3D SCENE AND OBJECT PARSING FROM A SINGLE IMAGE

BY

CHUHANG ZOU

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, 2019

Urbana, Illinois

Doctoral Committee:

Associate Professor Derek Hoiem, Chair
Professor David Forsyth
Assistant Professor Alexander Schwing
Dr. Alex Colburn, Amazon

ABSTRACT

The term 3D parsing refers to the process of segmenting and labeling the 3D space into expressive categories of voxels, point clouds or surfaces. Humans can effortlessly perceive the 3D scene and the unseen part of an object from a single image with a limited field of view. In the same sense, a robot that is designed to execute a few human-like actions should be able to infer the 3D visual world, from a single snapshot of a 2D sensor such as a camera, or a 2.5D sensor such as a Kinect depth equipment. In this thesis, we focus on 3D scene and object parsing from a single image, aiming to produce a 3D parse that is able to support applications like robotics and navigation.

Our goal is to produce an expressive 3D parse : *e.g.* , what is it, where is it, how can humans move and interact with it. Inferring such a 3D parse from a single image is not trivial. The main challenges are: the unknown separation of layout surfaces and objects; the high degree of occlusions and the diverse classes of objects in the cluttered scene; how to represent 3D object geometry in a way that can be predicted from noisy or partial observations, and can help assist reasoning like contact, support and extent. In this thesis, we put forward the hypothesis and prove in experiments, that a data-driven approach is able to directly produce a complete 3D recovery from 2D partial observations. Moreover, we show that by imposing constraints of 3D patterns and priors into the learned model (*e.g.* , layout surfaces are flat and orthogonal to adjacent surfaces, support height can reveal the full extent of an occluded object, 2D complete silhouettes can guide reconstructions beyond partial foreground occlusions, and a shape can be decomposed into a set of simple parts), we are able to obtain a more accurate reconstruction of the scene and a structural representation of the object.

We present our approaches at different levels of detail, from a rough layout level to a more complex scene level and finally to the most detailed object level. We start by estimating the 3D room layout from a single RGB image, proposing an approach that generalizes across panoramas and perspective images, cuboid layouts and more general layouts (*e.g.* , "L"-shape room). We then make use of an additional depth image, explore at the scene level to recover the complete 3D scene with layouts and all objects jointly. At the object level, we propose to recover each 3D object with robustness to possible partial foreground occlusions. Finally, we represent each 3D object as a 3D composite of sets of primitives, recurrently parsing each shape into primitives given a single depth view. We demonstrate the efficacy of each proposed approach with extensive experiments both quantitatively and qualitatively on public datasets.

To my beloved family members, for their love and support.

ACKNOWLEDGMENTS

First of all, I would like to thank my Ph.D. advisor Professor Derek Hoiem. Thanks for your continuous support of my research, especially when I was trying to work on something you didn't believe in (and it always turned out that you're right). Thanks for your patience, especially when I met obstacles in my research, or when it took longer time for me to understand your great point of view. I could not have imagined having a better advisor for my Ph.D. study. I've learned a lot from your sharp critics on research, your moderate ways of getting along with others not only in the research community but also in daily life.

I would also like to thank my Ph.D. committee members: to Professor David Forsyth, thanks for kindly offering vision lunch every week! I really admire your passion for research, your unique but inspiring research perspectives. I believe you're the only CS Professor I know who is a tea expert. To Professor Alexander Schwing, though it was only a short time working with you, I really enjoyed talking to you and I felt honored to have you join our CVPR workshop organizer team! To Dr. Alex Colburn, I believe that the popularity of our proposed LayoutNet demonstrates the great cooperation between us.

I am grateful to my family members. Though living overseas, my parents always try their best to express their love and support to me through FaceTime and WeChat. My father, who is always busy in weekdays, often chooses to call me in weekends early in the morning to talk about his research and ideas or just random things. He never explicitly expresses his concerns for me, but be there and say nothing (alright I know you just don't know what to say to encourage me properly). My mom spends more time with me. She shares tips to keep me healthy and always reminds me to be on time for everything. Sometimes we might have intense arguments. But I know that's caused by some of my still existing immature characteristics (well most of us want to act like a child when we are with our family right?). I hope My grandma is always happy. I like the piano melody she plays, which makes us calm. Also to my dear grandpa, although it's still a long journey for me to keep up with your academic achievements (or maybe I never thought I could, I don't want to compare myself with my elder generations but admire them), I hope my current performance will make you feel honored. I want to thank my boyfriend Dr. Yijun Li, who is also my precious family member. I feel lucky to have your support and encouragement during the last two years of my Ph.D. study – the most important and busiest period. I always know that you're there aside and you are also working hard to achieve a similar goal to mine.

I would like to express my gratitude to my labmates of the University of Illinois at Urbana-Champaign (UIUC) vision group, and to my friends who have supported me along the way.

With a special mention to Zhizhong Li (I caught a lot of your snacks, they're too sweet), Tanmay Gupta (please don't say that the maple squash soup we tried in Boston tastes good), Aditya Deshpande (happy wedding!), Jae Yong Lee (your Chinese pronunciation is good), Rajbir Kataria (gentleman), Theerasit Issaranon (nice working with you), and to Daniel McKee, Mariya Vasileva, Unnat Jain, Aiyu Cui, Jeffrey Zhang, Anand Bhattad, and past graduates: P. Daphne Tsatsoulis, Saurabh Singh, Qieyun Dai, Kevin Shih, Joseph Degol, Arun Mallya, Bryan Plummer, Liwei Wang. Oh dear can I just make special thanks to everyone named on the UIUC vision webpage? It was fantastic to have the opportunity to work with you all. On becoming the "elder generation" of UIUC visionaries (to be honest I will never admit this), I hope the young generations can enjoy their research in this great lab.

I am also grateful to the Professors and collaborators who have appeared in my Ph.D. life: Professors Svetlana Lazebnik, Steve Lavelle, Doctors Qi Shan, Ersin Yumer, Jimei Yang and Duygu Ceylan.

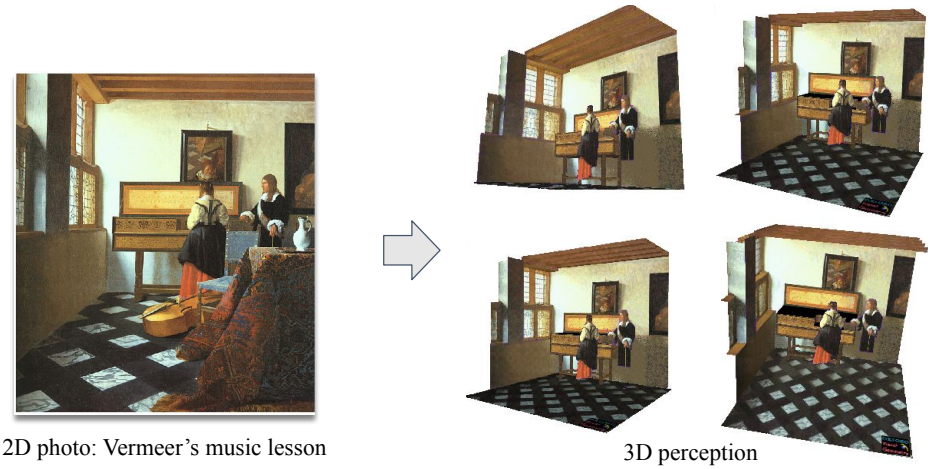
And finally, thanks again for all your encouragement and support. As a summary, I shall say, with you all, my Ph.D. life becomes sparkling.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

Humans are able to perceive the 3D world from a single view. When we see a photo of the Vermeer's music lesson in Figure 1.1, we know the 3D location of layouts such as walls, ceiling and floor, as well as the 3D shape of objects such as piano and painting. We can infer the spatial relationship between layouts and objects, *e.g.* , the piano is on the floor, near the back wall. We can also infer the 3D occupancy and free space inside the room. We are able to reason about the complete 3D shape of layouts and objects even they are partially occluded, *e.g.* , we know the complete shape of the piano, as shown in Figure 1.1, although the piano is partially occluded by the woman in front.



2D photo: Vermeer's music lesson

3D perception

Figure 1.1: Illustrations. Humans can effortlessly understand the 3D scene from a 2D photo, and are able to infer the unseen parts of objects. Images are adopted from [1].

This inspires us to build up a vision system that is able to understand the 3D world from a single image, which is highly desirable in the scenarios of robotics and navigation. For example, in Figure 1.2, the cute robot is instructed to help pick up the wine glass in the messy kitchen. The limited free space in the kitchen prevents the robot from a complete 3D scan of the scene, and the robot would rather choose to rely on the 3D inference from a limited field of view, *e.g.* , a single snapshot of the kitchen. In response to the request, as illustrated in Figure 1.3, the robot first localizes the position of the wine glass in the kitchen, estimates free space in front and moves towards it, finds the handle of the wine glass and poses a correct hand gesture to pick it up. Those successive actions require the robot to be able to infer the physical shapes of objects, surfaces, and

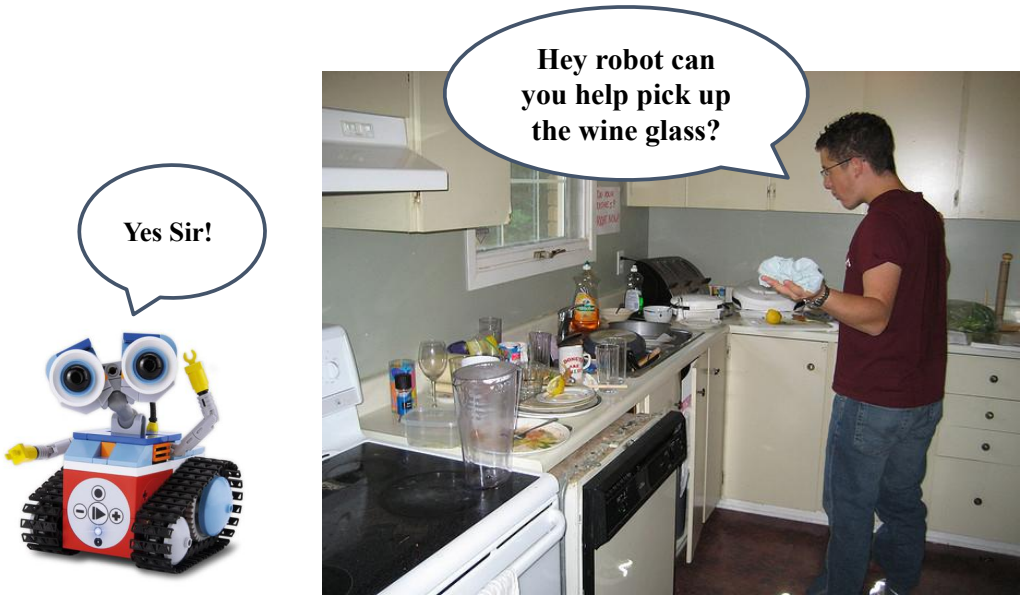


Figure 1.2: The cute robot is instructed to help pick up the wine glass on the counter in the messy kitchen.

the scene layout extent from sensors. Moreover, the robot should be able to understand what is in the scene and what could be done, and produce an expressive 3D parse that could support reasoning like support (*e.g.* , this table supports that mug) and interactions with humans or robots (*e.g.* , this is the chair seat that humans can sit on).

In this thesis, we aim to design a vision system that is able to infer the 3D visual world, from a single snapshot of a 2D sensor such as a camera, or a 2.5D sensor such as a Kinect depth equipment. However, inferring such a 3D parse from a single image is not always trivial. Generally, we face challenges in three folds: (1) how to infer the 3D room extent when the separation of layout surfaces and objects is unknown; (2) how to parse out the diverse set of objects, given the high degree of occlusions in cluttered scene; (3) how to represent 3D object geometry in a way that can be predicted from noisy or partial observations, and can help assist reasoning like contact, support and extent.

Instead of solving the whole task in one time, we explore the task at different levels of details, as shown in Figure 1.4. We start from the layout level to a more complex scene level and finally to the most detailed object level. At layout level, in order to infer the full extent of the scene, we estimate the 3D room layout and separate it from the foreground objects. Then, at the scene level, we jointly parse out the 3D extent of all objects and layouts in the scene, to localize every object and reason free space inside. At the object level, we reconstruct a complete 3D shape with robustness to possible partial foreground occlusions. Moreover, we further parse the object into

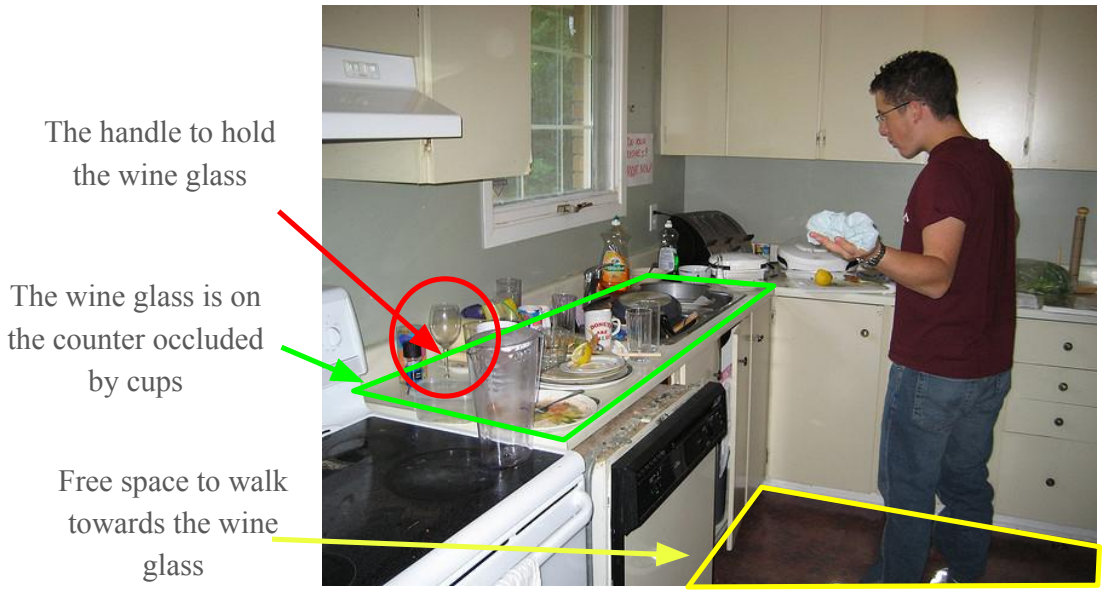


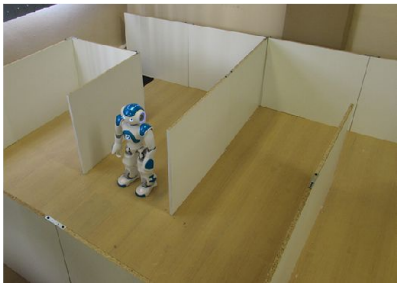
Figure 1.3: To pick up the wine glass, given a single image, the robot needs to localize the wine glass occluded by plates and cups on the counter, identify free space in front and move towards the counter, then find the glass handle to pick it up. Those successive actions require 3D scene and object parsing from a single image.

structural and compact representations (primitives) to support the functional interactions such as grasping and manipulation. We propose, at each level, an approach with proved efficacy based on extensive experiments both qualitatively and quantitatively on public datasets.

1.2 CONTRIBUTIONS

We present contributions of our approaches of 3D scene and object parsing at different levels of details: the layout level, the scene level and the object level. Details are as follows.

3D scene layout from a single RGB image. The first problem to be solved in 3D scene parsing is to estimate the 3D extent of background room layout: what's the scope of the scene? How high is the ceiling? How far are the walls? In 1999, Coughlan and Yuille [5] summarized the indoor scene statistics as "Manhattan world" assumptions: scenes were built on a Cartesian grid, as in indoor all walls are at right angles to each other and perpendicular to the floor. This representation makes up the loss of 3D geometry due to occlusions from objects in the scene, and is sparse and compact in representation that can speed up the reconstruction process. We propose LayoutNet [6], a more general RGB image to Manhattan layout algorithm that is suitable for both perspective and panoramic images. Our system compares well in speed and accuracy for panoramic images and



Layout level

Scene level

Object level

Figure 1.4: Three levels of details of 3D scene and object parsing from a single image. At layout level (left), we aim to parse out 3D layouts of wall, ceiling and floor to understand the full extent of the scene. At a more complex scene level (middle), we aim to have a complete 3D parse of all objects and the layouts in the scene, in order to localize objects and infer free space. At the most detailed object level (right), we aim to recover a complete 3D parse of objects with robustness to possible foreground occlusions, and decompose each object into a structural and functional representation for applications like robot grasping and manipulations. Images are adopted from [2, 3, 4]

achieves the second best (at the time of writing) for perspective images, while also being the fastest. We demonstrate gains from using precomputed vanishing point cues, geometric constraints, and post-process optimization, indicating that deep network approaches still benefit from explicit geometric cues and constraints. We also show that adding an objective to directly regress 3D layout parameters leads to better predictions of the boundaries and corners that are used to solve for the final predicted layout. Moreover, we extend the annotations for the Stanford 2D-3D dataset [7], providing room layout annotations that can be used in future work. Experiments on three evaluation metrics and two dataset show that these differences improve results. We further show notable improvements upon LayoutNet, noted as LayoutNet v2, by using better image feature encoder, refined training details and gradient ascent based post refinement step (Chapter 3). Our LayoutNet v2 achieves the state-of-the-art for cuboid room layout estimation.

