) (b) (c)

Figure 2.6: (a) The local occlusion boundary detection result (best viewed in color). Heat map indicates the beliefs for the occlusion boundary, and the redder the higher. (b) We gradually examine the edges with high beliefs and retrieve the loop. (c) We lower the beliefs of the edges that connected inside to this loop.

We explicitly model this property as follows: first we classify each edge and get its belief for the occlusion boundary, shown in Fig. 2.6 (a). Since each edge connects two junctions at its two ends, we gradually group these junctions to retrieve the loop: initially, each junction in the image forms an individual group. Then we sort all the edges by their predicted beliefs for the occlusion boundary in descending order. After that, we examine each edge from the top belief and its two junctions: if they belong to different groups, we merge them. Otherwise, we find a closed loop with the current maximum predicted belief. If the loop has the size L larger than a minimum requirement L_{min}, we set the beliefs for all the edges l that form the loop as $b_{new} = \sum_l b_l/L$, and lower beliefs by T of the edges connected inside to this loop. The algorithm stops until we examine all the edges with beliefs larger than B_{min}. We also encorage the long edges in a similar way: we group the neighboring edges if they share similar directions, and enhance their beliefs for the occlusion boundary if the group size is large

enough.

2.6 Experiments

We experiment on three different datasets: a synthetic dataset (**syn**), the occlusion boundary dataset provided in [34] (**occ**), and our depth order dataset (**d-order**). Quantitatively we evaluate the depth ordering results by the ordering accuracy: for any two neighboring segments in the image, we examine whether their depth orders are correctly labeled comparing to the ground truth. We compare our final depth ordering algorithm (**Global**) with the following approaches:

BF: uses the boundary features proposed in [34].

JA: We re-implement the algorithm proposed in [37] that orders the depth mainly by angles within a junction.

pBF: uses the proposed boundary features.

pJF: uses the proposed junction features.

Com: uses the combined the features. The above methods share the same depth reasoning in [37] that deletes the loop in the depth order graph by the lowest local predicted belief.

Global: This is our full algorithm. We use the combined features in **Com** and the proposed MRF graph model to ensure the global depth consistency.

We color each segment by its depth order in the image to visually display the results. Segments in front are darker (more black), and occlude the segments that are brighter (more white). Note that since we don't estimate the absolute

depth, but the relative depth order, the absolute color value does not hold a specific meaning. The relative color between segments is more important. Segments are marked by a red "x" if incorrectly labeled in the depth ordering[2].

Generating the object segments is a key step that precedes depth ordering. Since we rely on the occlusion boundary detection to generate the segmentation, we also quantitatively evaluate the average precision for different occlusion boundary detection algorithms. We compare our proposed algorithm **loop** with the following approaches:

bfeat: uses the low-level boundary features from [34] [3].

pfeat: uses the proposed BoW features in addition to **bfeat**.

graph: uses **pfeat** and a graph model (MRF) to enforce the continuity of occlusion boundaries, similar to [34].

loop: This is our full algorithm that uses **pfeat** and explicitly enforces closed-loops and long edges.

Synthetic dataset: We synthetically create a dataset to evaluate the depth ordering algorithms. For this dataset, we randomly place 6 to 10 abstract segments in a image, including rectangles, circles, ellipses etc., with different colors and sizes. Shapes placed later will overlay the previous ones, and in this way we know the ground-truth ordering. Examples are shown in 2.7. We generate 2000 synthetic images, and use half of them for training the depth ordering algorithms, and the other half for testing.

[2]In some cases, "incorrect depth" is a relative term between two segments, and we arbitrarily mark one of them.

[3]To follow the convention in this thesis and make a fair comparison, in this step we do not compare with the result from the high-level geometric context labels, which are also often inapplicable in the settings.

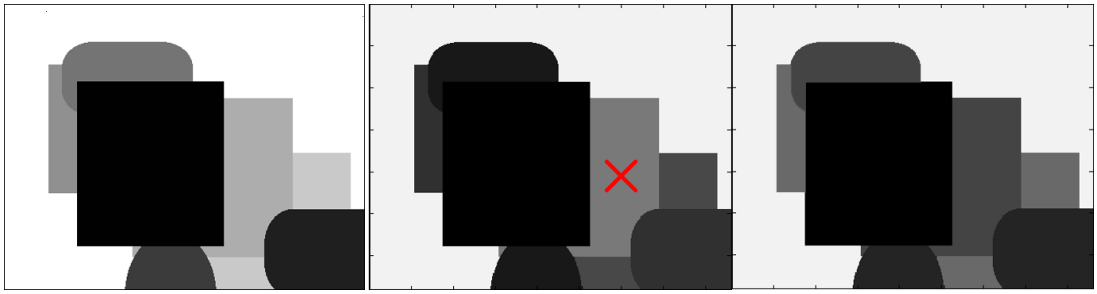
Figure 2.7: Examples of our synthetic dataset: color images are on the left and the ground truth depth orders are on the right, colored as the front segments are darker.

(a) (b) (c)

Figure 2.8: (a) Result from **Com**. Combined features can correctly label the depth order (darker segments are in front). (b) Results from **pJF**. (c) Results from **pBF**. Segments are marked by x if incorrectly labeled in the depth ordering.

This dataset has perfect segmentation, which enables us to directly compare the performance of different depth ordering algorithms. The depth ordering accuracies are presented in Table 2.1. The new features on boundaries (**pBF**) and junctions (**pJF**) improve around 3% in accuracy over the baseline feature sets (**BF** and **JA**), showing the effectiveness of our proposed features. Also combining them together (**Com**) achieves better performance over the individual feature set (10% over **pBF** and 4% over **pJF**). Our final algorithm (**Global**) has a clear advantage comparing to all the baselines. Overall **Global** achieves around 10% improvement over the previous works **BF** and **JA**.

Fig. 2.8 illustrates the advantage of the combined features. With only junction features, we cannot infer the depth order between the two rectangles, since



(a) (b) (c)

Figure 2.9: Our global reasoning algorithm can provide better depth ordering especially in complicated scenarios. (a) The ground truth depth ordering. (b) Result from **Com**. (c) Result from **Global**. Incorrectly labeled segments are marked by x.

their potential junction that can give the right depth order has been blocked, and the result is shown in Fig. 2.8 (b). On the other hand, using only boundary features makes it impossible to determine the depth order between the ellipse on the top and the rectangle below it, since the boundary in between is a straight line, and the result is shown in Fig. 2.8 (c). However, when combining these two features, we can correctly label the depth ordering of this image, as shown in Fig. 2.8 (a).

Our proposed **Global** algorithm outperforms the baselines, especially in the complicate cases when a segment interacts with multiple neighbors. Fig. 2.9 shows one example that when **Global** (shown in (c)) gives in a better depth ordering than **Com** (shown in (b)). The incorrectly labeled segment has four junctions with the segment behind it, and they produce inconsistent predictions. However after using the proposed model to enforce the consistency, we can produce a corrected depth order graph.

Occ dataset: We also experiment on the occlusion boundary dataset from [34]. This dataset includes 100 outdoor images with human-labeled segments and their quantized depth. For these natural images, object segmentation is the first

step before depth ordering. Therefore, two types of experiments are conducted: 1) we order the depth of manually-labeled ground-truth segments (**-gt**). 2) We automatically segment the image by using the occlusion boundary detection result, and then perform depth ordering (**-auto**)[4] [5]. We use 50 for training the occlusion boundary classifier and the depth ordering algorithms, and the other 50 for testing.

Table 2.1 shows the accuracies in depth ordering on the occ dataset, and example results are presented in Fig. 2.10. Since the variance in this dataset is large comparing to the limited number of training samples (only 50), the margins of the proposed algorithms over the baselines are smaller. However, still **pBF** and **pJF** outperform the baseline features **BF** and **JA** by 1.5% and 3%. **Com** further improves the result by 1%, and **Global** produces the best result.

For generating the segmentation, we show the average precision of the occlusion boundary detection in Table 2.2. The proposed BoW features give a 5% boost in detecting the occlusion boundary. Enforcing the closed loop (**loop**) marginally outperforms the baseline that uses the graph model (**graph**) and locally enforces the continuity. We believe the small increase is because this dataset is quite challenging. The output occlusion boundary result from the low-level feature **pfeat** is not reliable enough, and thus enforcing the loop may not be significantly better.

D-order dataset: Furthermore, to evaluate the depth ordering algorithms on natural images, we collect a new depth order (d-order) dataset. Various daily objects are placed to occlude each other in different configurations and scenar-

[4]The ground truth depth of each segment from the auto-segmentation is achieved by averaging the depth value over all the pixels in the segment.

[5]for this experiment only, geo-context information provided in [34] is necessary in order to generate usable segmentation.

ios. The dataset includes 1087 images. Each object is manually segmented, and its depth is acquired by using the Kinect sensor. Exemplar images are shown in Fig. 2.11.

We also use half of them for training and the other half for testing, and conduct two experiments: depth ordering on the ground-truth segmentation (**-gt**), and automatically generated segmentation from the occlusion boundary detection (**-auto**). The ground-truth depth order of each segment (from either human-labeled or auto-generated) is achieved by averaging the depth values within this segment.

Table 2.1 shows the depth ordering accuracy. The new features improve the performances from 1% to 3% over the baselines, and the combined features (**Com**) additionally boosts at least 4% in accuracy. **Global** gives the best performances in all the scenarios, achieving 10% improvement over the previous works in some cases. Fig. 2.10 and Fig. 2.13 show the ordering results.

Table 2.2 presents the average precision of the occlusion boundary detection, and Fig. 2.12 shows the example results. Our proposed new features outperforms the previous work by 7%, and our final algorithm (**loop**) produces additional 3% higher average precision comparing to the conventional graphical model (**graph**). More importantly, since our algorithm explicitly encourages the loop, it generates more reliable object segmentation for depth ordering.

Table 2.1: Average depth ordering accuracy (in %) of different methods on synthetic dataset (**syn**), occ dataset (**occ**), and our new depth order dataset (**d**). "-gt" : depth ordering is performed on the ground-truth segmentation. "-auto": the segmentation is auto-generated by the occlusion boundary detection.

	BF	JA	**pBF**	**pJF**	**Com**	**Global**
syn	81.0	86.6	83.0	89.9	93.7	**95.4**
occ-gt	70.9	63.3	72.4	66.9	73.2	**73.3**
occ-auto	66.4	58.2	69.5	64.5	69.4	**71.9**
d-gt	82.3	72.5	83.4	75.5	89.2	**91.7**
d-auto	75.0	62.2	75.3	68.3	79.3	**80.3**

Table 2.2: Average precision (in%) for the occlusion boundary detection on occ dataset (**occ-ap**) and our depth order dataset (**d-ap**).

	bfeat	**pfeat**	graph	**loop**
occ-ap	51.7	57.0	58.3	**58.6**
d-ap	65.5	73.0	75.7	**78.3**

2.7 Summary

We present a learning-based framework for depth ordering. We exploit new features on boundaries and junctions, and integrate them to form a better feature set for depth ordering. Furthermore, we propose a graph-based algorithm to encourage the global consistency in the depth ordering. We modify occlusion boundary detection algorithms to favor closed loops so that it is better suited for the ordering task at hand. We also collected a new dataset for the depth ordering task. Experiments in various scenarios show our proposed algorithms achieve better performances than the baselines.

For future work, we can further study how the depth ordering helps with segmentation and iteratively perform segmentation and depth ordering to see whether the interaction improves performances on both tasks. Additionally, we can employ our algorithm in tasks such as object recognition and scene understanding.

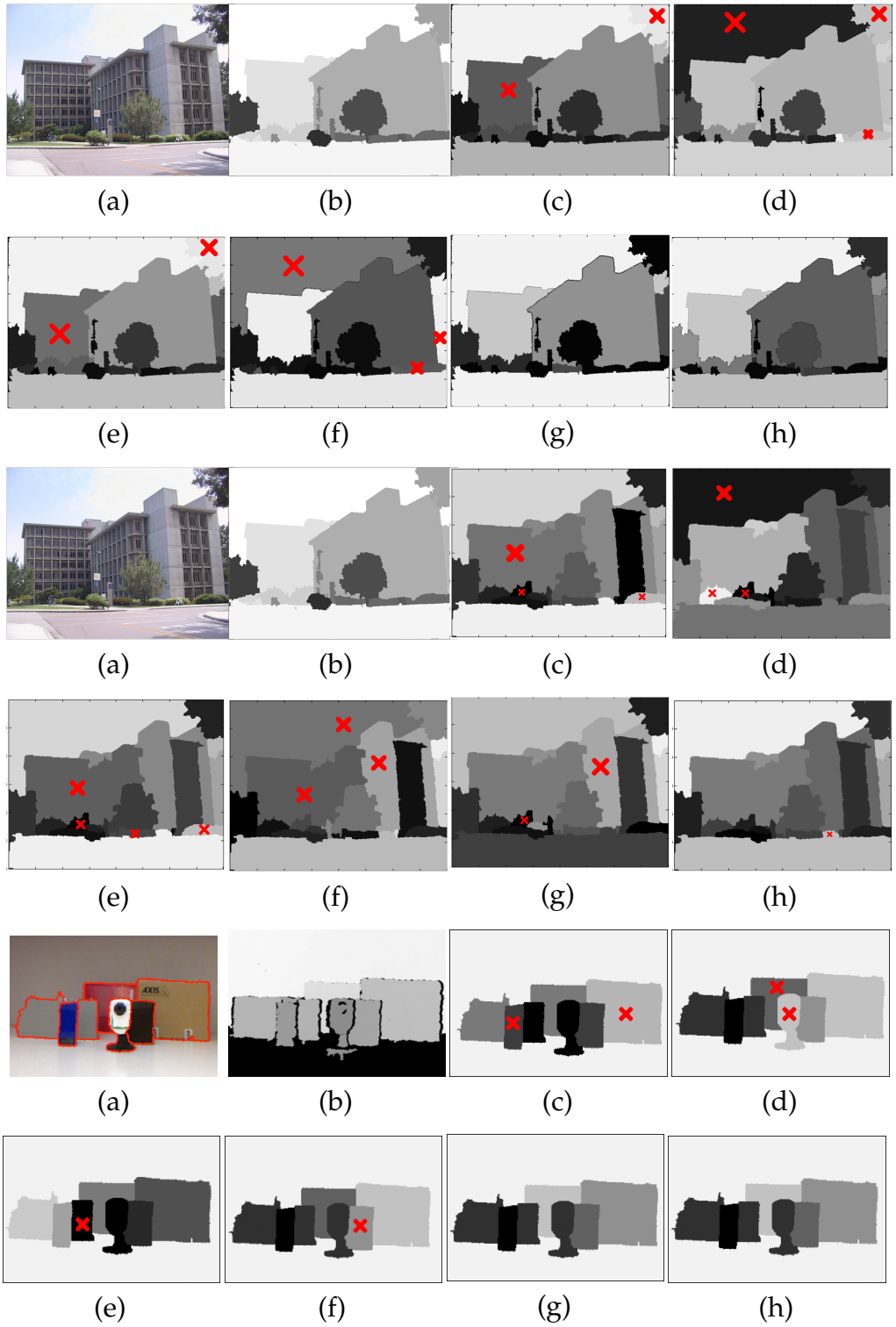
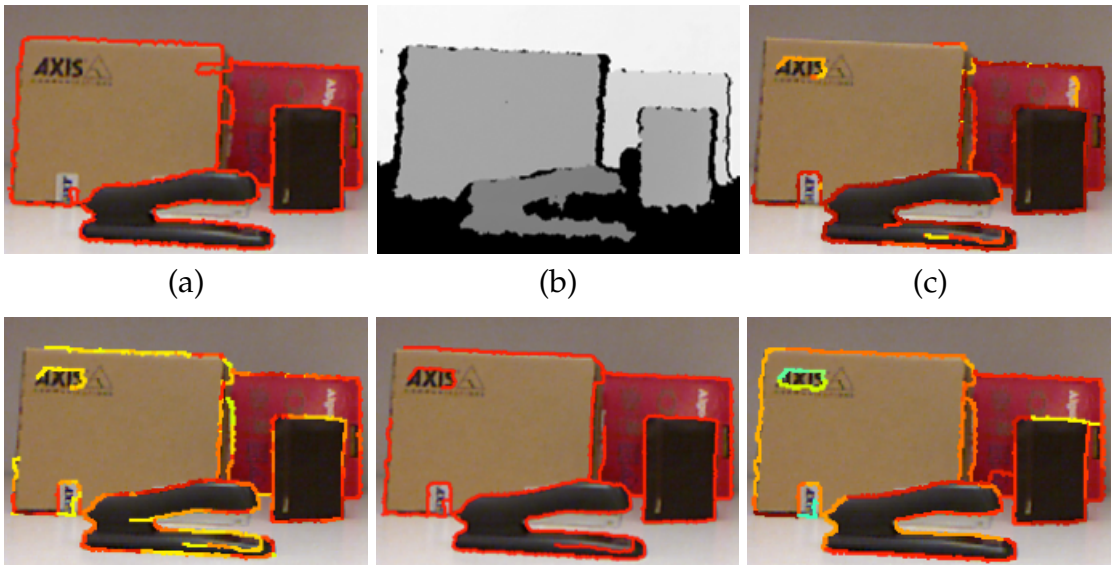


Figure 2.10: Results from occ datasets with ground-truth segmentation (**top row**), auto-segmentation (**middle row**), and from our depth order dataset with ground-truth segmentation (**bottom row**). (a) Input image. (b) Ground-truth segmentation and depth. (c) to (h) are results from different methods: (c) **BF**. (d) **JA**. (e) **pBF**. (f) **pJF**. (g) **Com**. (h) **Global**. Incorrectly labeled segments are marked by a red x.

Figure 2.11: Example images from our depth order dataset



(a) (b) (c)

(d) (e) (f)

Figure 2.12: Occlusion boundary detection result (best viewed in color): (a) the ground-truth occlusion boundary. (b) Depth image from Kinect. (c) Occlusion boundary detection result from **bfeat**. Red in color indicates higher beliefs for the occlusion boundary. (d) to (f) are results from (d) **pfeat**, (e) **graph**. and (f) **loop**.

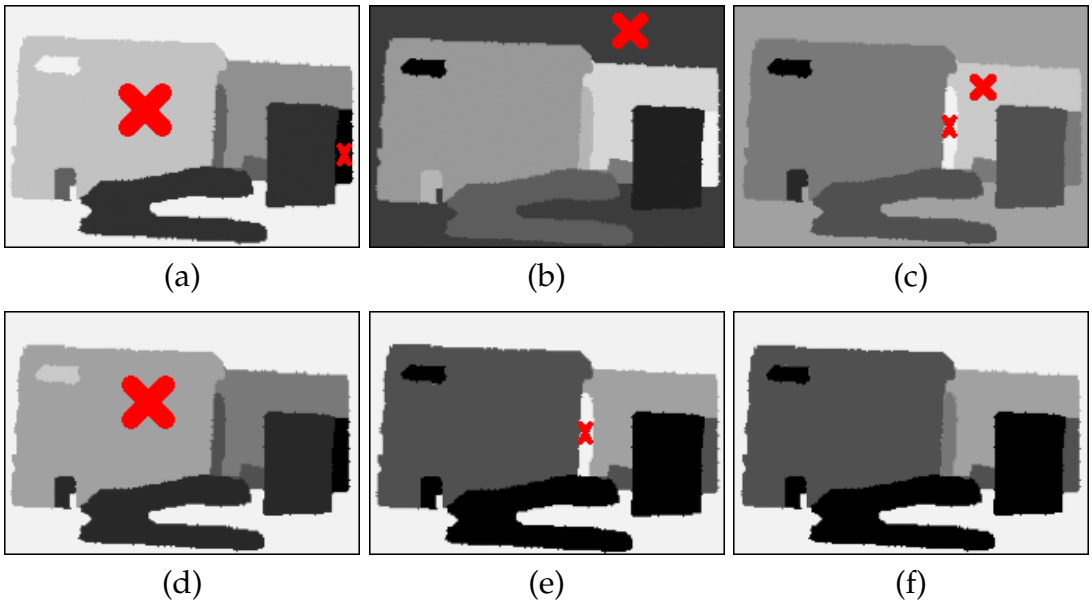


Figure 2.13: Depth orderings from auto-segmentation. (a) to (f) are results from (a) **BF**, (b) **JA**, (c) **pBF**, (d) **pJF**, (e) **Com**, (f) **Global**.

CHAPTER 3

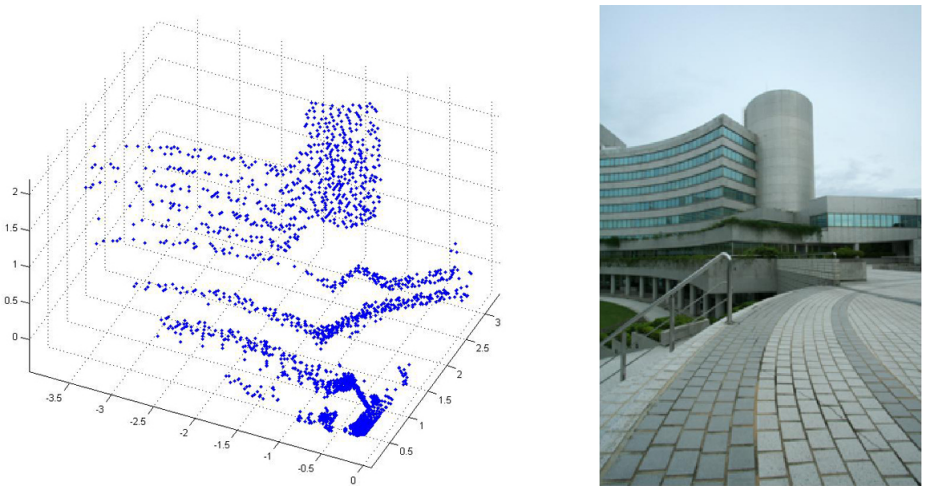
3D SURFACE SEGMENTATION

3.1 Overview

Dense 3D points become increasingly helpful for many different tasks, such as scene understanding and applications in robotics[1, 43, 44]. To get this 3D information, a laser scan is usually utilized. This method can generate thousands of 3D points in very good precision (shown Fig. 3.1 (a)).

