© 2014 Ruiqi Guo
SCENE UNDERSTANDING WITH COMPLETE SCENES AND STRUCTURED REPRESENTATIONS

BY

RUIQI GUO

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Derek Hoiem, Chair
Professor David Forsyth
Professor Dan Roth
Professor Raquel Urtasun, University of Toronto
Humans can understand scenes with abundant detail: they see layouts, surfaces, the shape of objects among other details. By contrast, many machine-based scene analysis algorithms use simple representation to parse scenes, mainly bounding boxes and pixel labels, and apply only to visible regions. We believe we should move to deeper levels of scene analysis, embracing more a comprehensive, structured representation.

In this dissertation, we focus on analyzing scenes to their complete extent and structured details. First off, our work uses a structured representation that is closer to human interpretation, with a mixture of layout, functional objects and clutter. We developed annotation tools and collected a dataset of 1449 rooms annotated in detailed 3D models.

Another feature of our work is that we understand scenes to their complete extent, even parts of them beyond the line of the sight. We present a simple framework to detect visible portion with appearance-based models and then infer the occluded portion with a contextual approach. We integrate contexts from surrounding regions, the spatial prior and shape regularity of background surfaces. Our method is applicable to 2D images, and can also be used to infer support surfaces in 3D scenarios. Our complete surface prediction quantitatively out-performs relevant baselines, especially when they are occluded.

Finally, we present a system that interprets from single-view RGB-D images of indoor scenes into our proposed representation. Such a scene interpretation is useful for robotics and visual reasoning but difficult to produce due to the well-known challenge of segmenting objects, the high degree of occlusion, and the diversity of objects in indoor scenes. We take a data-driven approach, generating sets of potential object regions, matching them with regions in training images, and transferring and aligning associated 3D models while encouraging them to be consistent with observed depths. To the best of our knowledge, this is the first automatic system capable of interpreting scenes into 3D models with similar levels of detail.
To my parents, for their love and support.
ACKNOWLEDGMENTS

I would like to first thank Derek Hoiem, for being my PhD advisor for the past five years. Derek has influenced me with his passion for solving vision problems, and provided me with encouragement and inspiration for my research. None of this work would be possible without him. Next, I would like to thank David Forsyth for sharing his unique perspective on research, his wonderful guidance and discussions, and for feeding the vision group with tasty vision lunches every Wednesday. I am deeply honored to have Dan Roth and Raquel Urtasun on my dissertation committee. Their advice and opinions have greatly helped the quality of this thesis. I also thank Lana Lazebnik, Julia Hockenmaier and Thomas Leung for their ideas and conversations.

Many thanks go to all of my lab mates: (In chronological order) Gang Wang, Ali Farhadi, Varsha Hedau, Duan Tran, Ian Endres, Zicheng Liao, Kevin Karsch, Qieyun Dai, Amin Sadeghi, Kevin Shih, Daphne Tsatsoulis, Saurabh Singh, Joeosph Degol, Jason Rock, Cecilia Mauceri, Bryan Plummer and Liwei Wang. Thanks to them, the vision group has been a very homey place for me the last five years. I am also thankful for Yunchao Gong for his late-night research discussions. I also thank my friends during my internship, Yangqing Jia, Yair Movshovitz-Attias, Yichang Shih and Kota Yamaguchi.

I am in debt to my family and friends. First, my parents, Yuming and Wen for their love, education, and letting me travel so far to the US to pursue my goals in life. I also thank Tong Meng, Ziyin Pan, Yifan Peng, Shuai Wang, Yuting Zhang and Xiaochuan Zhou for providing me with moral support during difficult times.
# TABLE OF CONTENTS

1 INTRODUCTION ............................................................. 1  
1.1 Motivation ............................................................... 2  
1.2 Our Approach .......................................................... 5  
1.3 Datasets ................................................................. 8  
1.4 Applications ........................................................... 9  
1.5 List of Content and Contributions ................................. 10  

2 RELATED WORK .......................................................... 12  
2.1 Historical Context ..................................................... 12  
2.2 Categories of Scene Understanding ................................. 15  
2.3 Relations to Our Work ............................................... 17  
2.3.1 Complete Extent ................................................... 17  
2.3.2 Modeling 3D Scenes ............................................... 18  
2.3.3 Integrating into Complete Scenes ............................... 20  

3 UNDERSTANDING COMPLETE SURFACE EXTENT ................. 22  
3.1 Labeling the Complete Scene Surfaces ............................ 22  
3.1.1 Labeling Visible Surfaces and Objects ......................... 23  
3.1.2 Using Context to Infer Labels of Hidden Surfaces .......... 24  
3.1.3 Region Overlay as a Scene and Shape Prior .................. 26  
3.1.4 Baselines ............................................................ 27  
3.2 Experiments ............................................................ 28  
3.2.1 StreetScene Dataset ............................................... 29  
3.2.2 IndoorScene Dataset .............................................. 30  
3.2.3 SUN09 Dataset .................................................... 31  
3.2.4 Per-category Analysis ............................................ 31  
3.2.5 Qualitative Results ............................................... 32  
3.3 Analysis ............................................................... 36  
3.3.1 Synthetic Example of Auto Context ............................. 36  
3.3.2 Analysis of Contextual Inference ............................... 36  
3.3.3 Scene Understanding as Polygon Detections ................. 38  
3.4 Discussion ............................................................. 40  

4 MODELING 3D SCENES .................................................. 42  
4.1 Modeling 3D Space ................................................... 42  
4.1.1 Preprocess ......................................................... 43
INTRODUCTION

Figure 1.1: How should we understand a scene? While current methods focus on labeling each pixel, we think of scene understanding as composing an interpretation with **complete extent** of layouts and objects, and with structured details on the 3D shape.

A visual scene is a view of an environment, and understanding it is crucial to our interaction with the real world. In order to plan our actions, we need to answer questions like “where can I walk” or “where can I sleep”. This requires that we find the layout surfaces of the room, identify major supporting planes and their extent, localize individual objects and recognize their shapes and functions. While many current computer vision algorithms are sophisticated in localizing and categorizing objects in isolation, solving the integrated task of understanding the whole scene remains a major challenge.

Most current systems posit the problem of scene understanding as pixel labeling (Figure 1.1b), sometimes with bounding box detection of a few categories of objects. Many of these systems developed sophisticated segment features and/or learning procedures and compete for labeling accuracies. Pixel labels are easy to obtain and straight-forward to evaluate. And they are also relatively simple from the perspective of algorithmic design. However, it is quite unintuitive for humans, since we tend to think of a scene as layout surfaces and objects as 3D models rather than separate labels of pixels. One cannot directly infer 3D models from 2D pixel labels, because pixel label maps do not encode individually the complete extent or shape details of layouts and objects. To
obtain deeper understanding of scenes, we believe that we should go a step further, by directly targeting full 3D models, as shown in Figure 1.1c.

In this thesis, we aim to address those two challenges: (1) interpreting the **complete extent** instead of only the visible, and (2) working in **structured, detailed representation**. We first highlight the plight of using only visible extent and simpler representation like bounding boxes and pixel labels. Then we leverage our two main ideas of using complete extent and detailed 3D models, by study these two problems in detail. Finally, we present a working, end-to-end system that automatically predicts such 3D models from single-view RGB-D images. To our knowledge, this is the first system capable of predicting similar 3D models.

### 1.1 Motivation

![Figure 1.2: Key Idea I: Complete Extent](image)

Scene parsing is often viewed as a problem of labeling pixels with visible categories. But these representations leave much of the underlying scene unknowable. For example, because the woman (top row) is occluded, we cannot determine what she is standing on without inferring that the bicycles are occluding the sidewalk. Likewise, finding paths through cluttered scenes is nearly impossible without reasoning about the underlying surfaces. Below, we project the ground into an overhead view (yellow=sidewalk; green=road; red=blocked by building or trees; gray=unknown). Without more complete estimates of the background, huge portions of the scene are left unknown.

The first question of designing a scene understanding system is “how much of
the scene does the system want to interpret?" The surrounding environment is infinite in its extent and shape details, and in vision we only observe a part of it. One of the most remarkable feats of the human visual system is how accurately and comprehensively it can understand the scenes even when some part of the scene is occluded. We can “see through” the foreground objects and infer the complete extent of surfaces. However, it is a problem that is seldom considered in machine vision: in many cases, the occluded part of the world is often simply ignored. Without the complete extent, the model of the scene has a lot of missing content and causes problems for higher levels of inference. This issue is illustrated in the case of Figure 1.2, where in outdoor cases, navigation is made much harder when the occluded area is incorrectly ignored. In the case of indoor scenes, the resulting model is not watertight and causes problems when interacting with them. For example, if we consider the scene in Figure 1.3a left and use it as the 3D model to play a physical simulation game, then a ball may fall through an occluded part of the floor (green area).

The second issue is finding adequate representation to interpret the scene. Most current scene understanding systems work with scene labels, object bounding boxes (mostly 2D and some 3D bounding boxes), and pixel labels. Considering how much effort the computer vision researchers have devoted into refreshing the records on recognition benchmarks, very little attention was paid to refreshing representation of the scene. We posit that a good scene representation is fully 3D, viewpoint independent, and complete. Visual data is limited by viewpoint. Portions of objects and surfaces are blocked from view, and the appearance of visible bits is subject to perspective. Moving the sensor changes the observations, yet the scene itself is not limited by viewpoint. We want our scene representation to be richer and fuller respecting shape details while representing different scene elements, and yet remain simple and realistic so that current vision techniques can work with it. Figure 1.3 showcases the strength of our proposed representation.

In short, our objective is to build a scene understanding system that is complete in its extent, and detailed in its representation. There are two challenges with this goal: (1) there is no scene dataset or annotation directly available with such complete and detailed representation, and (2) full 3D models are difficult to produce.
Figure 1.3: **Key Idea II: Full 3D Representation** In the original depth map, much of the space is incomplete because of occlusion or missing depth values. In (a), the behavior of objects when they interact with the green areas in the image on the left is completely unknowable. In our representation of the image on the right, scene models are complete and free from occlusion issues. In (b) we include a gallery of scene models under our representation. They accurately depict the details of the original image and can be used to represent a variety of complex scenes.
1.2 Our Approach

In the last section, we described criteria of a good representation of scene understanding in our mind, which extends the current mainstream scene understanding. But are we just arguing that “we have a long way to go”? We respond this question with algorithms and systems that work with our extended representation.

**Complete Extent** We formulate the problem of interpreting complete extent of surfaces as image parsing, or pixel labeling. And we label pixel into background labels for both visible and occluded extent of scenes. We offer a general, contextual approach to infer labels of occluded regions in both 2D and 3D. We incorporates several types of visual cues including estimates of visible surrounding regions, detected objects, and shape priors from transferred training regions. Then, we infer the full extent with a feedforward contextual classifier, in the spirit of Tu and Bai’s autocontext [1]. The proposed algorithm does not make restrictive assumptions about the scene structure and could be used as a default system and adapted to particular domains where appropriate. We analyze different components of the algorithm and study the performance with respect to different parameters of the feed-forward procedure. Also, the transferred polygons can be seen as a hypothesis of the configuration of scene and thus provide more structured understanding than pure pixel label predictions. To analyze the performance of such polygon prediction, we treat the problem as a polygon detection problem, evaluating how many polygons have been correctly predicted based on intersection over union (IOU) criteria.

**Structured, Detailed Representation** To support our study of complete extent and detailed 3D models of the scenes, we collected a dataset of 3D scene models which extends the current NYU Depth V2 Dataset [2]. We built a labeling tool which enables users to model scenes with a mixture of layout surfaces, extruded objects and predefined furniture models, creating a rich representation of the scene consistent with the existing detailed 2D annotations.

- **Layout structures** include floors, walls and ceilings. Because walls are always perpendicular to the floor, they are modeled as line segments in the overhead view and polygons in the horizontal view. Similarly, ceilings and floors are line segments in the horizontal view and polygons in the overhead view. We also model openings such as doorways and windows on the walls as polygons on the wall planes.
Figure 1.4: Examples of three kinds of scene elements we used for representation: (1) layout surfaces, which are walls, ceilings and floors represented by planar surfaces with openings; (2) furniture like chairs and tables modeled with CAD models; (3) extruded polygons for objects we do not have detailed CAD models for such as bottles and shoes.

- **Furniture** objects are common in indoor scenes and tend to have complicated 3D appearance. For example, chairs and sofas cannot be modeled using cubic blocks. To accurately model them, we use Google SketchUp models from the Google 3D Warehouse repository. We manually select 30 models to represent 6 categories of furniture that are most common: chair, table, desk, bed, bookshelf and sofa. The support surface of each object is labeled by hand on each predefined 3D model.

- **3D extruded models** are used to describe all other objects. Most clutter objects are small and do not have a regular shape, and we model them as polygons extruded from their support surface. Examples of these 3D models are shown in Figure 1.4.

Our pipeline of annotation is semi automated in the sense that users are guided by depth data and existing 2D annotation. We believe that the proposed dataset can benefit many research topics in scene understanding, such as surface layout prediction, object recognition and scene reconstruction. Such annotation has
been adapted by Reconstruction Meets Recognition Challenge (RMRC) [3] to provide 3D bounding boxes of object detection. We also present a pipeline for complete extent prediction of support surfaces, as an extension of the 2D version of the algorithm, by simply adapting the features to 3D space.

**Predicting Full 3D Models** We present a realistic system that takes a single RGB-D image as input and interprets it into our desired representation, similar to the annotation we are proposing in the collected dataset.

To infer full 3D models, we take a data-driven approach. First, we generate sets of potential object regions. The regions are then matched to exemplar regions in training images. Next, 3D models associated with the exemplar regions are transferred and aligned based on observed depths. Finally, we select a set of object and layout proposals that are most consistent with each other and with observed depth values. The problem is complex and multifaceted, and our approach likewise has several components, including parsing layout surfaces, automatic segmentation, region retrieval, fitting 3D objects and composing multiple scene elements. Each of these topics can be studied in its own right and perhaps in a separate thesis. Here, we focus on how to put these components together to construct an end-to-end system. To our knowledge, this is the first integrated system operating with similar representation.