Complete 3D scene parsing from a single RGBD image. Our next step is to jointly recover 3D objects and room layouts in the scene. We aim to parse both the visible and occluded portions of the scene and all observable objects, producing a $complete$ 3D parse. Such a scene interpretation is useful for navigation and free space reasoning, but difficult to produce due to the well-known challenge of segmentation, the high degree of occlusion, and the diversity of objects in indoor scenes. We choose to work with RGBD images so that we can focus on designing and inferring a useful representation, without immediately struggling with the added difficulty of interpreting geometry of visible surfaces. We model the extent of objects with CAD-like 3D shapes and the layout of orthogonal walls as planar surfaces that conform to a Manhattan structure. To obtain

a dataset suitable for our experiments, we refine the NYUd v2 dataset with detailed 3D shape annotations for all the objects in each image. The labeling achieves a better representation of both ground truth object shape and whole scene depth. We propose an approach to recover a 3D model of room layout and objects from an RGBD image, arguably producing the most detailed and complete 3D scene parses to date [8]. 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. We also investigate impact of support estimation on scene parsing. We hypothesize, and confirm in experiments, that support height information will help most for interpreting occluded objects because the full extent of an occluded object can be inferred from support height (Chapter 4).

Robust 3D object reconstruction beyond occlusion. One major challenge in 3D object reconstruction is to infer the complete shape geometry from partial foreground occlusion. Occlusion frequently occurs in natural scenes: *e.g.* , we often see an image of a sofa occluded by a table in front and a dining table partially occluded by a vase on top. Key challenges are the unknown $existence$ and $extent$ of an occluding region from limited view point. We propose a method to reconstruct the complete 3D shape of an object (represented by a set of point clouds) from a single RGB image, with robustness to occlusion. First, we propose a framework that predicts 3D point clouds from a single RGB image. We demonstrate performance gains by using a 2D re-projection loss on multiple synthetic views and a surface-based point cloud refinement step. Our approach achieves the state-of-the-art. We then extend our approach by predicting complete silhouettes for robustness to occlusion in real scenes. We propose a silhouette completion network that achieves the state-of-the-art. We show improvements for reconstruction of non-occluded and partially occluded objects. Experiments demonstrate the efficacy of our approach both quantitatively and qualitatively on a real scene dataset (Chapter 5).

Primitive-based 3D Object Parsing. We explore the structural and functional representation of 3D objects, in order to support applications such as robot grasping and manipulation. Inspired by the nature of human perception of 3D shapes as a collection of simple parts, we explore such an abstract shape representation based on primitives. Compared with the common voxel-based representation, primitives are much more *compact* in parameterization and are *holistic* in simplifying the reasoning about properties like stability and connectedness. One of the challenges of primitive-based 3D object modeling is to accommodate variable number of primitives within the object class they are trained for. We propose 3D-PRNN [9]: a generative recurrent neural network that reconstructs 3D shapes as sequences of primitives given a single depth image. We also propose an efficient method to fit primitives from point clouds based on Gaussian-fields and energy minimization. Our primitive representation provides enough training samples for 3D-PRNN in 3D reconstruction (Chapter 6).

1.3 DISSERTATION OUTLINE

We present our literature review of 3D scene and object parsing from a single image in Chapter 2. We introduce in Chapter 3, 4, 5 and 6 our propsoed approaches. Chapter 3 presents LayoutNet, a convolution neural network based method that predicts room layout from a single image that generalizes across panoramas and perspective images, cuboid layouts and more general layouts (*e.g.* , L-shape room). We also improve upon LayoutNet and present LayoutNet v2 with notable performance boost. Chapter 4 presents our approach of complete scene parsing from an RGBD image. Chapter 5 presents our solution to reconstruct 3D object with robustness to possible partial foreground occlusion. Chapter 6 presents 3D-PRNN, a generative recurrent neural network that parses an object as a composition of primitives to infer its structural and functional characteristics. Finally, in Chapter 7, we come to our conclusions and discuss a few directions of future work.

CHAPTER 2: LITERATURE REVIEW

2.1 SINGLE IMAGE 3D RECONSTRUCTION

In 1963, Larry Roberts defended his Ph.D. thesis at MIT to extract 3D information from 2D line drawings [10]. Roberts put forward the "Blocks World" (polyhedra) hypothesis to model the 3D structure of objects and scenes. For decades, based on the blocks world assumption, researchers have proposed various methods to improve the performance of single image 3D reconstruction.

In 1999, Coughlan and Yuille [5] summarized the indoor scene statistics as "Manhattan world" assumptions: scenes were built on a Cartesian grid, as in indoor all walls are at right angles to each other and perpendicular to the floor. This representation makes up the loss of 3D geometry due to occlusions from objects in the scene, and is sparse and compact in representation that can speed up the reconstruction process. Manhattan world assumptions have inspired researchers to perceive the 3D indoor scene as a "box world": both layouts and objects can be abstracted as oriented cuboids, inferring the rough 3D shape as well as the accurate 3D location and pose of objects and layouts, as shown in Figure 2.1. This can be applied for the task of 3D object detection [11], 3D scene reconstruction [12, 13, 14], and inferring of the free space and occupancy in the 3D scene [15, 16].

(a) (b)

Figure 2.1: Illustration of (a) utilizing the Manhattan world assumptions to estimate scene orientations and (b) the "box in the box" concept of 3D scene and object parsing from a single image. Images are adopted from [5, 13, 11].

More recent approaches continue to follow the concept of the "Manhattan world" assumptions and the "Blocks World" hypothesis, and attempt to solve 3D scene and object parsing from a single image at three different levels of details: layout level, scene level and object level. We discuss related work at each level of detail as follows.

2.2 3D LAYOUT FROM A SINGLE IMAGE

Single-view room layout estimation has been an active topic of research for the past decades. Delage *et al*. [17] fit floor/wall boundaries in a perspective image taken by a level camera to create a 3D model under "Manhattan world" assumptions [5]. A special case is the cuboid model, in which four walls, ceiling, and floor enclose the room. Lee *et al*. [18] produce Orientation Maps, generate layout hypotheses based on detected line segments, and select a best-fitting layout from among them. Hedau *et al*. [15] recover cuboid layouts by solving for three vanishing points, sampling layouts consistent with those vanishing points, and selecting the best layout based on edge and Geometric Context [19] consistencies. Subsequent works follow a similar approach, with improvements to layout generation [12, 20, 21], features for scoring layouts [20, 21], and incorporation of object hypotheses [11, 22, 23, 24, 25] or other context. The most recent methods train deep network features to classify pixels into layout surfaces like walls and ceiling [26, 27], boundaries [28], corners [29], or a combination [30].

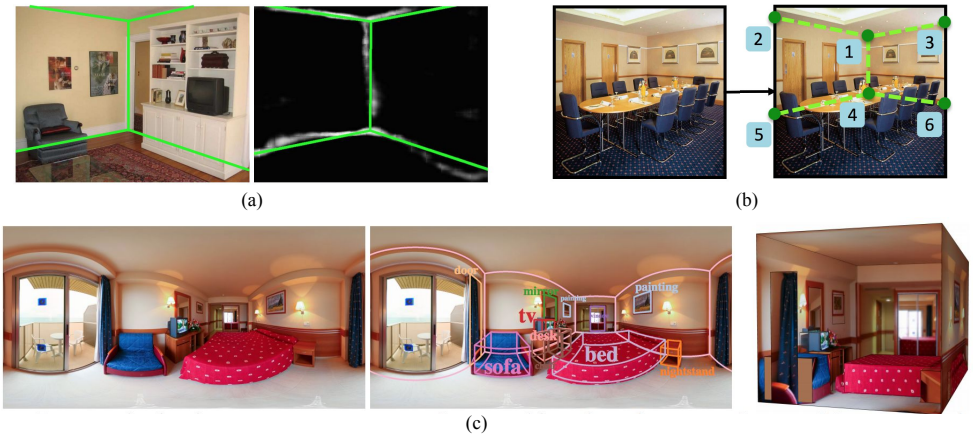


Figure 2.2: 3D geometric cues for single image layout reconstruction: (a) line (boundary) cues [28]; (b) corner cues [29]; (c) Manhattan layout assumptions (special case as a cuboid) for a single panorama [31]. Images are adopted from the cited paper.

Nearly all of these works aim to produce cuboid-shaped layouts from perspective RGB images. A few works also operate on panoramic images [31, 32, 33] or RGBD images [34, 14, 35, 36]. Methods also exist to recover axis-aligned, piecewise-planar models of interiors from large collections of images [37, 38] or laser scans [39], benefiting from more complete scene information with fewer occlusion. Rent3D [40] takes advantage of a known floor plan.

2.3 3D SCENE PARSING FROM A SINGLE IMAGE

The term 3D scene parsing refers to the process of recovering the layout and shape of surfaces and objects in a scene. 3D scene parsing from a single image is a foundational problem in computer vision. One major challenge is to cope with the huge diversity of layouts and objects under heavy occlusion, due to the complexity of natural scenes. Previous approaches utilize contextual relations and physical constraints to recover cuboid representations of layouts and objects from a single RGB image [25, 41, 42, 43], or from a single RGBD image [44]. However, cuboids do not provide good shape approximations to chairs, tables, sofas, and many other common objects. Therefore, researchers proposed approaches to fit CAD-like models to depicted objects. Song and Xiao [45] search for chairs, beds, toilets, sofas, and tables by sliding 3D windows and enumerating all possible poses. Song *et al.*then extend the method to detect a larger variety of classes [46] with CNN-based approach and interprets semantic voxel representations from a single depth map [47]. Izadinia *et al.* [27] infer the CAD models of layouts pieces and objects via render-and-match. Huang *et al.* [48] further improve the performance by involving latent human context – affordance and the functionality of the arrangement of the room.

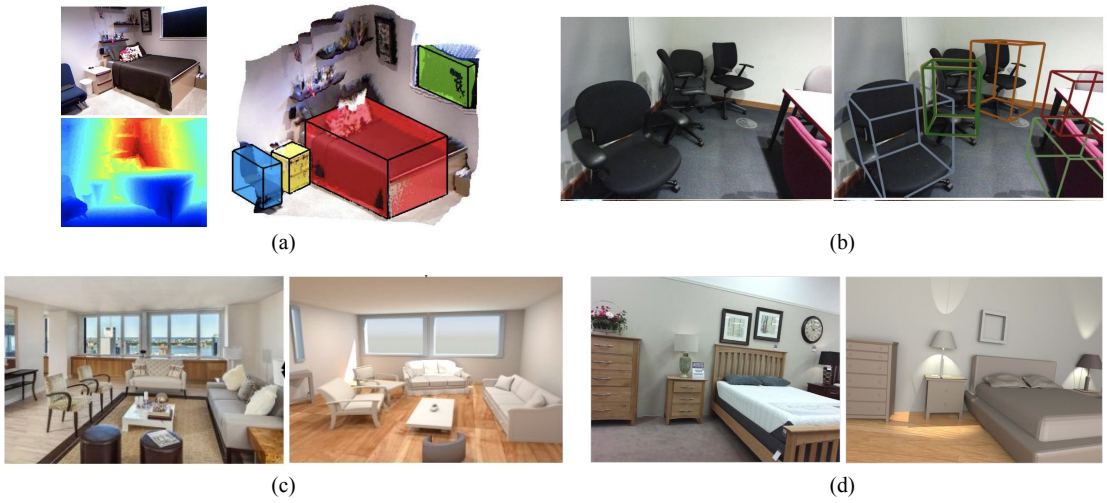


Figure 2.3: Approaches for 3D scene parsing from a single image: cuboid layouts and objects recovery from (a) a single RGBD image [42] or (b) a single RGB image [43]; using CAD-like or mesh models to fit 3D objects and layouts from a single image [27, 48] as in (c) and (d). Images are adopted from the cited papers.

In addition, object height priors have shown to be crucial geometric cues for better object detection [49, 50, 51, 45, 52, 53]. Deng *et al.* [54] apply height above ground to distinguish objects. Guo and Hoiem [55] localize the height and full extent of support surfaces from one RGBD image.

2.4 3D OBJECT PARSING

Many robotics and graphics applications require 3D shape interpretations from sensory data. For example, picking up a cup or moving a chair all rely on at least a vague idea of object position, shape, contact and connectedness. This can be attempted given an RGB image [56, 57, 58, 59, 60, 61, 62, 63, 64], a depth image [65, 66, 67, 68] or both [52, 69, 70]. Recent approaches mainly either focus on part- and object-based retrieval from large repositories [27, 71, 52, 66, 63, 72, 73], deformations from meshes [74, 75, 56] or a direct voxelized volumetric prediction via convolution neural networks [57, 61, 76, 64]. A better model fitting includes part deformation [77] and symmetry [78]. Wu *et al*.present preliminary results on automatic shape completion from depth by classifying hidden voxels with a deep network [67]. Wu *et al*.reconstruct shapes based on predicted skeletons [57].

Seeing beyond occlusions. Occlusions have long been an obstacle in both single-view and multi-view reconstruction. Solutions have been proposed to recover portions of surfaces from single views, *e.g*. with synthetic apertures [79, 80], or to otherwise improve robustness of matching and completion functions from multiple views [81, 82, 83]. Other work decompose a scene into layered depth maps from RGBD [36] or video [84] and then seek to complete the occluded portions of the maps. But errors in layered segmentation can severely degrade the recovery of the occluded region. Learning-based approaches [85, 86, 87] have posed recovery from occlusion as a 2D semantic segmentation completion task. Ehsani *et al*. [86] propose to complete the silhouette and color/texture of an occluded object.

Point clouds based 3D object representations. We explore 3D shape representations as a set of point clouds, which is flexible and easy to transform. Qi *et al*. [88] propose a novel deep net architecture suitable for consuming unordered point sets in 3D; Fan *et al*. [89] propose to generate point clouds from a single RGB image using generative models. More recent approaches improve point set reconstruction performance by learning representative latent features [90] or by imposing constraints of geometric appearance in multiple views [91].

Primitive-based 3D object representations. We also explore shape representations as aggregations of primitives that has the benefit for lower computational and storage cost. Biederman, in the early 1980s, popularized the idea of representing shapes as a collection of components or primitives called "geons" [94], and early computer vision algorithms attempted to recover object-centered 3D volumetric primitives from single images [95]. have long been popular in psychology [94], interactive graphics [96] or shape completion from point clouds [97, 98, 99]. In the case that shapes often have canonical parts like planes or boxes and efficient solution for large data is required, primitives are used in reconstructions of urban and architectural scenes [100, 101, 102, 103].

Figure 2.4: 3D object parsing and object representations: Given an RGB image, approaches explore to recover the complete shape represented by (a) 3D voxels [92], (b) a 3D mesh [56] or (c) 3D surfaces [59]; objects can also be decomposed into primitives (oriented cuboids) from a 3D volume as in (d) [93]. Images are adopted from the cited papers.

2.5 RELATION TO OUR WORK

We discuss the relationship between the previous approaches and our work at three levels of details: layout level, scene level and object level.

2.5.1 Layout level

At the layout level, our work is most similar in goal to PanoContext [31] and in approach to RoomNet [29]. PanoContext extends the frameworks designed for perspective images to panoramas, estimating vanishing points, generating hypotheses, and scoring hypotheses according to Orientation Maps, Geometric Context, and object hypotheses. To compute these features, PanoContext first projects the panoramic image into multiple overlapping perspective images, and then combines the feature maps back into a panoramic image. Our approach is more direct: after aligning the panoramic image based on vanishing points, our system uses a deep network to predict boundaries and corners directly on the panoramic image. In this regard, we are similar to RoomNet, which uses a deep network to directly predict layout corners in perspective images, as well as a label that indicates which corners are visible. Our method differs from RoomNet in several ways. Our method applies to panoramic images. Our method also differs in the alignment step (RoomNet performs none) and in our multitask prediction of boundaries, corners, and 3D cuboid parameters. Our final inference is constrained to produce a Manhattan 3D layout. RoomNet uses an RNN to refine 2D corner position predictions, but those predictions might not be consistent

with any 3D cuboid layout. Our experiments show that all of these differences improve results.