However, the 3D points from the laser scan are still sparse when compared with a normal image, which may contain pixels in the scale of million (shown in Fig. 3.1 (b)). Therefore one may want to get a denser 3D map by interpolating the 3D location of every pixel in the image. To do this, during the laser scan, an image is taken and paired to the laser scan data. Each 3D point can be registered and projected back to the image using the simple geometry of the camera. For the pixel that lacks its corresponding 3D point, we can use its 2D pixel neighbors who have their corresponding 3D points to interpolate the 3D position of the target pixel. This is called "3D-interpolation" or "3D-superresolution", and can be done by linear interpolation, or using the pixel color as a clue to weigh the neighbors [45, 46].

In this work, additional aspects are considered as contributions: we estimate the underlying 3D surfaces hidden behind the 3D point clouds, and combine them with the color information for 3D-interpolation. The proposed algorithm is inspired by the following intuitions: first, 3D points can be better segmented in addition to the color. In 3D we can pick better neighbors for the target pixel



(a) (b)

Figure 3.1: (a) the 3D point clouds scanned from a laser range sensor. (b) the scene image associated with the 3D point clouds

for interpolation. Take Fig. 3.1 as an example: pixels near the corner of the building are close in 2D space and have similar colors, but in 3D space they lie on different surfaces with changes in depth. Using only the pixel location and color will result in wrong neighbor points for interpolation. However, a clustering in 3D space based on surface can better solve this problem and lead to a lower interpolation error.

Second, estimating the 3D surface function will result in better interpolating. Previous works of 3D-interpolation are usually done in a local region. However, fitting larger surfaces to the 3D point clouds can produce better model. Also high-order surfaces with curvatures may fit the 3D points better, such as the cylinder structure of the building shown in Fig.3.1 (b). Therefore we propose an algorithm to segment the 3D points based on their underlying surfaces, and interpolate using these surface functions. The overview of our proposed algorithm is presented in Fig. 3.2

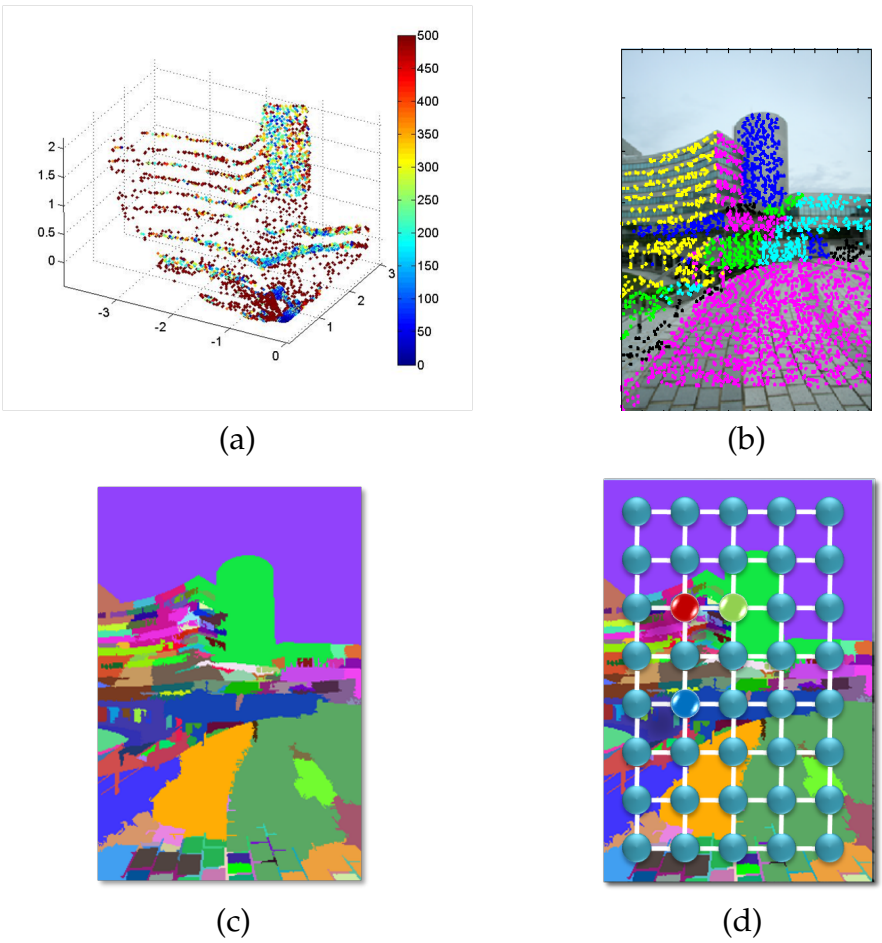


Figure 3.2: (a) Linear interpolation will introduce large errors when the depth changes within local region. In this figure the 3D points are presented in color by the fitting errors to the ground truth (error bar is on the right: the more blue, the lower error. (b) When projecting each 3D point to its corresponding 2D pixels, it shows that the 3D points are quite sparse compared to the number of pixels. (Color represents a preliminary 3D surface segmentation) (c) 2D segmentation can help identifying the real neighbor of each 3D points. (d) Using an MRF, we can infer the actual 3D geometry information of each pixel, and thus achieve better 3D point estimation.

3.2 Related works:

Dense Interpolation on 3D data using an image have been studied in [45] and [46]. [46] puts the color distance between the pixels into Markov Random Field.

This method initializes a bi-linear interpolation, and then iteratively updates the 3D result using the colors of the pixels. [45] comprehensively examines the existing methods for 3D-interpolation. However, previous algorithms rarely utilize the geometry information within the 3D points. We compare our method to these baselines and experiments show the improvement when using the 3D surface information. [1] proposes a pure vision-based approach for depth estimation, but the goals are different from us since they only estimate the depth of each pixel rather than the full 3D location. Thus the results are not comparable.

[43] combines the 2D color image and the 3D laser scan data to find a salient region in the scene. Their underlying idea is similar to ours in combining the image and the dense 3D point clouds together. However we are targeting at two different tasks and the approaches are significantly different. We focus on building a more precise 3D model, while they want to detect the saliency.

3.3 Surface segmentation and fitting

There are some related works of surface segmentation on triangle-meshes [47]. For 3D point clouds, surface-based segmentation has been used in the area of range image processing in [48] [49], and reverse-engineering in [50]. Generally there are three approaches: Split-and merge, region growing and clustering. In this thesis, we implement a new way of region growing using efficient graph-based method [19].

Surface segmentation: Surface segmentation is to group the 3D-points that: 1) are close to each other in Euclidean distance; 2) lie on one smooth surface. For surface segmentation, the normal vector of the latent surface that each 3D point

lies on becomes important. We cluster the 3D points by the angles of this normal vector.

The normal of a 3D-point is initially estimated by using its neighborhood. For each 3D point $P_i = [x,\ y,\ z,\ 1]$[1]), we calculate the normal vector $\vec{n}(P_i)$ by solving the following equation:

$$[P_{i1},\ P_{i2},\ \ldots,\ P_{in}][\vec{n}(P_i),\ d]^T = 0 \tag{3.1}$$

for $P_{ij} \in Neighbor(P_i)$ with the constraint that $|n(P_i)| = 1$. The neighborhood of a 3D point P_i can be determined by choosing the N nearest neighbors.

After estimating the normal vector $\vec{n}(P_i)$ of P_i, segmentation is performed based on surface. The intuition is that if the angles of two normal vectors are too different, then the corresponding two points may belong to different 3D surfaces. We use the efficient graph-based method [19] to generate the initial segmentation base on the normal vector. It returns a set of 3D segments $\{C_i\}$.

Incorporating the idea introduced in [19], we modify this efficient graph-based method to generate surface segmentation S as follows:

Surface fitting: We discover that generally most objects are not composed of complex surfaces, especially for structured and man-made things, such as buildings, cars, roads etc. Therefore we propose to use only the first and the second order surfaces (plane and quadratic surfaces) for fitting. Practically they provide very good approximation.

For each segment C_i, we solve Eq.(3.1) for the points $\{P_i = [x,\ y,\ z]^T\} \in C_i$ to estimate the normal vector $\vec{n}$ of C_i, then apply a rotation matrix R to align $\vec{n}$ to $\vec{z}$

[1]the fourth dimension is set to 1 to follow the convention in Structure from Motion

Algorithm 1: Surface segmentation on 3D points

Initialize $\{P_i\}$ as one individual group C_i, set $l_i = 0$ for each C_i. $S^0 = (C_1, C_2, \ldots, C_n)$. Calculate $\vec{n}(P_i)$ by Eq.(3.1)

Link P_i with its neighbor P_{ij} to form an edge $e_{ij} = \arccos(\vec{n}(P_i) \cdot \vec{n}(P_{ij}))$. Sort all the m edges in ascending order $\{e_q\}$.

for q=1 to m **do**

  if $e_q < min(l_i + \frac{k}{|C_i|}, l_j + \frac{k}{|C_j|})$ **then**

    Form a new segmentation S^q by

    a) merging C_i and C_j into a new group C_h;

    b) updating $l_h = e_q$ for C_h

  end if

end for

direction. This enables us to model the plane surface by,

$$f_{ip}(x, y, z) = z - [x,\ y,\ 1]\theta_p, \theta_p = [c_x\ c_y\ 1]^T$$

and the quadratic surface by,

$$f_{iq}(x, y, z) = z - [x^2\ y^2\ xy\ x\ y\ 1]\theta_q,$$

$$\theta_q = [c_{xx}\ c_{yy}\ c_{xy}\ c_x\ c_y\ 1]^T$$

θ_p and θ_q can be calculated by least square fitting on the 3D points $\{P_i\} \in C_i$.

Classifying the plane from the quadratic surface is a necessary step to prevent potential over-fitting. We adopt two methods for this classification: calculating the principal curvatures on the estimated surfaces [47], and testing the distribution of the fitting errors [48]. The details are omitted for brevity.

Unstable points: 3D points that lie on the boundary between the surfaces introduce errors in estimating the normal vector, and thus they make the surface segmentation unreliable. We call these points "unstable points". We note that compared with their neighbor points, these points have drastic changes in the normal vector or the fitting error, so we can classify these points as follows: first we calculate several criteria $S(P_i)$ of each point P_i to its neighbor points $N(P_i)$:

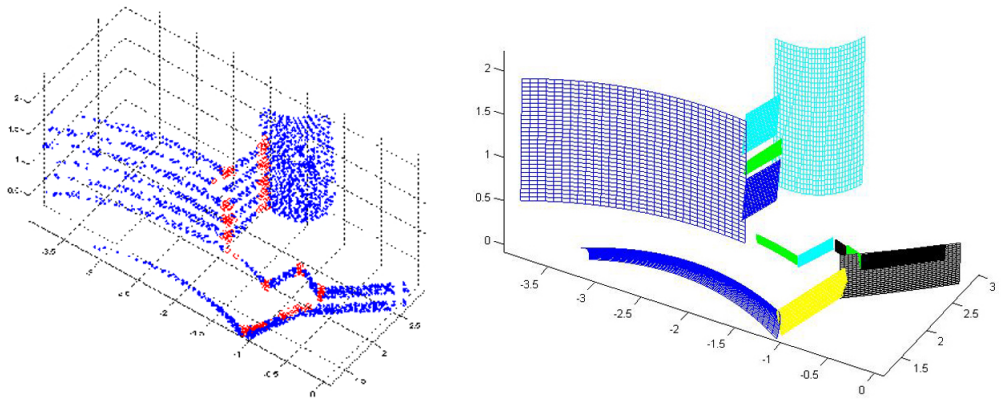
$$S(P_i) = \sum_{j=1}^{n} |X(P_i) - X(P_j)|, P_j \in N(P_i)$$

X can be the previously estimated normal vector, the fitting error or the 3D location of each point. We calculate the average $\bar{S}$ and the standard deviation $std(S)$ of S value for all the points, and choose those points with S value larger than $\bar{S} + 3std(S)$ as the unstable points. The selected result is shown in Fig. 3.3 (a).

Then we repeat surface segmentation/fitting on the remaining stable points, and filter out the unstable points iteratively. After two to three iterations the algorithm will produce reliable 3D surfaces $\{C_i\}$ with their surface functions, as shown in Fig. 3.3 (b).

3.4 Combining with color

Since the 3D laser scan data is sparser than its corresponding 2D image, only a subset of pixels in the image will have their corresponding 3D locations. After previous steps, these pixels are associated with their estimated surfaces $\{C_i\}$. We treat each surface as a label, and inference the latent surface label $\{C_i\}$ on every pixel in the image through Markov Random Field (MRF) [51], shown in Fig 3.5 (c).



(a) (b)

Figure 3.3: (a) we filter out the unstable points, e.g. the points between surface boundaries (points in red). (b) the surface segmentation result after iteratively fitting the surface and filtering out the unstable points.

Forming MRF: MRF uses two terms in modeling: the smoothness-term (defined on each edge) and the data-term (defined on each node (i.e. pixel)). In this work we connect 4-grid neighborhood of each pixel to form the edges. For the smoothness-term, the Potts model is applied on the edge between the pixel p_i with label C_i and the pixel p_j with label C_j:

$$V(C_i - C_j) = \begin{cases} 0, & C_i = C_j \\ d, & C_i \neq C_j \end{cases}$$

The data-term defines the potential on each pixel node. We use the RGB color feature to model this potential. Note that one 3D surface may possess of several different color mixtures. One example is shown in Fig. 3.1 (b), where green windows and grey walls are the two main colors for the building. Therefore, for the pixels $\{p_{C_i}\}$ that belong to the surface C_i in 3D, we estimate k Gaussian Mixture Models in color space, $N_{C_i,1,(\mu_1,\sigma_1)}, ..., N_{C_i,k,(\mu_k,\sigma_k)}$. Then the probability of assigning the pixel p_i (with RGB color $I(p_i)$) to the surface C_i is determined by the maximum likelihood of the pixel p_i assigning to each color mixture model

of the surface C_i:

$$D_{p_i}(C_i) = P(p_i|C_i) = \max_j N_{C_i,j}(I(p_i))$$

$D_{p_i}(C_i)$ is used as the data-term.

In sum, the inference on MRF is achieved by minimizing the following energy function:

$$E(C) = \sum_{p_i} D_{p_i}(C_i) + \sum_{(p_i,p_j)\in N} V(C_i - C_j)$$

where N represents the four-edge connected graph. Loopy Belief Propagation is implemented for the inference.

Efficiency: We improve the efficiency for MRF inference by reducing the possible surface labels for each pixel. First we segment the color image[2] and generate a set of super-pixels $\{g_i\}$. We prune the surface labels as follows: if within one super-pixel g_i, the pixels with known 3D points lie on T surfaces $\{C_{i1}, C_{i2}, \ldots C_{iT}\}$, then all the pixels within this super-pixel are limited to these T possible labels. This significantly decrease the number of possible labels for each pixel, and leads to a shorter inference time which is within minutes. Also the edge consistency in the color image produced by the segmentation algorithm can be better preserved.

3D-point interpolation: Having the surface label for each pixel, the dense 3D-interpolation is achieved by using the surface function f_i of each surface label C_i. For each pixel p_i with label C_i, its 2D location $[u,\ v]^T$ is related to its 3D location $P = [x,\ y,\ z]^T$ by:

$$[uw,\ vw,\ w] = M[x,\ y,\ z,\ 1]^T \tag{3.2}$$

where the projection matrix M can be easily estimated using the pixels with known 3D points. w is the only unknown variable and can be calculated by

[2] we use mean-shift algorithm [52]

using the surface function: $f_i(x(w), y(w), z(w)) = 0$. Then the 3D location $[x,\ y,\ z]^T$ is obtained by substituting the value of w into Eq. 3.2.

3.5 Experiments

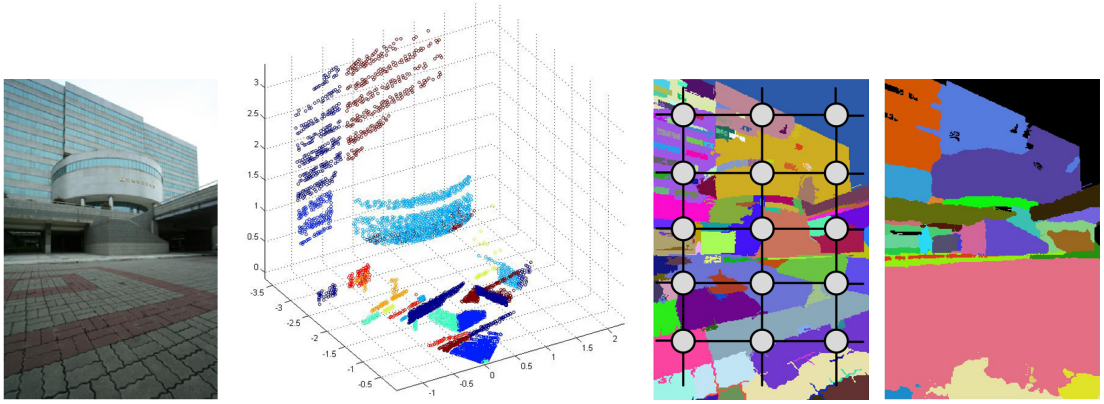
(a) (b)

Figure 3.4: Some sample images for the experiments. (a): ITRI dataset, including 73 indoor and outdoor scenes. (b): Make3D dataset [1]

.

We experiment on two dataset: Make3D dataset and ITRI dataset. Make3D[1] is a public dataset with well calibrated 3D laser scan data and images available. We use dataset [1] in Make3D, with 239 sets of images and laser scan data available (the others are missing the full 3D information, where only the depth is available). In addition, we manually collect another ITRI 3D dataset (Fig. 3.4), focusing on structured outdoor buildings and indoor environments. 73 indoor and outdoor scenes (Fig. 3.4) were scanned by the Sick laser range scanner LMS-291. Images were taken simultaneously and 3D points were manually calibrated with the image. For the parameter setting, the images are resized to 480 pixels in width for efficiency. We use 20 nearest points as the neighborhood for P_i. For MRF inferencing, we set $k = 10$ for GMM, $d = 10$ and $\lambda = 1$ for the energy function.

Two baseline methods are implemented for comparison: linear interpolation (**LP**): we linearly combine 20 nearest neighbor pixels whose 3D data are avail-



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

Figure 3.5: (a) the input image. (b) the surface segmentation result. (c) inferencing MRF on the image (segmentation in the color space is for efficiency). (d) MRF inference result.

Table 3.1: Interpolation error (in mm) on Make3D dataset (upper row) and on ITRI dataset (lower row). We also experiment with the down-sampled modeling set, from 100% to 50% of the total modeling 3D points.

percent	100%	90%	80%	70%	60%	50%
LP	63.0	67.9	70.1	73.4	77.2	80.7
cMRF	54.7	60.0	60.4	62.9	66.0	67.8
Prop	**37.8**	**37.9**	**38.0**	**38.1**	**39.7**	**44.1**
LP	164.7	170.4	180.5	190.4	199.4	211.1
cMRF	159.0	164.5	172.3	181.4	189.3	199.7
Prop	**117.7**	**117.8**	**118.7**	**122.0**	**124.2**	**132.3**

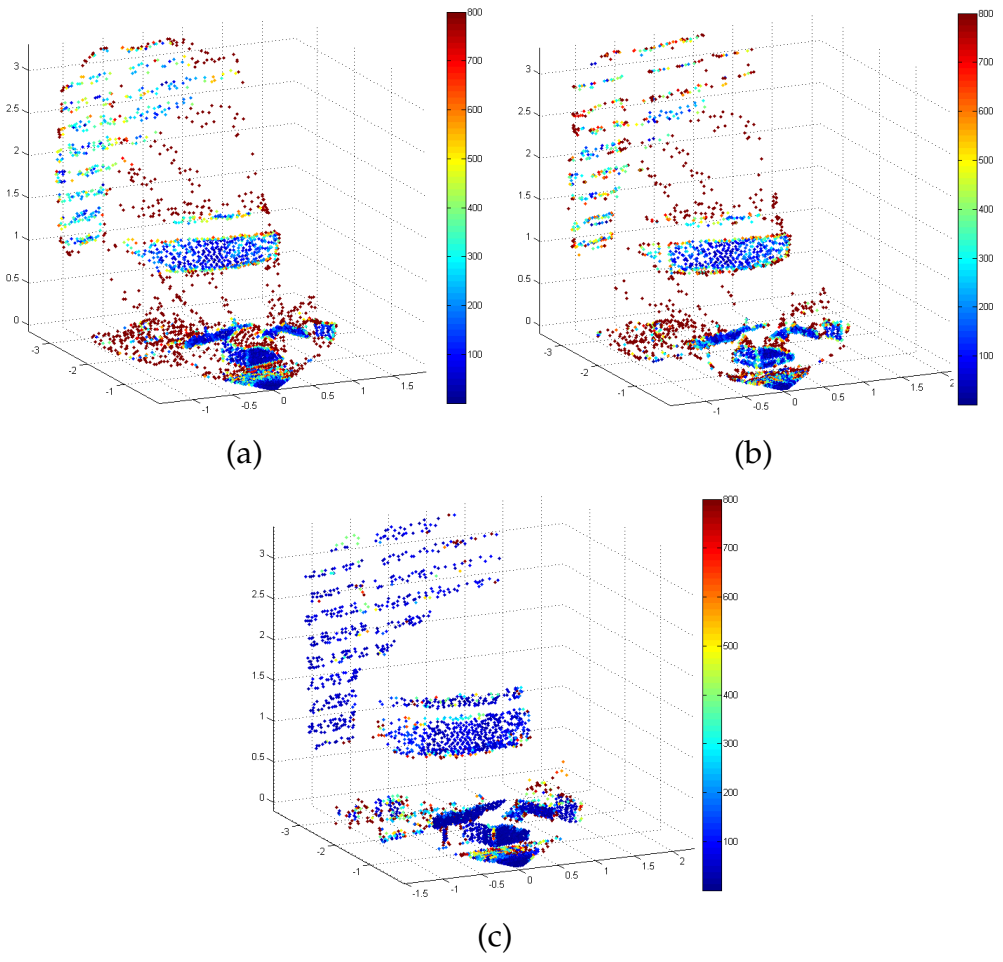


Figure 3.6: 3D-point interpolation error using different methods. The error is mapped in color, and the unit is mm. We show the result of Linear interpolation (LP) in (a), color-based MRF (cMRF) in (b), and the proposed algorithm (Prop) in (c).

able, and weigh them by their 2D distance to interpolate the target pixel; and interpolation using the color-based MRF (**cMRF**) [45, 46]. We evenly divide the 3D points into two sets, and each one includes around 7000 3D points. The first set is used for modeling; the other set is used as the ground truth for the interpolation testing. The average interpolation error of the testing points is reported in the first column of Table 3.1. On average our method (**Prop**) achieves more than 25% improvement over the baselines.