![Figure 1.5: Key Idea III: Full 3D Prediction](image)

Our representation is not just for annotation. We build a working system that predicts such full 3D models from RGB-D images, without human interaction. In the example above, our automatic prediction (middle) comes very close to the ground truth model (right). In fact, it selects more accurate models for sofa than the annotation.
1.3 Datasets

We work with two types of dataset: RGB and RGB-D datasets, as shown in Figure 1.6. Our complete extent algorithm works on both types of datasets. Our full 3D model prediction framework operates with RGB-D data, because it relies heavily on depth maps from sensors.

Figure 1.6: Examples of RGB and RGB-D dataset used in this thesis. The RGB datasets use LabelMe style polygon annotations while for RGB-D we annotate scenes with full 3D models.

**RGB dataset.** Clean, well-annotated data is indispensable to understand structured, complete scenes. Recent years saw a big increase of scene datasets. Image scene datasets like LabelMe [4], StreetScene [5] and Barcelona [6] provide crowd-sourced object annotation of hundreds of categories. The annotation process involves outlining the object boundary with a polygon and introducing a name from a set of tags [7]. When regions are not fully visible, people use their judgement to fill the occluded part. According to notes of the annotator of LabelME [7]: “As the boundaries of the walls and floor are not visible, one has to take into account the perspective of the scene.”
For us, using existing datasets is first of all very convenient since we can harvest the complete extent from current annotations without any additional effort. Secondly, we essentially think that these datasets are not utilized to their fullest potential, because most researchers simply work with the “flattened” pixel maps. In fact, these datasets convey more than pixel maps since they are annotated with polygons indicating their full extent. Polygon masks contain strictly more information than pixel maps, because one can easily convert masks to pixel maps but not the other way around. In contrast, by considering each polygon individually, we can evaluate our performance in terms of each individual object polygon as oppose to each pixel, giving a more structured understanding than pixel accuracy. For predicting complete extent on RGB images, we use Internet image datasets, including StreetScene [5], IndoorScene [8] and SUN09 [9], to evaluate our performance.

**RGB-Depth dataset.** We use our extended version of annotation on NYUv2 [10] since it is the only one with similarly annotated representation. NYUv2 [2] is an indoor scene dataset of RGB-Depth data. It consists of 1449 RGBD images, gathered from a wide range of commercial and residential buildings in three different US cities, comprising 464 different indoor scenes across 26 scene classes. A dense per-pixel labeling was obtained for each image using Amazon Mechanical Turk. We extend its 2D annotation with three kinds of scene components.

We create our dataset with the intention of representing “everything we want to infer” about the scene, to support a variety of evaluation criteria. Currently, we use it for extent prediction and voxel accuracy on occupancy and freespace to measure our performance. However, there is more in the dataset than we are using currently. For example, we include doorways and windows as well as the orientations of our furnitures, even though they are not currently being predicted.

### 1.4 Applications

Applications of complete and detailed 3D model predictions can be found in many areas including robotics, surveillance, computer graphics. For example, they can be used in:

**Robotic navigation.** In navigation, an overhead view of the scene is often used to help plan motion paths and avoid collision with objects. Since occlusion
often creates unknown areas in the overhead view, it is important that we obtain the complete extent of the surfaces like Figure 1.2.

**Architectural floor plan prediction.** In indoor scenes, floor plans are useful for designing houses, home decoration and housekeeping. Full 3D models of layouts and furniture from our prediction can be easily converted to floor plans by rendering the models in the overhead view.

**Scene editing.** Karsch et al. [11] rendered scenes with synthetic objects added into existing scenes. In industry, the furniture maker IKEA has also released an augment reality smartphone app [12] to help people make furnishing decisions.

**Augmented reality.** Video game designers have also tried to bring digital game experience to a real world environment, for example, Illumiroom [13]. Predicted 3D models from scenes are beneficial to such applications, since they give complete geometric details of the surrounding environment.

## 1.5 List of Content and Contributions

In this thesis, we first survey the background of our work in Chapter 2. We overview these previous literatures and categorize them into 5 different categories: (1) Holistic Scene Recognition (2) Object Centric Understanding (3) Surface Centric Understanding (4) Layout Recognition and (5) Functional Centric Understanding. Then we discuss the most relevant ones in technical detail and relate them to our work in terms of the underlying assumptions and difference in representations to emphasize the novelty of our work.

This thesis contains material from previous publications during my PhD program; these parts are described in more detail and are presented in later chapters. The most important contributions of this thesis include:

**Complete Extent** We systematically study the problem of parsing extent in images, which has not been done before. We formally define it as a parsing problem on the regions that are both directly visible and occluded based on polygon annotations of datasets. We address the problem of finding complete extents from RGB images. We propose a contextual approach to infer extents that are not directly visible and study its effectiveness as
compared to traditional methods like Markov Random Field. (Chapter 3. Work appeared in ECCV2012 [14]).

**3D Representation** We push the representation of 3D scenes beyond bounding boxes and pixel labels and propose a set of full 3D representation with CAD models, extruded objects and layout planes. We created tools and a dataset extending the NYUv2. To show the usefulness of our dataset, we present an adaptation of the method in Chapter 3 to the case of 3D support surfaces, which are the functional surfaces of objects capable of supporting other objects. Models can be used to infer extent of support surfaces in 3D space. (Chapter 4, work appeared in ICCV2013 [10].)

**Predict full 3D model** The final piece of our contribution is a fully automatic system to parse RGB-D images into full 3D models. Previous works used solid box-like layout surfaces, very few object categories with 3D bounding boxes.

Our predicted 3D models cover flexible layout surfaces (with openings), hundreds of categories of objects including 6 categories with CAD models and complete extent of the scene. To our knowledge, this is the first working system capable of predicting models with similar levels of detail and completeness like Figure 1.1. Prediction results are evaluated in form of layout prediction, semantic retrieval and 3D voxel accuracy. (Chapter 4, work may appear in future publications.)
2 RELATED WORK

2.1 Historical Context

Scene understanding, as suggested by the “scene” in its name, is concerned with the view of the environment. Unlike object recognition or segmentation, which give more attention to individual components that we act upon, scene understanding explores the extent of the space that we interact with. The denotation of the word “understanding” however, is often less well defined. Philosophically speaking, it can mean two things. One meaning is to give an objective, observer-independent view, or ground truth account of the scene. And the only goal of the vision system is to be able to predict exactly that. Most of the time computer vision researchers take this view, especially when it comes to experimental designs.

The other meaning is to generate an observer-dependent interpretation of the scene - not necessarily the ground truth one, but rather one of many representations that is consistent with one’s rationale and experience. After all, human interpretation may also vary among different people and they may not always agree. The BSDS500 segmentation dataset [15] followed this philosophy and allowed multiple ground truths. Similarly, for scene understanding, there can be multiple “correct,” or “acceptable,” explanations and lean towards not having a single “god-view” of the world. Therefore it is important that a vision system can produce multiple guesses of the configuration of layouts and objects in the scene. This is reflected in our final system for full 3D prediction.

Another very important issue about scene understanding is the scene representation. We can represent a scene with just a single label, for instance “indoor” or “outdoor.” Or, it is possible to hire professional 3D modelers and use high precision CAD models to model scenes. There is not a single answer for which representation is optimal. The decision largely depends on how many structures we need for specific applications. Human vision systems are good at memorizing structures of scenes, while forgetting a lot of the low level
Figure 2.1: Representation of an office as schematic map/plot [16], which characterizes both the category and spatial locations of objects. To this day, converting images (or RGB-D images) into such plots is still a major challenge to computer vision systems.

details [17]. This means we are inherently more interested in scene structures, including the extent of layout surfaces and object categories, than in fine geometric details.

It is interesting though, that pioneers of our field used representations often more sophisticated than those used now. Early cognitive scientists had developed symbolic or schematic maps [16, 18] like Figure 2.1. Some early works used line drawings [19] and stick models [20], as shown in Figures 2.2 (a) and (b). There is also a wide range of literature using other geometric objects like generalized cylinder [21, ?], industry parts [22]. Overall, object and scene recognition research was largely based geometric details until and late 1990s and early 2000s, and we refer readers to [23] for a more comprehensive review on early geometric works.

Since the late 1990s, bounding boxes and pixel labels (Figures 2.2 (c) and (d)) have become dominant because of their simplicity, and have over-shadowed all other representations. However, as computer vision techniques advance, we believe we will eventually move back to more detailed representations. For daily scenes, the main things to focus on are the overall layouts and prominent objects. In additional to their 3D locations, for surfaces, we also want to know whether or not they can support other objects, and whether they contain openings like doorways and windows. For objects, we also want to know their semantic categories, the CAD models they belong to, and the support relations between objects.
Figure 2.2: (a) Line drawings and (b) stick models are often richer in detail than (c) bounding boxes and (d) pixel labels. The problem is that (a) and (b) are not the easiest representation for prediction and evaluation.
2.2 Categories of Scene Understanding

Often, the philosophical stance of “interpreting” versus “understanding” is not
important because vision researchers are more concerned about “whether this is
useful and possible.” In practice, people simply use the word “scene
understanding” as an umbrella term for a lot of scene-related vision tasks. Often,
as long as a task tries to infer something about the scene, such as geometric
layout, illumination properties, objects’ shapes and locations, GPS locations, or
human activities, it can be seen as a type of scene understanding. Since the early
days of scene understanding, researchers have proposed a spectrum of
algorithms. According to the focus of these algorithms, they can be categorized
as follows:

**Scene Label Recognition** categorizes a scene, by naming its scene label (e.g.
forest, street, bedroom). Such works look at the global configurations of the
scene, and describe a scene using its “gist,” ignoring most of the local details.
Such techniques include Spatial Envelope [24] and Spatial Pyramid
Matching [25], which aim to describe the coarse structure of the scene using
frequencies and gradients. They have been applied to scene categorization, and
are evaluated based on the classification accuracies. Recently, there has been a
move to bigger image classification datasets such as ImageNet [26] and
SUN [27] dataset. Features and methods suitable for large datasets have also
been developed, such as fisher vectors [28] and convolutional neural
networks [29, 30, 31, 32].

**Object Centric Scene Understanding** focuses on recognition around object
detection in a scene. It measures performance of scene prediction in terms of
corrected localized tight bounding boxes, where the correctness of object
detection is defined on the intersection over union (IOU) scores between
predicted and ground truth bounding boxes. In many cases, researchers believe
that the tasks of scene recognition and object detection are mutually enhancing.
They aim to use scenes as context to improve object detection. They first detect
candidate objects, using object detectors such as Deformable Part-based Model
(DPM) [33], and then use contextual information to find the most likely joint
configurations. Researchers have proposed a myriad of ways to model the
relationship between objects and scene labels [34], objects and surfaces [35],
objects and objects [36, 9, 37, 38, 39], as well as objects and surrounding
regions [40]. In more recent years, a number of papers have also extended their
detections to 3D domain by using depth features on both outdoor [41] and indoor [42, 43, 44] imagery.

**Surface Centric Scene Understanding** labels scenes with surfaces and object mask. Different from object centric approaches, those who use these approaches believe pixel maps are more expressive than bounding boxes. They assign each pixel of the scene with a label of semantic categories [45, 46, 47], geometry [48] or illumination [49]. This task is also called semantic segmentation or image parsing. These methods usually start with over-segmenting the image into atomic regions [50], and then classify each region based on its appearance features such as textures and colors. Then contextual information is integrated using Markov Random Field (MRF) or Conditional Random Field (CRF) to enforce the label consistency over a local neighborhood [51, 6]. Finally, they measure performance in terms of pixel accuracy.

**Layout Recognition** departs from surface-based scene understanding in that it tries to create a more structured representation than pixel labels. Researchers on layout recognition utilize the fact that scenes, especially indoor ones, have strong regularity, such as planar surfaces that form right angles to each other. For example, Hedau et al. [8, 52] tried to recognize the scene in a constrained, box-like structure. A line of works from Schwing et al. [53, 54] showed that such a box layout can be exactly solved using branch and bounding inference. However, this assumption of the 3D world can be too restrictive; for instance, Hedau et al. [8] assumed that everything in the room had to be boxy and aligned with the axes of the room. Flint et al. [55] and Furukawa et al. [56] constructed room layouts with Manhattan structures, which is more flexible than box layout. They used explicit 3D models therefore they can infer complete extent of surfaces and objects. Nonetheless, most of the scenes they tested were clean and simple scenes and did not have foreground clutter. Like surface-based understanding, they also evaluate layout regions with pixel accuracy on layout classes like “floor,” “ceiling,” “front wall,” “left wall,” “right wall.”

**Functional Centric Scene Understanding** reasons about a scene by considering the functions of objects and their relationship to humans. It follows the intuition that scenes are often arranged in certain configurations to facilitate human interaction. Gupta et al. [57] considered the affordance of surfaces, and used human poses as cues for 3D scene understanding. Fouhey et al. [58] directly used the observed human actions from video sequences. Jiang et al. [59] jointly modeled human configuration and object affordance in a topic model. Such high
level inference can be potentially helpful in robotic applications such as guiding personal robots to arrange the house in ways preferred by humans.

2.3 Relations to Our Work

The problem we are studying is mostly related to object and surface, as well as layout centric understanding. However, we extend these approaches in two ways: (1) by giving surfaces complete extent, and (2) by representing objects with more geometric details. Next, we will contrast previous works with ours in terms of context and technical differences.

2.3.1 Complete Extent

Most image parsing algorithms work with the visible part of the scene. However, there are also exceptions. Some publications solved for the layout surfaces in indoor rooms [60, 8, 61], which do cover the complete extent of indoor surfaces. However, they assume boxy structures for all rooms, which in many cases is an over-simplified representation. Another problem is that layout estimation from single-view RGB images suffers from mistakes caused by vanishing point algorithms. Geiger et al. [62] worked with outdoor street scenes but again had strong assumptions about the shape of the road. Hsiao and Hebert [63] handled occlusion in an explicit way with no restrictive scene models, but their approach required very detailed object models.