More generally, we propose the first method, to our knowledge, that applies to both perspective and panoramic images. We also show that our method extends easily to non-cuboid Manhattan layouts. Thus, our method is arguably the most general and effective approach to date for indoor layout estimation from a single RGB image. Moreover, our approach simplifies reconstruction by estimating layout directly on a single RGB equirectangular panorama: our final output is a sparse and compact planar Manhattan layout parameterized by each wall's distance to camera, height, and the layout rotation.

2.5.2 Scene level

The most related work that recovers complete models from scene is proposed by Zhang *et al*. [31], where they predict 3D bounding boxes of the room and all major objects inside, together with their semantic categories from an RGB 360° full-view panorama. Different from Zhang *et al*.: we interpret whole-room 3D context with detailed 3D shapes and layout planes; our input is a single image, which has limited field of view; and we make use of depth information, which eases the difficulty of interpreting geometry of visible surfaces.

Different from the existing methods that try to recover detailed shape for cuboid layouts and the main furniture in the scene, our approach finds a detailed shape for any object and layout in the scene from a single RGBD image. Our approach is efficient and light-weight compared with ScanNet [104], which utilizes RGB-D video dataset to annotate instance-level semantic segmentations, since our model operates on single images instead of videos. Moreover, our detailed shape prediction can be utilized as an refining tool for existing dataset with 3D bounding box ground truth.

Our approach also relates to parsing [105, 106] of RGBD images and generation of object-like regions [107, 108, 109]. We propose to generate object-like regions and retrieve their related 3D shape via region transfer, which is inspired by the SuperParsing method of Tighe and Lazebnik [110]. Similar ideas have also been used in other modalities: Karsch *et al*. [111] transfer depth, Guo and Hoiem [112] transfer polygons of background regions, Yamaguchi *et al*. [113] transfer clothing items. Exemplar-based 3D modeling is also employed by Satkin and Hebert [114] to transfer 3D geometry and object labels from entire scenes.

2.5.3 Object level

One major challenge in 3D object parsing is to infer the complete 3D shape with robustness to partial foreground occlusions. Most of the approaches of 3D object parsing from a single image

are applied to non-occluded objects with clean backgrounds and no occlusions, which may prevent their application to natural images. Sun *et al*. [115] conduct experiments on real images from Pix3D, a large-scale dataset with aligned ground-truth 3D shapes, but do not consider the problem of occlusion. We are concerned with predicting shape of objects in natural scenes, which may be partly occluded. Our approach improves the state-of-the-art for object point set generation, and is extended to reconstruct beyond occlusion with the guidance of completed silhouettes. That is, We go further to try to predict the complete 3D shape of the occluded object.

Another major challenge in 3D object parsing is to have a functional and structural representation of objects for applications such as robot grasping. Recently, more compact and parametric representations in the form of template objects [57], and set of cuboid primitives [93] have been introduced. These representations, however, require non trivial effort to accommodate variable number of configurations within the object class they are trained for. This is mainly because of their single feed-forward design, which implicitly forces the prediction of a discrete number of variables at the same time. Different from previous approaches, we propose to sequentially generates primitives. Our approach is inspired by Graves' work [116] to sequentially generate parameterized handwriting strokes and the PixelRNN approach [117] to model natural images as sequentially generated pixels. To produce parameterized 3D primitives (oriented cuboids), we customize the RNN to encode explicit geometric constraints of symmetry and rotation. For example, separately predicting whether a primitive should rotate along each axis and by how much improves results over more simply predicting rotation values, since many objects consist of several (unrotated) cuboids.

CHAPTER 3: MANHATTAN ROOM LAYOUT FROM A SINGLE RGB IMAGE

The first problem to be solved in 3D scene parsing is to estimate the extent of room layout: whats the scope of the room? How high is the ceiling? How far are the walls? In this chapter, we propose an algorithm to predict room layout from a single image that generalizes across panoramas and perspective images, cuboid layouts and more general layouts (*e.g.* , "L"-shape room). Our method operates directly on the panoramic image, rather than decomposing into perspective images as do recent works. Our network architecture is similar to that of RoomNet [29], but we show improvements due to aligning the image based on vanishing points, predicting multiple layout elements (corners, boundaries, size and translation), and fitting a constrained Manhattan layout to the resulting predictions. Our method compares well in speed and accuracy to other existing work on panoramas, achieves among the best accuracy for perspective images, and can handle both cuboid-shaped and more general Manhattan layouts.

Figure 3.1: **Illustration.** Our LayoutNet predicts a non-cuboid room layout from a single panorama under equirectangular projection.

3.1 INTRODUCTION

Estimating the 3D layout of a room from one image is an important goal, with applications such as robotics and augmented reality. The room layout specifies the positions, orientations, and heights of the walls, relative to the camera center. The layout can be represented as a set of projected corner positions or boundaries, or as a 3D mesh. Existing works apply to special cases of the problem, such as predicting cuboid-shaped layouts from perspective images or from panoramic images.

We present **LayoutNet**, a deep convolution neural network that estimates the 3D layout of an

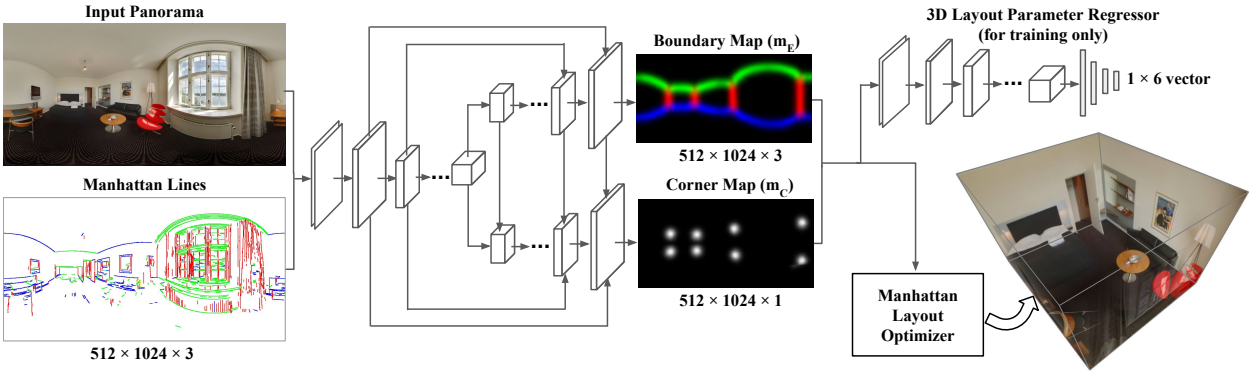


Figure 3.2: **Overview.** Our LayoutNet follows the encoder-decoder strategy. The network input is a concatenation of a single RGB panorama and Manhattan line map. The network jointly predicts layout boundaries and corner positions. The 3D layout parameter loss encourages predictions that maximize accuracy. The final prediction is a Manhattan constrained layout reconstruction.

indoor scene from a single perspective or panoramic image (Figure. 3.1). Our method compares well in speed and accuracy on panoramas and is among the best on perspective images. Our method also generalizes to non-cuboid Manhattan layouts, such as "L"-shaped rooms.

Our LayoutNet approach operates in three steps (Figure 3.2). First, our system analyzes the vanishing points and aligns the image to be level with the floor (Chapter 3.2.1). This alignment ensures that wall-wall boundaries are vertical lines and substantially reduces error according to our experiments. In the second step, corner (layout junctions) and boundary probability maps are predicted directly on the image using a CNN with an encoder-decoder structure and skip connections (Chapter 3.2.2). Corners and boundaries each provide a complete representation of room layout. We find that jointly predicting them in a single network leads to better estimation. Finally, the 3D layout parameters are optimized to fit the predicted corners and boundaries (Cfhapter 3.3). The final 3D layout loss from our optimization process is difficult to back-propagate through the network, but direct regression of the 3D parameters during training serves as an effective substitute, encouraging predictions that maximize accuracy of the end result.

3.2 APPROACH

3.2.1 Panoramic image alignment

Given the input as a panorama that covers a 360° horizontal field of view, we first align the image by estimating the floor plane direction under spherical projection, rotate the scene, and re-project it to the 2D equirectangular projection. Similar to Zhang *et al.*'s approach [31], we extract long line

segments using the Line Segment Detector (LSD) [118] in each overlapped perspective view, then compute the vanishing points using the Hough Transform. This alignment step eases our network training. The detected candidate Manhattan line segments also provide additional input features that improve the performance, as shown in Chapter 3.5.

3.2.2 Network design

An overview of the LayoutNet network is illustrated in Figure 3.2. The network follows an encoder-decoder strategy.

Deep panorama encoder. The input is a 6-channel feature map: the concatenation of single RGB panorama with resolution of 512×1024 (or 512×512 for perspective images) and the Manhattan line feature map lying on three orthogonal vanishing directions using the alignment method in Chapter 3.2.1. The encoder contains 7 convolution layers with kernel size of 3×3. Each convolution is followed by a ReLU operation and a max pooling layer with the down-sampling factor of 2. The first convolution contains 32 features, and we double size after each convolution. This deep structure ensures a better feature learning from high resolution images and help ease the decoding step. We tried Batch Normalization after each convolution layer but observe lower accuracy. We also explored an alternative structure that applies a separate encoder for the input image and the Manhattan lines, but observe no increase in performance compared with our current simpler design.

2D layout decoder The decoder consists of two branches as shown in Figure 3.2. The top branch, the layout boundary map ($\boldsymbol{m_E}$) predictor, decodes the bottleneck feature into the 2D feature map with the same resolution as the input. $\boldsymbol{m_E}$ is a 3-channel probability prediction of wall-wall, ceiling-wall and wall-floor boundary on the panorama, for both visible and occluded boundaries. The boundary predictor contains 7 layers of nearest neighbor up-sampling operation, each followed by a convolution layer with kernel size of 3×3, and the feature size is halved through layers from 2048. The final layer is a Sigmoid operation. We add skip connections to each convolution layer following the spirit of the U-Net structure [119], in order to prevent shifting of predictions results from the up-sampling step. The lower branch, the 2D layout corner map ($\boldsymbol{m_C}$) predictor, follows the same structure as the boundary map predictor and additionally receives skip connections from the top branch for each convolution layer. This stems from the intuition that layout boundaries imply corner positions, especially for the case when a corner is occluded. We show in our experiments (Chapter 3.5) that the joint prediction helps improve the accuracy of the both maps, leading to a better 3D reconstruction result. We experimented with fully convolutional layers [120] instead of the up-sampling plus convolutions structure, but observed worse performance with checkerboard artifacts.

3D layout regressor The function to map from 2D corners and boundaries to 3D layout parameters is simple mathematically, but difficult to learn. So we train a regressor for 3D layout parameters with the purpose of producing better corners and boundaries, rather than for its own sake. As shown in Figure 3.2, the 3D regressor gets as input the concatenation of the two predicted 2D maps and predicts the parameters of the 3D layout. We parameterize the layout with 6 parameters, assuming the ground plane is aligned on the $x - z$ axis: width s_w, length s_l, height s_h, translation $T = (t_x, t_z)$ and rotation r_θ on the $x - z$ plane. The regressor follows an encoder structure with 7 layers of convolution with kernel size 3×3, each followed by a ReLU operation and a max pooling layer with the down sampling factor of 2. The convolution feature size doubles through layers from the input 4 feature channel. The next four fully-connected layers have sizes of 1024, 256, 64, and 6, with ReLU in between. The output 1×6 feature vector $d = \{s_w, s_l, s_h, t_x, t_z, r_\theta\}$ is our predicted 3D cuboid parameter. Note that the regressor outputs the parameters of the 3D layout that can be projected back to the 2D image, presenting an end-to-end prediction approach. We observed that the 3D regressor is not accurate (with corner error of 3.36% in the PanoContext dataset compared with other results in Table 3.1), but including it in the loss objective tends to slightly improve the predictions of the network. The direct 3D regressor fails due to the fact that small position shifts in 2D can have a large difference in the 3D shape, making the network hard to train.

Loss function. The overall loss function of the network is:

$$\begin{aligned} L(\boldsymbol{m_E}, \boldsymbol{m_C}, \boldsymbol{d}) = &-\alpha \frac{1}{n} \sum_{p \in \boldsymbol{m_E}} \big(\hat{p} \log p + (1 - \hat{p}) \log(1 - p)\big) \\ &- \beta \frac{1}{n} \sum_{q \in \boldsymbol{m_C}} \big(\hat{q} \log q + (1 - \hat{q}) \log(1 - q)\big) + \tau \|\boldsymbol{d} - \hat{\boldsymbol{d}}\|_2 \end{aligned} \tag{3.1}$$

Here $\boldsymbol{m_E}$ is the probability that each image pixel is on the boundary between two walls; $\boldsymbol{m_C}$ is the probability that each image pixel is on a corner; d are the regressed cuboid parameters which have ground truth $\hat{\boldsymbol{d}}$; p and q are pixel probabilities of edge and corner with ground truth values of $\hat{p}$ and $\hat{q}$, respectively. The loss is the summation over the binary cross entropy error of the predicted pixel probability in $\boldsymbol{m_E}$ and $\boldsymbol{m_C}$ compared with ground truth, plus the Euclidean distance of regressed 3D cuboid parameters d to the ground truth $\hat{\boldsymbol{d}}$. Note that the RoomNet approach [29] uses L2 loss for corner prediction. We discuss the performance using two different losses in Chapter 3.5. α, β and τ are the weights for each loss term. In our experiment, we set $\alpha = \beta = 1$ and $\tau = 0.01$.

Training details. Our LayoutNet predicts pixel probabilities for corners and boundaries and regresses the 3D layout parameters. We find that joint training from a randomly initialized network sometimes fails to converge. Instead, we train each sub-network separately and then jointly

train them together. For the 2D layout prediction network, we first train on the layout boundary prediction task to initialize the parameters of the network. For the 3D layout regressor, we first train the network with ground truth layout boundaries and corners as input, and then connect it with the 2D layout decoder and train the whole network end-to-end.

The input Manhattan line map is a 3 channel 0-1 tensor. We normalize each of the 3D cuboid parameter into zero mean and standard deviation across training samples. We use ADAM [121] to update network parameters with a learning rate of e^{-4}, $\alpha = 0.95$ and $\epsilon = e^{-6}$. The batch size for training the 2D layout prediction network is 5 and changes to 20 for training the 3D regressor. The whole end-to-end training uses a batch size of 20.

Ground truth smoothing. Our target 2D boundary and corner map is a binary map with a thin curve or point on the image. This makes training more difficult. For example, if the network predicts the corner position slightly off the ground truth, a huge penalty will be incurred. Instead, we dilate the ground truth boundary and corner map with a factor of 4 and then smooth the image with a Gaussian kernel of 20×20. Note that even after smoothing, the target image still contains $\tilde{9}5\%$ zero values, so we re-weight the back propagated gradients of the background pixels by multiplying with 0.2.

Data augmentation. We use horizontal rotation, left-right flipping and luminance change to augment the training samples. The horizontal rotation varies from $0^o - 360^o$. The luminance varies with γ values between 0.5-2. For perspective images, we apply $\pm 10^\circ$ rotation on the image plane.

Algorithm 3.1 3D layout optimization

Given panorama I, layout corner prediction $\boldsymbol{m_C}$, and boundary prediction $\boldsymbol{m_E}$;
Initialize 3D layout L_0 based on Eq. 3.2;
$E_{best} = Score(L_0)$ by Eq. 3.3, $L_{best} = L_0$;
for $i = 1$:wallNum **do**
    Sample candidate layouts L_i by varying wall position w_i in 3D, fix other wall positions;
    for $j = 1 : |L_i|$ **do**
        Sample candidate Layouts L_{ij} by varying floor and ceiling position in 3D;
        Rank the best scored Layout $L_B \in \{L_{ij}\}$ based on Eq. 3.3;
        if $E_{best} < Score(L_B)$ **then**
            $E_{best} = Score(L_B)$, $L_{best} = L_B$;
    Update w_i from L_{best}, fix it for following sampling
return L_{best}

3.3 MANHATTAN LAYOUT OPTIMIZATION

The initial 2D corner predictions are obtained from the corner probability maps that our network outputs. First, the responses are summed across rows, to get a summed response for each column. Then, local maxima are found in the column responses, with distance between local maxima of at least 20 pixels. Finally, the two largest peaks are found along the selected columns. These 2D corners might not satisfy Manhattan constraints, so we perform optimization to refine the estimates.