Furthermore we test the robustness of the proposed algorithm. We down-sample the 3D points in the modeling set until only 50% of the original points are used, and keep the testing set constant. Table. 3.1 shows that the proposed algorithm still gives lower interpolation errors in all the cases.

Fig. 3.5 and Fig. 3.6 show one complete experiment example. The performance of our method is improved in the following cases: 1) between the boundary of the surfaces. In this case the propose algorithm can identify a better neighborhood for the target pixel, especially when the surfaces has similar color, but have a depth change in 3D. Baseline method will result in a high error between the surface boundaries, shown in (a) and (b), while our method is more reliable, shown in (c); 2) in the large and structured regions. Our algorithm makes use of a larger group of the 3D points to estimate the surface function, therefore the fitting error is lower than locally interpolating.

To test the robustness of our method, we also down-sample the estimation group. We start only using 20% of the original 3D points to build the model/for interpolation, and increase 10% each step until 100%, while keeping the testing group the same all the time. The overall error should be decreasing when more points are given for modeling/interpolation, as shown in Fig. 3.6 (g). However, the average fitting error is of our method is lower than the baseline method in all these tests, which indicates that even in the case of smaller samples, our method can still give robust modeling of the environment.

3.6 Summary

In this part of the thesis we propose an algorithm for the dense 3D point interpolation based on the 3D surface and the color information. We first perform the surface segmentation and fitting, and then combine the surface labels with color through MRF Framework. The experiments on various indoor/outdoor scenes show that our method has a better performance over the baselines, and is robust even with fewer modeling points.

Future works can be done on classifying different objects and interpolating the 3D map accordingly, e.g. trees are better segmented into small regions while buildings are better segmented into large surfaces. This may lead to a better model and lower interpolation errors.

CHAPTER 4

3D OCCLUSION BOUNDARIES

4.1 Overview

Object boundaries in images are important clues towards the high level interpretation of the scene [53] [8]. In general, three types of boundaries exist: (a) occlusion boundaries, which are the edges produced by one object occluding the other; (b) connected boundaries, which refer to the touching edges of two connecting objects; and (c) homogenous boundaries, which are produced by the texture from the object. One exemplar image of different boundaries is shown in Fig. 4.1. In this part of the thesis, we learn to detect boundaries on color and depth image pairs.

Occlusion and connected boundaries are important edges for understanding the geometry of a scene as well as the layout of objects within the scene. Occlusion boundaries can provide a segmentation of the image [54] and the depth ordering between objects [36]. Connected boundaries, which indicate the supporting relation between surfaces, are important for scene understanding [53] [30]. For example, in Fig. 4.1, once the occlusion boundaries (f) and connected boundaries (g) are known, it is easier to segment the objects and analyze supporting planes. This understanding in turn makes further applications possible, such as object manipulation or object placement [15].

Although these boundaries are important, identifying them in a robust manner is not an easy task. In some cases, prior semantic knowledge of the scene (e.g. “ground”, “sky” or geometric context) has to be introduced for occlusion

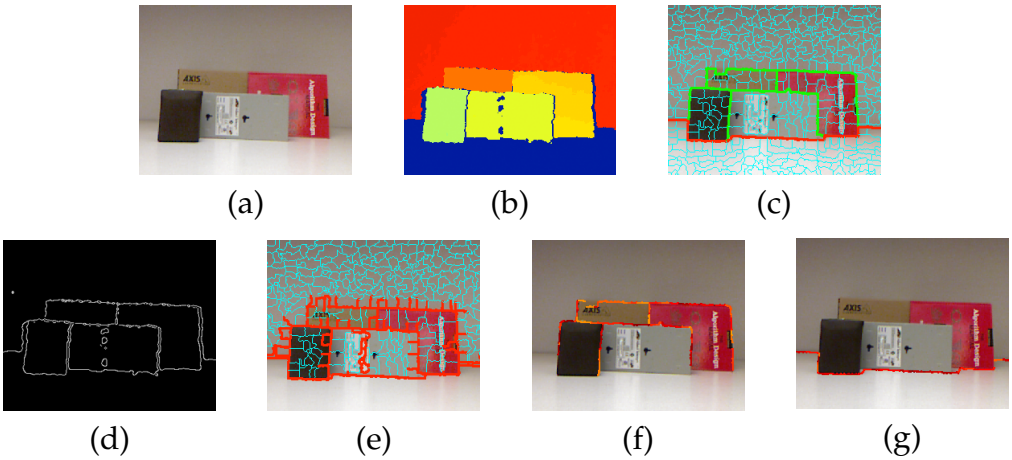


Figure 4.1: Boundary examples: (a) the color image and (b) the depth image from the structured light depth sensor (Kinect). (c) We extract all the possible edges by densely segmenting the color image, and label the following three types of boundaries: **homogeneous boundary** (cyan), **occlusion boundary** (green), and **connected boundary** (red). Directly using the depth data to extra the boundaries may fail because of the noisy in the boundary region. (d) is a typical Canny edge detector result performed on the depth image. It shows inaccurately detected edges due to noise. (e) presents the result when naively applying the depth edge detection result to label the occlusion boundary. However, using our learning based framework, we can better detect the occlusion boundary (f), and the connected boundary (g) , where the color indicates the classification beliefs for the labeling (more red → higher belief).

boundary recovery [53] [2]. This additional knowledge may not be applicable for generic and complex scene images, as in [55], or images of objects at a macro view, as shown in Fig. 4.1. This is where the depth can play an important role and help most [56].

3D depth data are increasingly popular as a help for many different vision tasks, such as object recognition, scene understanding, and vision for robotics [8] [1]. A laser scanner or structured-light sensor, like Kinect, is usually used to retrieve the depth information, along with a regular color camera for RGB images. Dataset combined with color and depth become quite available online

[55] [1]. Specifically, in this thesis we focus on the depth data from Kinect-like sensors. Compared with a laser scanner, they are inexpensive, widely available, and have higher resolution and faster speed in retrieving depth images.

However, to identify the occlusion and connected boundaries, simply "adding" the Kinect depth data may not solve the problem, because these depth information are quite noisy, especially in the region of the object boundaries [57] [58]. Fig. 4.1 (d) and (e) provide exemplar images. The depth image contains noisy boundaries, or even false ones because of the holes created by the non-reflected region. In general, depth images fail to produce the sharp edges common in color images, which are the regions that are most vital to our problem of reasoning about occlusion and connected boundaries. Therefore, we propose our learning-based framework and develop novel 3D features to address this problem. We use a 3D surface-based segmentation to overcome the noisiness of the depth data. This segmentation step can avoid local decision pitfalls, and forms a better joint interpretation of the surfaces.

Meanwhile, we also generate features in the color domain, and concatenate all the features to supervise a Support Vector Machine (SVM). The output of the SVM is used as the unary node in our graphical model. For a joint inference, we propose a Conditional Random Field (CRF) based framework, where pairwise potentials are learned by using the features computed on each junction of the boundaries.

Labeling the boundaries for learning is usually an intensive and laborious work for human, because there can be many occlusion and connected boundaries in a single image, and the human needs to label each one. This step largely restricts the size of the dataset for learning. For example, in [1], only 50 im-

ages were used for training, probably because of the labor involved. Therefore, we incorporate an active learning method with our new feature set to identify which boundaries would be most useful for training our model. In this way, we use fewer labels for training and achieve similar testing performances.

Our extensive experiments on two different datasets prove the effectiveness of our new features, and the proposed CRF framework improves the inference accuracy compared to solely local decisions. In addition, the active learning approach decreases the number of images required to label, while achieving a high level of testing accuracies.

In sum, our contributions are as follows:

1. We introduce mid-level features for boundary inference from color and depth images that are based on surface segmentation.
2. We propose a learning-based framework for both occlusion and connected boundary inference, and allow for active learning.
3. We propose a shared CRF model for occlusion and connected boundary inference.

4.2 Related Work

Our work is primarily related to two topics in the literature: boundary detection and depth imaging.

Image-based boundary detection and segmentation has a long history. In Martin et al [3] [59], low-level color and texture features are proposed for

learning the segmentation of natural images, using a proposed human-labeled dataset [9]. Hoiem et al. [53] then extended this learning-based segmentation algorithm to the area of occlusion boundary detection and scene understanding. [53] showed that by detecting the occlusion boundary and the geometric labelings of the scene, it is easy to estimate the depth of the testing image through analyzing the occlusion boundary between the object and the ground. Later [8],[30] and [7] demonstrated that this information can further help other high-level interpretation of the scene, such as the object recognition. In this work, we further explore the occlusion and connected boundary detection with the help from both depth and color image.

As a mass-market depth sensor, Kinect has received wide interest from the computer vision community. Since its introduction, the color and depth information from this sensor have been applied to a wide range of computer vision tasks, such as environmental reconstruction [60], object recognition [61] [62] [63], object segmentation [55], and robotics [15]. In estimating human pose, [56] completely ignore the color information and exclusively relies on simple depth features for recognition.

The Kinect depth image relies on infrared projection, and tends to have a limited depth range (about 5 meters), and has noise at object boundaries. To improve the depth map, [57] and [58] propose 3D denoising and interpolation algorithms. Further, by merging multiple depth images, highly clean depth maps can be produced [64]. In our work, we are interested in a scenario with single static color and depth images for boundary inference. We use surface segmentation and fitting [11] [49] for a higher-level interpretation of the 3D data. This process decreases the depth noise by forming larger surfaces from the surface-

fitting step. More importantly, this step produces a rough object segmentation, and can approximately locate the occlusion and connected boundary for the task. However, as we will show, this analysis, by itself, is not sufficient to accurately locate occlusion boundaries and infer their type. Our proposed new feature set captures the information from this surface segmentation to learn different boundaries.

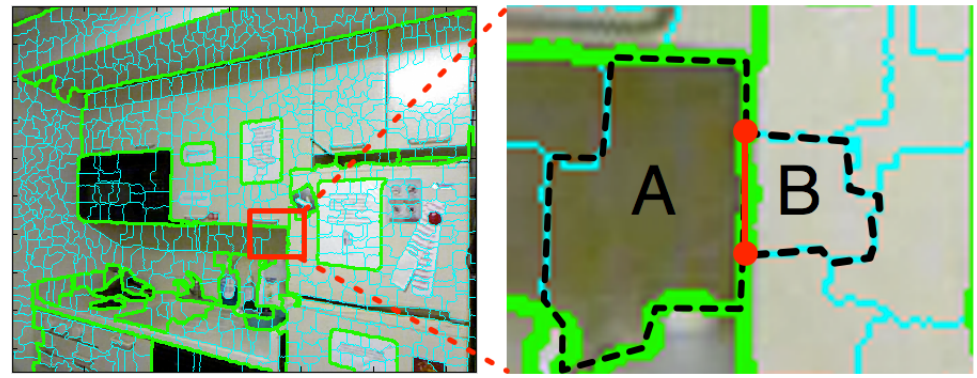
4.3 Color and Depth Features

Our algorithm mainly follow the flow from [53] for detecting boundaries. First, the depth sensor is calibrated with the color camera, which enables us to retrieve the 3D depth of each color pixel. Initially, we densely over-segment a color image into super-pixels using watershed algorithm, shown in Fig. 4.1 (c). Then the task is to classify each small edge into one of the three boundary categories. We propose a set of color features x_c and depth features x_d, and training a Support Vector Machine based on them.

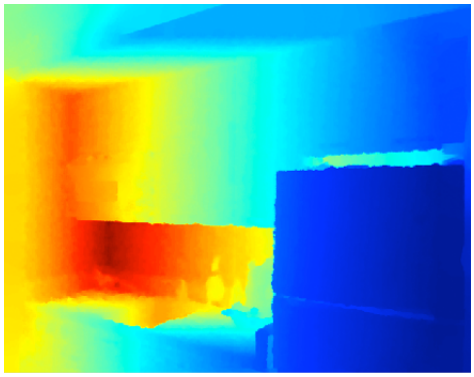
4.3.1 Color features

base features: we use the edge and segment features proposed in [53] as our base feature set in the color image. This set includes color, probability of boundary [3], segment position and others. The high-level geometric labeling is discarded because it is usually not applicable for the indoor scenarios in out testing and training sets.

Edge curvature: the curvature of each edge gives a strong clue for identifying a



(a)



(b)

(c)

Figure 4.2: (a) **left:** initially, we densely over-segment color images to extract all the possible boundaries. The cyan edges are produced by the over-segmentation, and the green ones are the ground-truth occlusion boundaries. **right:** Each edge lies between two segments, e.g. the red edge is between segment A and B. Features are computed based on the edge and its two segments. (b) The depth image. (c) The surface segmentation result from the depth data.

reliable boundary. In an indoor scene, most man-made objects have structured boundaries. Homogenous boundaries are usually produced by the texture or noise, and they can be shaky and irregular, while the actual occlusion or connected boundaries are composed of sharp straight lines. Examples are shown in Fig. 4.2 (a).

For each small edge from the over-segmentation, we explicitly describe its curvature as follow: first we connect its starting pixel p_s and ending pixel p_e,

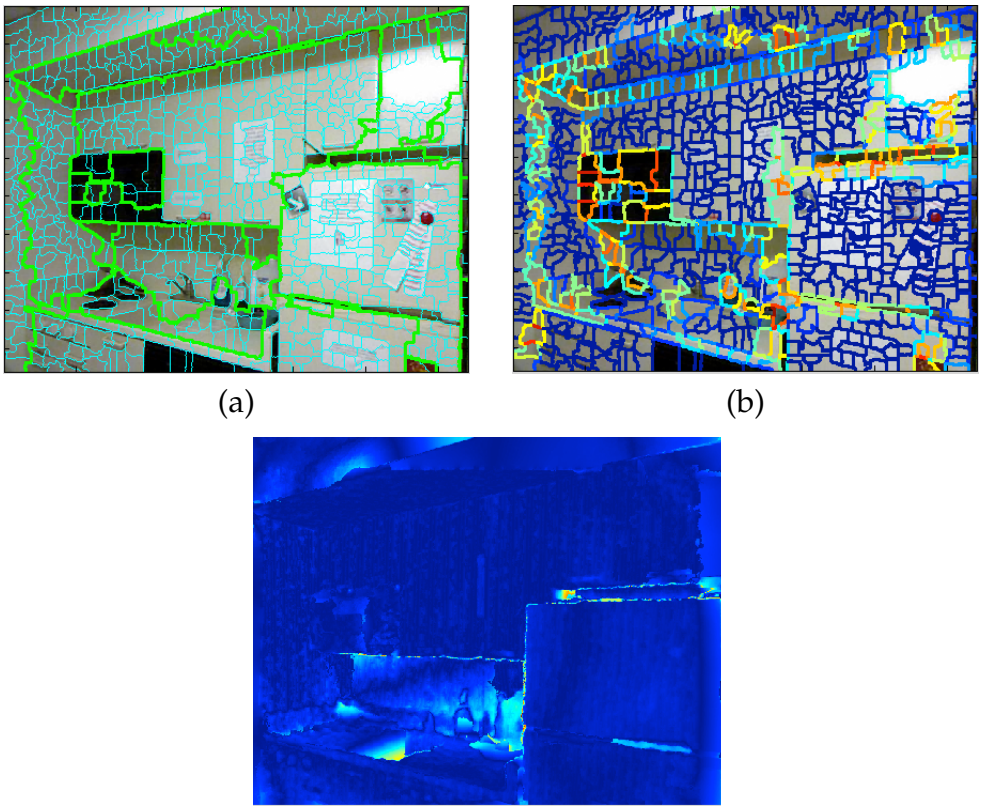
and form a vector $\vec{v}_e = p_e - p_s$. Then each pixel p_i along the edge is connected to the starting pixel p_s, which forms a vector $\vec{v}_i = p_i - p_s$. We calculate the angles θ_i between $\vec{v}_e$ and $\vec{v}_i$. We assign a positive angle value if $\vec{v}_i$ lies on the right side of $\vec{v}_e$, and then θ_i has the range from $-\pi$ to π. We form a histogram of these angles with 36 bins from $-\pi$ to π, and use this histogram as the feature that describe the edge curvature. In general, homogenous boundaries show more uniform angle histograms than occlusion or connected boundaries.

4.3.2 Depth features

We first perform the surface segmentation on the depth data, and then introduce our 3D features x_d computed based on surface segmentation and fitting.

Surface segmentation and fitting: We applied the surface segmentation and fitting algorithm proposed in [11]. The intuition is to cluster the sparse point clouds by their Euclidean distance and estimated surface normals, and then apply surface fitting to refine the segmentation result. Exemplar results are shown in Fig. 4.2 (c). After this step, for each pixel p_i and its 3D points P_i, we have acquired its 3D surface group C_i, and the corresponding surface function $f_{C_i}(x, y, z)$.

Surface segmentation can give a rough prediction for occlusion boundaries: if one edge lies between two different surfaces or on the edge of a surface in 3D, it is more likely to be an occlusion or connected boundary. However, some new issues emerge: the clustering step can also introduce errors from over/under segmentation. Therefore, we propose a set of depth features based on this surface segmentation result.



(a) (b)

(c)

Figure 4.3: (a) Occlusion boundaries labeled from the surface segmentation algorithm (section: **Surface segmentation label**). (b) Surface label distribution on each edge. (c) Surface fitting errors on each pixel.

Surface segmentation label: this simply uses the result from surface segmentation algorithm [11] to predict boundaries: for each edge e and its two segments A_e, B_e, we find the most frequent surface labels of the pixels within each segment, $C(A_e)$ and $C(B_e)$. If edge e lies on two different surfaces, we mark it as positive to indicate an occlusion or connected boundary, otherwise we label it negative to indicate a homogenous boundary. Fig. 4.3 (a) shows the labeling result from this method. In our experiments, although this method gives a better performance compared to the naive edge detection in the depth image, in complex scenes, many boundaries are still mistakenly labeled. Because of this, we treat this labeling as one-dimension feature for boundaries classification.

Surface distribution: for the segment A_e, B_e that edge e lies in between, we also retrieve the 3D surface label distribution for each segment, and include this as another feature.

For one segment, we calculate the ratio between the occurrence of the most frequent surface label C_{max} and the total number pixels. For example, if in segment A_e, 90% of its pixels belong to surface C_1, then the feature value for this segment will be $sd(A_e) = 0.9$. This feature effectively measures the confidence of the previous surface segmentation algorithm. We compute this feature on an edge basis by taking the average of the surface distribution value of each edge's two segments: $sd(e) = (sd(A_e) + sd(B_e))/2$. Fig. 4.3 (b) gives an example of the surface distribution value for each edge: the more red an edge is, the smaller its surface distribution value is, which indicates less confidence in the surface segmentation.

Fitting error: for each 3D point P, we also retrieve its surface function f_C that P lies on and compute the fit error, measured in 3D space. One example of the fit error distribution is shown in Fig. 4.3 (c), in which the red color indicates higher fitting errors, and the blue color indicates lower ones.

The surface segmentation errors usually occur at occlusion or connected boundaries, where a clear segmentation is harder and the surface function has a worse fit. Thus, the distribution of the fit errors gives a strong clue about the type of the boundary, e.g. for occlusion boundaries, the 3D points may have larger fitting errors than the points that lie on a connected boundary, because there is a large depth change from the occlusion. A connected boundary indicates the place where two surfaces are touching, and thus may have a smaller fit error for its 3D points. For a homogenous boundary, the fit errors of its 3D

points should be still lower.

We compute two types of fitting error for each edge e and its surfaces A_e and B_e: the pixel-wise fit errors along the edge and within each segment. We histogram the error distribution into 40 bins with equal intervals in log space from 0 to 10 centimeters, and use this as one of the depth features.

Neighboring surface difference: we compute two types of differences between edge e's segment A_e and B_e: (a) average depth difference, and (b) angle between the surface normals.

The average depth difference is straight forward: we first compute the average depth of segment A_e and B_e, and then calculate their difference. This value can help boundary classification, because occlusion boundaries may result in higher depth difference between their two sides, while connected and homogenous boundaries may expect lower values.

Furthermore we compute the angle between the surface normals for segments A_e and B_e. Since the segment here are super-pixels from a dense over-segmentation, we approximately fit a plane locally for the 3D points with each segment, and calculate the angle between their normals. The intuition is as follows: the two segments of a connected boundary may have an orientation difference around 90°. However, the occlusion and homogeneous boundaries tend to have their neighboring segments facing similar directions. Therefore the orientation difference of the neighbor surfaces can also help us for boundary classification.