We have proposed a feed-forward contextual framework for the complete extent of surfaces [14]. Our method makes no assumption about the underlying semantic categories or the structure of scenes and therefore can serve as the default method in many image parsing systems. In this sense, our work is similar to GraphCut which is also generally applicable to many domains.

MRF and CRF are often used in image parsing as a way to integrate local evidence and provide inference for the whole image. In particular, pairwise submodular MRFs [64] are often used as post-processing in image parsing systems. Despite the nice property that they can be minimized using Graph-cut, the main problem with them is that the pairwise submodular constraints restrict the form of the potentials, making the modeling process less flexible. When such constraints are met, the behavior of the random field often becomes just
smoothing. Higher order potentials may be added [51] to enable greater flexibility of the model but often at the expense of needing an approximate and complicated inference process. Our method for complete extent accounts for long-range interplay of the pixel labels, and we show this point through our experiments on datasets of indoor and outdoor scenes. We also visualize the different behavior of MRF and auto context on synthesized examples in Chapter 3. We later extend our complete extent algorithm for support surfaces in 3D in Chapter 4.

Some papers appeared after our approach was first published also carry similar ideas. Kosov et al. [65] examined the labeling problem of foreground and background pixels the same way as in [14], and used two layers of CRF. Isola and Liu [66] were interested in composing a scene with the right depth ordering using the a collection of polygons instead of pixels, which also addresses the problem of the complete extent. Recently, Silberman et al. [67] proposed a method for finding the complete extent of 3D support surfaces, which complete the occluded extent with lines and parabolas.

2.3.2 Modeling 3D Scenes

The biggest difference between our work and that of others is the use of more expressive representation.

Layouts. Many works modeled scenes as the layout of walls, floor, and ceiling with a 3D box (e.g., [8, 55, 53, 54]), which provided a good approximation to the space. Such box layouts can be inferred from an RGB image, based on vanishing point and other perspective cues. Others [68, 60], estimated a more flexible layout, with Manhattan-structured perpendicular walls, using visible floor-wall-ceiling boundaries. In robotics, some researchers [69] recovered wall layouts from a RGB-D image, based on plane fitting and dynamic programming.

Researchers in graphics are also interested in recovering axis-aligned, piecewise-planar models of indoor scenes from large collections of images [70] or laser scans [71]. The fact that the depth cloud is more reliable than single-view image cues often leads to more precise 3D models. On the other hand, they do not work with single view imagery, and some of their algorithms [71] required a Kinect-Fusion style full scan of the room [72].

We model the exterior structure of the room with a collection of axis-aligned
planes with cutouts for windows, doors, and gaps from single view RGB-D images. Such representation is flexible, because the layouts are axis-aligned, but not necessarily a box. We also look for openings, which are ignored by most previous literature. Despite the complexity of the model, we show that we can infer both visible and occluded surfaces with high accuracy by proposing sets of vertical and horizontal planes and jointly inferring a set of layout surfaces and objects that satisfies depth values observed from one viewpoint.

Some researchers consider the contextual interactions between surfaces and objects. The simple intuition that objects usually fall onto a sparse set of support surfaces was often used [35, 38, 73] to boost object detection performance. But the support planes are generally considered implicitly and indirectly – they were not detected using appearance models, nor were they evaluated. As long as the support planes improve object detection, whether the support planes themselves are accurate is not explicitly considered. We want to know more details about the support surfaces: is the support surface a layout surface? Or is it a support surface from a piece of furniture? How well are we predicting the extent of these surfaces?

**Objects.** Bounding box representation is a crude representation of objects when it comes to many daily objects like chairs, tables, sofas. These objects are not boxy and need more precise models, such as CAD models. Over the years, recognizing CAD models has attracted a lot of attention [74, 75, 76]. However, the scale and the quality of CAD datasets have always been a problem. Recently, as large-scale CAD model datasets such as Google/Trimble 3D Warehouse [77] become available, there is recurrent interest in the idea of recognizing CAD models. Lim et al. [78] finds IKEA furniture instances from RGB images and Aubry recognized chairs with DPM [33]. Because of the variation in object geometric shape, appearance, illumination and occlusion conditions, finding CAD models from single-view RGB images is not an easy task. Previous papers mainly considered just a few categories. For example, Aubry et al. [79] considered only one category - chair. Song and Xiao [44] considered three categories: chair, bed and toilet. This is quite limiting since so many other objects are not modeled. For example, there are more than 800 categories in NYUv2 [2]. This is because these detector-based approaches need a lot of examples for training. By contrast, transfer-based approaches require no training stage and can work with currently available datasets.

We want our system to be able to capture hundreds of object classes. To
accommodate large number of object categories, we take a transfer-based approach. Our method is inspired by the SuperParsing method of Tighe and Lazebnik [6], which transferred pixel labels from training images based on retrieval. Similar ideas have also been used with other modalities: Karsch et al. [80] transferred depths, Guo and Hoiem [14] transferred polygons of background regions, and Yamaguchi et al. [81] transferred clothing items. Satkin and Hebert [82] transferred 3D geometry and object labels of entire images. Our approach differ from all of them because we transfer 3D models of individual objects, and also allow each transferred object to scale, rotate and translate.

2.3.3 Integrating into Complete Scenes

There is a rich body of work on object-object relations. 2D co-occurrence and spatial relationships have been used to improve object detection and image parsing [34, 83, 84, 85, 40, 36]. However, they were designed for 2D images and were not applicable to RGB-D imagery. One paper that considered full 3D scenes, Satkins et al. [86, 82], used ranking and retrieval to 3D CAD models that correspond to complete room models according to single-view visual cues. On the positive side, their matching process is viewpoint invariant and the viewpoints of their output 3D models are always coherent (because all of the 3D objects come from the same exemplar scene). On the other hand, retrieving based on entire images is constraining because the chance of finding two rooms with the same configuration is small. The fact that they do not perform any “editing” to the matched scene greatly limits their ability to adapt to the new images. To make the retrieval process more flexible, Russel et al. [87] found confident partial matches, transferred 2D regions of individual objects and surfaces, and composed them into a coherent 2D model. Shao et al. [88] also composed scenes from individual objects, but their approach required user intervention. In contrast, our composition system, as described in Chapter 5, is fully automatic and requires no human interaction.

Instead of retrieving 3D models based on the appearance of entire images like Satkin et al. [86, 82], our approach retrieves 3D models of objects based on segmented regions. Super pixel regions used by image parsing literatures [35, 6] are not suitable for our purpose because super pixels tend to be small and may not correspond to complete objects. We propose a bag of object-like regions,
transfer the retrieved 3D models, and resolve conflicts of these 3D models in a final composition process that accounts for fidelity to observed depth points, coverage, and consistency. In that way, our approach also relates to work on segmentation [89, 90], parsing of RGB-D images [91, 92] and generation of bags of object-like regions [93, 94, 95]. After segmentation and region retrieval, we incorporate a per-object 3D alignment procedure based on Iterative Closest Points (ICP) which checks the consistency between the retrieved 3D models and the observed depths cloud, and also recover scaling, rotation and translation of the matched 3D models. The retrieved 3D models provide an over-complete pool of 3D models for objects in a scene, and we select a coherent subset of them to compose the final results.
3 UNDERSTANDING COMPLETE SURFACE EXTENT

When understanding a scene, we want to interpret both the part that is directly visible and the part that is being occluded. In this chapter, we detail our proposed contextual inference procedure for finding the complete extent on both these parts. We believe a good complete extent prediction algorithm should be:

1. Generally applicable to multiple domains and use less restrictive assumptions.
2. Efficient so that it can be used as a fast post-processing step.
3. Robust and able to outperform traditional inpainting methods.

Our approach incorporates existing techniques, such as region classification [48], object detection [33], feed-forward contextual prediction [1], and non-parametric label transfer [96]. We expect that contextual recognition algorithms and domain-specific scene layout algorithms would benefit from having our more complete scene layout estimates as a starting point for more complex reasoning.

3.1 Labeling the Complete Scene Surfaces

In this section, we describe our labeling algorithm for complete scene layout. We first describe how to label the visible part of the scene, using an off-the-shelf image labeling algorithm and pre-trained object detectors. We then incorporate visible information into a feed-forward contextual prediction to infer the occluded part of the background regions. Next, polygons are matched based on the current label confidences to provide a shape prior for each label. The final pixel prediction incorporates visible surface predictions and the transferred polygons to provide the final complete background labeling. The full pipeline to find the complete scene layout is summarized in Figure 3.1. We also describe three baseline methods for inferring occluded background labels.
3.1.1 Labeling Visible Surfaces and Objects

We apply the region classifier from Hoiem et al. [48] to label pixels into visible foreground and background regions. The image is first over-segmented into superpixels, which are then grouped into multiple segmentations. Color, texture, edge, and vanishing point cues are then computed for each superpixel. Finally a boosted decision tree classifier combines the prediction and estimates the likelihood of each possible label for each pixel, providing a confidence map for each label.

This algorithm is designed to parse geometric classes and works well for background regions such as building, road and trees, outperforming SuperParsing [6] on these classes (see Figure 3.4 in the experiment section). We use Felzenswalb et al.’s object detector [33] (pre-trained on PASCAL07 dataset [97]) to detect foreground objects such as “cars” and “pedestrians”. For each category, we convert the detection bounding boxes to a detection confidence map, where the value of each pixel is set to the maximum of the scores of the detection window that contains that pixel, as shown on the second image from the left on Figure 3.2.
Complete Scene Labeling Algorithm

/* Training */
Perform object detections and image parsing of visible surfaces in all training images
Let \( \{ V_i \} \) be the confidence maps of region classification and detection for each image \( i \)
/* Main loop of auto-context */
Sample image \( i \) and position \( x \) from \( V \) and \( Y \) to build training set \( S_0 = \{ Y_{ix}, V_{i,N(x)} \} \),
where \( Y \) is the ground truth map of the complete scene and \( N(x) \) is the radial distributed
neighborhood of position \( x \). Train classifiers using logistic regression and obtain parameters \( w_0 \).
For \( t = 1 \ldots T \)
   Apply previous classifiers with parameters \( w_{t-1} \) to all training images.
   The resulting probability maps of label predictions are \( P^{(t-1)} \).
   Build a new training set, now with feed-forward context \( S_t = \{ Y_{ix}, V_{i,N(x)}; P^{(t-1)}_{i,N(x)} \} \).
   Train classifiers using logistic regression on \( S_t \); the learned parameters are \( w_t \).
End
/* Incorporate shape prior */
For each training image \( i \), retrieve polygons from training images that match the \( P^{(T)}_i \).
Compute shape prior \( \{ Q_i \} \) by flattening retrieved polygons.
Construct final training set \( S_{\text{final}} = \{ Y_{ix}, V_{i,N(x)}; P^{(T)}_{i,N(x)}; Q_{i,N(x)} \} \)
Train the final classifier with \( S^{\ast} \) with parameter \( w^{\ast} \).

/* Testing */
For a testing image \( k \), do object detections and region classification of visible surfaces.
Let the result be \( V_k \). Apply classifier \( w_0 \) to \( V_{k,N(x)} \) for each position \( x \).
The resulting confidence map of label prediction is \( P^{(0)}_k \).
For \( t = 1 \ldots T \)
   Apply classifier \( w_t \) to \( [V_{k,N(x)}; P^{(t-1)}_{k,N(x)}] \) for each position \( x \). The result is \( P^{(t-1)}_k \).
End
Compute shape prior \( Q_k \) by retrieving polygons from training images.
Apply \( w^{\ast} \) to \( [V_{k,N(x)}; P^{(T)}_{k,N(x)}; Q_{k,N(x)}] \) to get final prediction map \( P^{\ast} \).

Figure 3.2: **Outline of our full algorithm.** We use three kinds of features: (1) visible parsing results, (2) object detection. Then, we apply our major intuitions: spatial contexts and retrieved shape prior are combined to infer the complete label map and the polygonal layout.

### 3.1.2 Using Context to Infer Labels of Hidden Surfaces

We base our approach to infer background labels of occluded pixels on the auto-context framework [1]. In the original auto-context paper, the algorithm starts by learning an appearance classifier using image patch features and a boosting algorithm. After applying the trained classifier, the confidence map is then fed as contextual information to train the next classifier. The surrounding label confidences of a pixel are used as features to construct a new training feature set. A new boosted classifier based on the training set is learned and it, in turn, updates the confidence map. The algorithm iterates this process, improving the ground truth likelihood of training labels in each iteration.

We use the same feed-forward idea. However, rather than using appearance features (which would only describe the visible surfaces), we instead directly
rely on the outputs of our region classifier and object detectors. As suggested in Tu and Bai [1], we sample the contextual features in the form of sparse, radially distributed points. We use logistic regression for classification instead of boosting as our classifier so that feature computation and applying of the classifier can be done by 2D convolution operations and linear additions over the whole image. This allows us to reduce the testing time to under 1 second per image (400×300 pixels), whereas the original algorithm runs at 30 to 70 seconds on a 300 × 200 image.

Due to its discriminative training, auto-context can go beyond simple smoothing. For example, in Figure 3.3, auto-context can recover the sidewalk region by looking at the prediction of nearby pixels: if there is building on top of it and road below it, then it is more likely to be sidewalk. An initially missed sidewalk region is recovered after three iterations of auto-context.

![Figure 3.3: Illustration of baseline methods. (a) shows the original image and (b) gives the parsing map of visible part; (c)-(e) Baseline methods do not go beyond smoothing. (f) and (g) shows incorporating context using classifier does more than a smoothing term, recovering the some sidewalk area. After polygons are retrieved, the flattened shape prior (h) helps to regularize the layout and gives final output (i).]
3.1.3 Region Overlay as a Scene and Shape Prior

Intuitively, the overall pattern of labels should be similar to other images observed in the training set, and the pattern of a particular type of label is likely to match some training image quite closely. We operationalize this intuition by finding polygons in the training set that match our current label predictions. These polygons provide a scene prior (because the training image that they come from should have similar labels overall) and a shape prior (because the transferred region maintains its shape). The transferred regions can be used to refine our per-pixel background labels, and the set of transferred regions provide alternative coherent yet compact hypotheses about the hidden portions of the scene.