Given the predicted corner positions, we can directly recover the camera position and 3D layout, up to a scale and translation, by assuming that bottom corners are on the same ground plane and that the top corners are directly above the bottom ones. We can further constrain the layout shape to be Manhattan, so that intersecting walls are perpendicular, *e.g.* , like a cuboid or "L"-shape in a top-down view. For panoramic images, the Manhattan constraints can be easily incorporated, by utilizing the characteristic that the columns of the panorama correspond to rotation angles of the camera. We parameterize the layout coordinates in the top-down view as a vector of 2D points $\boldsymbol{L_v} = \{\boldsymbol{v_1} = (0,0), \boldsymbol{v_2} = (x_1, y_1), \ldots, \boldsymbol{v_N} = (x_N, y_N)\}$. $\boldsymbol{v_1}$ resolves the translation ambiguity, and $|\boldsymbol{v_1} - \boldsymbol{v_2}| = 1$ sets the scale. Because the layout is assumed to be Manhattan, neighboring vertices will share one coordinate value: $y_1 = y_0 = 0, x_2 = x_1, y_3 = y_2 \ldots$, reducing the number of free parameters to $N - 2$. We denote the camera position as $\boldsymbol{v_c} = \{x_c, y_c\}$, where $0 \leq x_c \leq max\{x_N, x_N \in \boldsymbol{L_v}\}$ and $0 \leq y_c \leq max\{y_N, y_N \in \boldsymbol{L_v}\}$ since the camera position is within the layout extent. We recover the camera position $\boldsymbol{v_c} = \{x_c, y_c\}$ and $\boldsymbol{L_v}$ by minimizing the following energy function inspired by Farin *et al.* [122]:

$$E(\boldsymbol{L_v}, \boldsymbol{v_c}) = \sum_{i,j} \|\beta(\boldsymbol{v_i}, \boldsymbol{v_j}) - \alpha(\boldsymbol{v_i}, \boldsymbol{v_j})\|_2, i = j - 1, 0 \leq i \leq N - 1 \tag{3.2}$$

Where $\boldsymbol{v_i}, \boldsymbol{v_j}$ are a pair of neighboring vertices, and $\beta(\boldsymbol{v_i}, \boldsymbol{v_j}) = arccos\frac{\boldsymbol{v_i}-\boldsymbol{v_c}\cdot\boldsymbol{v_j}-\boldsymbol{v_c}}{\|\boldsymbol{v_i}-\boldsymbol{v_c}\|\|\boldsymbol{v_j}-\boldsymbol{v_c}\|}$ is the rotation angle between $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ centered at the camera $\boldsymbol{v_c}$ in top-down view. $\alpha(\boldsymbol{v_i}, \boldsymbol{v_j})$ denotes the horizontal distance (number of pixels) between $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ projected on the panorama (visualized as a pair of vertical axis aligned corners on the same wall-wall boundary) under rectangular projection, divided by the overall length of the panorama. The energy function is derived from the observation that the length (horizontal dimension) of the panorama is equal to the summation of the rotation angles of the camera between all neighboring wall-wall boundaries in top-down view, resulting in a total of 360 degrees. Note that the $L2$ minimization in Eq. 3.2 also applies to general Manhattan layouts with $N > 3$. We use L-BFGS [123] to solve for Eq. 3.2 efficiently.

We initialize the ceiling level as the average (mean) of 3D upper-corner heights, and then optimize for a better fitting room layout, relying on both corner and boundary information using the

following score to evaluate 3D layout candidate L:

$$\begin{aligned} Score(L) =& w_{junc} \sum_{l_c \in C} \log P_{\text{corner}}(l_c) + w_{ceil} \sum_{l_e \in L_e} \max_{p_e \in l_e} \Big(\log P_{\text{ceil}}(p_e) \Big) \\ &+ w_{floor} \sum_{l_f \in L_f} \max_{p_f \in l_f} \Big(\log P_{\text{floor}}(p_f) \Big) \end{aligned} \tag{3.3}$$

where C denotes the 2D projected corner positions of L. The cardinality of L is #walls$\times 2$. We connect the nearby corners on the image to obtain L_e which is the set of projected wall-ceiling boundaries, and L_f is the set of projected wall-floor boundaries (each with cardinality of #walls). $P_{\text{corner}}(\cdot)$ denotes the pixel-wise probability value on the predicted $\boldsymbol{m_C}$. $P_{\text{ceil}}(\cdot)$ and $P_{\text{floor}}(\cdot)$ denote the probability on $\boldsymbol{m_E}$. The 2nd and 3rd term take the maximum value of the pixel's log likelihood response in each boundary $l_e \in L_e$ and $l_f \in L_f$. w_{junc}, w_{ceil} and w_{floor} are the term weights, we set to 1.0, 0.5 and 1.0 respectively using grid search. This weighting conforms with the observation that wall-floor corners are often occluded, and the predicted boundaries could help improve the layout reconstruction. We find that adding wall-wall boundaries in the scoring function helps less, since the vertical pairs of predicted corners already reveals the wall-wall boundaries information.

Directly optimizing Eq. 3.3 is computationally expensive, since we penalize on 2D projections but not direct 3D properties. In this case, we instead sample candidate layout shapes and select the best scoring result based on Eq. 3.3. We use line search to prune the candidate numbers to speed up the optimization. Algorithm 3.1 demonstrates the procedure. In each step, we sample candidate layouts by shifting one of the wall position within $\pm\%10$ of its distance to the camera center. Each candidate's ceiling and floor level is then optimized based on the same sampling strategy and scored based on Eq. 3.3. Once we find the best scored layout by moving one of the walls, we fix this wall position, move to the next wall and perform the sampling again. We start from the least confident wall based on our boundary predictions. In total, ~ 1000 layout candidates are sampled. The optimization step spends less then 30 sec for each image and produces better 3D layouts as demonstrated in Chapter 3.5.

3.4 EXTENSIONS

With small modifications, our network, originally designed to predict cuboid layouts from panoramas, can also predict more general Manhattan layouts from panoramas and cuboid-layouts from perspective images.

General Manhattan layouts. To enable more general layouts, we include training examples that have more than four walls visible (*e.g.* , "L"-shaped rooms), which applies to about 10% of

examples. We then determine whether to generate four or six walls by thresholding the score of the sixth strongest wall-wall boundary. Specifically, the average probability along the sixth strongest column of the corner map is at least 0.05. In other words, if there is evidence for more than four walls, our system generates additional walls; otherwise it generates four. Since the available test sets do not have many examples with more than four walls, we show qualitative results with our additional captured samples in Chapter 3.5.2.

Note that there will be multiple solutions given non-cuboid layout when solving Eq. 3.2. We experimented with predicting a concave/convex label as part of the corner map prediction to obtain single solution, but observed degraded 2D prediction. We thus enumerate all possible shapes (*e.g.* , for room with six walls, there will be six variations) and choose the one with the best score. We found this heuristic search to be efficient as it searches in a small discrete set. We do not train with the 3D parameter regressor for the non-cuboid layout.

Perspective images. When predicting on perspective images, we skip the alignment and optimization steps, instead directly predicting corners and boundaries on the image. We also do not use the 3D regressor branch. The network predicts a 3-channel boundary layout map with ceiling-wall, wall-wall and wall-floor boundaries, and the corner map has eight channels for each possible corner. Since perspective images have smaller fields of view and the number of visible corners varies, we add a small decoding branch that predicts the room layout type, similar to RoomNet [29]. The predictor has 4 fully-connected (fc) layers with 1024, 256, 64 and 11 nodes, with ReLU operations in between. The predicted layout type then determines which corners are detected, and the corners are localized as the most probable positions in the corner maps. We use cross entropy loss to jointly train the layout boundary and corner predictors. To ease training, similar to the procedure in Chapter 3.2.2, we first train the boundary/corner predictors, and then add the type predictor branch and train all components together.

3.5 EXPERIMENT

We implement our LayoutNet with Torch and test on a single NVIDIA Titan X GPU. The layout optimization is implemented with Matlab R2015a and is performed on Linux machine with Intel Xeon 3.5G Hz in CPU mode.

We demonstrate the effectiveness of our approach on the following tasks: (1) predict 3D cuboid layout from a single panorama, (2) estimate 3D non-cuboid Manhattan layout from a single panorama, and (3) estimate layout from a single perspective image. We train only on the training split of each public dataset and tune the hyper-parameters on the validation set. We report results on the test set. Our final corner/boundary prediction from the LayoutNet is averaged over results with input

Figure 3.3: **Qualitative results (randomly sampled) for cuboid layout prediction on PanoContext dataset**. We show both our method's performance (even columns) and the state-of-the-art [31] (odd columns). Each image consists predicted layout from given method (orange lines) and ground truth layout (green lines).

of the original panoramas/images and the left-right flipped ones.

3.5.1 Cuboid layout for panorama

We evaluate our approach on three standard metrics: (1) 3D intersection over union (**IoU**), calculated between our predicted 3D layout and the ground truth and averaged across all images; (2) **Corner error**, the $L2$ distance between predicted room corner and the ground truth, normalized by the image diagonal and averaged across all images; (3) **Pixel error**, the pixel-wise accuracy between the layout and the ground truth, averaged across all images.

We perform our method using the same hyper-parameter on the following two datasets.

PanoContext dataset The PanoContext dataset [31] contains 500 annotated cuboid layouts of indoor environments such as bedrooms and living rooms. Since there is no existing validation set, we carefully split 10% validation images from the training samples so that similar rooms do not appear in the training split. Table 3.1 shows the quantitative comparison of our method, denoted as "ours full (corner+boundary+3D)", compared with the state-of-the-art cuboid layout estimation by Zhang *et al.* [31], denoted as "PanoContext". Note that PanoContext incorporates object detection as a factor for layout estimation. Our LayoutNet directly recovers layouts and outperforms the state-of-the-art on all the three metrics. Figure 3.3 shows the qualitative comparison. Our approach presents better localization of layout boundaries, especially for a better estimate on

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
PanoContext [31]	67.23	1.60	4.55
ours (corner)	73.16	1.08	4.10
ours (corner+boundary)	73.26	1.07	**3.31**
ours full (corner+boundary+3D)	**74.48**	**1.06**	3.34
ours w/o alignment	69.91	1.44	4.39
ours w/o cuboid constraint	72.56	1.12	3.39
ours w/o layout optimization	73.25	1.08	3.37
ours w/ $L2$ loss	73.55	1.12	3.43
ours full w/ Stnfd. 2D-3D data	75.12	1.02	3.18

Table 3.1: Quantitative results on cuboid layout estimation from panorama using PanoContext dataset [31]. We compare the PanoContext method, and include an ablation analysis on a variety of configurations of our method. Bold numbers indicate the best performance when training on PanoContext data.

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
ours (corner)	72.50	1.27	3.44
ours (corner+boundary)	75.26	1.03	**2.68**
ours full (corner+boundary+3D)	75.39	**1.01**	2.70
ours w/o alignment	68.56	1.56	3.70
ours w/o cuboid constraint	74.13	1.08	2.87
ours w/o layout optimization	74.47	1.07	2.92
ours w/ $L2$ loss	**76.33**	1.04	2.70
ours full w/ PanoContext data	77.51	0.92	2.42

Table 3.2: Evaluation on our labeled Stanford 2D-3D annotation dataset. We evaluate our LayoutNet approach with various configurations for ablation study. Bold numbers indicate best performance when training only on Stanford 2D-3D training set.

occluded boundaries, and is much faster in time as shown in Table 3.3.

Our labeled Stanford 2D-3D annotation dataset The dataset contains 1413 equirectangular RGB panorama collected in 6 large-scale indoor environment including office and classrooms and open space like corridors. Since the dataset does not contain applicable layout annotations, we extend the annotations with carefully labeled 3D cuboid shape layout, providing 571 RGB panoramas with room layout annotations. We evaluate our LayoutNet quantitatively in Table 3.2 and qualitatively in Figure 3.4. Although the Stanford 2D-3D annotation dataset is more challenging with smaller vertical field of view (FOV) and more occlusions on the wall-floor boundaries, our LayoutNet recovers the 3D layouts well.

Ablation study. We show, in Table 3.1 and Table 3.2, the performance given the different configurations of our approach: (1) with only room corner prediction, denoted as "ours (corner)";

Figure 3.4: **Qualitative results (randomly sampled) for cuboid layout prediction on the Stanford 2D-3D annotation dataset.** This dataset is more challenging than the PanoContext dataset, due to a smaller vertical field of view and more occlusion. We show our method's predicted layout (orange lines) compared with the ground truth layout (green lines).

Method	Average CPU time (s)
PanoContext [31]	> 300
ours full (corner+boundary+3D)	44.73
ours w/o alignment	31.00
ours w/o cuboid constraint	13.75
ours w/o layout optimization	14.23

Table 3.3: Average CPU time for each method. We evaluate the methods on the PanoContext dataset [31] using Matlab on Linux machine with an Intel Xeon 3.5G Hz (6 cores).

(2) joint prediction of corner and boundary, denoted as "ours (corner+boundary)"; (3) our full approach with 3D layout loss, denoted as "ours full (corner+boundary+3D)"; (4) our full approach trained on a combined dataset; (5) our full approach without alignment step; (6) our full approach without cuboid constraint; (7) our full approach without layout optimization step; and 8) our full approach using $L2$ loss for boundary/corner prediction instead of cross entropy loss. Our experiments show that the full approach that incorporates all configurations performs better across all the metrics. Using cross entropy loss appears to have a better performance than using L2. Training with 3D regressor has a small impact, which is the part of the reason we do not use it for perspective images. Table 3.3 shows the average runtimes for different configurations.

Comparison to other approaches: We compare with Yang *et al.* [32] based on their depth distribution metric. We directly run our full cuboid layout prediction (deep net trained on PanoContext + optimization) on 88 indoor panoramas collected by Yang *et al.*.

As shown in Table 3.4, our approach outperforms Yang *et al*. in L2 distance and is slightly worse in cosine distance. Another approach, Pano2CAD [33], has not made their source code available and has no evaluation on layout, making direct comparison difficult. For time consumption, Yang *et al*. report to be less than 1 minute, Pano2CAD takes 30s to process one room. One forward pass of LayoutNet takes 39ms. In CPU mode (w/o parallel for loop) using Matlab R2015a, our cuboid constraint takes 0.52s, alignment 13.73s, and layout optimization 30.5s.

Method	L2 dist	cosine dist
Yang *et al*. [32]	27.02	**4.27**
Ours	**18.51**	5.85

Table 3.4: Depth distribution error compared with Yang *et al*. [32].

3.5.2 Non-cuboid layout for panorama

Figure 3.5 shows qualitative results of our approach to reconstruct non-cuboid Manhattan layouts from single panorama. Due to the limited number of non-cuboid room layouts in the existing datasets, we captured several images using a Ricoh Theta-S 360° camera. Our approach is able to predict 3D room layouts with complex shape that are difficult for existing methods.

Figure 3.5: **Qualitative results for non-cuboid layout prediction.** We show our method's predicted layout (orange lines). Best viewed in color.

3.5.3 Perspective images

We use the same experimental setting as in [26, 29]. We train our modified approach to jointly predict room type on the training split of the LSUN layout estimation challenge. We do not train on the validation split. Our method takes 39ms (25 FPS) to process a perspective image, faster than the 52ms (19 FPS) of RoomNet basic [29] or 168ms (6 FPS) of RoomNet recurrent, under the same hardware configuration. Moreover, our method is generalized across both panorama and perspective images, and can predict both cuboid shape and non-cuboid shape room from single panorama.

Table 3.5 shows our performance compared with the state-of-the-art on Hedau's dataset [15].