4.4 Conditional Random Field

We propose a Conditional Random Field for a joint inference of boundaries. Given the initial over-segmentation, oo classify each edge e we define the unary potential, $\phi(y_i|x_i)$, and the pairwise potential $\psi(y_i, y_j|x_{i,j})$. y indicates the edge labels, e.g. homogenous or occlusion/connected boundaries, and x indicates the feature vector. i and j refer to the neighboring edges. Then the task is to minimize the following energy function E:

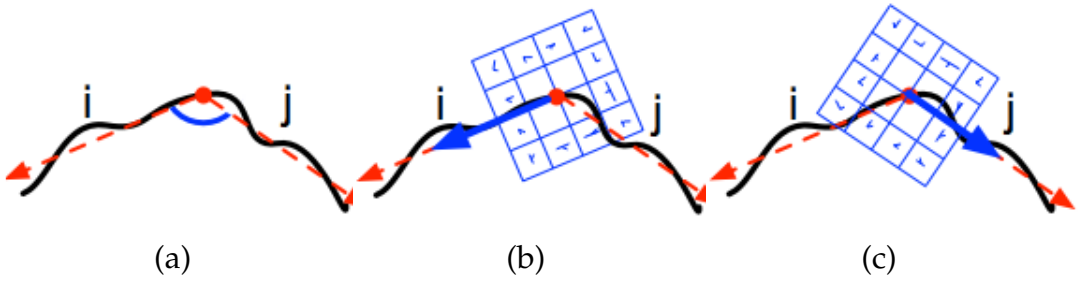
$$E = \sum_i \phi(y_i|x_i) + \sum_{i,j} \psi(y_i, y_j|x_{i,j}). \tag{4.1}$$

4.4.1 Unary potential

Since our color and depth features are computed on edge basis, we can concatenate them into one feature vector $x = [x_c, x_d]$, and train a Support Vector Regression f_u for the local prediction. We use linear SVM regression for fast training and testing speed. After that, we retrieve the probability $P(y|x)$ of the edge label y given the feature x, using the regression f_u, and use the negative log likelihood of this probability as the unary potential $\phi(y|x)$ in our CRF.

4.4.2 Pairwise potential

We learn the pairwise potential ψ for any two neighboring edge i and j that connected in the color image, meeting at a junction with position p_{jun}. First, we concatenate both color and depth features from edge i and j: $x_i = [x_{c,i}, x_{d,i}]$ and $x_j = [x_{c,j}, x_{d,j}]$. This serves as the basic feature set to learn the pairwise potential.



(a) (b) (c)

Figure 4.4: Additional pairwise features for edge i and j in the color image for learning pairwise potentials. Edge i and j are in solid black lines, and the edge directions are plotted with red arrows. The meeting junction p_{jun} is the red dot at center. (a): angle difference $\theta_{i,j}$ (blue half circle) between two edges. (b) and (c): oriented SIFT features aligned with the direction of each edge direction.

Furthermore, we develop additional features to describe the neighboring edge relation.

Edge direction: first, we estimate the direction of an edge by fitting a line to the pixels along this edge, and form a vector pointing outwards from the meeting junction p_{jun}. Fig. 4.4 (a) gives an illustration of this estimation, and the edge directions are plotted with red arrows.

Angle difference: we calculate the angle difference $\theta_{i,j}$ between the direction of neighboring edge i and edge j, shown in Fig. 4.4 (a) as a blue half circle. The intuition is that if one occlusion boundary meets another occlusion boundary, it is very likely that they lie on a straight line, especially for man-made structured objects. This should follow a continuous boundary of the object. It is the same case for two connected boundaries as neighbors. However, if one occlusion/connected boundary meets a homogenous boundary, then the angle difference can be of arbitrary value.

Oriented SIFT: different types of boundaries will give different texture shapes

at the meeting junction, and we compute a SIFT descriptor at the junction to capture such information. The underlying idea is as follows: if two edges are both occlusion/connected boundaries, then the SIFT descriptor will have a consistent large value along the boundary direction. In contrast, homogenous boundaries produce texture of random and irregular patterns, and lead to a more uniform distribution for each bin value in the SIFT descriptor. Therefore this descriptor can provide additional texture information at the junction where edges meet. Besides that, In computing the features, SIFT descriptors use a histogram approach, which can tolerate some the noise in the boundary as well as a little mis-alignment of the depth image.

We compute this feature as follows: the SIFT descriptor is centered at the meeting junction position p_{jun}, and aligned with the direction of each edge. Then we compute a fixed size (5 pixels per bin) SIFT descriptor for each edge on both the color (converted into gray scale to follow the convention of SIFT) and depth image. After that, we concatenate the descriptors on different image domains. This forms the oriented SIFT feature x_s to learn pairwise potentials.

Training and testing pairwise potentials: we combine all the previous features including the color and depth features for each individual edge, x_i, x_j, the angle difference between two edges, $\theta_{i,j}$, and the Oriented SIFT feature at the meeting junction x_s, and form them into one feature vector $x_{i,j}$. We use this final feature to train the pairwise potential $\psi(y_i, y_j|x_{i,j})$.

For training, we simplify the learning phase of CRF by training an individual classifier for each pair of labels. For example, given any two neighboring edge i and j, suppose their ground-truth labels are $y_i = 1$, and $y_j = 0$ (1 indicates the occlusion boundary, and 0 the homogenous boundary). Then their

feature $x_{i,j}$ is used as a positive training instance for the pairwise label $(1, 0)$, and a negative training instance for all the other label pairs. If we limit our labeling space to occlusion boundaries 1 and homogenous boundaries 0, this gives four pairwise potentials regressions: $f_{p,(0,0)}$, $f_{p,(0,1)}$, $f_{p,(1,0)}$, and $f_{p,(1,1)}$. During testing, we feed the pairwise feature $x_{i,j}$ to all the regressions, convert the outputs into pseudo-probabilities, and use the negative log values as the pairwise potentials $\psi(y_i, y_j|x_{i,j})$ between two neighboring edges i and j.

4.5 Active Learning

We use an active learning approach to decrease the amount of the labeling work. In this step, we initialize the learning framework with very few training instances, and update the classifiers by wisely selecting additional training instances. In this way, we can achieve the same performance with fewer training data, and thus decrease the workload of labeling the ground-truth.

Initially, we have the training set $X_{tr} = \{x_i, x_{i,j}\}$ without their ground-truth labels Y_{tr}, and want to actively pick out the training instances for a human to label. First, we randomly use a subset $X_0 \subset X_{tr}$ and retrieve their ground-truth labels from human. Then the training set X_{tr} is decreased into a remaining training set $X_{tr,0} = X_{tr} - X_0$. At each step t, we select K training instances out of the remaining set $X_{tr,t}$ and merge them to form the new training set X_{t+1}.

Training instances selection is made based on the classification margin. We apply the current classifiers to the remaining training set $X_{tr,t}$, and select the least confident ones for a human to label. For our CRF model, there are two types of classifiers to update: the unary potential f_u, and the pairwise potential f_p. For

the unary potential, we test f_u on the current remaining set $X_{tr,t}$, and find the instances with the least margin (close to 0.5 probability for a binary occlusion boundary classification). We sort the margin in increasing order, pick out the top K instances as the least confident instances, and ask for the ground-truth labels. After that, we merge them with the previous training instance to re-train f_u.

For the pairwise potential, we have a set of classifiers f_p. We use the entropy as the criteria to actively select the training instances. At step t, for each instance $x_{i,j}$ in the remaining set $X_{tr,t}$, we apply the current pairwise classifiers f_p and compute the probability for all the possible pairwise labels: $P(y_i, y_j|x_{i,j})$. Then the entropy is computed as:

$$S(x_{i,j}) = -\sum_{i,j} P(i, j|x_{i,j}) \log(P(i, j|x_{i,j})). \tag{4.2}$$

We sort all the entropies in decreasing order, and select the instances with top K large entropies to retrieve the human labels and update the pairwise classifiers.

4.6 Experiments

To evaluate the effectiveness of our proposed feature set and learning framework, we compare our final proposed approach (**crf**) with the following algorithms:

base: uses the color and texture features proposed in [53]. This serves as the basic feature set for color image boundary detection (no depth). For the following algorithms, we add different feature sets to this **base** approach, e.g. the following approaches are feature set in addition to **base**.

ec: (edge curvature) in addition to **base**, we add the edge curvature feature. Also this method is solely RGB features (no depth).

The following variants incorporate depth features:

sl: (surface label) this directly uses the surface segmentation algorithm proposed in [11] in addition to **base**. We incorporate the surface segmentation label as one additional depth feature.

sd: (surface distribution) we add the surface distribution feature set.

se: (segment fitting error) the histogram of the the fitting errors within each segment.

ee: (edge fitting error) the histogram of the fitting errors along each edge.

nd: (neighbor difference) the neighboring surface difference features.

all: we combine all the previous feature sets into one feature vector for boundary classification.

crf: final CRF that uses both the unary potential learned from the feature set **all**, and the pairwise potential learned from the proposed pairwise features.

We experiments on two different datasets: Kinect depth-order dataset collected by ourselves, and the public NYU Knect dataset of indoor scenes [55].

4.6.1 Depth order dataset

To produce this dataset, we place different objects in a distinct depth order, and on a supporting surface like a table or the ground. Color and depth images

are collected using Microsoft Kinect Sensor. In total there are 200 image pairs including four different scenes with various common objects. Examples are shown in Fig. 4.5. We manually label all the occlusion and connected boundaries in the scene. This dataset and the ground-truth boundary labels will be released to the public. We split the dataset into two halves for separate training and testing.

We evaluate different algorithm by comparing the average precision of detecting boundaries, and present the results in Fig. 4.6 (a) and (b). Overall, it proves that our proposed framework works for both occlusion and connected boundary detections. Without depth information, using the base features from [53] provides a lower bound on performance, and our edge curvature feature still improves by around 3% performance in average precision.

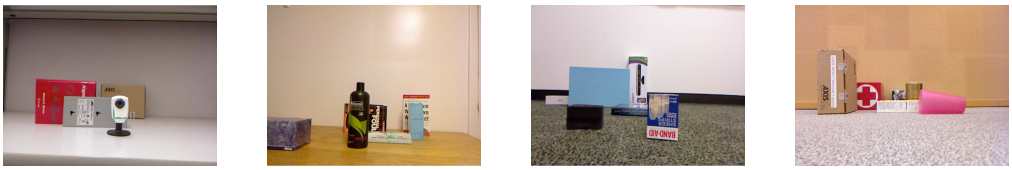


Figure 4.5: Example images of the kinect depth order dataset.

Adding depth features definitely help the tasks. Directly using the surface segmentation in [11] **sl** gives 6% boost for classifying connected boundaries, and 8% for occlusion boundaries. Besides, our proposed depth feature sets (**sd**,**se**,**ee**,**nd**) also help and generate better result than **base**, giving around 70% to 80% average precisions. When combining all the feature sets (**all**), it outperforms the individual feature set by a large margin, leading to an average precision of nearly 90% for both occlusion and connected boundary detection. Compared to the individual depth features (blue columns from **sl** to **nd**), the combined one (**all**) achieves at least a 10% improvement.

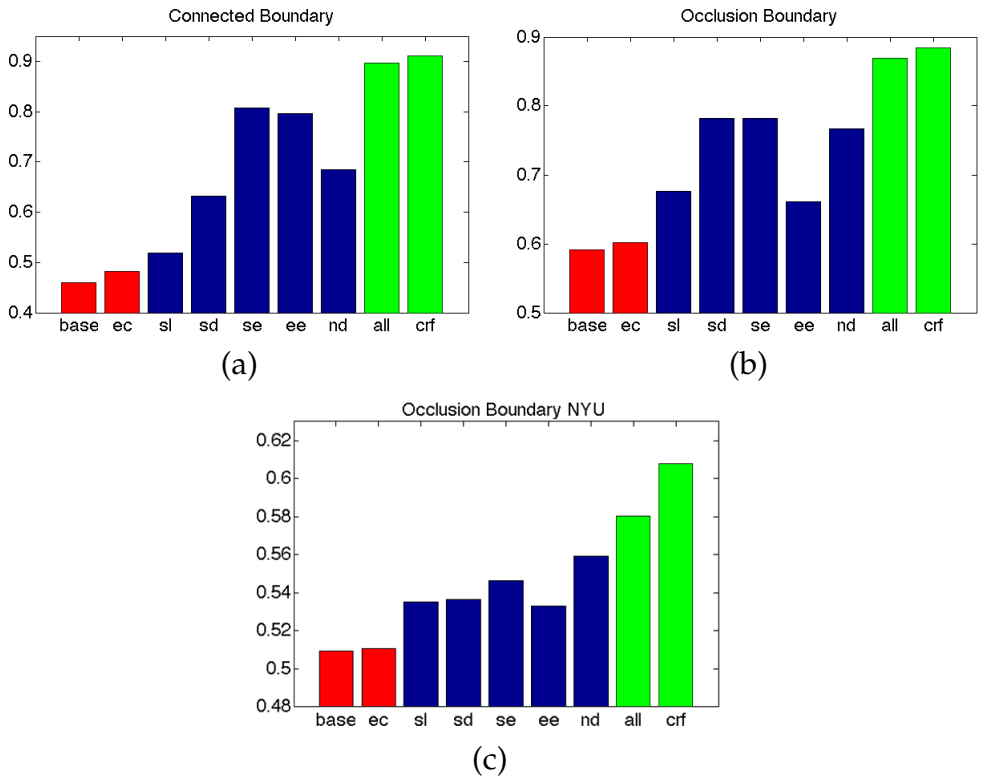


Figure 4.6: Average precision (y-axis) for different approaches (x-axis) on our kinect depth order dataset: (a) connected boundary. (b) occlusion boundary. (c) occlusion boundary detection result on NYU depth dataset. Red: color only feature set. Blue: adding individual depth feature sets. Green: the final combined approach (**all** and **crf**).

Finally, our proposed CRF model still improves the performance 2% compared with **all**, and gives the best result of all the approaches, because it encourages continuity between boundaries. Some example images of our boundary detection results using **crf** are shown in Fig. 4.7. It shows that our learning framework reliably identify both occlusion and connected boundaries in different scenarios.

We report the performance of our active learning approach by comparing with the baseline of randomly selecting the training instances. The results for

Figure 4.7: Boundary detection result using the proposed algorithm. It reliably detects the connected (left two) and occlusion (right two) boundaries in different scenarios.The color indicates the confidence in classification. The more red it is, the larger the belief.

this depth order dataset are shown in Fig. 4.8 (a) and (b). We keep the same testing set as the previous experiment, but use only 5% training instances initially, and add 5% at each step. This leads to 20 steps in training instances selection. We can see that for detecting occlusion and connected boundaries, the proposed active learning approach reaches the maximum plateau with less than 4 steps, using only around 20% of total training instances. This proves the effectiveness of our feature set and the active learning approach.

4.6.2 NYU dataset

We also experiment on the public NYU depth dataset [55]. This dataset only provides the object segmentation, and we approximately use it as the occlusion boundary to fit our task.

This dataset contains 2284 frames of Kinect image pairs with some human labeled object boundaries. However, many of them are of the same scene and near consecutive frames in a video. Therefore, we sample the dataset into 600 images, ensuring the remaining images are not too similar to each other. After that, we follow the same settings as the previous experiments. The NYU depth dataset is split into two halves for separate training and testing.

For this dataset, we compare with the baseline approaches and quantitatively evaluate the average precision in Fig. 4.6 (c). Our proposed edge curvature feature can improve the performance over the baseline color feature. The proposed depth feature sets (blue columns) show the benefit of bringing the depth information. They achieve around 55% in average precision, and all outperform the color-only scheme by 2% to 6%. The final combined CRF model gives the best performance, achieves near 10% absolute boost from 51% to 61% comparing to **base**, and has 5% improvements in average precision to the individual depth feature sets.

The overall performance in this dataset is lower. We believe this is due to the complexity of the scene in this dataset. Our feature sets are computed quite locally, and may not be able to well capture too many different structures. However the relative contribution of our CRF model is larger, and overall the detection results agree with the ground-truth in general. Some results are shown in Fig. 4.9.

We compare our active learning approach with random selection on this dataset in Fig. 4.8 (c), with the same experiment setting as before. Once again, our proposed approach also gives a better detection result with the same amount of training instances, and reaches the maximum performance using

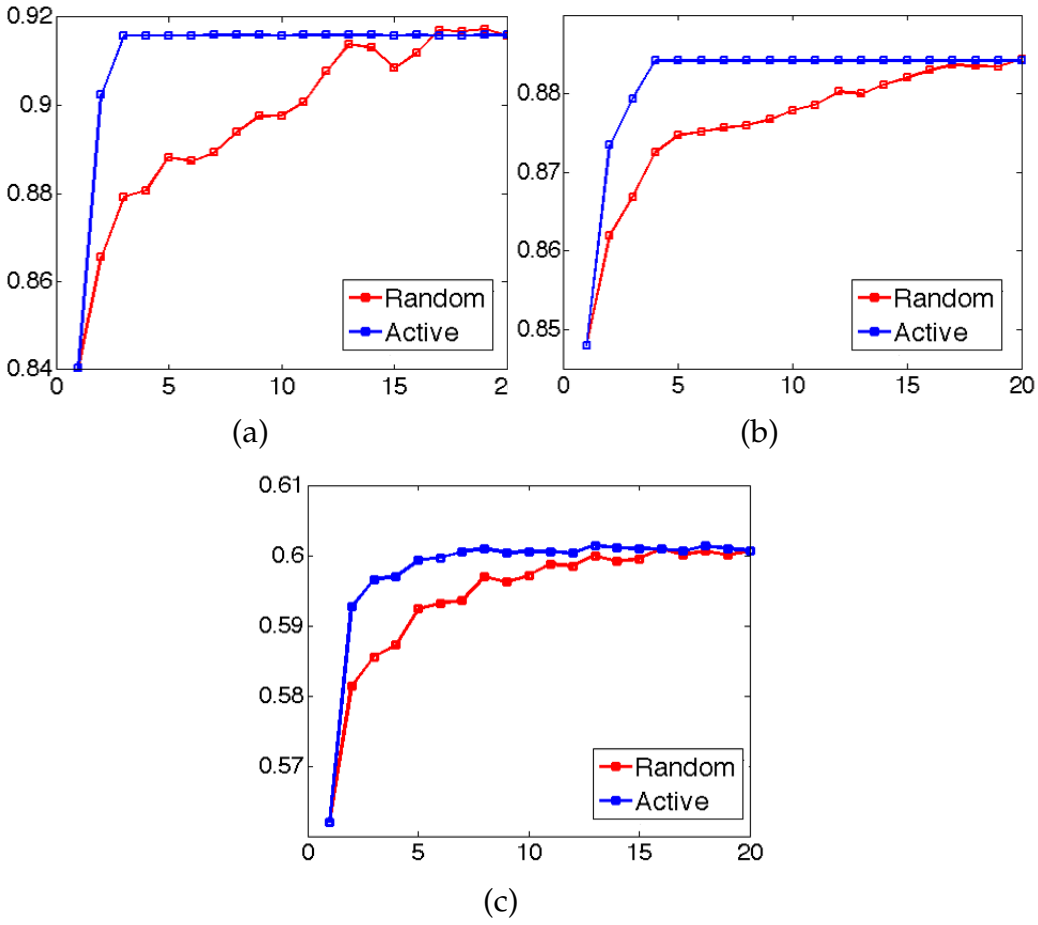


Figure 4.8: Active learning results. X-axis: step from 1 to 20. Y-axis: the average precision of detection for testing. Blue lines: the proposed active learning scheme. Red lines: randomly selecting the training instances. (a) to (c) are different tasks: (a) connected boundary on the depth order dataset. (b) occlusion boundary on the depth order dataset. (c) occlusion boundary on the NYU depth dataset.

around 20% of the selected training instances.

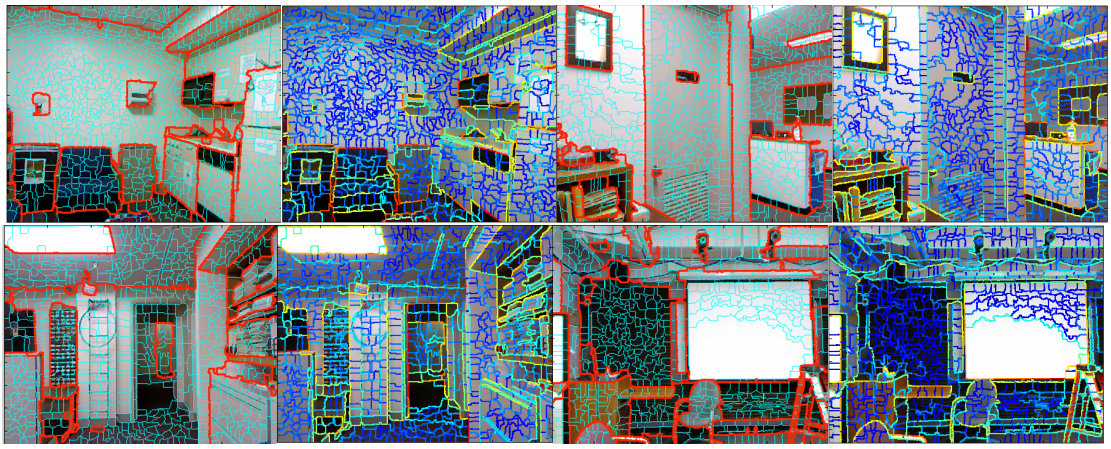


Figure 4.9: Experiment results on NYU dataset. Ground-truth labels are on the left, with red indicates the occlusion boundaries, and cyan indicates the homogenous boundaries. The testing results are shown on the right. Heat map indicates the belief: the more red an edge is, the more likely it is an occlusion boundary.

4.7 Summary

As the types of imaging modalities increase, it will be important to combine various types of data to solve vision problems. This part of the thesis demonstrates a solution for classifying image boundaries from color and depth that is significantly improved over using one or the other type of information exclusively. We first perform surface segmentation on the depth data, and generate a set of novel depth features based on the surface. After that, we propose a CRF framework for a joint inference on boundaries, and an active learning scheme for selecting the training instances. Experiments show that our proposed feature sets and the learning framework outperform the baselines. For further work, one possible approach can be a hierarchical multi-level CRF model, which incorporates features on larger region after merging segments. Other applications, such as scene understanding and object recognition, can also be built on top of this work.