We find polygons for each background class separately using the intersection over union criteria. From the previous steps, for each query image \( k \) we computed probability map of the background label \( l \) as \( P_{l,k} \). We then find the best polygons that match it and directly lay down the polygons as our region prediction for image \( k \), label \( l \).

The ground truth mask of background label \( l \) in training image \( i \) is \( G_{l,i} \), where \( G_{l,i,x} = 1 \) if pixel \( x \) of image \( i \) is of the label \( l \) in the ground truth annotation and 0 otherwise. Then the fitting score of image \( i \)'s polygons to \( P_{l,k} \) is defined as:

\[
Score(G_{l,i}, P_{l,k}) = \frac{\sum_x \min(G_{l,i,x}, P_{l,k,x})}{\sum_x \max(G_{l,i,x}, P_{l,k,x})}
\]

This can be interpreted as a weighted version of region overlap score for two polygons. For each class \( l \), we select the top image \( i \) from the training images set whose matching score is highest for query image \( k \) and is bigger than a threshold \( t = 0.3 \).

Similarly, we define global matching score as:

\[
Score(G_i, P_k) = \frac{\sum_l \sum_x \min(G_{l,i,x}, P_{l,k,x})}{\sum_l \sum_x \max(G_{l,i,x}, P_{l,k,x})}
\]

To preserve global layout similarity, we only consider matching polygons whose image global matching scores \( Score(G_i, P_k) \) are among the highest \( R = 200 \) images.

Our initial set of polygons might overlap or leave some pixels unlabeled. However, they preserve the simplicity of the real world regions and can be
Table 3.4: (a) Pixel accuracy on StreetScene dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Complete</th>
<th>Occluded</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Confident</td>
<td>0.795</td>
<td>0.601</td>
<td>0.810</td>
</tr>
<tr>
<td>Nearest</td>
<td>0.798</td>
<td>0.619</td>
<td>0.812</td>
</tr>
<tr>
<td>Graphcut</td>
<td>0.803</td>
<td>0.615</td>
<td>0.818</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.833</strong></td>
<td><strong>0.715</strong></td>
<td><strong>0.843</strong></td>
</tr>
<tr>
<td>Superparsing</td>
<td>0.775</td>
<td>0.453</td>
<td>0.800</td>
</tr>
</tbody>
</table>

(a) Pixel accuracy on StreetScene dataset

(b) The effectiveness of different cues of our framework

<table>
<thead>
<tr>
<th>Method</th>
<th>Complete</th>
<th>Occluded</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Confident</td>
<td>0.795</td>
<td>0.601</td>
<td>0.810</td>
</tr>
<tr>
<td>Shape Prior Only</td>
<td>0.818</td>
<td>0.713</td>
<td>0.826</td>
</tr>
<tr>
<td>Ours, w/o Shape Prior</td>
<td>0.832</td>
<td>0.705</td>
<td>0.842</td>
</tr>
<tr>
<td>Ours, w/o Detection</td>
<td>0.831</td>
<td>0.705</td>
<td>0.841</td>
</tr>
<tr>
<td><strong>Ours, Full</strong></td>
<td><strong>0.833</strong></td>
<td><strong>0.715</strong></td>
<td><strong>0.843</strong></td>
</tr>
</tbody>
</table>

(b) The effectiveness of different cues of our framework

Figure 3.4: (a) shows our result compares favorably to the baselines. The “Most Confident”, “Nearest” and “Graphcut” are based on the confidence maps of training the classifier of Hoiem et al.. The SuperParsing results shown are produced in the Most Confident fashion. Using Nearest or GraphCut on SuperParsing results yields similar performance. (b) Explores the effectiveness of each component in our framework, also on StreetScene dataset. “Shape Prior Only” correspond to the pixel map of the best polygonal layout guess, as shown in Figure 3.3 (h). “Ours, w/o Shape Prior” has everything except transferring shape prior. “Ours, w/o Detection” uses everything except object detection cues.

helpful for further inference. To resolve the overlapping issue, we assign the overlapping region exclusively to the polygon which has the highest confidence in the overlapping part.

We then create a polygonal shape prior by putting down the polygons and assign the unlabeled pixel to its nearest polygon region. This gives a clean, polygonal layout of the complete scene and is fed back into our context classifier for final training. Since visible part of the sky and trees cannot be occluded by other background regions and usually has complicated boundaries, we do not put any shape prior on those regions.

3.1.4 Baselines

Other general-purpose methods to infer labels of occluded regions do not exist in the literature, so we provide several baselines. Each method attempts to predict the labels of the underlying surfaces, given the label confidences for the visible
surfaces.

**Most confident background** assigns each foreground pixel to the most confident background label.

**Nearest** method assigns occluded background pixels to the nearest (in image location) visible background pixel.

**Graph-Cut** implements a pixel-wise MRF, which is often used in post-processing stage of semantic segmentation. The setup of the MRF is similar to that of Shotton et al. [46], where each pixel is represented a node in the graph. The unary term contains the log probability that a pixel $x$ of a query image $k$ has background class label $l$: $\psi_{\text{unary}}(x, l) = \log P(l; x) = \log P_{k,l,x}$ directly from the output of our visible parser. The pairwise term enforces contrast-sensitive boundaries in visible regions and uniform smoothing in occluded regions for adjacent pixels $x_1$ and $x_2$:

$$
\psi_{\text{pairwise}}(x_1, x_2, l_1, l_2) = 1(l_1 \neq l_2)[\lambda_1 P(fg|x_1) \\
+ \lambda_2(1 - P(fg|x_1))e^{-(l(x_1)-l(x_2))^2/\sigma^2}]
$$

where $P(fg|x_1)$ is probability that $x_1$ is in a foreground region. $\sigma$ is a parameter that controls the amount of smoothing. $\lambda_1$ and $\lambda_2$ modulate how much label smoothing we want for visible and occluded portions of the image. We use alpha-beta swaps [98] to solve the MRF.

### 3.2 Experiments

In this section we show both quantitative and qualitative results for predicting scene layout on three different datasets, StreetScenes, IndoorScenes and SUN09. We evaluate the pixel labeling accuracy of the background categories on all three datasets. For each experiment, we use 3 iterations of feed-forward training. The ground truth pixel map is created with only complete background classes, as if the foreground objects are not there. Unlabeled regions in the ground truth are ignored for evaluation. Our pixel accuracy is reported as the average pixel accuracy over images.
<table>
<thead>
<tr>
<th>Method</th>
<th>Complete Scene</th>
<th>Occluded</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Confident</td>
<td>0.700</td>
<td>0.658</td>
<td>0.710</td>
</tr>
<tr>
<td>Ours</td>
<td>0.739</td>
<td>0.729</td>
<td>0.742</td>
</tr>
<tr>
<td>Hedau et al. [8]</td>
<td>0.796</td>
<td>0.698</td>
<td>0.821</td>
</tr>
</tbody>
</table>

(a) Pixel accuracy on IndoorScenes

<table>
<thead>
<tr>
<th>Method</th>
<th>Complete scene</th>
<th>Occluded</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Confident</td>
<td>0.639</td>
<td>0.498</td>
<td>0.665</td>
</tr>
<tr>
<td>Ours</td>
<td>0.691</td>
<td>0.661</td>
<td>0.695</td>
</tr>
</tbody>
</table>

(b) Pixel accuracy on SUN09

Figure 3.5: The overall accuracy on testing set of IndoorScenes and SUN09 dataset, our method greatly helped occluded part of the scene, and made slight improvement on the visible portion.

3.2.1 StreetScene Dataset

The StreetScenes dataset consists of 3547 high quality images of urban environments, in which 2837 images are for training and the rest for testing. The dataset is labeled with polygonal, complete region labels by trained annotators. In this dataset, background classes as “road” and “sidewalk” is often heavily occluded by foreground objects like “car” and “pedestrian”.

All three baselines described in Section 3.1.4 are tested on this dataset and provide very similar performance (Figure 3.4(a)). Most Confident and Nearest baselines methods leave the pixels predicted as background intact; they cannot correct any mistakes on the visible part of the scene. The Nearest method mainly works if the visible prediction is smooth and correct, but the method fails when the visible prediction is cluttered. Notice that Graphcut also made almost no improvement compared to Most Confident, because it does not go beyond smoothing the label map. As shown on Figure 3.3(e), Graphcut smooths out the sidewalk region just like the other two baselines. However, using the feed forward discriminative learning approach, the sidewalk is correctly recovered. We also tested SuperParsing [6], which performs similarly with the baseline methods on the visible part, but poorly on occluded regions, because it depends heavily on local visible features. Our full system outperforms all baselines, eliminating 18% of the error from the Most Confident method. A paired T-test shows that our final results is better than Most Confident baseline, and reject the hypothesis that the mean of two results are the same with a p-value of $2.1934e − 10$. We evaluated the effectiveness of different components of our system in Figure 3.4(b).
3.2.2 IndoorScene Dataset

The IndoorScene dataset [8] has 308 indoor images with 5 ground truth layout surfaces, annotated also in polygons: floor, left wall, middle wall, right wall, and ceiling. The challenge in the IndoorScene dataset is that the foreground “clutter” such as furniture occludes background regions.

One adaption we made for this dataset is that we transfer polygons all from one image to the query image instead of transferring from different images so that the box surfaces agree on geometry with each other in the final prediction. For evaluation, we compare to the original baseline, the confidence map trained using Geometric Context classifier. Our method increases the labeling overall by 3.9% (Figure 3.5(a)). For the occluded region, the pixel accuracy was improved from 65.8% to 72.9%. Hedau et al.’s method [8] was specifically designed for this task and incorporates vanishing point estimates, and obtained 79.6% on overall accuracy. However, we outperform them on the occluded portion of the scene by 3.1%. Consider the fact that they enforce much stronger geometric prior (boxy room layout), this implies that our algorithm works strongly on occluded
Figure 3.7: **Per-class pixel accuracy on all 3 datasets we experimented with.**

Note that there is no number on “ceiling” for occluded IndoorScene dataset because they are never occluded. For SUN09, our biggest improvement comes from background classes that are often occluded, such as wall, floor and road.

3.2.3 **SUN09 Dataset**

We also experimented on SUN09 dataset, which contains 8684 images of both indoor and outdoor scenes. We manually cleaned up the tags, among which “vehicle”, “chair”, “people” and “object” are foreground; “building”, “ceiling”, “floor”, “ground”, “field”, “road”, “sky”, “tree”, “wall”, “water” and “sidewalk” are background. The overall pixel accuracy are reported Figure 3.5(b). Similarly to the previous two experiments, we see significant improvement on occluded regions (from 49.8% to 66.1%) and modest increase on the visible regions.

3.2.4 **Per-category Analysis**

We also studied the performance of our system with respect to different background region categories, shown in Figure 3.7. On StreetScene dataset, the performance boost come from the categories of road and sidewalk, which are often occluded by pedestrians and cars. On SUN09, the gain come from floor, walls and roads, which suffers the most from foreground occlusions. Most of the performance gain is due to the use of context. The polygonal shape prior has
better accuracy on occluded portion but does not improve the labeling of visible portions, as shown Figure 3.6.

3.2.5 Qualitative Results

We show qualitative results from three datasets: StreetScenes (Figure 3.8), IndoorScene dataset (Figure 3.9) and SUN09 dataset (Figure 3.10). We first show our visible surfaces and detected objects. The gray area indicates background regions occluded by the foreground objects. Then using feed-forward inference, the missing background regions are completed, and then polygons are fit to those regions creating complete polygonal layout proposals. Finally those polygons are used as shape prior to refine the pixel labels.
Figure 3.8: **Qualitative results on street scenes:** Left to right: ground truth; labeling into visible surfaces and detected objects; labeling of completed surfaces with first polygon guess; same labeling with second polygon guess. In each image, the region colors indicate pixel labels. The polygons in the right two columns indicate the transferred regions, representing different hypotheses about individual structures. For example, top row: red polygons indicate the possibility that the building region is composed of one building or two. Bottom row: sidewalk is incorrectly hallucinated to cross the road. Note that our system is often able to infer sidewalk regions that are nearly fully occluded.
Figure 3.9: **Qualitative results on Indoor dataset:** Left to right: ground truth; labeling into visible surfaces; labeling of completed surfaces with first polygon guess; same labeling with second polygon guess. In each image, the region colors indicate pixel labels. We can infer the room structure using the same process as for outdoor scenes. Although our method does not outperform Hedau et al.’s domain-specific method [8] that incorporates strong geometric priors, our method does outperform the initial surface labeler used by them.
Figure 3.10: **Qualitative results on SUN09 dataset:** Left to right: ground truth; labeling into visible surfaces; labeling of completed surfaces with first polygon guess; same labeling with second polygon guess. In each image, the region colors indicate pixel labels. SUN09 features a variety of both indoor and outdoor images, and a broader ranges of foreground and background labels.
3.3 Analysis

3.3.1 Synthetic Example of Auto Context

To better show our argument about long-range interaction and larger neighborhood, we show a synthetic example in Figure 3.11, applied auto-context to it and then compared it to a simple MRF. For training, we have another image corrupted by Gaussian noise with zero mean and the variance of 1. In testing, we have the image of the same pattern, but with unknown translation and corrupted by noise and occlusion. The task is to recover the clean image from the corrupted input.

We compare the result of our contextual method to a simple 4-connected MRF with submodular pairwise potential. When the noise is not heavy, the graph cut does a good job de-noising the image. However, when heavy noise is added, the MRF does not capture the pattern and incorrectly over-smooths the image, as illustrated in the third image on the left column of Figure 3.11. It also does not try to recover the occluded portion, because it does not account for long range interaction. On the other hand, auto-context is capable of capturing such interactions. In the first iteration, context has already been used to recover the occluded part, but it is still a little blurry. However, as more iterations are applied, the image become “clearer” and the original pattern is largely recovered. Note that the context here is implicit and learned, as opposed to the patch-based completion method which uses training texture patches. True, this is a rather contrived example because the contextual cues in this example are much stronger than what one would expect to see in real-world data, making it particularly suitable for auto-context. MRFs with high-order potential should theoretically also be able to address this problem. But it will be significantly harder to model and solve.