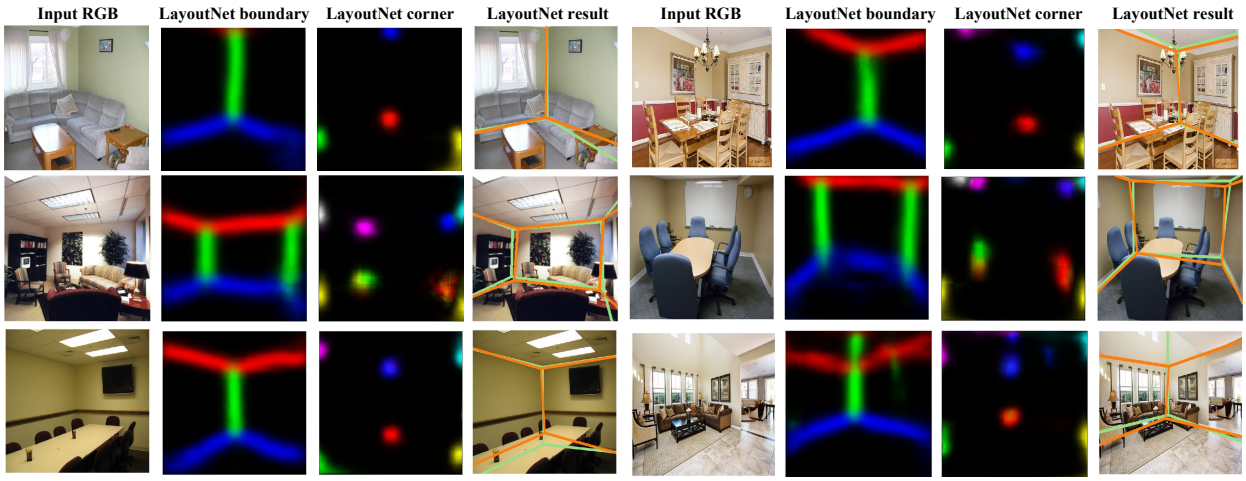


Figure 3.6: **Qualitative results for perspective images**. We show the input RGB image, our predicted boundary/corner map and the final estimated layout (orange lines) compared with ground truth (green lines). Best viewed in color.

Method	Pixel Error (%)
Schwing *et al*. [12]	12.8
Del Pero *et al*. [24]	12.7
Dasgupta *et al*. [26]	9.73
LayoutNet (ours)	9.69
RoomNet recurrent 3-iter [29]	**8.36**

Table 3.5: Performance on Hedau dataset [15]. We show the top 5 results, LayoutNet ranks second to RoomNet recurrent 3-iter in Pixel Error (%).

Our method ranks second among the methods. Table 3.6 shows our performance compared with the state-of-the-art on the LSUN dataset. Our method ranks second in Keypoint Error (%) and ranks third in Pixel Error (%). We also report results of the RoomNet basic approach [29] that does not apply recurrent refinement, which is closer in design to our approach. The lower accuracy in pixel error mainly results from our simplified room keypoint representation. Different from RoomNet [29] that assumes all keypoints are distinguished across different room types, our LayoutNet directly predicts the 8 keypoints, and selects among them based on the room type to produce the final prediction. Moreover, we do not apply layout optimization step as explained in this chapter that incorporate predicted boundary probability to the perspective image task. Figure 3.6 shows qualitative results on the LSUN validation split. Failure cases include room type prediction error (last row, right column) and heavy occlusion from limited field of view (last row, left column).

Method	Keypoint Error (%)	Pixel Error (%)
Hedau *et al*. [15]	15.48	24.23
Mallya *et al*. [28]	11.02	16.71
Dasgupta *et al*. [26]	8.20	10.63
LayoutNet (ours)	7.63	11.96
RoomNet recurrent 3-iter [29]	**6.30**	**9.86**
RoomNet basic [29]	6.95	10.46

Table 3.6: Performance on LSUN dataset. LayoutNet ranks second comparing with RoomNet recurrent 3-iter in Keypoint Error (%) and ranks third in Pixel Error (%). We also report the RoomNet basic approach that does not apply recurrent refinement step, which is closer in design to our approach.

3.6 LAYOUTNET V2

So far, we've discussed our LayoutNet approach that combines neural network based pixel-wise predictions and a post optimization step to produced Manhattan layouts. More recent approaches [124, 125, 126] for single 360 panorama 3D layout reconstruction also build upon this framework, and all show promising results.

In this section, we show our own improvements over the original LayoutNet in three folds: (1) a better image encoder to capture room corners and boundaries; (2) refined implementation details that incorporate new data augmentations like random stretching [126] and (3) a more efficient gradient ascent based optimization instead of sampling based optimization. We call the improved version "LayoutNet v2", which, at the time of writing, achieves the state-of-the-art.

3.6.1 Improvements upon LayoutNet

ResNet based network architecture. Instead of using a vanilla convolution and up-sampling based image encoder as in the original LayoutNet, we use ResNet [127] as our image encoder. The decoder remains as a two-branch architecture that jointly produces a corner map and a boundary map of the same resolution as the input RGB image. Both two branch consist 5 up-sampling layers, each with a convolution layer (kernel size 3×3) and a ReLU operation afterwards. We also add the same skip links as the original LayoutNet architecture. We exclude the 3D box parameter regressor branch as we observed no significant performance changes, but the training time was increased.

Refined implementation details. We experiment with ResNet-18, ResNet-34 and ResNet-50 image encoders respectively. The encoder outputs a bottleneck feature size of 1024. We use pre-trained ResNet on ImageNet to initialize the encoder. During training, we use the same optimization method and hyper-parameters like learning rate as the original LayoutNet. The training

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
LayoutNet	75.12	1.02	3.18
LayoutNet v2 (ResNet-18)	84.13	0.65	1.92
LayoutNet v2 (ResNet-34)	**85.02**	**0.63**	**1.79**
LayoutNet v2 (ResNet-50)	82.44	0.75	2.22

Table 3.7: Quantitative results on cuboid layout estimation from a single 360 panorama using PanoContext dataset [31]. All methods are trained with extra data from Stnfd. 2D-3D dataset. We show our LayoutNet v2 performance with three ResNet encoders respectively.

scheme differs slightly: we first train the layout boundary prediction branch, then fix the weights of boundary branch and train the corner prediction branch, and finally we train the whole network end-to-end. We slightly decrease the dilation factor to 3 for ground truth smoothing, making the ground truth corner/boundary pixel be the peak response after a Gaussian smooth of $\sigma = 20$. Despite the data augmentations used in the original LayoutNet, we change image intensity with a minimum intensity ranges between 0-127 and a fixed maximum intensity of 255. We also add color jittering to each RGB channel independently by multiplying a factor ranges in 0.8-1.2. Moreover, we perform random stretching introduced by Sun *et al*. [126] with stretching factors $k_x = 1$ and $k_z = 2$. Each augmentation parameter is uniformly and randomly sampled from the defined range. To avoid the unstable learning of the batch normalization layer in ResNet encoder due to smaller batch size, we freeze the parameters of the batch normalization layer when training end-to-end.

Gradient ascent optimization. Our original LayoutNet's post optimization step (Chapter 3.3) is sampling based, with is time consuming and is constrained to the pre-defined sampling space. We instead use stochastic gradient ascent [128] to search for local optimum of the cost function [1]. Moreover, we use the revised Equ. 3.3 that computes the average response across layout lines instead of the maximum response.

$$Score(L) = w_{junc}\frac{1}{|C|}\sum_{l_c \in C} P_{\text{corner}}(l_c) + w_{ceil}\frac{1}{|L_e|}\sum_{l_e \in L_e} P_{\text{ceil}}(l_e) + w_{floor}\frac{1}{|L_f|}\sum_{l_f \in L_f} P_{\text{floor}}(l_f) \quad (3.4)$$

We set $w_{\text{junc}} = w_{\text{ceil}} = w_{\text{floor}} = 1$.

3.6.2 Experiment

To verify the efficacy of our proposed LayoutNet v2, we (1) first conduct ablation study to verify the performance improvement of each modification to the original LayoutNet, then (2) compare

[1] We use the SGD based optimization implemented by Sun (with different loss term weights): https://github.com/sunset1995/pytorch-layoutnet

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
LayoutNet v2 (ResNet-34)	**85.02**	**0.63**	**1.79**
- random stretching	83.97	0.65	1.92
- freeze bn layer	83.98	0.7	2.01
- gradient ascent	83.60	0.73	2.12

Table 3.8: Quantitative results on cuboid layout estimation from a single 360 panorama using PanoContext dataset [31]. All methods are trained with extra data from Stnfd. 2D-3D dataset. We show our LayoutNet v2 performance with the best performed ResNet-34 encoder, comparing with different training configurations. "-random stretching" stands for training without random stretching data augmentation, "-freeze bn layer" stands for not freezing batch normalization layer when training end-to-end, "-gradient ascent" stands for using original LayoutNet's gradient

our LayoutNet v2 with other state-of-the-art.

Ablation study. Different from LayoutNet, our LayoutNet v2 uses ResNet as encoder. Since there are several ResNet variations: ResNet-18, ResNet-34, ResNet-50 and ResNet-101, we compare the performance with different ResNet encoder in Table 3.7. We train each network with a maximum batch size using a single GPU of 12 GB: 8 for ResNet-18/34 and 4 for ResNet-50 when training each branch respectively, and 4 for ResNet-18/34 and 2 for ResNet-50 when training end-to-end. In Table 3.7 we can see that with ResNet-34 LayoutNet v2 performs better than networks with ResNet-18 or 50. This demonstrates the importance of a denser encoder (ResNet-18 v.s. ResNet-34), and the importance of a larger batch size to train the batch normalization layer (ResNet-34 v.s. ResNet-50). We do not include the performance of ResNet-101, as the network size is too large to have a reasonable batch size when training on a single GPU of 12 GB, producing similar results as using ResNet-50 as encoder. Note that the performance drop for denser encoder is due to the GPU memory limitation, it's reasonable to hypothesis that given a GPU of larger memory or applying parallel computing of batch normalization layer can resolve this problem and produce better results than ResNet-34.

Another ablation study is the modifications of network training details as mentioned in Chapter 3.6.1. We show in Table 3.8 quantitative comparison when excluding each of the modification. As we can see, using gradient ascent to as post optimization contributes the most for the performance boost, while adding random stretching data augmentation contributes the less. Freezing batch normalization layout when training end-to-end can avoid unstable training of this layer when batch size is small. Including all modifications together produces the best performance and outperforms others in a large margin.

Qualitative comparison with LayoutNet. We show in Figure 3.7 and Figure 3.8 the qualitative results of our LayoutNet v2 comparing with our original LayoutNet approach on the two datasets

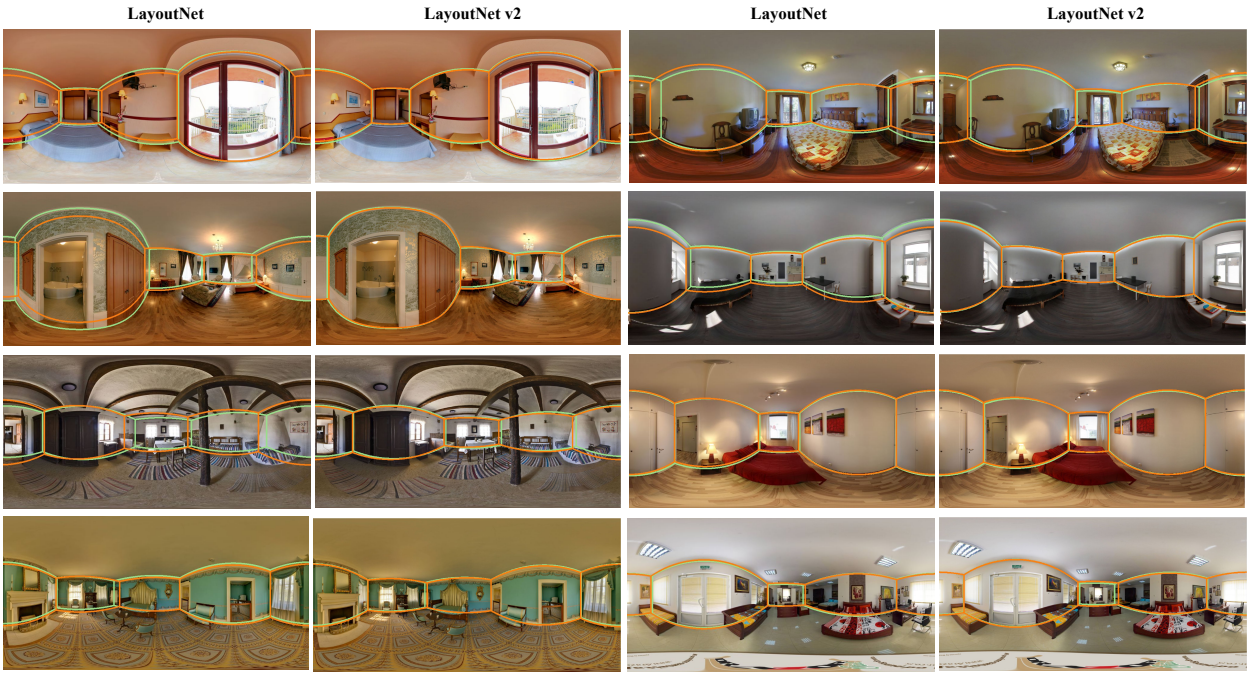


Figure 3.7: Qualitative results for cuboid layout estimation on the PanoContext dataset. We compare on the performance between LayoutNet (odd column) and LayoutNet v2 (even column). Each method's prediction is marked with orange lines and the ground truth is marked with green lines. Our LayoutNet v2 shows better performance than our original LayoutNet, especially when layout boundaries are highly occluded.

respectively. LayoutNet v2 can better estimation the layout boundaries and corners, especially when the boundaries are highly occluded.

Comparison with state-of-the-art. We compare our LayoutNet v2 with the original LayoutNet and other state-of-the-art on PanoContext dataset and Stanford 2D-3D respectively in Table 3.9 and 3.10. We report the best performed LayoutNet v2 with ResNet-34. HorizonNet [126] uses ResNet-50 as encoder with an RNN-based refinement branch. DuLa-Net [125] performs pixel-wise semantic predictions and is based on ResNet-18. For fair comparison, we report DuLa-Net's best performed network using ResNet-34 (DuLa-Net also suffers performance drop with ResNet-50). On both PanoContext and Stanford 2D-3D dataset, LayoutNet v2 outperforms the original LayoutNet. Our LayoutNet v2 achieves the state-of-the-art on PanoContext dataset, outperforming the second best HorizonNet with a large margin on all three metrics. On Stanford 2D-3D dataset, LayoutNet v2 outperforms HorizonNet on 3D IoU and just slightly falls behind DuLa-Net. We also rank the second for corner error and pixel error in a close margin comparing with HorizonNet. In general, LayoutNet v2 on cuboid layout estimation achieves the state-of-the-art.

We show in Table 3.11 the comparison of time consumption of each state-of-the-art: LayoutNet,

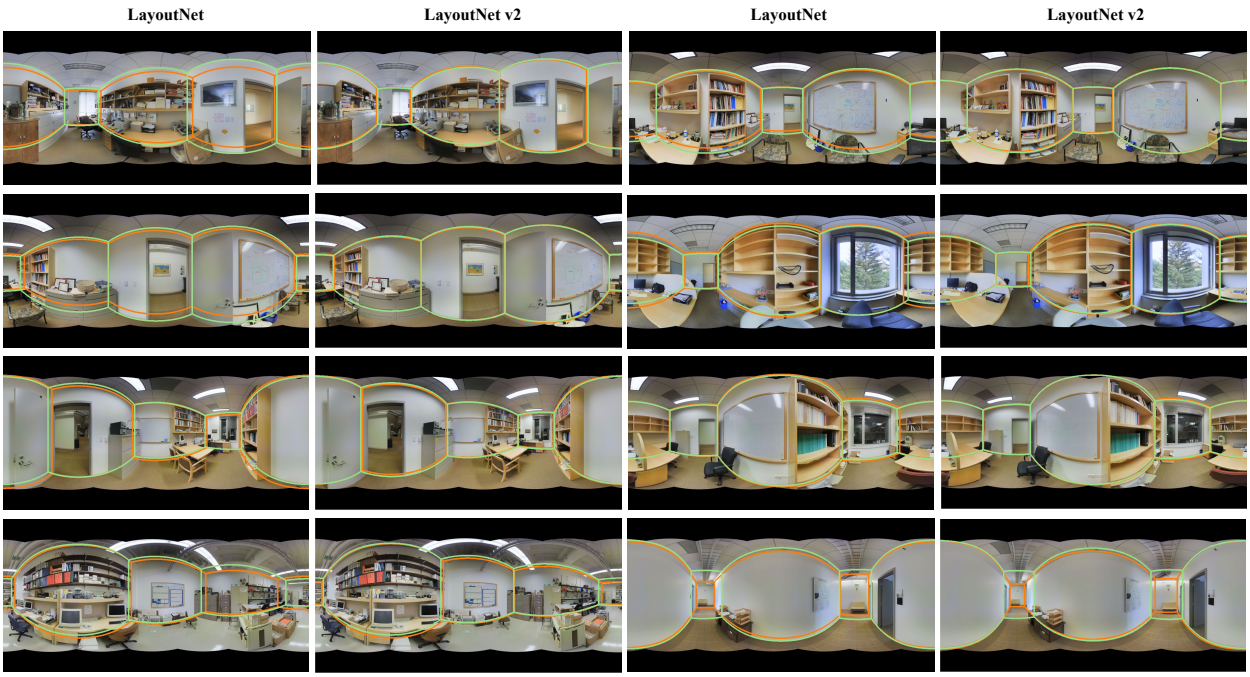


Figure 3.8: Qualitative results for cuboid layout estimation on the Stanford 2D-3D annotaiton dataset. We compare on the performance between LayoutNet (odd column) and LayoutNet v2 (even column). Each method's prediction is marked with orange lines and the ground truth is marked with green lines. Our LayoutNet v2 shows better performance than our original LayoutNet, especially when layout boundaries are highly occluded. Not that in the bottom right example, the ground truth has error annotation at the end of the corridor. Both two methods tend to predict the true end, and LayoutNet v2 is more accurate.