CHAPTER 5

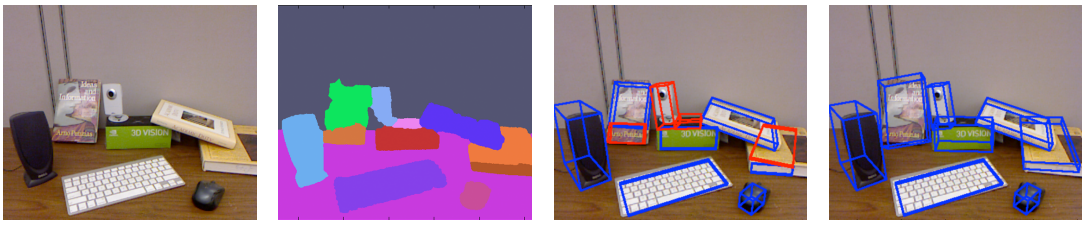
3D VOLUMETRIC REASONING

5.1 Introduction

3D reasoning is a key ingredient for scene understanding. A human perceives and interprets a scene as a collection of 3D objects. Rather than groups of 'flat' color patches, we perceive objects in space with perspective. In static scenes, we understand that objects occupy volumes in space, are supported by other objects or the ground, are typically stable (i.e., not falling down or toppling), and occlude farther objects. These physical properties are usually not considered in traditional object recognition.

In this thesis, we propose a framework for 3D segmentation and scene reasoning with volumetric blocks that incorporates the physical constraints of our natural world. Our algorithm takes RGB-D data as input, performs 3D box fitting of proposed object segments, and extracts box representation features (such as box intersection and stability inference) for a physically-based scene reasoning. Our final output is the object segmentation of the scene, and its block representation (shown in Fig. 5.1 (d)).

Past works for producing 3D interpretations represent the world as a "pop-up" model [65], as point-wise depth-grid [66], as piece-wise planar segments [1, 67], or as blocks constrained to rest on the ground [68]. However, inferring a 3D interpretation is only part of the picture, a good scene interpretation should also follow physical rules: assuming the image captures a static scene, objects should be placed stably. If we attempt to segment the scene purely based on appearance



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

Figure 5.1: (a) The input RGB-D image. (b) Initial segmentation from RGB-D data. (c) A 3D bounding box is fit to the 3D point clouds of each segment, and several features are extracted for reasoning about stability. Unstable boxes are labeled in red. (d) The segmentation is updated based on the stability analysis and produces a better segmentation and a stable box representation.

or shape, we may end up with segmentations that do not make physical sense, as shown in Fig. 5.1 (b). Reasoning about stability brings physics into our model, and encourages more plausible segmentations and block arrangements, such as the example presented in Fig. 5.1 (d).

The challenge is that objects can be arranged in complicated configurations. While some recent work considers notions of support (e.g., [15, 18, 68]), they are limited to single support or isolated objects on a flat surface. Although these methods work well on larger structures such as furniture and buildings, they do not apply to more complicated stacking arrangements of objects that can occur, for example, on desks or other cluttered situations.

In our algorithm, we first fit a 3D box to the point-cloud of each segment, and then extract several features for further reasoning about the scene: 1) we define the box fitting error based on the 3D points and box surfaces; 2) we ensure that 3D points lie on the visible surfaces of the boxes given the camera position; 3) we find space violations when neighboring boxes intersect one another; 4) we propose supporting relations and the stability of the scene given the boxes. This

evaluation of the box representation allows us to refine the segmentation based on these box properties through a process whose parameters are learned from labeled training images.

The block representation provides us many useful features, such as the box fitting error and the object stability, and we learn the importance of each feature through supervised learning. We design an energy function to describe the quality of the segmentation given the RGB-D image pairs. By minimizing this energy function value, we achieve a better scene segmentation and volumetric block representation. For minimization, we use a sampling algorithm that incorporates randomized moves including splitting and merging current segments.

We experiment on several datasets, from a synthetic block dataset to the NYU dataset of indoor scenes. We also propose a new Supporting Object Dataset (SOD) with various configurations and supporting relations, and a Grocery Dataset (GD) extended on SOD in order to demonstrate more application scenarios. Experimental results show that our algorithm improves RGB-D segmentation. Further, the algorithm produces a 3D volumetric model of the scene, and high-level information related to stability and support.

To summarize, our major contributions are:

1. A volumetric representation of the RGB-D segments using blocks.
2. The use of physics-based stability for modeling an RGB-D scene.
3. A learning-based framework for inferring object segmentation in an RGB-D scene.
4. New supporting objects datasets including human segmentation labels

and support information.

The rest of the thesis for this topic is organized as follows: we discuss the related work in Section 5.2. An overview of the approach is presented in Section 5.3. After that, we present our approach for single box fitting in Section 5.4, and the features to model the pairwise box relations in Section 5.5. The stability reasoning process is presented in Section 5.6. We introduce our energy function for segmentation in Section 5.7, including the sampling algorithm with splitting and merging. The experimental results are presented in Section 5.8. We conclude the this part in Section 5.9.

5.2 Related work

3D Understanding from Color Image: Object segmentation on a single color image is one of the most studied computer vision problems, and many methods have been proposed, for example, [69], [70], [71], [19] and [72]. These methods group pixels into objects by clues such as color, texture or semantic classification results. They operate on a 2D image, but it is natural next step to incorporate the 3D understanding into object segmentation.

The first attempts for geometric inference from a single color image were proposed in [1, 66] and [67] for estimating the depth of each segment using only color features. Usually, a ground plane is detected, and then the depth of a segment that stands on the ground can be estimated by the touching position. The results appear either as "pop-up images" [65]: segments stand like billboards in different depth layers and have empty space behind them, as a point-wise

depth-grid [66] or as piecewise planar segments [1]. The limitation is obvious: these models do not align with our understanding of the scene, where each object actually occupies a *volume* in 3D, which we explore in this work (Fig. 5.1 (d)).

To overcome this limitation, Gupta et al. [68] propose a block-world representation to fit 2D color segments. Segments in outdoor scenes are represented by one of eight predefined box types representing a box viewed from various positions. Although buildings in these outdoor scenes often fit nicely into one of the block categories, this assumption is not true for general images of stacked objects, where the orientations of objects are not limited to eight. Zheng et al. [73] also use blocks representation for objects, but required interactive human labelings for non-box objects. Xiao et al. [74] detect 3D cuboids with arbitrary orientations solely in RGB images, Bleyer et al. [75] show box fitting for improved stereo, and Jiang et al [76] propose a linear programming for fitting cubiods in depth images. In this work, we use RGB-D data and fit boxes with depth information for volumetric and stability reasoning.

In addition, researchers have studied indoor environment reasoning on color images, where the 3D geometric inference can be approximated as a Manhattan World [77] [78] [79] [80]. Further, the 3D structure of indoor scenes has been studied through affordances, as in [81] [82] and [83]. Indoor images have the strong clues of lines and planes as well as a fixed composition of ceiling, wall and ground. These approaches posit that indoor spaces are designed by humans, so furnature items and objects are arranged in ways to facilitate usefulness of these spaces by humans. These approaches are complementary to ours.

RGB-D Scene Understanding: Previous work has shown that integrating depth

with color information improves the performances of many vision tasks, such as segmentation (in [18]), contour detection (in [84]), object recognition (in [17], [85], and [16]), scene labeling (in [55], [63], [86] and[87]), and activity detection (in [88], [14] and [89]). These algorithms usually treat depth as another information channel without explicitly reasoning about the space that an object occupies. For example, when an object is partially observed from a single viewpoint, it remains hollow inside. In this way, segmentation and supporting inference are transformed into a classification problem in a 2.5D space. In contrast, we explicitly reason about full 3D models by fitting boxes to objects. This leads to a more natural interpretation of the scene, facilitated by better segmentation and support inference.

Support and Stability: Grabner et.al. [90] analyze the interaction between humans and objects such as chairs in 3D space. The algorithm finds object support, and shows that a 3D model can predict well where a chair supports the person. This also helps chair detection. However, in this thesis, we perform a more general analysis of the 3D objects in the scene through box fitting and stability reasoning.

Jiang et al. [15] [91] reason about stability for object arrangement, but their task is different from ours: given a few objects, their goal is to *place* them in the environment stably.

In other recent work, Silberman et al. [18] identify which image segments support which other segments. However, reasoning about support and stability are two different things. Past work on support pre-supposes that segmentations are already stable, and implicitly assumes that all regions need only one region to support them, without checking any physics-based model of stability. We

use stability reasoning to verify whether a given volumetric representation of a scene could actually support itself without toppling, and adjust the segmentation accordingly.

In concurrent work, Zheng et al. [92] reason about stability in a depth image. They use geometric primitives, including voxels, to represent object volumes, and merge together neighboring voxels until stability is achieved. Their approach focuses only on the depth domain. In contrast, our work fuses both color and depth features. We model each object with cubic volumes and combine this representation with color information for reasoning about support, stability and segmentation in one framework.

We use a simple model for evaluating the stability of our block arrangements, although more complicated physics-based simulators [93] could be employed. One approach could be to consider all possible reasonable segmentations, and plug each into a simulator. However, this would result in an exponential number of evaluations, and would still be susceptible to noise and other unknown physical parameters (e.g., coefficients of friction). Our approach for stability evaluation is based on a simple Newtonian model: the center of gravity of each adjacent object subset must project within its region of support. This simple model is justified by the ideas of intuitive physics [94] that humans even have a sense of stability at a glance. Our algorithm is not a perfect reflection of the physical world, but it is accurate enough to achieve our goal of improving parsing 3D scenes.

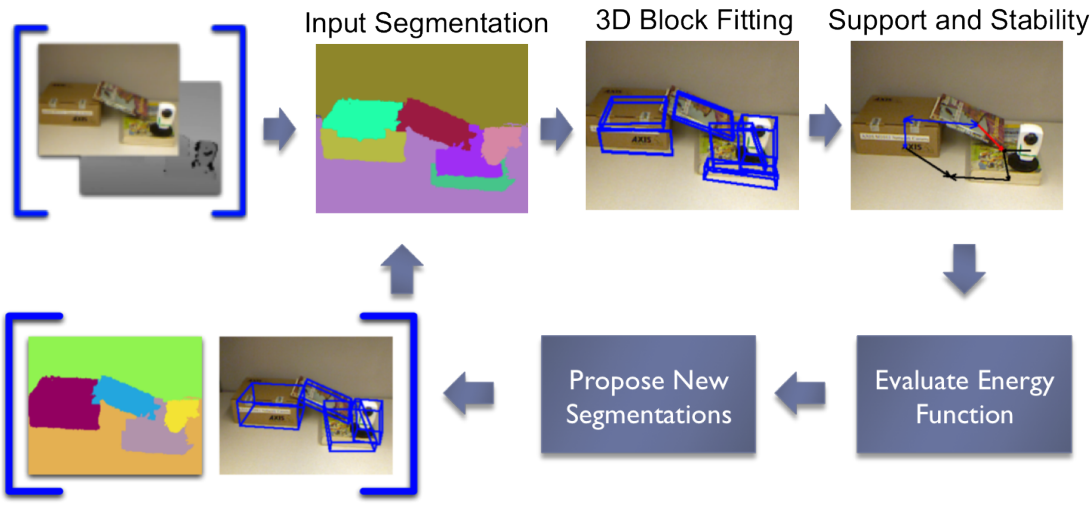
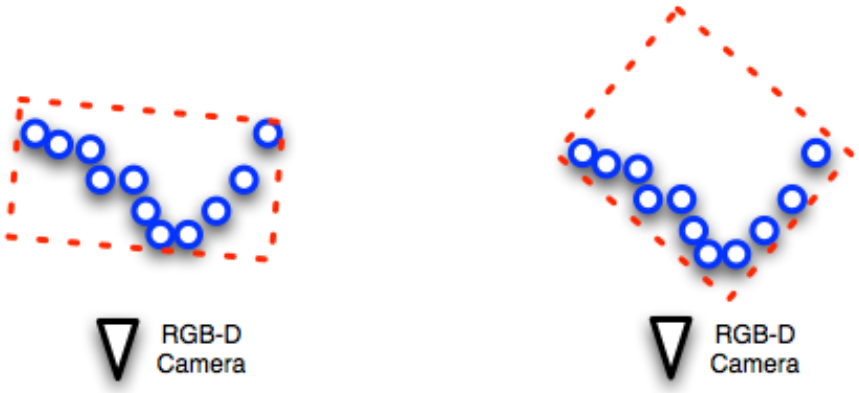


Figure 5.2: An overview of our algorithm.

5.3 Approach Overview

Our input is an initial RGB-D segmentation, generated from an algorithm proposed in the literature [18]. First, we fit a 3D bounding box to the 3D point-cloud points corresponding to each segment. Next, we compute features for single boxes and between pairs of boxes and propose supporting relations, perform stability reasoning, and adjust the box orientation based on the supporting surfaces. Finally, we model the segmentation with an energy function based on learned regressors that are trained using these features. The segmentation is optimized by minimizing this energy function using randomized splitting and merging. The output is the segmented RGB-D image along with volumetric representation using the fitted boxes and support information. See Fig. 5.2 for an overview.



(a) (b)

Figure 5.3: (a) A bounding box fit based on minimum volume may not be a good representation for RGB-D images, where only partially observed 3D data is available. (b) A better fit box not only occupies a small volume, but also has many 3D points near the box surface. Data points are projected to 2D for illustration.

5.4 Single box fitting

In this section, we describe the procedure for representing a segment from an RGB-D image with a box. RGB-D data is observed from only one viewpoint, and fitting 3D bounding boxes with minimum volumes [95] may fail to produce box representations that align well with the actual objects in the scene. Fig. 5.3 (a) gives an illustration. A minimum volume box covers all the data points but might not give the correct orientation of the object, and fails to represent the object well. A well-fit box should have many 3D points near box surfaces, as shown in Fig. 5.3 (b).[1] We propose a RANSAC-based algorithm (details below) to fit boxes to the point cloud.

[1]Recent related work [96] considered cylinder fitting of 3D points to the surface but also did not consider visibility.

(a) (b)

Figure 5.4: (a) To fit the 3D points, we use RANSAC to find the first plane S_1. (3D points are projected on 2D for a simpler illustration, and the plane S_1 is presented as red line). (b) For the 3D points that do not belong to S_1, we fit another plane S_2 to them, enforcing that S_2 is perpendicular to S_1.

5.4.1 Minimum surface distance

The orientation of a 3D bounding box is determined by two perpendicular normal vectors (the third normal is perpendicular to these two vectors). The idea is to find the two principle orientations of the 3D bounding box so that the 3D points are as close as possible to the box surfaces. Given a set of 3D points $\{P_i\}$ and a proposed 3D box, we calculate the distance of each point to the 6 surfaces of the box, and assign each point to its nearest-face distance $\{D_{min}(P_i)\}$. The objective for our box fitting algorithm is to minimize this sum for all the 3D points: $\sum_i D_{min}(P_i)$.

The input to this step is the 3D points within one segment. First, we use RANSAC to find a plane to fit all the 3D points, providing the first surface S_1, shown in Fig. 5.4 (a). Next, we collect the outlier 3D points that do not belong to S_1, and then fit a plane, S_2, to them also using RANSAC. We constrain that the surface orientation of S_2 is perpendicular to S_1, shown in Fig. 5.4 (b).

The above steps give the orientations that align with many points. The mini-

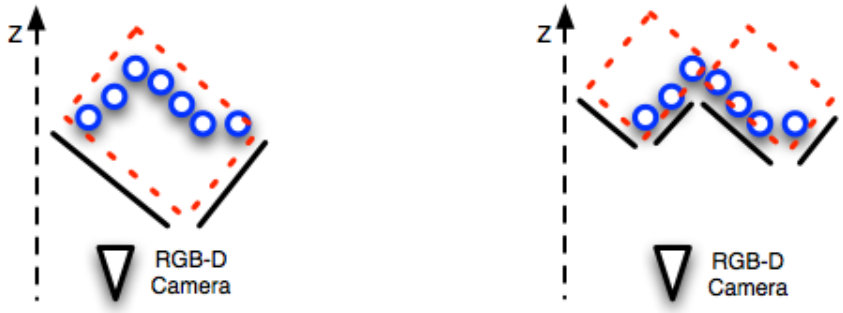
mum volume is determined by finding the extent of the 3D points given the box orientation. Note that there are usually noisy depth points: If a segment mistakenly includes a few points from other segments in front or behind, a large increase of the box volume can occur. Therefore, we allow for up to 5% outliers in the 3D points, requiring that $\geq$ 95% of a segment's 3D points are enclosed within its box.

With the final 3D bounding box, the sum of the minimum surface distance of the point, $\sum_i D_{min}$, is calculated. The whole process is repeated several times and the best fitting box (smallest distance $\sum_i D_{min}$) is chosen.

5.4.2 Visibility

We identify the box surfaces that are visible to the camera. If the objects in the scene are mostly convex, then most 3D points should lie near visible box surfaces instead of hidden faces.

Fig. 5.5 illustrates the visibility feature for our box fitting. Surface visibility is determined by the position of the camera center and the surface normal. We define the positive normal direction of a surface as the normal pointing away from the box center, and then a surface is visible if the camera center lies at its positive direction. Each box has at most three visible surfaces. We compute the percentage of the points that belong to visible surfaces, and use this as the feature for later processing.



(a) (b)

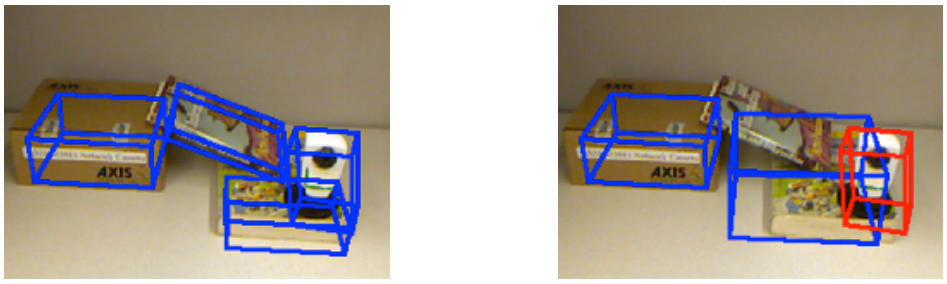
Figure 5.5: Given the camera position and a proposed bounding box, we determine the visible surfaces of the box, shown as a solid parallel black line to the box surface. (a) This box may give a compact fit, but most of the points lie on the hidden surfaces. (b) With a better box fit, most of the points lie on the visible surfaces of the two boxes.

5.5 Pairwise box interaction

We examine two pairwise relations between nearby boxes: box intersection and box support. These features are important because they encode agreement between neighboring segments and provide additional clues for refining the box representation.

5.5.1 Box intersection

Box intersection gives an important clue for volume reasoning. Ideally, a box fit to an object should contain the object's depth points, and not intrude into neighboring boxes. If a proposed merging of two segments produces a box that intersects with many other boxes, it is likely an incorrect merge. An example is shown in Fig. 5.6.



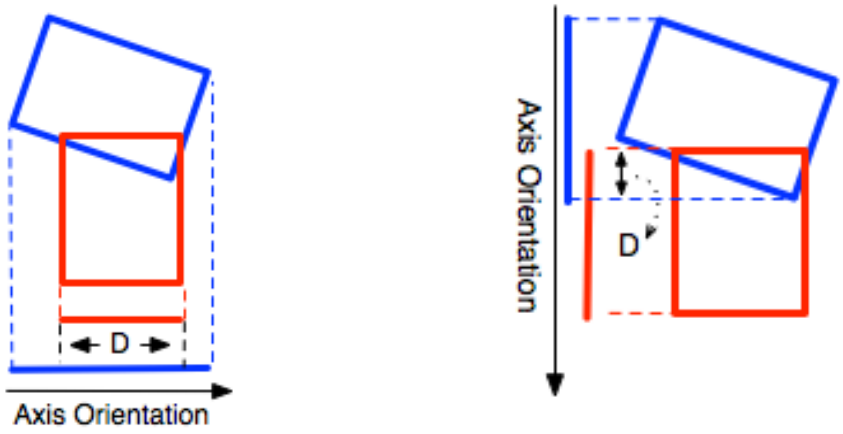
(a) (b)

Figure 5.6: (a) Well-fit boxes should not intersect much with neighboring boxes. (b) If two segments are merged incorrectly, e.g., the two books in the image, then the new box fit to the segment is likely to intersect with neighboring boxes, e.g., the box shown in red.

We explicitly compute the box intersection, and the minimum separation distance between box pairs and direction. Since 3D bounding boxes are convex, we apply the Separating Axis Theorem (SAT) [97], used in computer graphics for collision detection. We present a 2D illustration for finding the distance of the box intersection in Fig. 5.7. The distance D shown in Fig. 5.7 (b) is the minimum moving distance to separate two intersecting boxes.

Extending this algorithm to 3D bounding boxes is straight-forward: since three surface orientations of a box are orthogonal to one another, we examine a plane parallel to each surface, and project the vertexes of the two boxes to this plane. We compute the convex hull of the projection of each box, checking whether the two convex hulls intersect to find the minimum separating distance D.

This process gives both separating distance and the orientation θ_{sep} to separate the two boxes with the minimum distance. θ_{sep} is used when determining the pairwise supporting relations between boxes. For non-intersecting boxes, we choose the orientation and the distance that maximally separate the two



(a) (b)

Figure 5.7: Separating Axis Theorem in 2D: (a) in order to separate two boxes, we rotate the axis perpendicular to any of the edge, and project all the vertices to this rotated axis. (b) If two bounding boxes are separate, there exists an axis that has a zero overlap distance (*D* in the image). We examine all the possible axis rotations (in this case four possibilities), and choose the minimum overlap distance. This gives the orientation and the minimum distance required to separate two boxes.

boxes as their intersection features.

5.5.2 Box supporting relation

In order to address various object-object support scenarios, we define three supporting relations between the boxes: 1) surface on-top support (an object is supported by a surface from below); 2) partial on-top support (an object is tilted and only partially supported from below); 3) side support. Examples are shown in Fig. 5.8 (a) to Fig. 5.8 (c).