3.3.2 Analysis of Contextual Inference

In this section, we examine design decisions we make in the proposed algorithm. Specifically, we want to understand how the performance changes with respect to different parameters in our feed-forward inference. The main parameters of the inference procedure described in this paper are (1) the template it uses, and (2)
Figure 3.11: **Synthetic Example.** The task is a mixture of de-noising and occlusion recovery. While 4-connected MRF over-smooth the corrupted data, auto-context recover a clearer image in a few iterations. See detailed discussion in the text.

The number of iterations. With larger templates, we consider a bigger neighborhood and therefore capture longer-range spatially varying interactions of semantic labels. More iterations help the label prediction to converge. However, they also mean a higher computation cost, which grows linearly with the template size and the number of iterations.

In Figure 3.12, we show the performance of the feed-forward procedure with respect to different template size and iteration numbers. Overall, the procedure
converges within less than 5 iterations. When the template is small (under the radius of 10 pixels), the performance does not change much after just two iterations. The impact of iterations increases when the template is bigger, but the most improvement comes from the first two iterations. The template size, on the other hand, makes a huge difference. The improvement on pixel accuracy is 2.1% when the radius of the template is 1 (thus $3 \times 3$ template, blue line in Figure 3.12). However it becomes 5.7% as the template increases to the radius of 169 pixels (gray line in Figure 3.12), which is almost three times as much as the improvement when the radius is 1. For the occluded part, the accuracy gain is 4.0% and 18.2% for a template of radius 1 and 169, respectively.

It is interesting that the performance gain keep increasing with the template size, until the template increases to the size of whole image. The template with a radius of 169 is $338 \times 338$ in size while the image is $400 \times 300$. This confirms our intuition that we need to consider long-range interactions: the bigger the size of the template, the better the results. And the big template gives the algorithm an advantage to the pixel-wise MRFs with non spatially varying weights, such as the GraphCut baseline described in Section 3.2.

Our feed-forward procedure is more efficient than the original auto-context in [1], because it does not do “auto” feature selection. We tried to use different template configurations, instead of a rigid, ray-shape template. We sampled the templates from 2D Gaussian, Laplacian or triangle distributions to produce a randomized template and then cross-validate for the best-performing one. However, we found it does not yield noticeable improvement, and thus decided to keep things simple by using a ray-shape template. The other concern is the computation speed. For consistency and speed reasons, we use a template with the radius of 49 and 3 iterations in all our previous results.

3.3.3 Scene Understanding as Polygon Detections

The shape matching procedure can give us both the shape prior of the scene and the transferred polygons. The transferred polygons can be collaged together to make a configuration of the scene (the third and the fourth column of Figure 3.8, 3.9 and 3.10). Predicting polygons is by itself an interesting way to understand scenes. Compared to bounding boxes, polygons are more expressive and can represent a greater variety of shapes. Compared to pixel labeling,
polygons are more structured because they are closer to human annotation/understanding and respect the shape regularity of individual entities (e.g. two trees is not equivalent to a bunch of disconnected tree pixels).

![Diagram showing performance change on StreetScene dataset with respect to parameters.](image)

**Figure 3.12:** The performance change on StreetScene dataset with respect to the parameters. (1) template size, and (2) number of iterations, on both the overall and the occluded portion of the scene. In both cases, the performance increases as the template becomes larger, until the radius becomes 169, which translates to 339 pixel, which almost covers the complete image, which has the size of 400 × 300.

We believe polygon detection should be studied in its own right. To the best of our knowledge, this is the first time this task has been systematically evaluated. We follow the evaluation paradigm as in the PASCAL object detection challenge [97]: a polygon detection is considered correct if the intersection-over-union (IOU) score is over 0.5 and multiple detections are penalized. Note that this is a stricter criteria than IOU of bounding boxes, because general polygons are flexible and tend to have low IOU scores. We report the Precision-Recall curve in Figure 3.13(a). Note that the performance is very high for “road” and “building,” since they are often large and continuous in outdoor scenes. “Trees” and “sidewalk,” however, are usually separate and small, and thus 0.5 IOU becomes a harsh criteria for these categories.

We show the plot of recall versus the number of polygon layouts we guess in Figure 3.13(b). With an increasing number of polygon guesses, it is more likely that some of the detected polygons will have an IOU score of 0.5 with the ground truth polygons. However, since this is a very simple baseline which does not consider diversity of proposals or the consistency between polygons, the recall quickly saturates with the increasing number of guesses. We believe it is possible
Figure 3.13: **Evaluation of polygon detection.** (a) Precision-Recall curve of polygon detections in StreetScene dataset, similar to the way bounding box detections are evaluated. (b) Highest possible recall versus the number of polygon proposals. With more proposals, it is more likely that we detect the right object polygons.

To extend this framework with a better collaging method to form a more diverse set of viewpoint-consistent, non-overlapping polygons. We will leave it open as a future work.

### 3.4 Discussion

In summary, our main contribution is to systematically propose the problem of parsing the complete extent of surfaces, with its definition and evaluation measures. We proposed a generic contextual approach for solving this problem, and the proposed algorithm can serve as a good starting point for recovering the complete extent in domain-specific scenes. We showed through experimental results on three datasets that the proposed method is particularly effective on the occluded portion of the scene, and contexts can help us predict the content of the scene that is blocked from the view.

For future directions, it is possible to extend our proposed work in three ways. (1) Concrete examples of the higher level inference tasks using complete extent prediction and the improvement made on those high level tasks. For example, in
robotics, we can show the difference in path lengths of the path planned using visible extent and complete extent of surfaces. (2) In terms of algorithm design, logistic regression is used in the auto-context framework, which limits its power to adapt to more complex contextual models. One possibility is to use other classifiers, especially more sophisticated classifiers like random forest or neural networks. (3) Study the convergence properties of the proposed algorithm. In the original auto-context paper [1], the authors briefly described how auto-context algorithm can be compared to belief-propagation. Our proposed method can also be seen as a way to solve higher-order MRF with a large neighborhood in a stage-wise manner. Since it almost always converges in practice, probably some theoretical guarantee can be derived from the algorithm.
MODELING 3D SCENES

To support high-level scene understanding, we need to define a set of rules of scene representation and a dataset annotated in such a representation. Previous single-view work interprets the scene as a pixel map or a box layout, with simple geometric primitives like cuboids for a small subset of interior objects. Previous multi-view works recover a volumetric or surface model but do not separate into distinct objects. The reason why no prior work takes a data-driven approach to recover complete and detailed 3D scene models is in part because a large-scale dataset of 3D scene models was not available. We undertook an extensive effort to create full 3D models that correspond to scenes in the NYUv2 dataset [2]. In the following sections, we first detail our representation, describe our annotation tools and the collected dataset, and finally show the extent prediction of support surfaces as an application of the proposed dataset.

4.1 Modeling 3D Space

A good scene representation should also provide an abstraction that facilitates understanding, movement and interaction. The scene should be decomposed into separable objects and surfaces and interpreted according to shape and function. We categorize scene elements into three classes:

- **Layout structures** include floors, walls and ceilings. Because walls are always perpendicular to the floor, they are modeled as line segments in the overhead view and polygons in the horizontal view. Similarly, ceiling and floors are line segments in the horizontal view and polygons in the overhead view. We also model openings such as doorways and windows on the walls as polygons on the wall planes.

- **Furniture** objects are common in indoor scenes and tend to have a complicated 3D appearance. For example, chairs and sofas cannot be modeled using cubic blocks, and their support surfaces are often not the
top part. To accurately model them, we use Google SketchUp models from the Google 3D Warehouse repository. We manually select 30 models from these models to model 6 categories of furniture that are most common: chair, table, desk, bed, bookshelf and sofa. The support surface of each object is labeled by hand on the 3D model.

- **3D extruded models** are used to describe all other objects. Most clutter objects are small and do not have regular shape, and we simply model them as vertically extruded polygons.

We created an annotation tool that annotates scenes into this representation without using professional modeling tools or spending much time. It 1) automatically aligns a scene into dominant room axes and annotates in overhead and horizontal views; 2) allows the users to be guided by depth data and existing 2D annotation. On average, it takes about 5 minutes to model a scene with more than 10 objects, depending on the complexity of the scene.

4.1.1 Preprocess

The RGBD scenes are first preprocessed to facilitate annotation. We start with the cropped RGBD scene, which corresponds to the area where information from both sensors are available. Then the surface normals are computed at each pixel, by fitting local planar surfaces in its neighborhood. These local planar surfaces take into account color information as well, in a procedure similar to bilateral filtering, which improves robustness of plane fits compared to using only depth information.

Next, we compute the dominant room orientation by iteratively aligning the surface normals to the \( x, y \) or \( z \) axes. Initially, surface normals of each point is assigned to the nearest axis. Then we fix the assignment and compute optimal rotation matrix \( R \) of the room is computed using SVD, based on all points that are aligned within a given threshold (we empirically set it to 80\%). The point cloud is then rotated using \( R \) and we repeat from the alignment step again until the rotation matrix does not change any more.
4.1.2 Annotation Procedure

After the data is pre-processed, annotators click on regions to build 3D models with. If the floor is visible, our tool can estimate the floor height directly from the floor mask. If necessary, the annotator can correct the floor height by clicking on a scene point and indicating its height with respect to the floor. The annotator then alternates between labeling in an overhead view (from above the scene looking down at the floor) and horizontal view (from the camera looking at the most frontal plane) to model the scene:

1. The annotator is asked to click on one region from the 2D annotated image plane.

2. In the overhead view, the annotator is shown highlighted 3D points and an estimated bounding box that correspond to the object. The annotator can fit a polygon to the footprint of the object.

3. The horizontal view is then shown, and the annotator specifies the vertical height of the object by drawing a line segment at the object’s height.

4. Additionally, the annotator can supply more details of the object, such as adding openings to the wall or placing a detailed SketchUp model for furniture. The user can choose the predefined SketchUp model and orientation by a few keypresses.

Additionally, the annotator can supply more details of the object, such as adding openings to the wall or placing a detailed furniture models for furniture categories. The user can choose the predefined SketchUp model and orientation by a few keypresses. The tool also has a number of handy features that help users annotate quickly including 1) a **snapping feature** that snaps the polygon vertices to neighboring support surfaces or corners, and/or aligns annotations with the room axes; 2) an initial guess of the extent by fitting **default bounding boxes** to depth points included in the region. Therefore the users can often trust the default model configuration and have to edit only when the system’s estimate is poor. Figure 4.1 shows the procedure of annotating a scene into full 3D models.
Figure 4.1: Our annotation tool let users model the RGBD data (top left) into 3D scene models (bottom left) by drawing footprint on overhead (top right) and frontal view (bottom right). The resulting 3D models are consistent with the existing 2D annotation in NYUv2.
4.2 Extension of Complete Surface Algorithm

In this section, we explore the problem of using our annotation to infer 3D support surface. Our goal is to infer the heights and extents of support surfaces in the scene from a single RGBD image. We describe an approach to predict the extent and height of supporting surfaces such as tables, chairs, and cabinet tops from a single RGBD image. Given a RGBD image, we want to localize the height and full extent of such surfaces in 3D space.

One limitation of our algorithm in Chapter 3 is that it works in 2D image plane and therefore it is possible that the polygons we produce do not agree in terms of viewpoint and geometry. Therefore we extent our inference into 3D space and work with RGB-D data. We propose a similar algorithm to find complete extent of surfaces in 3D scenes. In this section, we consider the problem of finding complete extent of support surfaces in indoor scenes. First, we align the room with the dominate directions of the surfaces and then detect the heights where support surfaces occur. Next, we formulate problem of finding the complete extent of support surfaces as parsing in the overhead view of the indoor scene. To this end, we adapt our current 2D approach to 3D space, by making two modifications: (1) use 3D features instead of 2D features (2) allow objects to translate and shift when doing the shape matching.

4.2.1 Approach

First, we label visible pixels into “floor”, “wall”, “ceiling” and “object” using the RGBD region classifier from Silberman et al. [2] and then project these pixels into an overhead view using the depth signal, based on the same scene rotation matrix found in previous section. We then predict which heights are likely to contain a support surface based on a variety of 2D and 3D features. This includes (1) observed upward planar surfaces, (2) observed geometric labels, (3) edgemap, (4) voxel occupancy, (5) volumetric difference, (6) support height prior, (7) relative location prior. We refer the readers to the original paper [10] for details on the 3D features.

When predicting support heights, the features are aggregated, followed by SVM classification and non maximum suppression. For extent prediction, the above feature maps are used and the iterative prediction procedure is applied. In addition to the previous predictions of the current plane, we also look at the
Figure 4.2: The input is an aligned RGBD image. We first classify features in the image plane and then project them into an overhead view. Next, horizontal planes are classified as “support” or “non-support”. The horizontal extent of a supporting surface is then estimated for each support plane based auto-context. Template matching is performed with the footprints of training objects.

Figure 4.3: Overhead scene parsing. The feature set we used in our support surface prediction: observed up-pointing points, 3D geometric labels and edgemaps are computed in the image plane and then projected. Volumetric difference, occupancy and location/view prior are directly computed on the overhead grid. Auto context inference is applied to each predict support height.
Figure 4.4: **Quantitative Evaluation** on support surface prediction. (a) PR curve of support extent prediction with and without auto-context, and the baseline accuracy. (b) Prediction on visible and occluded portion of the surface extent.

Prediction of support planes above and below it. The parameters of the feed-forward contexts used here is the same as the in the 2D cases: 3 iterations and 49 radius. The procedure of the inference is shown in Figure 4.5(a). As in the case of 2D labeling, template matching can also serve to help. This time, we want to modify the matching procedure so as to allow translation and is done through FFT. Essentially, we want to encourage the template to overlap with high probability area in the probability map and penalizes the overlap with the free space, as illustrated in Figure 4.5(b).