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
LayoutNet	75.12	1.02	3.18
DuLa-Net [125]	81.45	0.90	2.73
HorizonNet [126]	84.23	0.69	1.90
LayoutNet v2	**85.02**	**0.63**	**1.79**

Table 3.9: Quantitative results on cuboid layout estimation from a single 360 panorama using PanoContext dataset [31]. All methods are trained with extra data from Stnfd. 2D-3D dataset. We show our LayoutNet v2 performance with three ResNet encoders respectively, comparing with the original LayoutNet and the other state-of-the-art.

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
LayoutNet	77.51	0.92	2.42
DuLa-Net [125]	**84.32**	0.74	2.55
HorizonNet [126]	83.51	**0.62**	**1.97**
LayoutNet v2	84.17	0.71	2.04

Table 3.10: Quantitative results on cuboid layout estimation from a single 360 panorama using Stanford 2D-3D dataset [7]. All methods are trained with extra data from PanoContext dataset. We show our LayoutNet v2 performance with three ResNet encoders respectively, comparing with the original LayoutNet and the other state-of-the-art.

Method	Optimization average CPU time (s)	Network average GPU time (ms)
LayoutNet	44.73	39
DuLa-Net [125]	0.022	35
HorizonNet [126]	**0.012**	50
LayoutNet v2	1.222	**30**

Table 3.11: Time consumption comparison.

DuLa-Net, HorizonNet and LayoutNet v2. We report the time consumption of LayoutNet v2 with ResNet-34 encoder. We report the time consumption of HorizonNet with RNN, and note that HorizonNet without RNN only costs 8ms for network prediction, but produces less accurate result comparing with other approaches. We compare on a single forward pass of the network and the post-process optimization step. Our LayoutNet v2 is the fastest for network prediction.

3.7 CONCLUSION

In this chapter, we propose LayoutNet, an algorithm that predicts room layout from a single panorama or perspective image. Our approach relaxes the commonly assumed cuboid layout limitation and works well with non-cuboid layouts (*e.g.* , "L"-shape room). We demonstrate how pre-aligning based on vanishing points and Manhattan constraints substantially improve the quantitative results. Our method operates directly on panoramic images (rather than decomposing into perspective images) and is among the state-of-the-art for the perspective image task. Future work includes extending to handle arbitrary room layouts, incorporating object detection for better estimating room shapes, and recovering a complete 3D indoor model recovered from single images. We further show improvements with a better encoder and a gradient ascent based post optimization step (named as LayoutNet v2), achieving on par performance comparing with the state-of-the-art.

CHAPTER 4: COMPLETE 3D SCENE PARSING FROM A SINGLE RGBD IMAGE

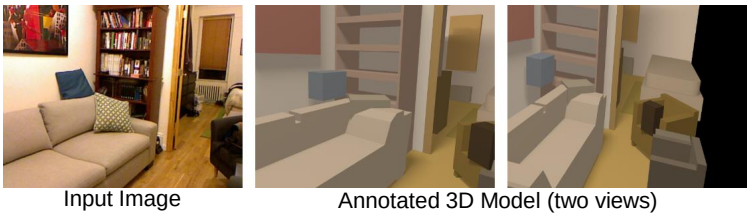


Figure 4.1: Our **goal** is to recover a 3D model (right) from a single RGBD image (left), consisting of the position, orientation, and extent of layout surfaces and objects.

We've described our approach to estimate the 3D layout from a single image (Chapter 3). Our next step is to jointly infer all 3D objects and layouts in the scene (figure 4.1). One major goal of vision is to infer physical models of objects, surfaces, and their layout from sensors. In this chapter, we aim to interpret indoor scenes from one RGBD image. Our representation encodes the layout of orthogonal walls and the extent of objects, modeled with CAD-like 3D shapes. We parse both the visible and occluded portions of the scene and all observable objects, producing a *complete* 3D parse. Such a scene interpretation is useful for robotics and visual reasoning, but difficult to produce due to the well-known challenge of segmentation, 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 to regions in training images, and transferring and aligning associated 3D models while encouraging fit to observations and spatial consistency. We use support inference to aid interpretation and propose a retrieval scheme that uses convolutional neural networks (CNNs) to classify regions and retrieve objects with similar shapes. We demonstrate the performance of our method on our newly annotated NYUd v2 dataset [129] with detailed 3D shapes.

4.1 INTRODUCTION

This chapter introduces our approach to recover complete 3D models of indoor objects and layout surfaces from an RGBD (RGB+Depth) image (Figure 6.1). Recovering 3D models from images is highly challenging due to three ambiguities: the loss of depth information when points are projected onto an image; the loss of full 3D geometry due to occlusion; and the unknown separability of objects and surfaces. In this chapter, we choose to work with RGBD images, rather than RGB images, so that we can focus on designing and inferring a useful representation, without immediately struggling with the added difficulty of interpreting geometry of visible surfaces. Even so, ambiguities due to occlusion and unknown separability of nearby objects make 3D reconstruc-

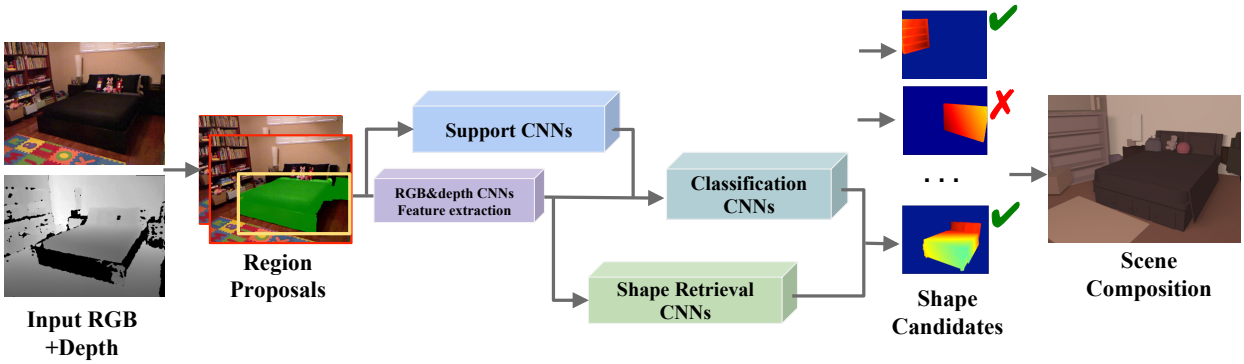


Figure 4.2: **Overview of our approach.** Given an input RGB-D (left), we propose possible layouts and object regions. We predict each object proposal's support height and class by our support and classification CNNs. We then retrieve a similar object shape, by using our shape retrieval CNNs to match the object region to the most similar region in the training set and transferring and aligning associated 3D models to the input depth image. The subset of proposed objects and layouts are then optimally selected based on consistency with observed depth, coverage, and constraints on occupied space. We show an example result (upper-right) and ground truth annotations (lower-right).

tion impossible in the abstract. If we see a book on a table, we observe only part of the book's and table's surfaces. How do we know their full shape, or even that the book is not just a bump on the table? Or how do we know that a sofa does not occlude a hole in the wall or floor? We don't. But we expect rooms to be enclosed, typically by orthogonal walls and horizontal surfaces, and we can guess the extent of the objects based on experience with similar objects. Our goal is to provide computers with this same interpretive ability.

Scene representation. We aim to infer a 3D geometric model that encodes the position and extent of layout surfaces, such as walls and floor, and objects such as tables, chairs, mugs, and televisions. In the long term, we hope to augment this geometric model with relations (*e.g.* , this table supports that mug) and attributes (*e.g.* , this is a cup that can serve as a container for small objects and can be grasped this way).

Our approach. We propose an approach to recover a 3D model of room layout and objects from an RGBD image. A major challenge is how to cope with the huge diversity of layouts and objects. Rather than restricting to a parametric model and a few detectable object classes, as in previous single-view reconstruction work, our models represent every layout surface and object with a 3D mesh that approximates the original depth image under projection. We take a data-driven approach that proposes a set of potential object regions, matches each region to a similar region in training images, and transfers and aligns the associated labeled 3D models while encouraging their agreement with observations. During the matching step, we use CNNs to retrieve

objects of similar class and shape and further incorporate support estimation to aid interpretation. We hypothesize, and confirm in experiments, that support height information will help most for interpreting occluded objects because the full extent of an occluded object can be inferred from support height. The subset of proposed 3D objects and layouts that best represent the overall scene is then selected by our optimization method based on consistency with observed depth, coverage, and constraints on occupied space. The flexibility of our models is enabled through our approach (Figure 4.2) to propose a large number of likely layout surfaces and objects and then compose a complete scene out of a subset of those proposals while accounting for occlusion, image appearance, depth, and layout consistency.

Detailed 3D labeling. Our approach requires a dataset with labeled 3D shape for region matching, shape retrieval, and evaluation. We make use of the NYUd v2 dataset [129] which consists of 1449 indoor scene RGBD images, with each image segmented and labeled with object instances and categories. Each segmented object also has a corresponding annotated 3D model, provided by Guo and Hoiem [55]. The 3D labeling provides ground truth 3D scene representation with layout surfaces as 3D planar regions, furniture as CAD exemplars, and other objects as coarser polygonal shapes. However, the polygonal shapes are too coarse to enable comparison of object shapes. Therefore, we extend the labeling by Guo and Hoiem with more detailed 3D annotations in the object scale. Annotations are labeled automatically and are adjusted manually as described in Chapter 4.2. We evaluate our method on our newly annotated groundtruth. We measure success according to accuracy of depth prediction of complete layout surfaces, voxel occupancy accuracy and semantic segmentation performance.

In contrast to multiview 3D reconstruction methods, our approach recovers complete models from a limited viewpoint, attempting to use priors and recognition to infer occluded geometry, and parses the scene into individual objects and surfaces, instead of points, voxels, or contiguous meshes. Thus, in some sense, we provide a bridge between the goals of interpretation from single-view and quantitative accuracy from multiview methods.

This chapter presents an extension of our previous work [35] that predicts full 3D scene parsing from an RGBD image. Our main new contributions are the refinement of the NYUd v2 dataset with detailed 3D shape annotations, the use of CNNs to classify regions and retrieve object models with similar shapes to a region, and use of support inference to aid region classification. We also provide more detailed discussion and conduct more extensive experiments, demonstrating qualitative and quantitative improvement.

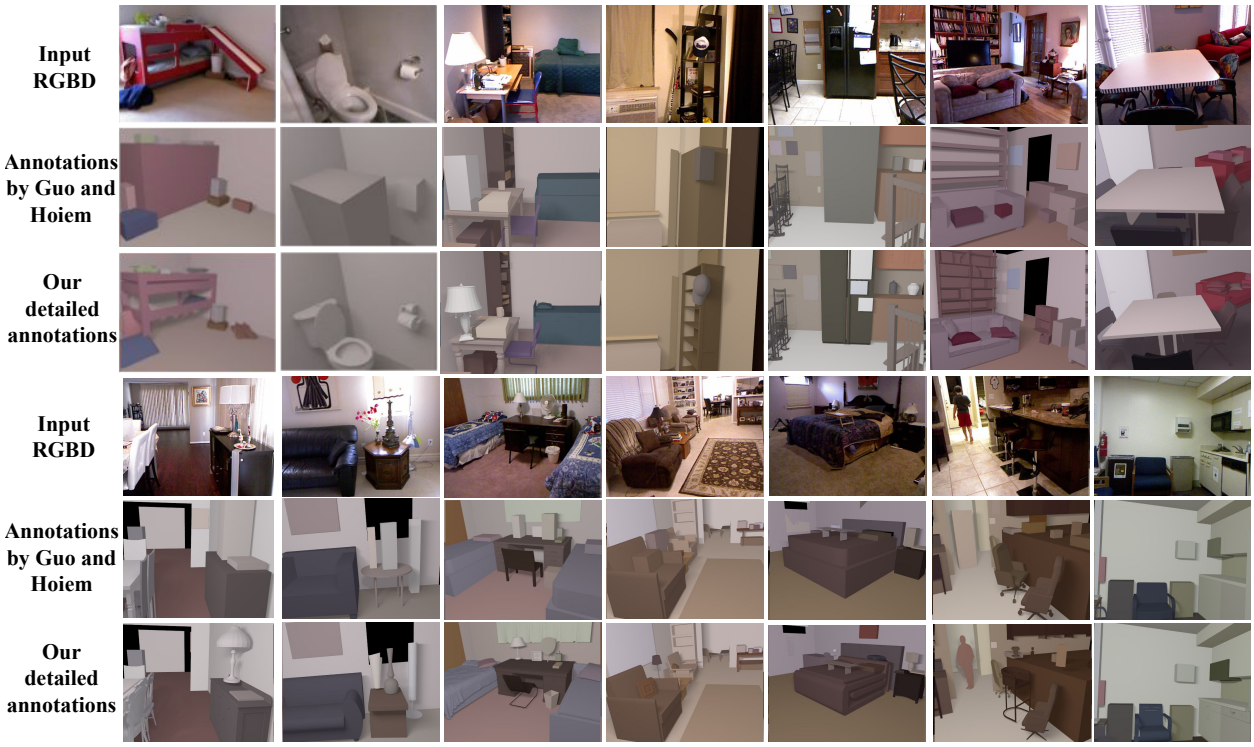


Figure 4.3: **Samples of our detail 3D annotation in NYUd v2 dataset**.We show from the left column to the right with different levels (larger to smaller) of relative depth error improvements compared with Guo and Hoiem[55]. Our annotations are much detailed in object shape scale.

4.2 DETAILED 3D ANNOTATIONS FOR INDOOR SCENES

We conduct our experiments on the NYUdv2 dataset [129], which provides complete 3D labeling of both objects and layouts of 1449 RGB-D indoor images. Each object and layout has a 2D segment labeling and a corresponding annotated 3D model, provided by Guo and Hoiem [55]. The 3D annotations use 30 models to represent 6 categories of furniture that are most common and use extruded polygons to label all other objects. These models provide a good approximation of object extent but are often poor representations of object shape, as shown in Fig 4.3. Therefore, we extend the NYUd v2 dataset by replacing the extruded polygons with CAD models collected from ShapeNet [130] and ModelNet [67]. To align name-space between datasets, we manually map all model class labels to the 633-class 3D object labels in NYUd v2 dataset. The shape retrieval and alignment process is performed automatically and then adjusted manually, as follows.

Coarse alignment. For each ground truth 2D region r_i in the NYUd v2 dataset, we retrieve model set $M = \{M_i\}$ from our collected models that have the same class label as r_i. We also include the region's original coarse 3D annotation by Guo and Hoiem [55] in the model set M, so that we can preserve the original labeling if no provided CAD models are better fit in depth. We

initialize each M_i's 3D location as the world coordinate center of the 3D annotation labeled by Guo and Hoiem. We resize M_i to have the same height as the 3D annotation.

Fine alignment. Next, we align each retrieved 3D object model M_i to fit the available depth map of the corresponding 2D region r_i in the target scene. The initial alignment is often not in the correct scale and orientation; *e.g.* , a region of a left-facing chair often resembles a right-facing chair and needs to be rotated. We found that using Iterative Closest Point to solve for all parameters did not yield good results. Instead, we enumerate 16 equally-spaced orientations from -180 to 180 from top-down view and allows 2 minor scale revision ratio as $\{1.0, 0.9\}$. We perform ICP to solve for translation initialized using scale and rotation, and pick the best ICP result based on the following cost function:

$$\begin{aligned}\text{FittingCost}(M_i, T_i) =& C_{depth} \sum_{j \in r_i \cap s(M_i,T_i)} |\mathcal{I}_d(j) - \hat{d}(j; M_i, T_i)| + \sum_{j \in r_i \cap \neg s(M_i,T_i)} C_{missing} \\ &+ C_{occ} \sum_{j \in \neg r_i \cap s(M_i,T_i)} \max(\hat{d}(j; M_i, T_i) - \mathcal{I}_d(j), 0) \end{aligned} \tag{4.1}$$

where T_i represents scale, rotation, and translation, $s(.)$ is the mask of the rendered aligned object, $\mathcal{I}_d(j)$ denotes the observed depth at pixel j and $\hat{d}(j)$ means the rendered depth at j. The first term encourages depth similarity to the ground truth RGBD region. The second penalizes pixels in the proposed region that are not rendered. We loosely allow the rendered object depth in the scene to be further away than the sensor depth. This is because depth map only gives the depth of the closest object, not all objects, due to possible occlusion. We do not want to penalize predicting a larger depth than is observed in the scene. The third term penalizes pixels in the rendered model that are closer than the observed depth image (so the model does not stick out into space known to be empty).