To classify supporting relations, we detect the ground and compute the ground orientation following [18]. We define the 3D axis as the follows: the

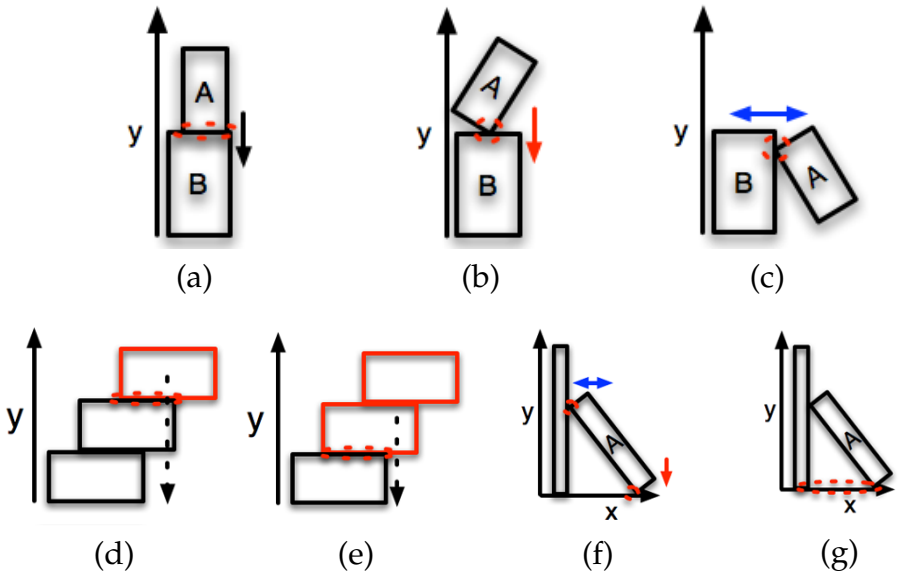


Figure 5.8: (a) to (c): three different supporting relations: (a) surface on-top support (black arrow); (b) partial on-top support (red arrow); (c) side support (blue arrow). Different supporting relations give different supporting areas as plotted in red dashed circles. (d) to (e): stability reasoning: (e) considering only the top two boxes, the center of the gravity (in black dashed line) intersects the supporting area (in red dashed circle), and appears (locally) stable. (e) When proceeding further down, the new center of the gravity does not intersect the supporting area, and the configuration is found to be unstable. (f) to (g) supporting area with multi-support: (f) one object can be supported by multiple other objects. (g) The supporting area projected on the ground is the convex hull of all the supporting areas.

xz-plane is parallel to the ground plane, and $y = -1$ is the downward gravity vector. We align the point-cloud with this axis.

Given the box representation of the scene, we classify pairwise supporting relations with the following set of rules: 1) we use the separating orientation θ_{sep} to distinguish between "on-top" support and the "side" support: an "on-top" support has a separating direction nearly parallel to y axis ($< 20°$), while

the "side" support has a separating direction close to parallel to the xz-plane (ground plane); 2) for "on-top" supporting relations, there are two possibilities: an even on-top support, shown in Fig. 5.8 (a), and a tilted on-top support, shown in Fig. 5.8 (b). We distinguish these two types by examining the two closest surfaces of the pairwise boxes. If these two surfaces have a large angle difference ($> 20^{\circ}$) with each other, and have different orientations to the ground plane, then it is classified as a partial "on-top" support, i.e., the object on top is tilted. Otherwise it is a "surface on-top" support.

Reasoning about stability requires that we compute centers of mass for object volumes, and determine areas of support (i.e., regions or points of the object that are supported, either on side or beneath). Stability requires that the projection of the center of mass of the object along the gravity vector falls within the region of support. We use an object's supporting relation to find the supporting area projected on the ground, and different supporting relations provide different supporting areas. For "surface on-top" support, we project the vertexes of the two 3D bounding box to the ground, compute the convex hull for each projection, and use their intersection area on the ground plane as the supporting area. For "partial on-top" and "side" support, we assume there is only one edge touching between two boxes, and project this touching edge on the ground plane as the supporting area. Examples of the supporting areas are shown as red dashed circles in Fig. 5.8 (a) to Fig. 5.8 (c).

5.6 Global stability

Box stability is a global property: boxes can appear to be fully supported locally, but still be in a globally unstable configuration. Fig. 5.8 (d) and Fig. 5.8 (e) provide an illustration.

We perform a top-down stability reasoning by iteratively examining the current gravity center and supporting areas. This process is shown in Fig. 5.8. For simplicity we assume each box has the same density. This assumption is usually valid for daily objects, e.g. books, boxes, or bottles. They have similar densities, and can either support other objects or be supported.

We begin with the top box by finding the box center of mass, and check whether its gravity projection intersects the supporting area. If so, we mark the current box stable, and proceed to another box beneath for reasoning, this time finding the center of mass of the set of boxes already found to be stable with the one under consideration. Assuming constant density, the center of mass $P_c = [x, y, z]$ for a set of boxes is calculated by averaging the volume V_i of each box i:

$$P_c = \left(\sum_i P_{c,i} \cdot V_i \right) / \sum_i V_i \tag{5.1}$$

We iteratively update the center of mass by adding the boxes from top to bottom until the ground is reached. If we found that the current supporting area does not support the center of mass, we label the current box (or collection of boxes) unstable, shown in Fig. 5.8 (e). For the set of boxes with multiple supports, we compute the convex hull of the multi-supporting areas as the combined supporting area, shown in Fig. 5.8 (f) to Fig. 5.8 (g).

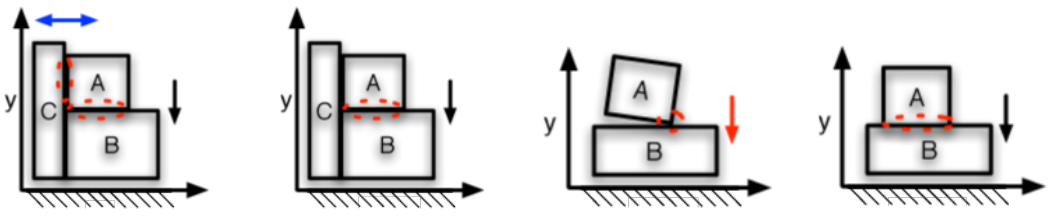


Figure 5.9: (a) Near-touching objects, e.g., objects A and C do not necessarily support one another. (b) After stability reasoning, we find that object A can be fully supported by object B beneath it through a surface on-top support. Therefore, we delete the unnecessary side support between A and C. (c) 3D oriented bounding boxes can be ill-fit because of noise, and this may lead to incorrect support relation inference. For example, between object A and B, a partial on-top support is proposed, although it should have been a surface on-top support. (d) After stability reasoning, we adjust the higher box if it is only supported from beneath, and then correct the support relation accordingly.

Support reasoning: Stability reasoning helps delete unnecessary supports. For example, side-to-side *nearly touching* objects do not necessarily support one another. We trim these unnecessary supporting relations by examining the support relations in the order: surface on-top, partial on-top and side support. If the object has a "surface on-top" support and the configuration can be stable, then additional support relations are unnecessary and can be trimmed. If not, we find a minimum combination of the on-top supports (both surface and partial) and at most two side supports examine whether the object can be stable. If so, all other support relations for the object are deleted. One example is shown in Fig. 5.9 (a) to (b).

Box fitting: Stability reasoning and supporting relations are used to refine the orientation of a box. If the box is fully supported through a "surface on-top" relation, then we re-fit the 3D bounding box of the top object, confining the

rotation of the first principle surface S_1 to be the same as the supporting surface. One example is illustrated in Fig. 5.9 (c) to (d). We perform this adjustment on box fitting every time after inferring the supporting relation and stability. This improves the box representation and support interpretation of the scene.

5.6.1 Integrating box-based features for segmentation

To incorporate all the box-based features, one baseline we implement is to start with an over-segmentation, and merge the pairwise segments based on learning [2]. We begin with initial segments generated with features from [18]. During training we use the ground-truth segmentation and label the segments that should be merged as $y = 1$, and the others as $y = 0$. We extract a set of features x based on the box fitting, pairwise box relation, and the global stability, shown in Table 5.1. For example, to compute one type of features (surface distance) for a merge, we record the minimum surface distances of two neighboring boxes before merging (2 dimensions, noted as **B**), and the minimum surface distance of the box after merging (1 dimension, noted as **A**), as well as the difference of this criterion before and after merging (1 dimension for each box before merging, 2 dimensions in total, denoted as **D**).

For this baseline model (labeled as **Stability** in the following sections), we train an SVM regression $y = w_{svm}^T x$ based on the features x and labels y. During testing, we greedily merge the neighboring segments based on the output prediction of the regression f, fit a new bounding box for each newly merged segment, recompute the stability reasoning, and re-extract the features for regres-

[2] Another possible implementation would be to start with an under-segmentation and perform splitting on each segment.

sion. We repeat the above steps until the classifier does not classify any pair of segments as a pair that should be merged. Note that this baseline merges pairs of segments, has no backtracking, and must begin with an over-segmentation of the image.

5.7 A Learned Energy Function

In this section, we improve the baseline model (**Stability**) from the previous section by introducing an energy function with unary and pairwise terms based on the volumetric boxes, their support relations, and stability (this method is labeled as **MCMC** in the following sections). This model provides the framework for exploring the space of an energy function that represents the goodness-of-fit of a particular box representation and corresponding segmentation for a scene with the corresponding RGB-D input. We define two different moves, *splitting* to split a segment, and *merging* to merge two adjacent segments. These moves allow us to traverse the space over which the energy function is defined. We explore the space with a partical-based filter to discover a local minimum that, hopefully, corresponds to a good segmentation and box representation of the scene.

We use s_i to represent one individual segment in a segmentation, and denote a segmentation as $S = \{s_1, ..., s_N\}$ with N segments and M pairs of neighboring segments. We define a pool of segmentations as $\{S\}$, which includes a set of possible different segmentations given the RGB-D input. $\{S\}_{all}$ indicates the space of all possible segmentations. Given one particular segmentation S, we define the energy function:

Table 5.1: Features based on volumetric and stability reasoning. **B**: the feature before a merge; **A** the feature after a merge; **D**: the difference of the feature before and after a merge.

Single/Pairwise features	**dim**
Box orientation with respect to the ground (B, A)	3
mean of the minimum surface distance (B, A, D)	5
Percentage of the visible points (B, A)	3
Percentage increase in the invisible points after a move	1
Number of intersecting boxes (B, A, D)	5
Average intersecting distance of the boxes (B, A, D)	5
Average intersecting distance of the boxes (B, A, D) with respect to volume	5
Pairwise supporting relations	1
Stability features	**dim**
Global stability (B, A, D)	3
Stabilities of the objects (B, A)	3
Distance of the projected gravity center to the supporting area center (B, A, D)	5
Difference for the three supporting relations	3
Average over number segments of the three supporting relations (B, A)	6

$$E(S) = \frac{1}{N}\sum_{i}\phi(s_i) + \frac{1}{M}\sum_{i,j}\psi(s_i, s_j), \tag{5.2}$$

where $\phi(s_i)$ is a regression score of a segment s_i describing the quality of this segment, and it is learned using single box features including box fitting errors, volumes, and stability, described as x_i in Table 5.2. Formally, $\phi(s_i)$ is defined as:

$$\phi(s_i) = w_s^T x_i, \tag{5.3}$$

where w_s is the learned regression parameters.

Similarly, $\psi(s_i, s_j)$ is a regression score of two neighboring boxes. It is learned using pairwise box features including box intersection distance, pairwise support relations, and pairwise box features, x_{ij}, described in Table 5.3. $\psi(s_i, s_j)$ is formally defined as:

$$\psi(s_i, s_j) = w_p^T x_{ij}, \tag{5.4}$$

where w_p represents the learned regression parameters.

5.7.1 Single and pairwise potentials

In the following section we further explain the training and testing processes for our single and pairwise potentials that comprise our energy function. The input at this step is a mid-step segmentation S, including N segments and M pairs of neighboring segments. This initial segmentation can be generated using the algorithm proposed in the literature, e.g. [18], or the previously proposed algorithm **Stability**.

Table 5.2: Features for single potentials. The "relative" feature values are the features divided by the volume of the box, instead of the absolute value.

Single potential $\phi(s_i)$ features x_i	**dim**
Box orientation with respect to the ground	1
Mean and variance of the minimum surface distance	2
Mean and variance of the relative minimum surface distance (divided by box volume)	2
3D point density over volume	1
Percentage of the visible points	1
Number of intersecting boxes	1
Global Stability	1
Stabilities of the objects	1
Average (and relative) intersecting distance of the boxes	2
Distance (and relative distance) of the projected gravity center to the supporting area center	2
Distance (and relative distance) of the projected gravity center to the projected vertexes	16

Table 5.3: Features for pairwise potentials. The "relative" feature values are the features divided by the volume of the box, instead of the absolute value.

Pairwise potential $\psi(s_i, s_j)$ features x_{ij}	**dim**
Number of intersection of each box	2
Relative of collision of each box (divided by each box volume)	2
Stability of each box	2
Pairwise supporting relations	1
Is one supporting another	1
Pairwise volume center distance	1
Projected gravity center to the supporting area center (if supported)	1
RGB-D features proposed in [18]	51

First, we learn the quality of each single segment s_i through a SVM regression as the single box potential $\phi(s_i)$. This is done through a supervised learning process on a held-out training set, and we generate the positive and negative training samples as follows: in the training images, we first use the ground-truth segmentation from human labeling as the positive training samples. We also make some random modifications from these ground-truth segmentations by splitting and merging, providing more positive and negative training instances. Then, we compute the segmentation score (the intersection-over-union ratio) of each segment s_i to the ground-truth segment $s_{j,gt}$:

$$score(s_i) = \max_{s_{j,gt}} \frac{Intersect(s_i, s_{j,gt})}{Union(s_i, s_{j,gt})}, \tag{5.5}$$

and consider a segment s_i as positive training sample if $score(s_i) \geq 90\%$, otherwise this segment is a negative training sample. After getting the training label, a 3D bounding box is then fit to this segment, and then the proposed box-related

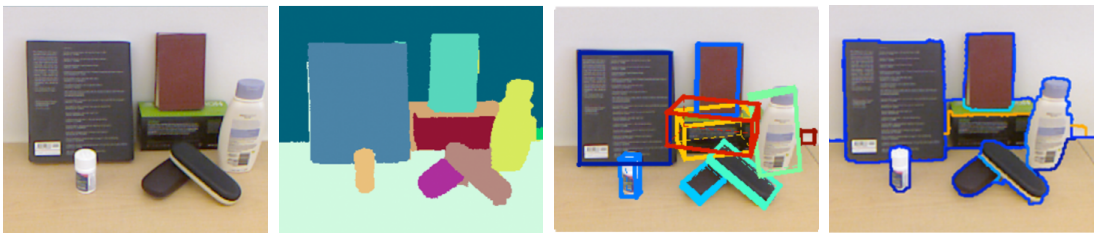
features x_i are computed for training.

During testing, we fit a 3D bounding box to each segment s_i, compute the features x_i, and perform the regression in Eq. 5.3 to calculate the single box potential value $\phi(s_i)$. Fig. 5.10 (c) presents one example of our single box potentials during testing. The boxes of the segments are color-coded in a way that the lower potential value $\phi(s_i)$ of segment s_i is, the more blue its corresponding box is. It shows that our proposed single box potential value captures the segment quality and classifies the ill-fit boxes, e.g., the boxes with yellow and red colors.

The pairwise potential is trained and tested following the similar manner: multiple randomly generated segmentations as well as the ground-truth ones are processed during training. A boundary is considered a positive training instance if the two segments it lies between both have segmentation scores (proposed in Eq. 5.5) larger than 90%. During testing, 3D bounding boxes are also first fit to all the segments, and then the pairwise features described in Table 5.2 and Table 5.3, bottom part, are computed. We perform regression $\psi(s_i, s_j)$ on the pairs of the segments sharing a boundary. Fig. 5.10 (d) presents one example of pairwise potentials $\psi(s_i, s_j)$. This potential gives a good indication of which pairs of segments, if merged, might produce a reduction to the global energy function.

5.7.2 Minimizing through splitting and merging

During testing, our goal is to minimize this energy function and find the optimal segmentation S^* that has the minimum energy value:



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

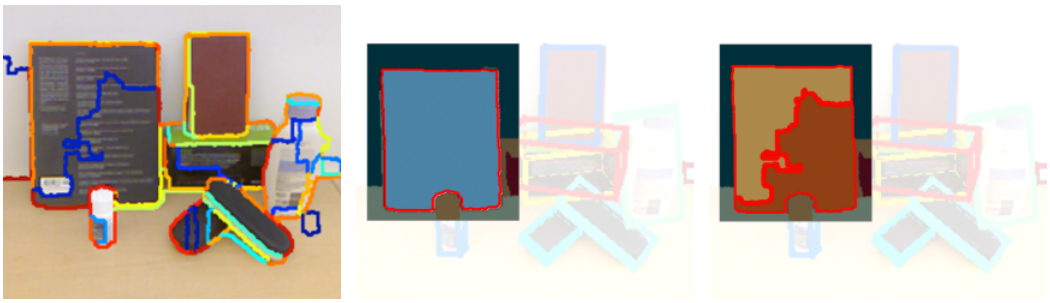
Figure 5.10: (a) Input image. (b) Mid-step segmentation during testing. (c) and (d) are exemplar testing results for (c) single potential $\phi(s_i)$ and (d) pairwise potential $\psi(s_i, s_j)$. The color of the boxes and boundaries is coded as the better quality the segments are, the more blue the boxes and boundaries are, with lower potential values. Our proposed features capture the quality of each segment and boundary.

$$S^* = \arg\min_S E(S). \tag{5.6}$$

Note that this energy function is non-convex, and the space of possible segmentations $\{S\}_{all}$ is very large, therefore it is infeasible to perform an exhaustive search to find the global minimum.

To explore the space, we adopt a Markov-Chain-Monte-Carlo (MCMC) [98] approach to this problem, where we design appropriate moves to explore the space. We start with an initial segmentation, and move to a new set of segmentations by either: (a) splitting one segment into two smaller segments, or (b) merging two neighboring segments into one segment. We use the potentials $\phi(s_i)$ and $\psi(s_i, s_j)$ to indicate which segments should be split or merged while designing the MCMC moves. We keep a pool of possible segmentations as the particles to explore this energy space, and keep track of the ones with the minimum energy values as we iterate for optimization.

Splitting: The single box potential $\phi(s_i)$ indicates the quality of each individual



(a) (b) (c)

Figure 5.11: (a) We pre-compute all the possible boundaries given RGB-D image. (b) The selected segment before splitting. (c) The selected segment after splitting. The splitting move is constrained to split one segment into two.

segment. This value guides the splitting moves so that we explore the segmentation space in a more efficient manner.

We calculate the single box potential for all the segments in the current segmentation, and then randomly choose one segment s_i to split based on its potential $\phi(s_i)$: the higher $\phi(s_i)$ is, the more likely s_i is going to be selected for splitting, because it represents a worse segmentation quality for s_i, and thus s_i needs to be modified. The final likelihood of selecting one segments is linearly mapped from $\phi(s_i)$ by converting $\phi(s_i)$ into probabilistic prediction [99].

Specifically, we split one segment s_i as follows: we pre-compute a boundary map of all the possible edges given the RGB-D images using [18]. One example is shown in Fig. 5.11 (a): all the possible boundaries are presented in this boundary map, including the false ones. This map provides us the basis for splitting one segment. Then given the selected segment s_i, this segment is forced to be split into two segments based on the boundary map, as illustrated in Fig. 5.11 (b) and Fig. 5.11 (c). The boundaries within s_i are merged from lower values to higher values based on the pre-computed boundary map, until only two seg-

ments remain in s_i.

Merging: We merge the segments with a similar principle: first we compute all the pairwise potentials $\psi(s_i, s_j)$ given the current segmentation, and then we randomly sample a pair of segments based on their pairwise potential value: if two segments s_i and s_j have a higher pairwise potential $\psi(s_i, s_j)$, they have a higher chance to be selected for merging, because $\psi(s_i, s_j)$ indicates a worse quality boundary between two segments. After the boundary and its pair of segments are chosen, we merge the neighboring two segments by deleting the boundary between them and group all the pixels into one segment.

Minimization: The energy function in Eq. 5.2 is devised in the way that the smaller the value is, the better segmentation is. We find a better segmentation with a lower energy value by maintaining a segmentation pool $\{S\}$, and repeatedly finding the segmentations with smaller energy values within this pool. Splitting and merging compose our basic moves for minimization. Given one initial segmentation, we propose $2N$ (we use $N = 5$) new segmentations by N splitting moves and N merging moves, and then re-evaluate all segmentations using Eq. 5.2. We take the K (we use $K = 5$) segmentations with the smallest energy values for the next iteration, and discard the remaining segmentations. We repeat this step again, so that the top K segmentations will branch, producing KN new moves, and then be evaluated together to choose the top K segmentations for the next step. We repeat this sampling step until we reach the maximum number of iterations M. In practice, this algorithm optimizes our energy function to a reasonable local minima in about 10-15 iterations. The details of the algorithm are presented in Alg. 2.

Algorithm 2: Energy Minimization

Given constants N, K, and M.

Initialize segmentation pool $\{S\}$ with initial segmentation S_{init}.

for i=1 to M **do**

  for each segmentation S_t in the pool $\{S\}$ **do**

    Compute $\phi(s_i)$ and $\psi(s_i, s_j)$ for S_t.

    for j=1 to N **do**

      Sample one segment s_i by $\phi(s_i)$ and split it, producing new segmentation S_j

      Add S_j to $\{S\}$

    end for

    for j=1 to N **do**

      Sample one pair of segments by $\psi(s_i, s_j)$ and merge them, producing new segmentation S_j

      Add S_j to $\{S\}$

    end for

  end for

  Evaluate the energy function E for all the segmentations in $\{S\}$.

  Keep top K segmentations in $\{S\}$ with smallest $E(S)$.

end for

Output S^*_{final} with the minimum energy value $E(S)$ in the $\{S\}$.

5.8 Experiments

We perform experiments on four different types of datasets: a block dataset, a supporting object dataset (SOD), a grocery dataset, and a public dataset of indoor scenes proposed in [18]. We evaluate the box fitting accuracy, the support relation prediction, and the segmentation performance.

5.8.1 Block dataset

We apply our algorithm to a toy block dataset. This dataset has 50 RGB-D images of blocks, shown in Fig. 5.12. For each block, we manually provide the ground-truth segment labels, as well as the orientations of two perpendicular surfaces[3]. Ground-truth surface orientations are labeled by manually clicking at least 8 points on the same surface, and fitting a plane to these labeled 3D points. Supporting relations of each block are also manually labeled.