### 4.2.2 Evaluation

Support heights and extent can be naturally extracted from the 3D scene annotations in [10]. To make evaluation less sensitive to noise in localization, we make the area around boundary of support surface within a thickness of $\epsilon$ to be “don’t care”. $\epsilon$ is empirical error of Kinect over the dataset, which is set to 0.15m. We also do not evaluate the area that is out of the field of view. In all, there are 5495 support surfaces in 1449 RGBD images, so on average 3.79 support surfaces per scene. In those support surfaces, 5095 are below the camera while 400 are above it.

We evaluate accuracy of support extent prediction with precision-recall curves on support extent prediction. And all other pixels are labeled as negative so that duplicate detections are penalized. The results are shown in Figure 4.4. We also
Figure 4.5: **Extension to RGB-D surfaces.** (a) Adapting auto-context in 3D surface extent prediction: we consider the previous predictions of surfaces in the scenes all as feature maps (b) Shape matching in 3D: we compute the score of template matching based on the overlap, with positive score with the high probability area and negative scores on freespace.

compare performance for occluded support surfaces to un-occluded (visible) ones. In qualitative results, we show predictions that have confidence greater than the value corresponding to the 0.5 recall threshold. For support extent prediction we compare to a baseline of plane-fitting, based on the Silberman et al. [2] code for plane segmentation. We used their plane estimation which comes from a RANSAC and graph cut procedure, and post-process them using the appearance and surface normals. We see that our method outperforms the baseline by 17% precision at the same recall level or 13% recall at the same precision. In addition, we also see that the performance of the visible regions are much better than that of the occluded areas, as expected.

In the qualitative results of Figure 4.6, we see that the transferred support planes can give us a rough estimation of individual support objects. Only the best configuration is displayed here, but our system can also generate the best $K$ configurations. Because the configurations are generated in real 3D space, the support detected surface are guaranteed to be consistent in 3D geometry.
Figure 4.6: **Overhead visualization.** Green and blue and red areas are estimated walls, floor and support surfaces respectively. The brighter colors of support surfaces indicate higher vertical heights relative to the floor. Dark areas are out of the field of view. The first and second column shows in the original scene and its corresponding ground truth support surfaces. The third column shows our final prediction, by thresholding the probability map at the threshold which correspond to 0.5 recall. The fourth column shows the most confidently matched surface configuration.
4.3 Discussions

We have shown the definition of our scene representation. Our surface model go beyond the boxy layout and represent openings and doorways. Our object models use a mixture of CAD models and extruded polygons. Based on this representation, we constructed a dataset of full 3D models, extending the current NYUv2 dataset. This dataset is used to support our study of full 3D models, which previous researchers were not able to conduct because of the lack of data. This dataset can be used in a variety of computer vision tasks including object detection and layout estimation. As an illustration, we studied the problem of support surface estimation in 3D space on the proposed dataset, in which we localize the complete extent of support surfaces in RGB-D scenes.

As future work, it is an interesting to develop new labeling tools that do better jobs on automatic generation of an initial scene. Such a new labeling tool can enable users to edit on more than two views. Also, we can increase the number of CAD models in repository, and/or provide some simple mechanisms of editing CAD models (e.g., change the height of a chair’s seat). For better support surfaces, one possible direction is to take into account the objects being supported to improve the support surface extent prediction.

Simple pop-up of the detected support surfaces can be used a crude 3D model (Figure 4.6). However, the resulting 3D models only encode large objects with support surfaces like kitchen counters or big dinning tables. Other objects like TVs and bottles that have no support surfaces are not captured by these predicted models. Nor do predicted models encode geometric details like chair legs and sofa armrests. This motivates us to work on a system capable of predict full 3D models (Chapter 5).
In Chapter 4 we have laid out our representation of a complete and detailed scene. The proposed representation encodes the layout of walls, which must conform to a Manhattan structure but is otherwise flexible, and the layout and extent of objects, modeled with CAD models and approximate 3D shapes. Such a scene interpretation is useful for robotics and visual reasoning, but difficult to produce due to the well-known challenge of segmenting objects, the high degree of occlusion, and the diversity of objects in indoor scenes. However, we believe that with more 3D models becoming available, we should gradually move from 2D prediction of layout and objects towards full 3D model prediction. The ultimate goal will be predicting 3D models from single-view 2D images. As the first step, we should learn to do it with RGB-D images. Few many previous works attempted to directly predict 3D models, partly because datasets of such annotation were not available. Fortunately, the dataset we created from Chapter 4 can now support our inference.

In this chapter, we present an automatic approach for predicting such 3D models. We take a data-driven approach, generating sets of potential object regions, matching to regions in training images, and transferring and aligning associated 3D models while encouraging fit to observations and overall consistency. The problem is complex and multifaceted, and our approach likewise has several components, each of which could be the sole topic of multiple papers. Therefore, our primary goal at this stage is to establish the representation, a reasonable instantiation of the approach and a sound evaluation methodology. We hope to stimulate improved ideas and techniques for parsing, matching, alignment, context, and integration of multiple viewpoints or modalities.

Additionally, we propose an evaluation for 3D model prediction. Visual data is limited by viewpoint, occlusion and the appearance of visible bits is subject to perspective. Moving the sensor changes the observations, yet the scene itself is not limited by viewpoint. Objects remain present or absent regardless of whether
we see them. A dropped egg breaks even if it falls out of view. The 3D shapes
that generate perspective effects do not change with viewpoint. Therefore, we
want an evaluation that is invariant to occlusion or viewpoint changes. That is,
when the camera moves, our evaluation of a scene should not change. 2D pixel
labeling is traditionally used to evaluate estimation of layouts and objects, but is
viewpoint dependent. Thus, we want to our scene evaluation to be in 3D space.

5.1 Overview

Our philosophy of predicting 3D models is that such systems should be based on
composition. We believe that since there are many elements interacting with each
others in a scene, it is difficult to achieve high accuracy for predicting these
elements individually. Therefore, we can come up with an over-complete set of
hypotheses of layouts and objects, then select among them to compose a
coherent scene. Once the hypotheses are generated, a joint inference is applied to
select a combination of them as an interpretation. The framework is modular and
one can decide the number of hypotheses depending on the trade-off between
accuracy and speed.

First we give the formulation our scene prediction problem. Given a number of
layout and object proposals, we wish to find the best combination of the
proposals $T_i$ and $\Theta$ that transform them into the the target scene, so that the
following energy function is minimized:

$$ T^*, \Theta^* = \arg\min_{T, \Theta} \sum_{T_i} \text{AppErr}(M_i, T_i) + \sum_{T_i} \text{DepthErr}(M_i, \Theta_i(T_i)) $$

$$ + \sum_{T_i, T_j} \text{Overlap}(\Theta_i(T_i), \Theta_j(T_j)) + \sum_{T_i} \text{Coverage}(\Theta_i(T_i)) $$

(5.1)

where $M_i$ is the supporting region of proposal $T_i$. $T_i$ can come from either the
layout or the object proposals. The only difference is that for layout surfaces, $\Theta_i$
is always identity because the layout surfaces have already fit into the scene. We
will explain the intuition behind having each term and then expand in the
sections that follow:

- $\text{AppErr}$ is the appearance error of the proposal and $M_i$ is the supporting
  region of proposal $T_i$. We want each of our proposals to agree with the
  appearance of its supporting region. We define $\text{AppErr}$ as 0 for objects

53
Figure 5.1: Framework. Given an input RGB-D (upper-left), we propose possible wall layouts and object regions. Each object region is matched to regions in the training set, and corresponding 3D model exemplars are transferred and aligned to the input depth image. The subset of proposed objects and walls is then selected according to consistency with observed depth, coverage, and constraints on occupied space. We show an example result (upper-right) and ground truth annotations (lower-right).

that are within the TopK retrievals, and $\infty$ otherwise. For layout surfaces, each detected surface will have an $AppErr$ of 0, and $\infty$ otherwise. It is possible to use retrieval scores in the appearance term, but we did not use them for simplicity. We talk about layout detection in Section 5.2 and object retrieval in Section 5.4.

- $DepthErr$ encodes how well the transferred 3D model agrees with the observed depth points. The point cloud of the observed depth map and the rendered depth map from the 3D model should be consistent. The transferred 3D model will have a penalty if it occupies space observed to be free space. However, there is no penalty if the the space is occluded. We will explain this term in Section 5.5.

- $Overlap$ penalizes objects for overlapping each other in the 3D space. Since our representation contains detailed models instead of just aligned 3D boxes, we compute the overlap of the actual objects by rendering them in 3D space and intersecting their shape, not by finding box intersections.
This process is detailed in Section 5.6.

- **Coverage** encourages the solution to cover the complete scene, so that each pixel is explained by some $T_i$ in the solution. Because $\arg\min$ will prefer empty solutions, we add a coverage term so that unexplained depth points are penalized. The term is explained in Section 5.6.

### 5.2 Layout Surfaces

To propose layout surfaces, we consider only the planes that are aligned with the dominant room directions. The proposal has two steps. First, we find the layout planes in 3D space, by sliding Manhattan planes along the $x$, $y$ and $z$ axes and determining whether the plane contains a “floor,” “ceiling,” “left wall,” “right wall” or “front wall.” Next, we check the observed depth point to find surface extent and openings.

Finding layout planes is posed as a detection problem. We aggregate appearance, depth and location features and train a separate classifier for each of the 5 layout categories to detect planes. When estimating, we apply these trained classifiers, followed by non-maximum suppression. The features used for layout planes include:

1. **Weighted Pixel**: the weighted number of pixels that belong to this plane.
   
   The weight is defined by the probability that the point falls onto the plane and its normal agree with the plane:
   
   $$P(x|p) = N(dist(x,p); 0, \sigma_d)N(|N_x \cdot N_p|; 1, \sigma_n),$$
   
   where $dist(x,p)$ point to plane distance from the point $x$ to the plane $p$, and $N_x, N_p$ their normals, and $\sigma_d$ and $\sigma_n$ are estimated from training data.

2. **Weighted Pixel Per Category**: the weighted number of pixels, also weighted semantic score of belonging to “floor,” “ceiling,” “wall” and “object.” Such weights are obtained from the appearance parsing from [2].

3. **Pixels Behind the Plane**: the count of depth points that fall behind the plane. Solid layout surfaces should have very few points behind it.

4. **Location Prior**: distribution of spatial location of the planes, modeled by truncated normal distributions, which have been estimated separately for each of the 5 layout categories.
After classification and non-maximum suppression of the layout planes, we consider layout planes whose classification score is above the threshold that corresponds to 0.5 recall. Next, for each proposed layout plane, we compute its extent. We apply a simple heuristic to find the extent: we first compute the probability of depth points belonging to this plane, $P(x|p)$ as mentioned earlier. Then, we threshold the probability map with $2\epsilon$ and compute the connected components, where $\epsilon$ is the sensor error at that depth. These components represent all depth points that are confidently not a part of this plane. Finally, we project those points back to the corresponding layout plane and compute a 2D bounding box for each connected component. These bounding boxes are carved out from the layout plane and become openings.

The procedure is visualized in Figure 5.2. In the first step, the planes are first detected from the classifiers. At this point, the planes do not have extent: they
Layout Surface Evaluation

To quantitatively measure the quality of our layout proposal, we perform two kinds of evaluations. First, we evaluate layout plane detection, which evaluates the localization of the planar surface. This is very similar to the evaluation of detecting support surfaces in [10]. We consider a plane to be detected if there is detection within some threshold and multiple detections are penalized. The
threshold is set to be 0.15m, the average depth error of the scenes, as in [10]. We plot the Precision Recall curve of the plane detection. We can see “floors” can be very confidently detected; walls and ceilings are harder to detect than the floor; and frontal walls are easier to detect than left and right walls.

We also consider semantic labeling class, labeling the scene into left wall, right wall, front wall, ceiling and floor, a 5-way classification problem. The ground truth comes from rendering the annotation of the full 3D scene. We compare to directly using appearance-based features [2]. For pixels that are labeled as “openings” in our prediction, we use the observed depth point to determine its type. Overall, our method has a clear advantage over pure appearance based methods in terms of labeling accuracy, shown in Figure 5.3(b), with a 26% absolute improvement in overall accuracy. It is relatively easy to predict surface labels when they are visible, with an error of 6.8% just using appearance features. Our method further reduces it to 4.1%, a 40% reduction. The problem is much more challenging when layout surfaces are occluded; still, our method has an accuracy of 87.6%, much higher than that of appearance features.

Next, we consider the problem of room depth prediction: predicting the depth of the complete room as if there are no foreground objects (Figure 5.3(c)). On the visible portion of the surfaces, the performance is almost the same. This is expected because the sensor depth is the true depth on the visible surfaces, and the error is just sensor noise. However, on occluded surfaces, our depth error is less than a quarter of the sensor depth error (0.167m versus 0.739m), which means that on average we can predict the layout surfaces within 0.167m of the an actual surface, even if we cannot see it.

5.3 Object Proposals

Our object region proposals involve a separate procedure from layout proposal. While we are aware of the numerous off-the-shelf region proposals algorithms out there [93, 94, 95], we find none of them suit our purpose immediately. Some of them need to adapted to RGB-D data, which is not a easy task (simply combining RGB edges and depth edges is not the best practice in our experience). One domain-specific advantage in indoor scenes that most of these algorithms do not consider is layout-object separation. Layout and objects are easily separable from appearance and normals, and such separation can help to
Figure 5.4: **Object proposals.** On the left is the original RGB-D image. The 2nd column from top: RGB-D edge detection [90]; the probability of foreground object from parsing [2]; the merging costs weighted by objectness. The 3rd and 4th columns show object proposals from our approach (light green=seed region; green=full proposed region).

avoid including layout surfaces in object proposals.