Based on the fitting cost of Eq. 4.1, our algorithm picks the model M_i with the best translation, orientation, and scale T_i. The fitting scales T_i along different dimensions are the same. This helps simplify the search procedure for the alignment. We found that the large variety of CAD models in the ShapeNet dataset already provide enough shape candidates to select from, in regardless of the scales along the three dimensions of each shape.

We set the term weights C_{depth}, $C_{missing}$, C_{occ} as $1.0, 0.9, 0.5$ using grid search in the validation set. The search criteria is to find the set of term weights that minimize both the rendered depth difference to the ground truth depth and the rendered 2D segmentation difference to the ground truth 2D region annotation. For original 3D polygonal labeling, we fix M_i and use equation 1 to search for the best fitting scale and translation that minimize the cost. For efficiency, we first obtain

the top 5 models based on the fitting cost, each maximized only over the 16 initial orientations before ICP. For each of these models, we then solve for the best translation T_i for each scale and rotation based on Eq. 4.1 and finally select the aligned model with the lowest fitting cost.

Post-processing. Automatic fitting may fail due to high occlusion or missing depth values. We manually conduct a post-processing check and refine bad-fitting models. Using a GUI, an annotator checks the automatically produced shape for each region. If the result is not satisfactory, the annotator compares to other top model fits, and if none of those are good matches, then the fitting optimization based on Eq. 4.1 is applied to the original polygonal 3D labeling. This helps to ensure that our detailed shape annotations are a strict improvement over the original course annotations. In total, we fit 3792 different CAD models to the object regions and through manual checking observe failures in only 10% of cases for automatic fitting. These failures are manually corrected.

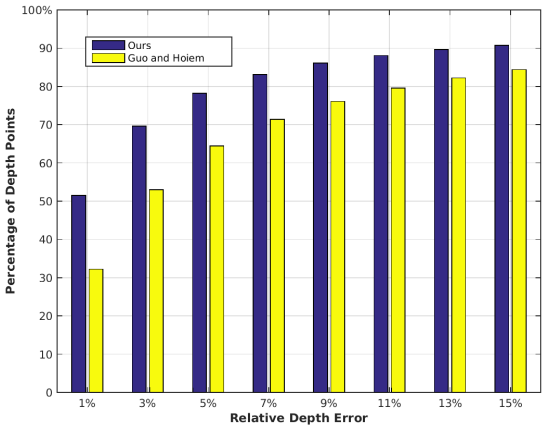


Figure 4.4: Cumulative relative depth error of our detailed 3D annotations and the 3D annotations by Guo and Hoiem [55] in NYUd v2 dataset.

Validation. Figure 4.4 reports the cumulative relative error of the rendered depth of our detailed 3D annotations compared with ground truth depth in NYUd v2 dataset. The relative error r_D is computed as:

$$r_D = \frac{1}{|S_I|} \sum_{I \in S_I} \sum_{p \in I} \frac{|d_p - \hat{d}_p|}{d_p} \tag{4.2}$$

where $S_I = \{I_1, I_2, \ldots, I_N\}$ is all the RGBD images in the dataset; p represents a pixel in each image I; d_p is the ground truth depth of pixel p from sensor; and $\hat{d}_p$ is the rendered depth of the 3D label annotation at pixel p. For comparison, we report the r_D of the 3D annotations by Guo and Hoiem [55]. Our annotations have more points with low relative depth error, and achieve a better modeling of depth for each image.

4.3 APPROACH

Given an RGBD image as input, we aim to find a set of layout and object models that fit RGB and depth observations and provide a likely explanation for the unobserved portion of the scene. Our resulting 3D parse is consistent with the observation and reasonable in 3D. We formulate this as:

$$\{\mathbf{M}, \theta\} = \arg\min(\text{AppearanceCost}(\mathcal{I}_{\text{RGBD}}, \mathbf{M}, \theta) + \text{DepthCost}(\mathcal{I}_{\text{D}}, \mathbf{M}, \theta) + \text{ModelCost}(\mathbf{M}, \theta)) \tag{4.3}$$

$\mathcal{I}_{RGBD}$ is the RGB-D image; $\mathcal{I}_D$ is the depth image alone; $\mathbf{M}$ is a set of candidate 3D layout surfaces and object models; and θ is the set of parameters for each surface/object model, including translation, rotation, scaling, and whether each candidate model is included. AppearanceCost encourages that object models should match underlying region appearance, rendered objects should cover pixels that look like objects (versus layout surfaces), and different objects should have evidence from different pixels. DepthCost encourages similarity between the rendered scene and observed depth image. ModelCost penalizes intersection of 3D object models.

We propose to tackle this complex optimization problem in stages (Figure 4.2): (1) propose candidate layout surfaces and objects; (2) retrieve 3D models of the proposed candidate and improve the transformation of each surface/object model to the depth image; (3) choose a subset of models that best explains the scene. Layout elements (wall, floor, and ceiling surfaces) are proposed by scanning for planes that match observed depth points and pixel labels and then finding boundaries and holes. Parsing 3D objects is particularly difficult. We propose an exemplar-based approach, matching regions in the input RGBD image to regions in the training set, and transferring and aligning corresponding 3D models (Chapter 4.3.2). We then choose a subset of objects and layout surfaces that minimizes the depth, appearance, and model costs using a specialized search (Chapter 4.3.3). Despite the challenges of matching an optimization, we experimentally find that our approach produces results from automatic regions that are nearly as good as those produced from ground truth regions, even though using ground truth regions greatly simplifies the matching and selection process.

4.3.1 RGBD image pre-processing and shape proposals

We conduct the same pre-porcessing step and the layout proposal step as the one used in Guo *et al*. [35]. To produce object candidate regions, we use the method for RGBD images by Gupta *et al*. [131] and extract top ranked 2000 region proposals for each image. Our experiments show that

this region retrieval is more effective than the method based on Prims algorithm [109] used in our previous work [35]. Likely object categories and 3D shapes are then assigned to each candidate region.

4.3.2 CNN-based shape retrieval

We train and use CNN networks to predict the object category and support height of each region, as shown in Figure 4.5 and Figure 4.6. The support height is used as a feature for the object classification. We also train and use a CNN with a Siamese network design to find the most similar 3D shape of a training object, based on region and depth features.

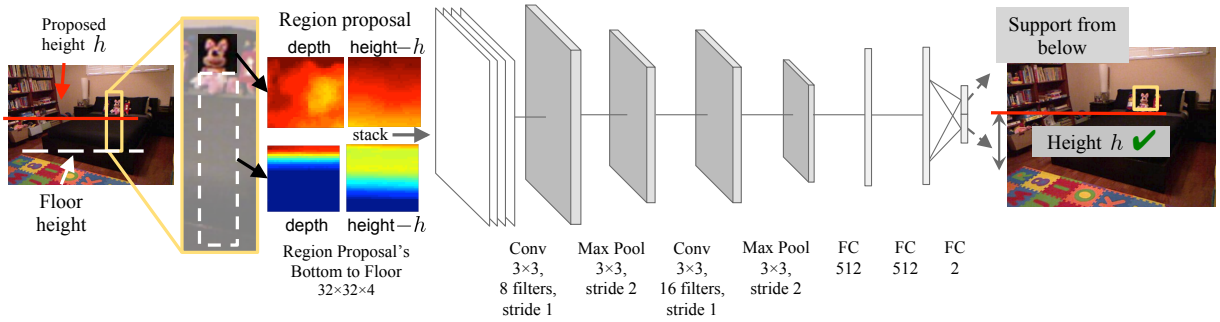


Figure 4.5: The CNN for predicting a candidate object's support height. We perform ReLU between the convolutional layer and the max pooling layer. Local response normalization is performed before the first fully connected (FC) layer. We add dropout with 0.5 before each FC layer during training.

Support height prediction. We predict support height for each object with the aim of better predicting class and position. As shown in Figure 4.5, our support height prediction network jointly predicts the candidate support height probability and the support type: whether the object is supported from below or behind. We first find candidate support heights using the method of Guo and Hoiem [55] and use the network to estimate the probability of each candidate being the correct height. The support type prediction and the candidate support height probability prediction share the same convolution structure and the first two fully connected layers. Our network input consists two features: crops of 1) the depth maps and height maps of the region proposal and 2) the depth maps and height maps of the region that extends from the bottom of the region proposal to the estimated floor position. The former feature helps infer support type through its shape and the latter helps infer the support height through the vertical change in texture and depth. We found that having those two inputs together performs better for both suport height and support type predictions. To create the input feature vector, we subtract the candidate support height from the

height cropped images, re-size all four crops to 32×32 patches, and concatenate channel-wise.

In the test set, we identify the closest candidate support height with 92% accuracy, with an average distance error of 0.18m. As a feature for classification, we use the support height relative to the camera height, which leads to slightly better performance than using support height relative to the estimated ground. This is because dataset images are taken from consistent heights but estimated ground height may be mistaken.

Categorization. Our classification network gets input of the region proposal's support height and type, along with CNN features from both RGB and depth. The network predicts the probability for each class as shown in Figure 4.6. To model the various classes of shapes in indoor scene, we classify the regions into the 78 most common classes, which have at least 10 training samples. Less common objects are classified as "other prop", "other furniture" and "other structure" based on rules by Silberman *et al*. [129]. In addition, we identify a region proposal that is not representative for an object shape (*e.g.* , a piece of chair leg region when the whole chair region is visible) as a "bad region" class. This leads to our $78 + 3 + 1 = 82$-class classifier. The input support height and type are directly predicted by our support height prediction network. To create our classification features, we copy the two predicted support values 100 times each (a useful trick to reduce sensitivity to local optima for important low-dimensional features) and concatenate them to the region proposal's RGB and HHA features from Gupta *et al*. [131] in both the 2D bounding box and masked region as in [132]. Experiments show that using the predicted support type and the support height improves the classification accuracy by about 1%, with larger improvement for occluded objects.

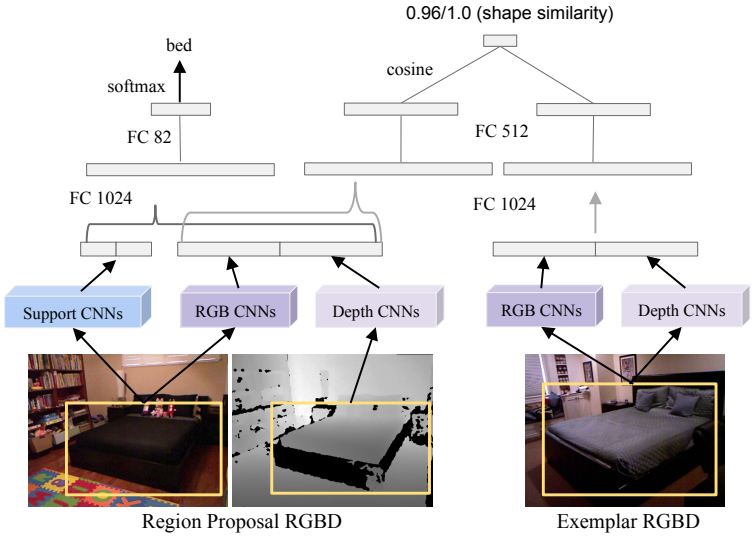


Figure 4.6: The CNNs for region classification (left) and similar shape retrieval (right). We perform ReLU and dropout with 0.5 after the first FC layer for both of the networks during training.

Shape candidate retrieval. Using a Siamese network (Figure 4.6), we learn a region-to-region similarity measure that predicts 3D shape similarity of the corresponding objects. The network embeds the RGB and HHA features used in our classification network into a space where cosine

distance correlates to shape similarity, as in [133]. In training, we use surface-to-surface distance [66], the normalized point cloud distance between the densely sampled points on each of the 3D meshes, as the ground truth similarity. The surface-to-surface distance is calculated from two centered complete 3D shapes complete 3D meshes, and thus no self-occlusion shall be considered during ground-truth computation. We train the network to penalize errors in shape similarity orderings. Each region pair's shape similarity score is compared with the next pair's among the randomly sampled batch in the current epoch and penalized only if the ordering disagrees with the ground truth similarity. We attempted sharing embedding weights with the classification network but observed a 1% drop in classification performance. We also found predicted class probability to be unhelpful for predicting shape similarity.

Candidate region selection. We apply the above retrieval scheme to each of the 2000 region proposals in each image, obtaining shape similarity rank compared with all the training samples and 81-object class and non-object class probability for each region proposal. In order to reduce the number of retrieved candidates before the scene interpretation in Chapter 4.3.3, we first reduce the number of region proposals using non-maximal suppression based on non-object class probability and threshold on the non-object class probability. We set the threshold to obtain 190 region proposals for each image, on average. We select the two most probable classes for each remaining region proposal and select five most similar shapes for each class, leading to ten shape candidates for each region proposal.

Then, we further refine these ten retrieved shapes. We align each shape candidate to the target scene by translating the model using the offset between the depth point mass centers of the region proposal and the retrieved region. We then perform ICP and use a similar method to that described in Section. 4.2 to speed up the process. We use a grid of initial values for rotation and scale and pick the best one for each shape based on the fitting energy in Eq. 4.1. We set the term weight: $C_{missing}$ as 0.6, C_{depth} as 1.0, C_{occ} as 0.9 based on a grid search on the validation set. The overall selection costs on average 3.7 seconds for each object. We tried using the estimated support height for each region proposal for aligning the related 3D shape models but observed a worse performance in the scene composition result. This is because a relatively small error in object's support height estimation can cause a larger error in fitting.

Finally, we select the two most promising shape candidates based on the following energy function,

$$E_l(m_i) = w_f E_{fitting}(m_i, t_i) + w_c \log P(c_i|r_i) + w_b \log P(b_i|r_i) \tag{4.4}$$

$E_{fitting}$ is the fitting energy defined in Eq. 4.1 that we used for alignment. $P(c_i|r_i)$ and $P(b_i|r_i)$ are the softmax class probability and the non-object class probability output by our classification

network for the regon proposal r_i. We normalize $P(c_i|r_i)$ to sum to 1, in order not to penalize the non-object class twice in the energy function. We set the term weights $w_f = 1.0$, $w_c = -1500$, $w_b = 1300$ using a grid search. Note that $E_{fitting}$ is on the scale of the number of pixels in the region.

Training details. We first find meta-parameters on the validation set after training classifiers on the training set. Then, we retrain on both training and validation in order to report results on the test set. We train our networks with the region proposals that have the highest (and at least 0.5) 2D intersection-over-union (IoU) with each ground truth region in the train set. We train the support height prediction network with the ground truth support type for the region proposal and set the ground truth support height as the closest support height candidate that is within 0.15 meters from the related 3D annotation's bottom. For training regions that are supported from behind, we do not penalize the support height estimation, since our support height candidates are for vertical support. For training the classification network, we also include the non-object class region proposals that have < 0.3 IoU with the ground truth regions. We randomly sample the same number of the non-object regions as the total number of the object regions during training. To avoid unbalanced weights for different classes, we sample from the dataset the same number of training regions for each class in each epoch. When training the shape similarity network, we use the ground truth detailed CAD annotation obtained from Chapter 4.2. When conducting the shape candidate retrieval, the pool of 3D CAD models is the detailed annotation from the training split of the NYUd v2 dataset. We translate each 3D model to origin and re-size to $200 \times 200 \times 200$-voxel cuboid before computing the surface-to-surface distance. We use ADAM [134] to train each network with the hyper-parameter of $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate for each network is: support height prediction 0.0008, classification 0.003 and Siamese network 0.0001.