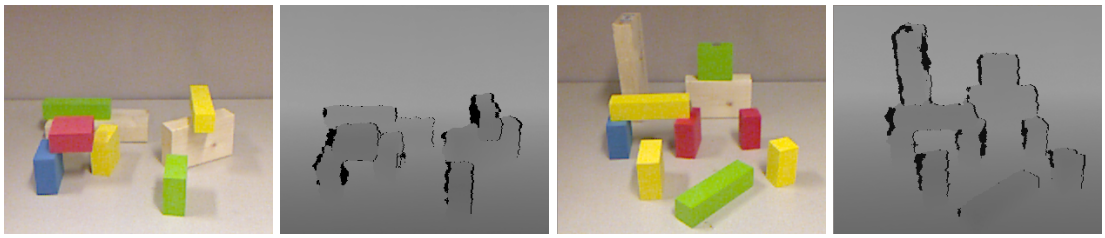


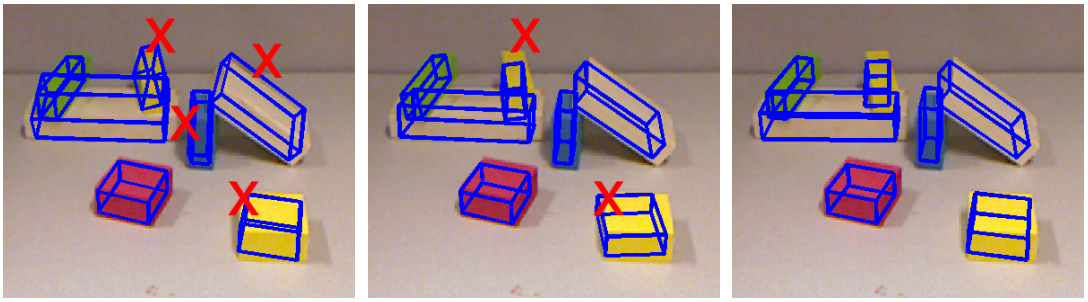
Figure 5.12: Examples of the RGB-D Block Dataset with color (left) and depth (right) images.

First, we evaluate our box fitting algorithm. The following algorithms are compared:

[3]The third surface orientation is perpendicular to the first two, and thus determined after providing the first two surface orientations.

Table 5.4: Average angle error on the bounding box orientation.

	Block Dataset
Min-vol	15.41°
Min-surf	9.75°
Supp-surf	7.02°



(a) (b) (c)

Figure 5.13: Fitting results on the block dataset. (a): **Min-vol**. (b): **Min-surf**. (c): **Supp-surf**. Blocks with large fitting error in orientation are labeled as a red "x".

Min-vol: the baseline algorithm from [95] of fitting minimum volume bounding box.

Min-surf: the proposed box fitting algorithm of finding the minimum surface distance.

Supp-surf: use our proposed algorithm **Min-surf** to find the initial boxes, and adjust the orientation of the box based on the supporting relations and stability.

We compare the orientation of the bounding box from each algorithm to the ground-truth, and calculate the average angle difference. Table 3 shows that our proposed minimum surface distance provides a better box fitting compared

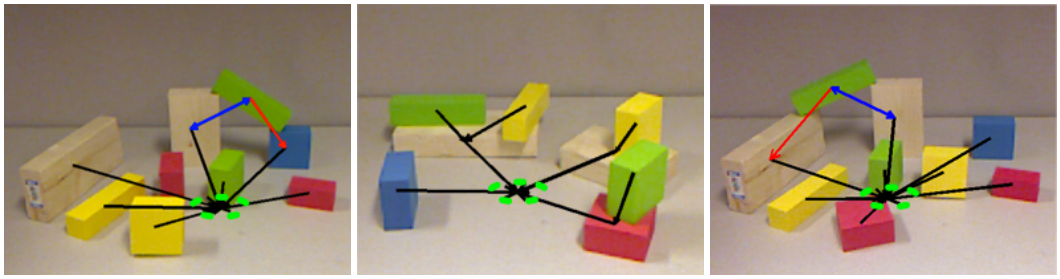


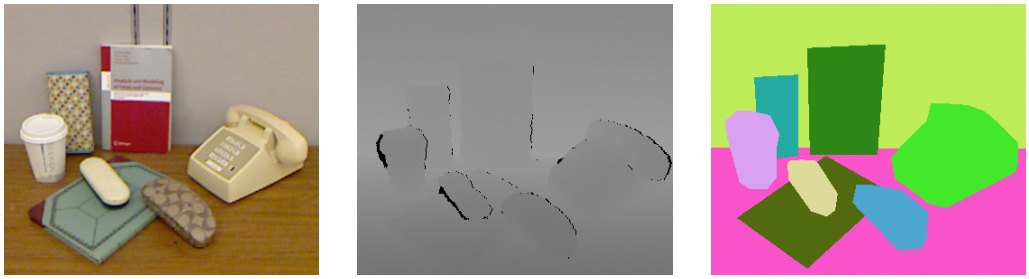
Figure 5.14: The predicted supporting relations on block dataset. Three different types of the supporting relations are colored in black (surface-top), red (partial-top), and blue (side). The ground plane center is plot as a green dashed circle.

to the minimum volume criteria, reducing the errors in angle from 15.41° to 7.02°, a 40% improvement. With stability reasoning, the fitting decreases error by another 2° in absolute value, a 15% improvement.

We then analyze the performance of our stability reasoning. We compare with the ground truth supporting relations, and count an object as correct if all its supporting objects are predicted. We compare our proposed algorithm (**Stability Reason**) that reasons about the stability of each block and deletes the false supporting relations with the baseline (**Neighbor**) that assumes one block is supported by its neighbors, i.e., the initialization of the supporting relations.

Table 5.6, left column reports the supporting relation accuracy for this block dataset. Since the segments in the dataset are perfect blocks, the neighboring rule gives a high accuracy at over 80% for predicting support. However, our proposed stability reasoning improves the supporting relation accuracy by an absolute 10%, achieving over 90% of accuracy. Exemplar images of the predicted supporting relations are shown in Fig. 5.14.

5.8.2 Supporting object dataset



(a) (b) (c)

Figure 5.15: Our supporting object dataset (SOD) includes (a) the color image, (b) the depth image, and (c) manually labeled segments.

Many of the daily objects can be approximated as 3D volumetric blocks with similar densities, following our stability reasoning assumption. Thus we collect a new Supporting Object Dataset (SOD) composing of 307 RGB-D images. Various daily objects are randomly placed in scenes in different configurations of support. For each object, we manually label the segment and the objects supporting it. Fig. 5.15 gives one exemplar RGB-D image pairs and ground-truth segmentation labeling of our supporting object dataset.

First, we measure the prediction of the supporting relations with the ground truth segmentation. The results of using the baseline **Neighbors** and our stability reasoning **Stability Reason** are shown in Table. 5.6, right column. In this dataset with irregular shaped objects and complicated support configurations, using the touching neighbors to infer supporting relations has an accuracy of 52%. Stability reasoning gives an absolute 20% boost, reaching over 72% accuracy. Fig. 5.18 presents the exemplar results of our box fitting and support prediction from the supporting object dataset.

We also evaluate the segmentation performance with our proposed features

Table 5.5: Pixel-wise segmentation score.

	SOD	GD	NYU
[18]	60.2%	65.9%	60.1%
S/P	64.7%	68.1%	60.8%
Stability	66.7%	69.2%	61.0%
MCMC	**70.0%**	**72.3%**	**61.7%**

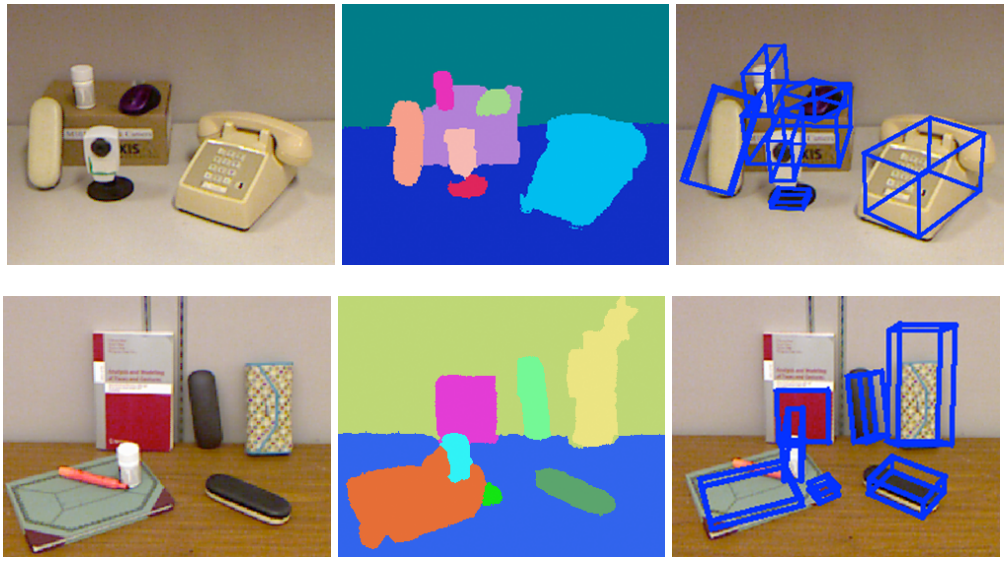


Figure 5.16: Segmentation and box fitting results of our proposed algorithm on the Support Object Dataset (SOD) testing images.

based on box properties. We randomly choose half of the images for training, and the other half for testing. We follow the procedure in [18] and use their color and depth features as the baseline. Then we add our features using the single and pairwise box relations (**S/P**), and our full feature set with stability reasoning (**Stability**) with the model proposed in Section 7. Finally we perform our final model based on the energy function with MCMC sampling allowing

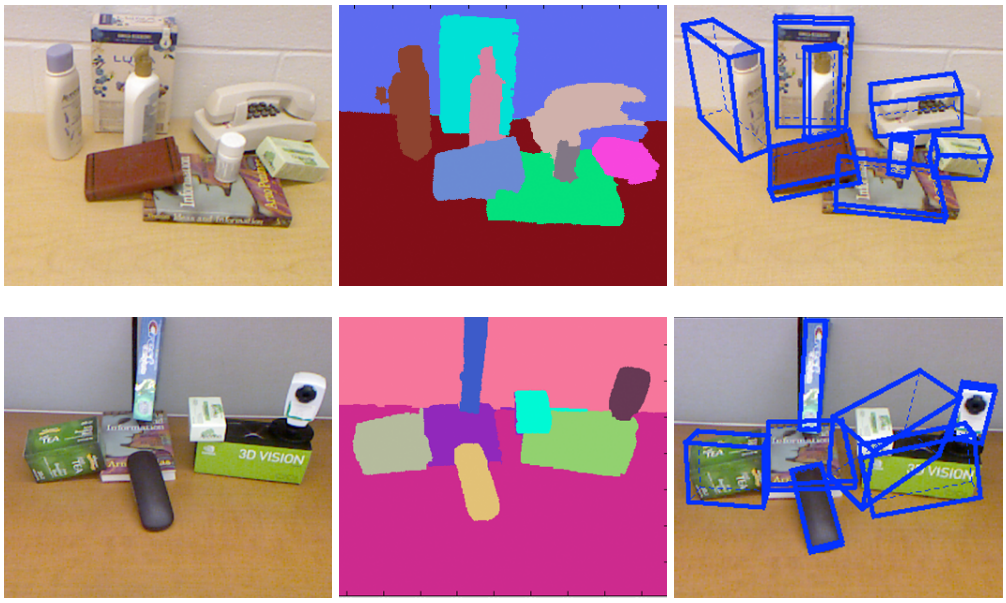


Figure 5.17: Segmentation and box fitting results of our proposed algorithm on the Grocery Dataset (GD) testing images.

both merging and splitting (**MCMC**).

The segmentation accuracy is scored by pixel-wise overlapping with the ground-truth segments, proposed in [67] and [18]. Table 5.5, first column, shows the performance comparison with different feature sets for our proposed dataset (evaluating only on the object segments because the background is shared across the images). Reasoning about each object as a box gives around 4% boost in segmentation accuracy, and adding the stability features further improves the performance by 2%. Our final energy model with MCMC sampling gives the best results with another 3% improvement. Testing results with block fitting are presented in Fig. 5.16.

The final algorithm (**MCMC**) performs better as we further iterate the sampling steps. For this dataset, the overall average segmentation performance over the iteration steps are presented in Fig. 5.20 (a), in blue curve. At the same

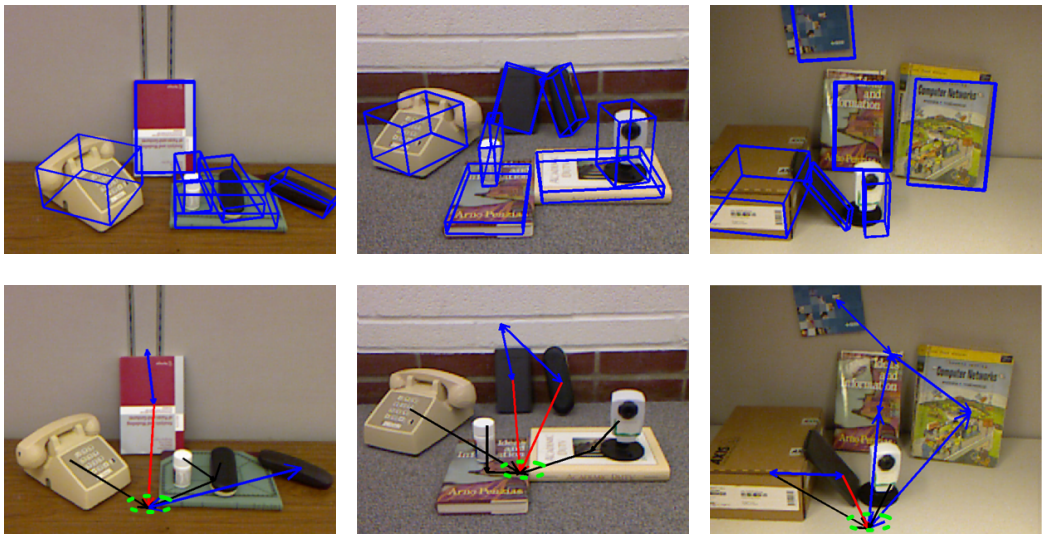


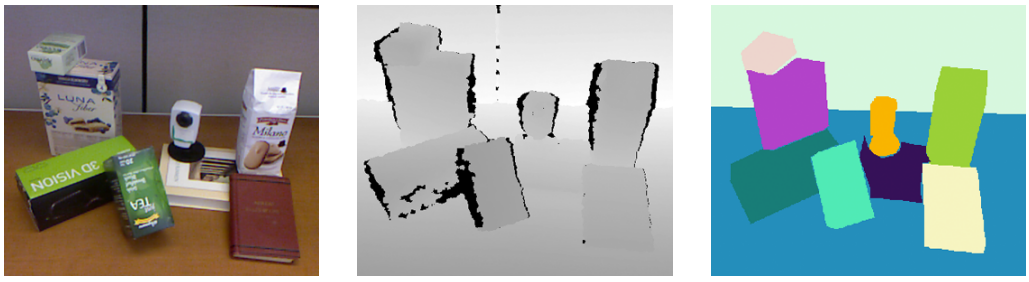
Figure 5.18: We qualitatively show our box fitting algorithm (left) on daily objects with ground-truth image segmentation and the supporting relation prediction after stability reasoning (right). Boxes for large surfaces (like the back wall and the ground) are not displayed for better visualization. The ground plane is plotted as a green dashed circle for showing the support inference results.

time, the average energy function is minimized, shown in Fig. 5.20 (a) as a green curve. It shows that the accuracy of the segmentation increases as we minimize the energy function through our **MCMC** sampling process. Therefore, it provides evidence that our energy function accurately represents the quality of the segmentation.

Fig. 5.21, top row, presents one particular sequence of the top segmentations (smallest energy values) at each step as we minimize energy function. The initial segmentation input is displayed in Fig. 5.21 (c). In detail, the top part of the camera is mistakenly merged with the book underneath it initially. During the middle step, our splitting moves successfully separate these two objects into individual segments, however the book is still over-segmented. In the final step our merging moves correctly group the book lying on the ground, and the

overall segmentation improves.

5.8.3 Grocery dataset



(a) (b) (c)

Figure 5.19: Our grocery dataset (GD) extended on support object dataset (SOD) also includes (a) the color image, (b) the depth image, and (c) manually labeled segments.

One possible application scenario of our proposed algorithm is a supermarket, where many objects are contained in regular boxes. We collect an extended Grocery Object dataset (GD) based on the Support Object Dataset (SOD) to demonstrate this application. This dataset mimics the environment of a grocery store, and includes a variety of common grocery objects, such as cereal boxes, shampoo bottles, etc. The dataset contains 609 RGB-D images with human-labeled ground-truth segmentation. Some exemplar RGB-D images with ground-truth segmentation are presented in Fig. 5.19.

We evaluate the segmentation accuracy on this dataset, and compared it with the baseline algorithm proposed in [18]. Full quantitative results of different algorithms are presented in Table 5.5, middle column. The results show that our proposed new feature set increases the segmentation accuracy, and the final sampling algorithm (**MCMC**) with merging and splitting moves gives the best

result. Some example testing images with final block representations are presented in Fig. 5.17. Our final algorithm produces more reasonable segmentation results as well as the volumetric block representations. This provides a richer interpretation of the object in the scene.

For this dataset, the average segmentation accuracies of our sampling algorithm (**MCMC**) over the iteration steps are presented in Fig. 5.20 (b), along with the energy function values, in blue and green curves respectively. It presents a similar pattern that as we minimize the energy function, the overall performance of segmentation improves.

We show two particular testing examples in Fig. 5.21, middle and bottom rows, to illustrate the iteration steps of mixed merging and splitting. The initial segmentation includes errors of both incorrect over segmentations and under segmentations. During our steps of MCMC sampling, some mistakes in the initial segmentations are corrected, while the other new errors arise, e.g., the book (in the middle row) and the mouse (in the bottom row) are merged to the ground. However, in the final step our proposed minimization method tend to find better overall segmentations.

Table 5.6: Supporting relation accuracy for different dataset.

	Block	SOD
Neighbor	80.59%	52.88%
Stability Reason	**91.68%**	**72.86%**

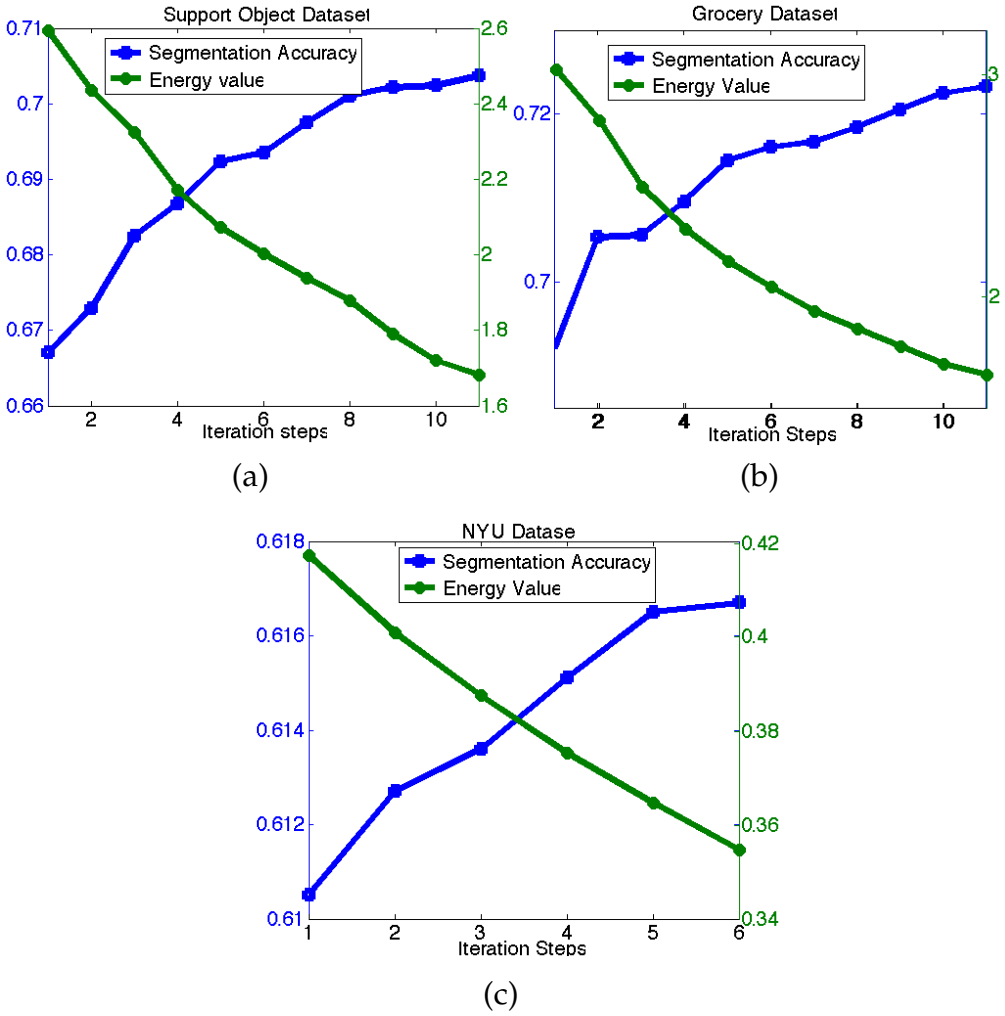
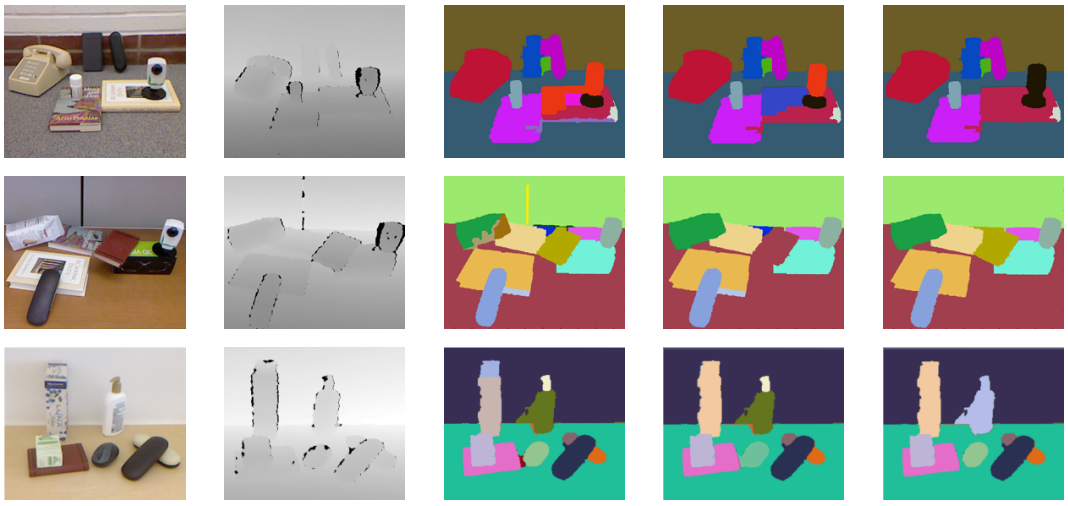


Figure 5.20: Segmentation results of our proposed sampling algorithm (**MCMC**) over each iteration on the SOD dataset (a), GD dataset (b) and NYU-2 dataset (c). As the energy value decreases through the minimization steps, the accuracy of the segmentation increases.