We make use of the state-of-the-art RGB-D edge detector with a structured random forest [90] and use it to initialize watershed segmentation to provide a superpixel over-segmentation. The super pixels are iteratively merged based on the RGB-D edge, weighted by the maximum of the objectness of two super pixels. This encourages layout-layout segment pairs to be merged first, and prevents objects from being merged with any layout region (bottom of the 2nd column in Figure 5.4). We iteratively merge the segmentation to a smaller number of segments and then apply the randomized Prim’s algorithm [95]. The seed region of the Prim’s algorithm is sampled according to the objectness of the segment, so that the segments that are confidently layout are never sampled as seeds. We also sample for size constraints and merging threshold to produce a more diverse set of segmentation. We suppress the regions having too big an overlap with another region to make the set of proposals more compact. The proposal procedure is visualized in Figure 5.4.
Segmentation Evaluation

We evaluate the quality of object proposals by the average intersection over union of each ground truth region with its best matched proposal. We evaluate the performance on object categories only. We compare our results to that of hierarchical segmentation, with the same number of proposals. For hierarchical segmentation, we used the algorithm from [48]. The score of our method is 0.533 versus the 0.480 of hierarchical segmentation. The score weighed by area is 0.615 versus 0.572.

5.4 3D Region Retrieval

After object segments are proposed, we use them to retrieve similar objects and transfer the associated 3D models. Because automatic segmentation/object proposals are never perfect, we want to evaluate our approach independent of segmentation error. The following sections are written so that the segmentation can either come from automatic proposals or ground truth segmentation.

Given a region from the segmentation, we extract both the 2D appearance and 3D shape features, and then retrieve similar regions from the training set using nearest-neighbor search. This is essentially in the same spirit as super-parsing [6] or paper doll parsing [81]. The difference is that (1) we are transferring the 3D object models, rather than semantic labels; (2) we make use of bigger, possibly overlapping object proposals, versus smaller, disjoint super pixels; and (3) we do not use the common nearest neighbor search, but instead we learn a distance function to improve the retrieval, which we will explain below.

Distance metric learning has been used to improve image nearest neighbor search [99], and here we apply these approaches to object proposals. In the case of Mahalanobis distance, these approaches aim to learn a distance metric $d(x, y)$ between two feature vectors $x$ and $y$, defined as:

$$d_W(x, y) = \sqrt{(x - y)^T W(x - y)}$$

and the metric is parameterized by a covariance matrix $W$. We have experimented with two approaches to learn $W$.

Relative comparison of Triplets. In [99, 100], Frome et al. and Schultz et al. first define triplets such as $(x_i, x_j, x_k)$, where $x_i$ is more similar to $x_j$ than $x_k$,
and formulate them into constraints: \( \forall (i, j, k) \ d_{A,W}(x_i, x_k) > d_{A,W}(x_i, x_j) \). As in most cases, \( x_i \) and \( x_j \) are similar when they belong to the same semantic category and \( W \) is a non-negative diagonal matrix. An SVM solver is used to determine \( W \).

**Canonical Correlation Analysis.** An alternative approach is to use CCA which is often used to find embeddings of visual features to improve retrieval [101]. CCA finds pairs of linear projections of the two views \( \alpha^T X \) and \( \beta^T Z \) that are maximally correlated. In our case, \( X = \{x\} \) are the feature vectors of the regions and \( Z = \{z\} \) is the label matrix, which contains the indicator vectors of the semantic labels of the corresponding region. The distance metric on the common space of the feature and semantic labels, inducing a distance metric: \( d_W(x, y) = \sqrt{(x - y)^T W (x - y)} = \sqrt{(x - y)^T \alpha \alpha^T (x - y)} \).

To describe the 2D appearance and 3D shapes, we also experimented with a number of feature descriptors for RGBD regions:

<table>
<thead>
<tr>
<th>Method</th>
<th>Raw</th>
<th>Triplets [100]</th>
<th>CCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>0.169</td>
<td>0.210</td>
<td>0.226</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.160</td>
<td>0.162</td>
<td>0.218</td>
</tr>
<tr>
<td>CNN</td>
<td>0.155</td>
<td>0.160</td>
<td>0.212</td>
</tr>
<tr>
<td>3D+SIFT</td>
<td>0.192</td>
<td>0.228</td>
<td><strong>0.265</strong></td>
</tr>
</tbody>
</table>

**Figure 5.5: Region Retrieval.** (a) Evaluating semantic category prediction with 1-NN under different features and learning approaches (using all categories in NYUv2). (b) Feature ablation test. (c) Top-1 and Top-10 confusion matrix of retrieval. Note this is a very small sample of the complete confusion matrix with more than 600 categories, so the row does not sum to 1.
1. **Depth features** from [2]. This includes histograms of surface normals, 2D and 3D bounding box dimensions, color histograms and relative depth.

2. **SIFT features** also from the set of structure class in [2].

3. **CNN features.** We have also tried with newly emerged convolutional neural network features for regions, by wrapping region and extract features with pre-trained network, as described in Girshick et al. [102] with implementation from [103].

**Retrieval Evaluation**

For retrieval experiments, we evaluate our results based on the retrieved semantic category with ground truth segmentations. In the training set of NYUv2, there is a total of 15,742 regions, from 665 categories, making it a reasonable sized retrieval set. In Figure 5.5(a), we report the one-nearest-neighbor (1-NN) retrieval accuracy. One clear observation from the retrieval results is that supervision can greatly boost the performance of the NN region search across different feature combinations. For the combined features, the accuracy rises from 19.2% of the original features to 26.5% of CCA, a relative improvement of 38%. Another observation is that CCA outperforms Triplets. This is probably because Relative Comparison [100] learns $W$ only on the diagonals, making Triplet less flexible than that of CCA.

We also look at the usefulness of different features in Figure 5.5(b). We noticed that the set of 3D features, despite their low dimensionality, are the single most useful set of features. Within the set of 3D features, bounding box features, such as size and location, are the most important. SIFT and CNN performs very similarly (0.5% difference). To simplify dependencies, we stick to the set of 3D+SIFT features used in [2].

In Figure 5.5(c), we show the confusion matrices of some popular object categories for Top-1 and Top-10 retrieval. The TopK confusion matrix is computed as such: if there is a correct prediction in TopK retrieved items, it is considered to be correct; otherwise, it is confused with the highly ranked retrieval. Looking at the confusion matrix, we can see that there is a good chance we will have at least one region retrieved from the correct semantic category if we retrieve 10 candidate for each region.
5.5 Fitting 3D Models

Next, we need to transform the 3D objects retrieved from the segmentation to fit the depth points of the target scene. First, for each retrieved object, we roughly estimate its location and scale based on the 3D bounding box of the source object and the region mask for the proposed object, and then render the transferred object in 3D space to obtain the inferred 3D points.

One problem is that the object proposed may not be of the right scale or orientation: a region of a left-facing chair often resembles a right-facing chair and needs to be rotated. Therefore, to solve for the right placement, we search for scales and rotations and use Iterative Closest Point (ICP) to solve for translation. We also tried to use ICP to find the scaling and rotation in the ground plane, but that did not yield good results. Therefore, we enumerate scales and rotations, and find the one with best transformation based on the following cost function. The depth error term in Equation 5.1

$$\text{DepthErr}(M_i, \Theta(T_i)) = \sum_{x \in M_i \cap \Theta(T_i)} |d(x) - \hat{d}(x)| + \sum_{x \in M_i \cap \Theta(T_i)} C_{\text{missing}}$$

$$+ \sum_{x \in -M_i \cap \Theta(T_i)} \max(\hat{d}(x) - d(x), 0)$$  \hspace{1cm} (5.2)$$

where $\Theta(T_i)$ denotes the mask of rendered object after ICP. $d$ is scene depth observed from the camera, $\hat{d}$ is the rendered depth of the transferred object observed in the same viewpoint, and $C_{\text{missing}}$ is a constant penalty for not explaining the depth points in the segment. The first term of the function enforces the depth of the visible portion of the transferred object to be consistent with the observed; the second term penalizes regions included in the segmentation but missing in the transferred object; the last term requires that the occluded portion of the transferred object may not violate the constraints of observed empty space (i.e. the object does not “stick out” into the empty space). In practice, we set $C_{\text{missing}} = 0.3$. For each transferred object $T_i$, we find the best transformation that minimizes the above error:

$$\Theta_i(T_i) = \arg \min \text{DepthErr}(M_i, \Theta(T_i))$$
5.6 Composing 3D Models

Finally, we will integrate the object and layout proposals together, composing a complete scene, as shown in Eqn. 5.1. The first term $DepthErr$, the same as that described in Equation 5.2, accounts for depth discrepancy between the observed depth and rendered object. The term $Overlap(\Theta_i(T_i), \Theta_j(T_j))$ penalizes objects for overlapping each other in the 3D space. In principle, overlap is voxel overlap in the 3D space between the two transferred objects $\Theta_i(T_i)$ and $\Theta_j(T_j)$. In practice, to speed up the computation, we compute the intersected part of their minimum and maximum depth image, and use that as a proxy. The last term $Coverage$ encourages the solution to explain what we have seen in the image plane, defined as $\sum_p \max_i C_{\text{empty}} \text{Covers}(\Theta_i(T_i), p)$, where $p$ is the pixels in the image plane. In practice, we work with super pixels and assign a weight to each super pixel covered by at least one object in the current solution. We set $C_{\text{empty}} = -1$ in the experiment.

Notice that the whole objective function can be seen as a holistic depth error function in the image plane, and is measured in meters: $DepthErr$, and $Overlap$ are measured in meters, missing a pixel on the proposed segment gives a penalty of $C_{\text{missing}}$ meters, and having an unexplained pixel is equivalent of having $C_{\text{empty}}$ meters of depth error.

Depending on which segmentation we are using, the minimization problem needs to be solved differently. For ground truth segmentations, we do not need the “emptiness” term because we know the ground truth segmentation explains all of the scene. Moreover, we can enforce additional constraints that exactly one of the proposed objects from one ground truth segment can be chosen in the solution. Therefore the objective function can be framed as an Integer Programming (IP) problem:

$$z^* = \arg \min_z z^T A + z^T B z$$

subject to  

$$z_i \in \{0, 1\}$$

$$\forall i \sum_j z_j = 1, \text{ where } z_j \text{ is proposed from } m_i$$

Here, $z_i$ corresponds to the indicator variable of $T_i$ in $T$, $A_i$ is the $DepthErr$ of $T_i$, and $B_{ij}$ the $Overlap$ of $T_i$ and $T_j$. With the ground truth segmentation and the additional constraints, we find that the IP problem can often be solved by its Linear Programming relaxation.
When automatic object proposals are used, however, we need the “emptiness” term to encourage objects so that it explains more visible depth points. Thus, we have the following objective:

$$z^* = \arg\min_z \text{loss}(z) = \arg\min_z z^T A + z^T Bz + \sum \max(C \circ (I z))$$

subject to $z_i \in \{0, 1\}$

where $C_{ki}$ is the number of pixels $\Theta_i(T_i)$ occupies in super pixel $S_k$ (in the image plane), $\circ$ is the entry-wise matrix multiplication and the max function is a max on each row. Relaxation to $[0, 1]$ is not useful because it often get stuck in local minima, giving non-integer solutions. Such relaxations usually have difficulty making moves such as removing two or three smaller objects and adding a bigger one. However, heuristic searches can easily deal with such situation. We use a randomized beam search algorithm, combined with stochastic hill climbing. We maintain a working set of the solutions, which are initialized by hill climbing. Then, in each iteration, we pick a random solution from the working set and add one object into the solution, removing all objects that are overlapping with it. After this, we improve the solutions by hill climbing again to obtain a new solution. If the new solution improves the loss function over the original one, we add the new solution into the working set. We repeat this procedure until no improvement can be made, or stop after some number of iterations. In practice, we find this simple procedure to be very effective against local minima. We discuss the details in the supplemental material.

Scene Composition Evaluation

We evaluate our scene prediction performance based on voxel prediction. The scope of evaluation is the space surrounded by annotated layout surfaces. Voxels that are out of the viewing scope or behind solid walls are not evaluated. We render objects separately and convert them into a solid voxel map. The occupied space is the union of all the voxels from all objects; free space is the complement of the set of occupied voxels.

Our voxel representation is constructed in a fine grid with 0.03m spacing to
Randomized Beam Search

Input: loss function \( \text{loss}(z) \)

\( N \) the size of working set, \( K \) the number of variables

Output: The best solution \( z^+ \)

For \( i = 1 \ldots N \) /* Initialize working set */

\( z_i = \text{StochasticHillClimb}(z_0 = 0); \)

End

/* Randomized improving objective function until convergence*/

Until No Further Improvement

/* Pick an object an object \( j \) */

\( j = \text{randi}(K); z^+ = z_i; \)

For \( i = 1 \ldots N \)

/* Remove all objects that overlap with object \( j \) */

\( z^+ = 1; z^+_u = 0 \forall u \text{ where } B_{uj} > 0; \)

/* Do Hill Climbing again \( j \) */

\( z^+ = \text{StochasticHillClimb}(z^+); \)

If \( \text{loss}(z^+) \) \(<\) \( \text{loss}(z_i) \)

Add \( z^+ \) to the working set \( Z \)

End

End

\( z^+ = \arg \min_{z \in Z} \text{loss}(z) \)

Figure 5.6: **Randomized Beam Search** is used to infer which set of object proposals to include when composing scenes with automatic object proposals. This is essentially a heuristic idea: put one object in, remove all overlapping objects and put in other objects, then go again.

provide the resolution to encode shape details of 3D model objects we use. The voxel prediction and recall are presented in Figure 5.7. The voxel representation has advantages of being computable from various volumetric representations, viewpoint-invariant, and usable for models constructed from multiple views (as opposed to depth- or pixel-based evaluations). Requiring a direct match of voxel maps may be too sensitive of a criteria. There is inherent annotation and sensor noise in our data, which is often much greater than 0.03\( m \) in many objects. Objects, when they are small, of nontrivial shape, or simply far away, result in very poor voxel accuracy, even though they agree with the input image. Therefore, we introduce prediction with a tolerance, proportional to the depth of the voxel, for which we use \( \epsilon = 0.03 \times \text{depth} \), the sensor resolution of Kinect. Specifically, if a voxel is within \( \epsilon \) of a ground truth voxel, it is considered to be a hit; if a ground truth voxel has a predicted voxel in its neighborhood, it is consider to be recalled.