5.8.4 NYU indoor dataset

We evaluate segmentation performance on the newly released RGB-D NYU-2 indoor dataset [18], and report the performance in Table 5.5, right column. This dataset is proposed for scene understanding, rather than object reasoning,



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

Figure 5.21: The segmentation results improve along with more iterations of the proposed algorithm **MCMC**. Given the color image (a), and the depth image (b), the initial segmentation (c) may have some mistakes. Some of these mistakes are corrected during middle steps as iteration goes on, shown in (d). In the final iteration, the segmentations are corrected into more reasonable ones, presented in (e).

and many large surfaces, such as counters and drawers, and are sometimes labeled as two or more distinct objects, i.e., one for each surface, instead of one for the entire object. Although these conditions limit the evaluated performance of our proposed algorithm, adding the proposed features still improves the segmentation performance. The performance of our sampling algorithm (**MCMC**) gives the best results, also the performance improves throughout iteration steps. The detailed segmentation accuracy of each step, as well as the energy function value, are presented in Fig. 5.20 (c), in blue and green curves. Some examples of the segmentation results are shown in Fig. 5.23.

We find that although proposed for modeling small object interactions, this block representation and stability reasoning framework can also be extended to

some indoor scenarios, e.g., for furniture sitting on the ground or supported on the wall. We qualitatively present the box fitting and supporting inference result with ground-truth segmentation for a indoor bedroom scenario in Fig. 5.22.

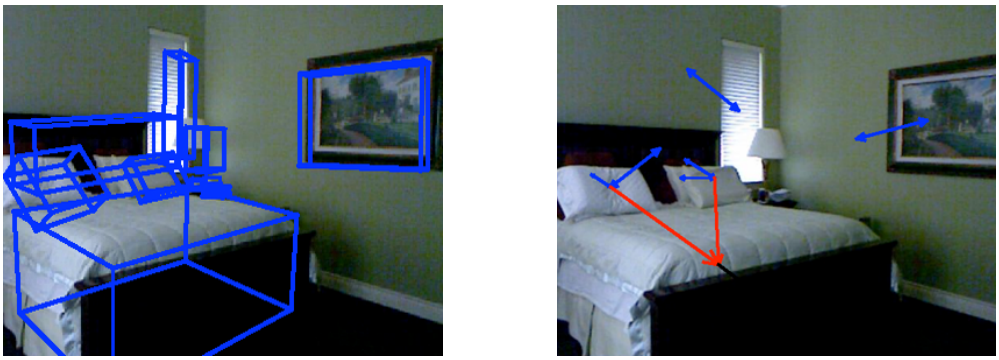


Figure 5.22: Qualitative result of box fitting (left) and supporting relation inference (right) on indoor scenes. For better visualization, boxes that are too large (wall, ground) or too small are not displayed.

Figure 5.23: Segmentation results of our proposed algorithm on NYU-2 indoor scene dataset.

5.9 Summary

In this thesis, we propose analyzing RGB-D images through physically-based stability reasoning. We begin with box fitting on partially observed 3D point clouds, and then introduce pairwise box interaction features. We explore global stability reasoning on proposed box representations of a scene. Segmentations associated with unstable box configurations are not physically possible and are subsequently modified for consideration in later iterations. Stability reasoning produces better estimates of supporting relations (by requiring enough support to provide stability for each object) and improved box orientation estimates (by knowing when objects are fully or partially supported from below). Experiments show that our proposed algorithm works for both synthetic and real world scenes, and leads to improvements in box fitting, support detection, and segmentation.

CHAPTER 6

CONCLUSION AND DISCUSSION

In this thesis we presented a complete framework to reason RGB-D images from segments to volumes. We started from 2D color images, and inferred depth ordering using clues from boundary shapes and junction textures. After that, we incorporated 3D information from depth sensor, such as laser scan data or consumer inferred sensors. Additional 3D data provided us further geometry understanding, such as surfaces, occlusion and connected boundaries. Finally, we combined all the inferences and clues to achieve a complete 3D volumetric understanding of the scene. This higher level interpretation allows us achieve more semantic 3D understanding, such as support and physical stability between objects, and gives better performance in the task of RGB-D segmentation.

We believe that physics-based stability reasoning in segmentation could be useful in several applications with RGB-D data, for example, activity detection, object detection and tracking, scene modeling, and so on. We mention a few possible future directions that can be extended based on the algorithm proposed in this thesis.

3D oriented block fitting with color image: The current block fitting algorithm in this thesis solely relies on the 3D point clouds. However as presented in contemporary work [74], color channel provides informative edge clues, which can also be incorporated for fitting the bounding box. It is possible to combine the color features with our proposed 3D point-cloud based algorithm and improve the 3D bounding box fitting.

Extending primitive shapes: Although blocks are good approximations for many convex objects, there are cases when they limit the performance of scene reasoning. For example, a basketball may be failed to be presented as a 3D oriented bounding box, and therefore its stability cannot be correctly estimated using the simple blocks that we propose. Extending the primitive shapes from blocks to cylinders (e.g., [96]) spheres, or non-parametric shapes, along with corresponding advancements to the stability reasoning module, may improve the support and stability reasoning, as well as the final object segmentation.

Combining with semantic classification: Previous work has shown that combining different tasks improves performance of individual vision tasks [100, 101]. We believe that combining the block representation with semantic classification will further improve the 3D scene understanding. Concave objects, such as chairs, are not well represented by a single box. In these situations, we can use multiple boxes to build the objects. However, to prevent the system from over-fitting, prior knowledge is helpful, e.g., the categories or the semantic labels of the objects. Therefore, we could detect objects as a pre-processing step, and then propose potential category hypothesis for the target objects. After that, we can choose the correct number of blocks to approximate the object, and produce an improved scene parsing.

Different block densities: Semantic classification can also be performed on other attributes, for example, to estimate the densities of the blocks. Future work can relax the even density assumption used in this thesis, and classify the weight of one block. This will enable us to adjust the block representation accordingly, allowing only the heavy boxes support the light ones, and thus making the scene more stable.

Hidden support: In this work we assume all the support relations are visible in the scene. However it is possible to analyze the hidden supports that are invisible or occluded, and only use the assumption that the scene is static. There are possible clues that enable us infer these hidden supports: for example, if one box is tilted with no other neighboring support, it is likely that the object is supported by an invisible object, e.g., a glass, or the supporting object is completely occluded. Analyzing the hidden support will unify the stability reasoning with the concept of occlusion.

Completing the physical model: For reasoning about stability, our model makes broad assumptions about objects in the scene. We assume objects are constant density and that objects are supported when their center of gravity projects into the convex hull of support, effectively ignoring friction. Further, we only reason about stability in a top-to-bottom fashion. Other, more sophisticated physical modelers (e.g., [93], Bullet [102] or Open Dynamics Engine [103]), though computationally more expensive, could be also explored. We expect they would provide a more complete analysis of the physics in the scene and lead to better RGB-D segmentations.

APPENDIX A

RELATED PUBLICATIONS

- Zhaoyin Jia , Andrew Gallagher, and Tsuhan Chen,"Camera and Gravity: Estimating Planer Object Orientation" IEEE International Conference on Image Processing (ICIP), 2013.
- Zhaoyin Jia , Andrew Gallagher, and Tsuhan Chen,"Learning Boundaries with Color and Depth" IEEE International Conference on Image Processing (ICIP), 2013.
- Zhaoyin Jia , Andrew Gallagher, Ashutosh Saxena and Tsuhan Chen,"3D-Based Reasoning with Blocks, Support, and Stability" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.
- Zhaoyin Jia , Andrew Gallagher, Yao-Jen Chang and Tsuhan Chen,"A Learning Based Framework for Depth Ordering" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
- Zhaoyin Jia , Ashutosh Saxena and Tsuhan Chen,"Sharing Utility Between Objects for Active Recognition" Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.
- Zhaoyin Jia , Yao-Jen Chang, Tzung-Han Lin and Tsuhan Chen,"Dense Interpolation of 3D Points Based on Surface and Color" IEEE International Conference on Image Processing (ICIP), 2011.
- Zhaoyin Jia , Ashutosh Saxena and Tsuhan Chen,"Robotic Object Detection: Learning to Improve the Classifiers using Sparse Graphs for Path Planning" International Joint Conferences on Artificial Intelligence (IJCAI), 2011.
- Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen,"Image Retrieval with Geometry-Preserving Visual Phrases" IEEE Conference on Computer Vision

and Pattern Recognition (CVPR), 2011.

- Zhaoyin Jia , Yao-Jen Chang, and Tsuhan Chen,"A general boosting-based framework for active object recognition" British Machine Vision Conference (BMVC), 2010.
- Zhaoyin Jia , Yao-Jen Chang, and Tsuhan Chen,"Active View Selection for Object and Pose Recognition" Workshop on 3D Representation for Recognition, International Conference on Computer Vision (ICCV), 2009.
- Zhaoyin Jia , Yao-Jen Chang, and Tsuhan Chen,"Dense 3D-Point Estimation based on Surface Fitting and Color Information", Western New York Image Processing Workshop (WNYIP), 2009.

BIBLIOGRAPHY

[1] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *PAMI*, 2009.

[2] D. Hoiem, A. Efros, and M. Hebert, "Geometric context from a single image," in *ICCV*, 2005.

[3] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *PAMI*, 2004.

[4] S. Maji, N. Vishnoi, and J. Malik, "Biased normalized cuts," in *CVPR*, 2011.

[5] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *CVPR*, 2009.

[6] J. Tighe and S. Lazebni, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013.

[7] D. Hoiem, A. Efros, and M. Hebert, "Putting objects in perspective," *IJCV*, vol. 80, no. 1, 2008.

[8] D. Hoiem, A. A. Efros, and M. Hebert, "Closing the loop in scene interpretation," in *CVPR*, 2008.

[9] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[10] Z. Jia, A. Gallagher, Y. Chang, and T. Chen, "A learning based framework for depth ordering," in *CVPR*, 2012.

[11] Z. Jia, Y. Chang, T. Lin, and T. Chen, "Dense interpolation of 3d points based on surface and color," in *ICIP*, 2011.

[12] Z. Jia, A. Gallagher, and T. Chen, "Learning boundaries with color and depth," in *ICIP*, 2013.

[13] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, "3d-based reasoning with blocks, support, and stability," in *CVPR*, 2013.

[14] H. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *IJRR*, 2013.

[15] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *IJRR*, vol. 31, no. 9, 2012.

[16] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *ICRA*, 2011.

[17] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3-D object dataset: Putting the kinect to work," in *ICCV workshop*, 2011.

[18] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012.

[19] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, 2004.

[20] Z. Jia, Y. Chang, T. Lin, and T. Chen, "Dense 3d-point estimation based on surface fitting and color information," in *WNYIP*, 2009.

[21] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, "3d reasoning from block to stability," in *In submission to PAMI*, 2013.

[22] Z. Jia, A. Gallagher, and T. Chen, "Camera and gravity: Estimating planer object orientation," in *ICIP*, 2013.

[23] Z. Jia, A. Saxena, and T. Chen, "Sharing utility between objects for active recognition," in *Workshop on Fine-Grained Visual Categorization, CVPR*, 2011.

[24] Z. Jia, A. Saxena, and T. Chen, "Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning," in *IJCAI*, 2011.

[25] Y. Zhang, Z. Jia, and T. Chen, "Image retrieval with geometry-preserving visual phrases," in *CVPR*, 2011.

[26] Z. Jia, Y. Chang, and T. Chen, "A general boosting-based framework for active object recognition," in *BMVC*, 2010.

[27] Z. Jia, Y. Chang, and T. Chen, "Active view selection for object and pose recognition," in *Workshop on 3D Representation for Recognition, ICCV*, 2009.

[28] B. Packer, S. Gould, and D. Koller, "A unified contour-pixel model for figure-ground segmentation," in *ECCV*, 2010.

[29] F. F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *CVPR*, 2005.

[30] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *CVPR*, 2010.

[31] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh, "How well do line drawings depict shape?," *ACM Transactions on Graphics*, 2009.

[32] H. Barrow and J. Tenenbaum, "Retrospective on "interpreting line drawings as three-dimensional surfaces"," *AI*, vol. 59, 1993.

[33] D. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," Tech. Rep. AI271, MIT, 1972.

[34] D. Hoiem, A. Efros, and M. Hebert, "Recovering occlusion boundaries from an image," *IJCV*, vol. 91, no. 3, 2011.

[35] N. Apostoloff and A. Fitzgibbon, "Automatic video segmentation using spatiotemporal T-junctions," in *BMVC*, 2006.

[36] M. Dimiccoli and P. Salembier, "Exploiting T-junctions for depth segregation in single images," in *ICASSP*, 2009.

[37] G. Palou and P. Salembier, "Occlusion-based depth ordering on monocular images with binary partition tree," in *ICASSP*, 2011.

[38] X. Ren and C. Gu, "Figure-ground segmentation improves handled object recognition in egocentric video," in *CVPR*, 2010.

[39] X. Ren, C. Fowlkes, and J. Malik, "Figure/ground assignment in natural images," in *ECCV*, 2006.

[40] P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik, "Occlusion boundary detection and figure/ground assignment from optical flow," in *CVPR*, 2011.

[41] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[42] T. Joachims, "Making large–scale SVM learning practical," in *Advances in Kernel Methods — Support Vector Learning*. 1999, MIT Press.

[43] G. Kim, D. Huber, and M. Hebert, "Segmentation of salient regions in outdoor scenes using imagery and 3-D data," in *WACV*, 2008.

[44] Y. Q. Ma, Z. Wang, M. Bazakos, and W. Au, "3D scene modeling using sensor fusion with laser range finder and image sensor," in *Applied Imagery Pattern Recognition Workshop*, 2005.

[45] R. Triebel H. Andreasson and A. Lilienthal, "Non-iterative vision-based interpolation of 3d laser scans," in *Autonomous Robots and Agents, Studies in Computational Intelligence*, 2007.

[46] J. Diebel and S. Thrun, "An application of markov random fields to range sensing," in *NIPS*, 2005.

[47] S. Petitjean, "A survey of methods for recovering quadrics in triangle meshes," *ACM Computing Surveys*, 2002.

[48] P.J. Besl and R.C. Jain, "Segmentation through variable-order surface fitting," *PAMI*, 1988.

[49] T. Yoshimi, Y. Kawai, and Fumiaki Tomita, "Range data segmentation with principal vectors and surface types," in *MVA*, 1996.

[50] P. Benkö and T. Várady, "Segmentation methods for smooth point regions of conventional engineering objects," *Computer-Aided Design*, 2004.

[51] R. Chellappa and A. K. Jain, *Markov Random Fields: Theory and Applications*, Academic Press, 1993.

[52] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *PAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[53] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *ICCV*, 2007.

[54] I. Endres and D. Hoiem, "Category independent object proposals," in *ECCV 2010*, 2010.

[55] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *ICCV Workshops*, 2011.

[56] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011.

[57] B. Huhle, T. Schairer, P. Jenke, and W. Strasser, "Robust non-local denoising of colored depth data," in *Workshop of Time of Flight Camera based Computer Vision, CVPR*, 2008.

[58] I. Reisner-Kollmann and S. Maierhofer, "Consolidation of multiple depth maps," in *ICCV Workshops on Consumer Depth Cameras for Computer Vision*, 2011.

[59] E. Borenstein and S. Ullman, "Learning to segment," in *ECCV*, 2004.

[60] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox, "Interactive 3D modeling of indoor environments with a consumer depth camera," in *UbiComp*, 2011.

[61] L. Bo, K. Lai, X. Ren, and D. Fox, "Object recognition with hierarchical kernel descriptors," in *CVPR*, 2011.

[62] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Dar-

rell, "A category-level 3-D object dataset: Putting the kinect to work," in *ICCV Workshops on Consumer Depth Cameras for Computer Vision*, 2011.

[63] H. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3D point clouds for indoor scenes," in *NIPS*, 2011.

[64] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon, "Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera," in *UIST*, 2011.

[65] Hoiem D, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *IJCV*, vol. 75, no. 1, pp. 151–172, 2007.

[66] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *NIPS*, 2005.

[67] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *ICCV*, 2007.

[68] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *ECCV*, 2010.

[69] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *CVPR*, 2012.

[70] P. Arbelaez, M. Maire, C. Fowlkes, and J Malik, "Contour detection and hierarchical image segmentation," *PAMI*, 2011.

[71] S. Maji, N. Vishnoi, and J. Malik, "Biased normalized cuts," in *CVPR*, 2011.

[72] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *ICCV*, 1998.

[73] Y. Zheng, X. Chen, M. Cheng, K. Zhou, S. Hu, and N. J. Mitra, "Interactive images: cuboid proxies for smart image manipulation," *ACM Trans. Graph*, vol. 31, no. 4, pp. 99, 2012.

[74] J. Xiao, B. C. Russell, and A. Torralba, "Localizing 3D cuboids in single-view images," in *NIPS*, 2012.

[75] M. Bleyer, C. Rhemann, and C. Rother, "Extracting 3D scene-consistent object proposals and depth from stereo images," in *ECCV*, 2012.

[76] H. Jiang and J. Xiao, "A linear approach to matching cuboids in rgbd images," in *CVPR*, 2013.

[77] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *CVPR*, 2006.

[78] A. Flint, D. W. Murray, and I. Reid, "Manhattan scene understanding using monocular, stereo, and 3D features," in *ICCV*, 2011.

[79] V. Hedau, D. Hoiem, and D. A. Forsyth, "Recovering free space of indoor scenes from a single image," in *CVPR*, 2012.

[80] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," in *NIPS*, 2010.

[81] D. Fouhey, V. Delaitre, A. Gupta, A A. Efros, I Laptev, and J Sivic, "People

watching: Human actions as a cue for single view geometry," in *ECCV (5)*, 2012.

[82] Y. Jiang, M. Lim, and A. Saxena, "Learning object arrangements in 3d scenes using human context," in *ICML*, 2012.

[83] A. Gupta, S. Satkin, A. Efros, and M. Hebert, "From 3D scene geometry to human workspace," in *CVPR*, 2011.

[84] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *NIPS*, 2012.

[85] Y. Jiang, H. Koppula, and Saxena A, "Hallucinated humans as the hidden context for labeling 3d scenes," in *CVPR*, 2013.

[86] A. Anand, H. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for 3d point clouds," *IJRR*, 2012.

[87] X. Ren, L. Bo, and D. Fox, "RGB-(D) scene labeling: Features and algorithms," in *CVPR*, 2012.

[88] H. Koppula and A. Saxena, "Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation," in *ICML*, 2013.

[89] H. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," in *RSS*, 2013.

[90] H. Grabner, J. Gall, and L. J. Van Gool, "What makes a chair a chair?," in *CVPR*, 2011.

[91] Y. Jiang and A. Saxena, "Infinite latent conditional random fields for modeling environments through humans," in *RSS*, 2013.

[92] B. Zheng, Y. Zhaoy, J. C. Yuy, K. Ikeuchi, and S.C. Zhu, "Beyond point clouds: Scene understanding by reasoning geometry and physics," in *CVPR*, 2013.

[93] D. Baraff, "Physically based modeling: Rigid body simulation," Tech. Rep., Pixar Animation Studios, 2001.

[94] M. McCloskey, "Intuitive physics," *Scientific American*, vol. 248, no. 4, pp. 114–122, 1983.

[95] C. Chang, B. Gorissen, and S. Melchior, "Fast oriented bounding box optimization on the rotation group SO(3, R)," *ACM Transactions on Graphics*, vol. 30, no. 5, 2011.

[96] D. Ly, A. Saxena, and H. Lipson, "Co-evolutionary predictors for kinematic pose inference from rgbd images," in *GECCO*, 2012.

[97] S. Gottschalk, "Separating axis theorem," *Technical Report*, 1996.

[98] J. Chang and J. W. Fisher, "Efficient MCMC sampling with implicit shape representations," in *CVPR*, 2011.

[99] R. C. Weng C. Lin, "Simple probabilistic predictions for support vector regression," in *Tech Report*, 2004.

[100] L.J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding:classification, annotation and segmentation in an automatic framework," in *CVPR*, 2009.

[101] C. Li, A Kowdle, A. Saxena, and T. Chen, "Towards holistic scene understanding: Feedback enabled cascaded classification models," *PAMI*, 2012.

[102] "http://bulletphysics.org," in *Bullet*.

[103] "http://www.ode.org/," in *Open Dynamics Engine*.