We present two simple baselines for comparison. For free space, we evaluate
### Table 1: Voxel Prediction Evaluation

<table>
<thead>
<tr>
<th>Method</th>
<th>Sensor</th>
<th>Ours-Annotated</th>
<th>Ours-Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>1.000</td>
<td>0.944</td>
<td>0.945</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>0.804</td>
<td><strong>0.936</strong></td>
<td>0.926</td>
</tr>
<tr>
<td><strong>F-score</strong></td>
<td>0.891</td>
<td><strong>0.940</strong></td>
<td>0.936</td>
</tr>
</tbody>
</table>

(a) Free space Evaluation

<table>
<thead>
<tr>
<th>Method</th>
<th>Bbox</th>
<th>Ours-Annotated</th>
<th>Ours-Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>0.527</td>
<td><strong>0.634</strong></td>
<td>0.575</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>0.260</td>
<td><strong>0.367</strong></td>
<td>0.328</td>
</tr>
<tr>
<td><strong>F-Score</strong></td>
<td>0.348</td>
<td><strong>0.465</strong></td>
<td>0.418</td>
</tr>
<tr>
<td><strong>Prec.-ε</strong></td>
<td>0.669</td>
<td><strong>0.781</strong></td>
<td>0.714</td>
</tr>
<tr>
<td><strong>Recall-ε</strong></td>
<td>0.425</td>
<td><strong>0.497</strong></td>
<td>0.490</td>
</tr>
<tr>
<td><strong>F-Score-ε</strong></td>
<td>0.520</td>
<td><strong>0.607</strong></td>
<td>0.581</td>
</tr>
</tbody>
</table>

(b) Occupancy Evaluation

---

Figure 5.7: **Voxel Prediction.** (a) Voxel precision and recall of free space estimation, using our method with ground truth segmentation (Ours-Annotated) or automatic proposals (Ours-Auto), compared to putting voxels around observed depth points (Sensor); (b) Voxel precision and recall of object occupancy, as compared to the baseline of fitting bounding boxes to ground truth regions.

The free space from observed sensor depth predicts 100% of the visible free space but recalls none of the free space that is occluded. From Figure 5.7(a), the baseline recalls 80.4% of the ground truth; therefore, almost 20% of free space is hidden. Our method captures a lot more hidden space: estimation with annotated segmentation recalls 93.6% of the free space, and automatic segmentation recalls 92.6%, with a small drop in precision.

For occupied space, we use a bounding box baseline. The bounding boxes are generated in the rectified view, with ground truth segmentation and 0.1 outlier rejection. The results in Figure 5.7(b) show: (1) exact voxel prediction, which requires small voxels to be predicted at the exact 3D position, is very difficult, with recall of 0.367, even when the visual rendering appears correct; (2) precision is higher than recall, which means that missing objects is more of a problem than false detections; and (3) with a small ε tolerance of one voxel (0.03m at 1.0m depth), the F-score increases by roughly 0.15 for both our method and the baseline.
5.7 Discussion and Remarks

We have proposed an approach to create a full 3D scene model of separate objects and surfaces from a single RGB-D image. Although much effort went into each component of our approach, there are still a lot of opening opportunities to improve the results. We have devised a variety of sub-task performance measures to aid comparison in later work. It is also possible to evaluate performance of the full 3D model prediction task using voxel-based accuracy measure. We outline a few open directions:

- **Improve segmentation.** We are using object proposals to generate complete regions of objects to seed retrieval. The current algorithm is essentially a randomized one, varying seed and sizes. One can always use the appearance and 3D geometry of the regions and rank them to get a better set.

- **Use object detection.** We found that a proposal+retrieval strategy works better than expected when 3D fitting is used. This is a new insight considering that this method never worked well for 2D. Our method has the advantage of being able to work with hundreds of categories (NYUv2 has more than 800) and very few exemplars. Another alternative is to use tried-and-true sliding window detectors, or proposal+classification framework like that of Girshick et al. [102], which are top performers in object detections. It may be the case that we can use detectors for well-defined object categories and use retrieval for the rest.

- **Contextual Inference.** We are not using spatial or co-occurrence relations for objects. For example, chairs are likely to be found near tables, and pots are likely to be found in the kitchen. There is a large line of contextual inference work that may produce better 3D models.

- **Support Relations.** At this point, there are no constraints that objects needs to be supported by other planes or object-supporting surfaces. Objects do not float, so an interpretation with a floating object needs to translate or refine the object model. This is probably very important to the visual appearance of the rendered objects.

- **Self-Similarity** Many items, such as chairs, tables, and glasses are duplicated within a scene, and imposing preference for less diverse
populations could improve results. It is possible to encourage fewer number of object models to be used within a single scene.

Our work infers a detailed, non-box layout model with doorways and openings, and we model all interior objects with detailed CAD models or approximate polygons. No prior work takes a data-driven approach to recover complete 3D scene models like ours, in part because the data only recently became available in [10] and no existing systems produce 3D models like Figure 5.8. Despite these many remaining challenges, we believe that we have made good progress on this difficult problem.
Figure 5.8: **Qualitative Result on NYUv2.** Each column is described in the captions above. The 4th column shows the set of region proposals that has been selected by our algorithm to generate the composed scene. The resulting scene parses usually appear to provide a good approximation of occupied space, with improved detail and parsimony when manually segmented regions are used. In some cases (coat in third row, toilet in last row), the retrieved 3D approximations fit the object shape better than the ground truth annotation. One common problem for retrieval based on automatic regions is that objects are split into smaller pieces, which enables a better approximation of 3D occupancy and does provide the same understanding of which objects are separable. Major directions for improvement include incorporating support estimates and constraints, encoding semantic context of objects and scenes (chairs near tables, microwave in kitchen), and encoding regularity (multiple chairs within a room are likely to be similar).
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original Image" /></td>
<td><img src="image2" alt="Manual Segmentation" /></td>
<td><img src="image3" alt="Composition w. Manual Seg." /></td>
<td><img src="image4" alt="Auto Proposal" /></td>
<td><img src="image5" alt="Composition w. Auto Proposal" /></td>
<td><img src="image6" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image7" alt="Original Image" /></td>
<td><img src="image8" alt="Manual Segmentation" /></td>
<td><img src="image9" alt="Composition w. Manual Seg." /></td>
<td><img src="image10" alt="Auto Proposal" /></td>
<td><img src="image11" alt="Composition w. Auto Proposal" /></td>
<td><img src="image12" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image13" alt="Original Image" /></td>
<td><img src="image14" alt="Manual Segmentation" /></td>
<td><img src="image15" alt="Composition w. Manual Seg." /></td>
<td><img src="image16" alt="Auto Proposal" /></td>
<td><img src="image17" alt="Composition w. Auto Proposal" /></td>
<td><img src="image18" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image19" alt="Original Image" /></td>
<td><img src="image20" alt="Manual Segmentation" /></td>
<td><img src="image21" alt="Composition w. Manual Seg." /></td>
<td><img src="image22" alt="Auto Proposal" /></td>
<td><img src="image23" alt="Composition w. Auto Proposal" /></td>
<td><img src="image24" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image25" alt="Original Image" /></td>
<td><img src="image26" alt="Manual Segmentation" /></td>
<td><img src="image27" alt="Composition w. Manual Seg." /></td>
<td><img src="image28" alt="Auto Proposal" /></td>
<td><img src="image29" alt="Composition w. Auto Proposal" /></td>
<td><img src="image30" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image31" alt="Original Image" /></td>
<td><img src="image32" alt="Manual Segmentation" /></td>
<td><img src="image33" alt="Composition w. Manual Seg." /></td>
<td><img src="image34" alt="Auto Proposal" /></td>
<td><img src="image35" alt="Composition w. Auto Proposal" /></td>
<td><img src="image36" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image37" alt="Original Image" /></td>
<td><img src="image38" alt="Manual Segmentation" /></td>
<td><img src="image39" alt="Composition w. Manual Seg." /></td>
<td><img src="image40" alt="Auto Proposal" /></td>
<td><img src="image41" alt="Composition w. Auto Proposal" /></td>
<td><img src="image42" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image43" alt="Original Image" /></td>
<td><img src="image44" alt="Manual Segmentation" /></td>
<td><img src="image45" alt="Composition w. Manual Seg." /></td>
<td><img src="image46" alt="Auto Proposal" /></td>
<td><img src="image47" alt="Composition w. Auto Proposal" /></td>
<td><img src="image48" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image49" alt="Original Image" /></td>
<td><img src="image50" alt="Manual Segmentation" /></td>
<td><img src="image51" alt="Composition w. Manual Seg." /></td>
<td><img src="image52" alt="Auto Proposal" /></td>
<td><img src="image53" alt="Composition w. Auto Proposal" /></td>
<td><img src="image54" alt="Ground Truth Annotation" /></td>
</tr>
<tr>
<td><img src="image55" alt="Original Image" /></td>
<td><img src="image56" alt="Manual Segmentation" /></td>
<td><img src="image57" alt="Composition w. Manual Seg." /></td>
<td><img src="image58" alt="Auto Proposal" /></td>
<td><img src="image59" alt="Composition w. Auto Proposal" /></td>
<td><img src="image60" alt="Ground Truth Annotation" /></td>
</tr>
</tbody>
</table>

Figure 5.8: **Qualitative Result on NYUv2. (cont.)**
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image31.png" alt="Image" /></td>
<td><img src="image32.png" alt="Image" /></td>
<td><img src="image33.png" alt="Image" /></td>
<td><img src="image34.png" alt="Image" /></td>
<td><img src="image35.png" alt="Image" /></td>
<td><img src="image36.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image37.png" alt="Image" /></td>
<td><img src="image38.png" alt="Image" /></td>
<td><img src="image39.png" alt="Image" /></td>
<td><img src="image40.png" alt="Image" /></td>
<td><img src="image41.png" alt="Image" /></td>
<td><img src="image42.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image43.png" alt="Image" /></td>
<td><img src="image44.png" alt="Image" /></td>
<td><img src="image45.png" alt="Image" /></td>
<td><img src="image46.png" alt="Image" /></td>
<td><img src="image47.png" alt="Image" /></td>
<td><img src="image48.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image49.png" alt="Image" /></td>
<td><img src="image50.png" alt="Image" /></td>
<td><img src="image51.png" alt="Image" /></td>
<td><img src="image52.png" alt="Image" /></td>
<td><img src="image53.png" alt="Image" /></td>
<td><img src="image54.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 5.8: Qualitative Result on NYUv2. (cont.)
6 CONCLUSIONS AND DISCUSSIONS

In this thesis, we study scene understanding with full 3D models, with the complete extent and full 3D models. We represent scenes with flexible layout surfaces and hundreds of classes of objects. Annotation tools are created to enable 3D scene modeling with the proposed representation. We collected a novel dataset which is now publicly available and can be used in a variety of scene understanding tasks. We introduce an algorithm for finding the complete extent of surfaces in both RGB and RGB-D images. The algorithm outperforms existing methods by considering surrounding spatial contexts and the shape prior of regions, while remaining efficient and applicable to multiple domains. Finally, we offer an end-to-end system for predicting full 3D models from single-view RGB-D images, which has not been previously attempted because of the lack of annotated scene data. The proposed framework is based on model transfer and accommodates hundreds of object categories and flexible layout surfaces. Final viewpoint-independent evaluation on layout prediction and free space estimation is presented to measure the performance of our final system.

I believe our work, especially the proposed end-to-end full 3D model prediction system, is fundamentally novel and I hope the ideas and practices presented in this thesis will be useful for researchers pursuing the same direction of research. A lot of follow-up work can be done in this direction. In particular, I feel that it is interesting to improve our current work from the following angles:

- **Dataset collection.** There are constant opportunities for better data collections. One may consider RGB or RGB-D images from more than one viewpoint. Our proposed dataset contains single-view RGB-D images just like NYUv2. Often, the quality of RGB-D images can be greatly improved when a continuous scanning system like KinectFusion [72] is used. The number of predefined CAD models can also be expanded. We currently use 30 CAD models, but this can be improved. Finally, newer hardware equipment, such as new Kinect2 may also help in collecting more accurate RGB-D data.
Improvement on the full 3D prediction system. Because of the scope of the problem, many aspects of the proposed full 3D prediction system can be modified or added. Fortunately, our proposed system has a modular design and each component can be improved and evaluated. Here is a list of possible ideas for improvement:

1. Introduce contextual inference and physical support relations in model selection stage. These cues have been used in 2D systems with success and I believe they can provide a sizable boost the performance of 3D prediction as well.

2. Use top-down, sliding-window object detectors for generating object 3D model proposals. Such discriminatively-trained detectors have shown their power to accurately detect 2D objects on many benchmarks. With reasonable innovation to adapt them to 3D cases, they can potentially generate better 3D model proposals for objects.

3. Further study RGB-D image segmentation methods for closing the performance gap between the full 3D models using manual segmentation and that of automatic segmentation.

4. Consider self-similarity cues. Objects of the same CAD models are more likely to reappear within a scene (e.g., multiple chairs of the CAD model may appear in the same room). This may help predict more consistent and visually appealing full 3D models.

5. Feature learning and metric learning of 3D region retrieval with more sophisticated machine learning algorithms.

6. Predict depths from RGB images, and then apply the proposed framework on the predicted RGB-D image.

Applications of full 3D model prediction. Predicting full 3D models is interesting by itself and also has many applications in related areas. I believe that systems that predict 3D models have the potential to generate an industrial significance when they are adopted in these areas:

1. Robotics. Predicted 3D models can be directly used in navigation and help robots to avoid obstacles and plan their paths. For example, robotic floor cleaners can use the predicted indoor room model to determine their route. The scene models can also provide guidance
on how to explore the surrounding environment. For example, a quadrotor can use the predicted 3D model to guide where to explore next.

2. Gaming. Predicting 3D models of people’s living rooms and bedrooms opens many opportunities for computer game designs. Game designers will be able to bring gaming experience to players with scenes of players’ real lives.

3. Graphics. 3D modelers and interior designers can benefit from the proposed system as well. They can use the automatically constructed full 3D model as a starting point for more precise 3D modeling and designing. Such an initial approximate model can greatly reduce the amount of work needed and help them focus on more creative activities.
REFERENCES


[12] “Ikea virtual room.”