4.3.3 Scene composition

Finally, given a set of candidate objects from Chapter 4.3.2 and layout plane candidates by Guo *et al*. [35], we need to choose a subset that closely reproduces the original depth image when rendered, adhere to pixel predictions of object occupancy, and correspond to minimally overlapping 2D regions and 3D aligned models. Instead of fitting each model to the depth map separately, we integrate the object and layout proposals together, optimize on proposals and compose a complete scene. The models $M = \{M_i\}$ and their alignment $T = \{T_i\}$ are fixed. The remaining parameters are $\mathbf{y}$ with $y_i \in \{0, 1\}$ indicating whether each layout or object model is part of the

scene. We choose $\mathbf{y}$ to minimize Equation 4.5:

$$\begin{aligned}\text{selectionCost}(\mathbf{y}) = &\sum_j \text{clip}(|\log_2 \frac{\hat{d}(j;\mathbf{M},\mathbf{T},\mathbf{y})}{\mathcal{I}_d(j)}| - \log_2(1.03), [0\ \ 1]) \\ &+ \sum_j |\text{isObject}(j;\mathbf{M},\mathbf{T},\mathbf{y}) - \text{P}_{\text{object}}(j;\mathcal{I}_{RGBD})| \\ &+ \sum_j \max(\sum_i y_i r_i(j) - 1, 0) + \sum_{i,k>i} y_i y_k \text{overlap3d}(M_i, T_i, M_k, T_k) \end{aligned} \quad (4.5)$$

Here j represents the rendered pixel for each candidate shape. $\hat{d}$ renders object models selected by $\mathbf{y}$. The first term minimizes error between rendered model and observed depth. We use log space so that the error matters more for close object. We subtract $\log_2(1.03)$ and clip at 0 to 1 because errors less than 3% of depth are within noise range, and we want to improve only reduction of depth if the predicted and observed depth is within a factor of 2. The second term encourages rendered object and layout pixels to match the probability of object map (P_{object}) of each image based on [129]. $r_i(j)$ is whether pixel j belongs to the model M_i rendered in 2D. The function $\text{isObject(j)} = 1$ identifies that a pixel corresponds to an object, rather than a layout model. The third term penalizes choosing object models that correspond to overlapping region proposals (each selected object should have evidence from different pixels). The second and third term, combined with region retrieval, account for the appearance cost between the scene model and observations. The fourth term penalizes 3D overlap of pairs of aligned models, to encourage scene consistency. To improve efficiency of this terms' computation, we approximate the object occupancy by rendering the closest and furthest points of the object at each pixel. We assume that all pixels between are occupied, and penalize using 3D overlap – the length of the range of overlapping occupied depth, summed over each pixel.

The depth rendering and consistency terms of Equation 4.5 involve high-order dependencies, leading to a hard binary problem. We initialize $\mathbf{y} = \mathbf{0}$ and estimate the solution in three steps. First, we perform greedy search to minimize depth error (minimize selectionCost while weighting first term by a factor of 10) by iteratively adding the next candidate that maximizes selectionCost gain. Then, this solution is used to initialize a hill-climbing search on selectionCost (experiments indicated insignificant benefit to weighting terms). The hill-climbing search iteratively adds ($y_i = 1$) or removes ($y_i = 0$), where i indicates the candidate model that results in the greatest cost reduction, until no change yields further improvement. Finally, for all layout proposals and a subset of object proposals that are not yet selected, our algorithm tries adding the proposed model and removing all models whose renderings overlap and keeps the change if the cost is reduced. In experiments, we found this search procedure to outperform a variety of other attempted methods

including general integer programming algorithms and relaxations.

4.4 EXPERIMENT

The Overall performance is evaluated on the final complete 3D scene prediction for objects and layouts using the ground truth detailed 3D annotation (in Chapter 4.4.2). We compare our current approach to our previous approach (Guo *et al*. [35]). For some measures, we also compute the performance of the detailed 3D ground truth annotations as an upper bound. We investigate the effectiveness of the design choices in each of the main steps: region proposal, classification and shape retrieval, and scene composition. We also investigate the effects of incorporating support inference. To avoid overfitting to the NYUv2 dataset, we train our approach on the standard training split, tune all the parameters on the validation split and report results on the test split.

Improving on Guo *et al*. [35]. Our current approach improves on our previous work (**"Guo *et al*."**) in two aspects. First, we use region proposals from Gupta *et al*. [131] ("MCG") instead of the region proposals generated by randomized Prims algorithm [109]. In Guo *et al*.'s method, region proposals are generated by first starting with an oversegmentation and boundary strengths by Silberman *et al*. [129]. A neighborhood graph is then created, with superpixels as nodes and boundary strength as the weight connecting adjacent superpixels. Finally, the randomized Prims algorithm is applied on the graph to obtain a set of region proposals. The seed region of the Prims algorithm is sampled according to the objectness of the segment, so that the segments that are confident layout are never sampled as seeds. Size constraints and merging threshold are used to produce a more diverse set of segmentations. Regions that are near-duplicates of other regions are suppressed to make the set of proposals more compact. For each image, 100 candidate region proposals are generated. Second, our method considers both object category and shape similarity for retrieval, while Guo *et al*.use only object category as a proxy for object similarity. Guo *et al*.apply canonical-correlation analysis (CCA) to find embeddings of visual features that improve retrieval [135]. CCA finds pairs of linear projections of the two views $\alpha^T X$ and $\beta^T Z$ that are maximally correlated. Here $X = \{x_i\}$ are the feature vectors of the regions. In experiments, Guo *et al*.use the combination of 3D features and RGBD-SIFT, which were found to provide better performance than pre-trained CNN features. $Z = \{z_i\}$ are the label matrix, which contains the one-hot indicator vectors of the object category labels of the corresponding region. The similarity weight matrix is then computed as $W = \alpha\alpha^T$, which is used to parametrize the distance metric:

$$\text{dist}_W(x_i, x_j) = \sqrt{(x_i - x_j)^T W (x_i - x_j)} \tag{4.6}$$

In the experiments for Guo *et al*., the top 3 object models nearest to each region proposal according

to dist_W are retrieved.

Detailed ground truth labeling as an upper bound. We include the evaluation of our detailed ground truth annotation for the pixel-wise measures in Chapter 4.2, denoted by **"3D ground truth"**. Note that even the "3D ground truth" sometimes does not achieve high accuracy, because rendered models may not follow image boundaries and some small objects are not modeled in annotations.

Comparison with Deep Sliding Shapes [46] ("DSS"). We compare our approach with DSS, the state-of-the-art amodal 3D object detector in RGBD images. Given single RGBD image, DSS predicts multiple (around 300 after NMS) overlapping tight 3D bounding boxes and a confidence score for each object class. Each object class is predicted independently, therefore there is no constraint on 3D overlap between them. The evaluation criteria is the mean average precision (mAP) for each class. Different from DSS, our approach aims at producing an exact reconstruction of scene composed by all coherent objects and layouts with detailed CAD representation, while penalizing on object 3D overlap to ensure consistency. We evaluate on final scene composition performance to match observed RGB and depth. Though our approach have different goal, metrics and optimization strategy compared with DSS, we can still convert the results to each other's format and compare.

Comparison with Semantic Scene Completion [47] ("SSCNet"). We compare our approach with the scene-level approach SSCNet, which is the state-of-the-art semantic scene completion method from single depth image. SSCNet predicts occupancy in a voxel space of $60 \times 36 \times 60$ within the view frustum of the scene. Each predicted voxel is assigned to either one of the 3 layout labels (wall, ceiling, floor) or one of the 8 common furniture classes (*e.g.* , bed, chair, table). Both SSCNet and our approach predict semantic labeling of the observed scene. Different from SSCNet, our framework utilizes RGB information in addition to observed depth. Our approach finds a mesh-based representation for any object and layout in the scene, compared with SSCNet which produces a limited number of classes under the voxel-based representation and is not flexible in rendering the scene with a higher resolution. For comparison, we align and voxelize our predicted 3D scene model to the same configurations as SSCNet and evaluate on their metric.

4.4.1 Performance measures

We evaluate our 3D scene completion approach on our newly annotated NYUd v2 dataset. We use both 2D and 3D quantitative measures to evaluate different aspects of our solutions: (1) instance segmentation and semantic segmentation performance induced by rendering; (2) label/depth accuracy of predicted layout surfaces; (3) voxel accuracy of occupied space and free space. Among these, the depth accuracy of predicted layout surfaces and voxel accuracy of object occupancy are

the most direct evaluations of our 3D model accuracy. The instance segmentation and semantic segmentation indicates accuracy of object localization and classification. Where possible, we compare on these measures to prior work and sensible baselines.

For the comparison with DSS, we directly use DSS's 3D bounding box prediction on the 654 test images of the NYUd v2 dataset. We use our detailed annotated ground truth to compare on our metrics. Since DSS has no layout prediction, we only compare on the 19 classes that DSS concerns. We use the following three metrics for parsing objects: 3D voxel object occupancy, 2D semantic segmentation and instance segmentation. Note that DSS produces multiple overlapping predictions with confidence. We first filter out detections with score lower than 0.5 in each image. We then apply NMS with an IoU threshold of 0 for each class separately to obtain non-overlapping top ranked 3D boxes. DSS has no 2D segmentation mask for each detection, we thus project the 3D box on the 2D image to get the predicted object segmentation and depth. We ignore the layouts and other object classes in both our 3D/2D predictions and the ground truth. On the other hand, to compare on DSS's amodal 3D object detection metric [45], we extract a tight 3D bounding box around each of our predicted detailed shape and set the confidence of each box to 1. We also compare with the result of Sliding Shapes by Song *et al*. [45]. Sliding Shapes predicts detailed shapes, while the DSS approach only predicts 3D bounding boxes. Same as [45] and [46], we evaluate on the ground truth of five main furniture on the test set which is the intersection of NYUd v2 and the Sliding Shapes test set.

For the comparison with SSCNet, we evaluate on SSCNet's metric given the input of the kinect depth map and RGB observations on the test split of the NYUd v2 dataset. Since our predicted complete scene model is aligned with the camera coordinate, we first align our predictions to the world coordinate given groundtruth floor plane. We then voxelize the scene into $60 \times 36 \times 60$ voxel grids to obtain the same resolution as in SSCNet and then compare. We map our 84-class semantic labeling to the 11-class used in SSCNet. We then evaluate on two metrics [47]: (1) scene completion with precision, recall and voxel-level IoU; (2) semantic scene completion with voxel-level IoU. Though our method is not optimized for voxel-based evaluation, we can still compare and evaluate to validate our approach on the scene-level 3D parsing performance.

4.4.2 Evaluation of scene composition

Qualitative results. We show representative examples of predicted 3D scenes in Figures 4.7 and 4.9 of our approach, 3D ground truth, and Guo *et al*.'s method. Compared with the ground truth annotation, our method produces a similar layout parsing and a reasonable prediction and localization of main furniture in the scene. Compared with our previous approach of Guo *et al*., our method performs better, which results from the better region classification and shape estimation

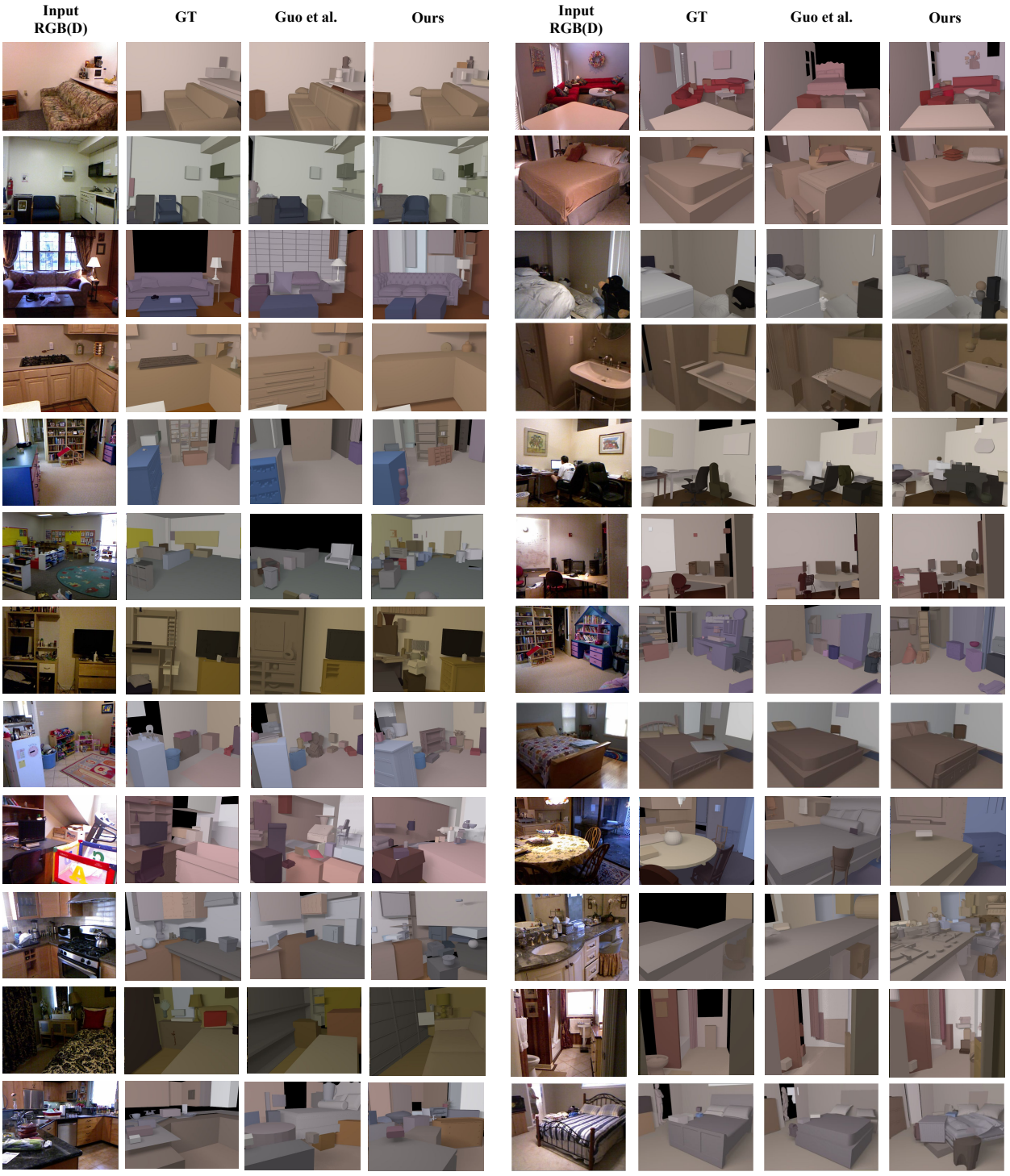


Figure 4.7: Qualitative results on scene composition with automatic region proposals. We randomly sample images from the top 25% (first four rows), medium 50% (row 6-9) and worst 25% (last four rows) based on 84-class semantic segmentation accuracy.

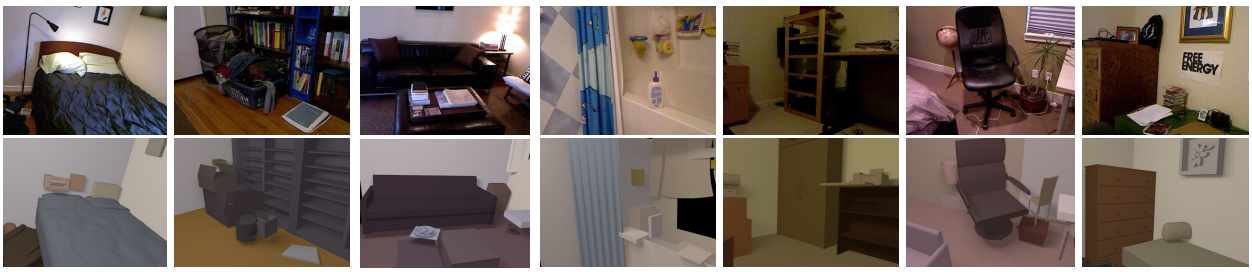


Figure 4.8: Qualitative results on SUN-RGBD dataset. Our method is able to estimate accurate 3D layout and the 3D occupancy of main furniture.

Method	Mean coverage weighted	Mean coverage unweighted
[35]	48.12	29.75
MCG+CCA	50.43	32.91
Ours w/o support	**50.43**	**34.34**
Ours	50.23	33.92
Ours w/ GT support	50.52	34.32
Ours w/ GT-region	52.39	35.61
Detailed ground truth labeling	65.42	49.15

Table 4.1: Results of 84-class instance segmentation on both automatic region proposals and ground truth regions.

Method	Objects+Layouts		
	avg class	avg instance	avg pixel
[35]	6.96	30.39	43.85
MCG + CCA	8.04	32.56	46.17
Ours w/o support	10.51	32.45	45.94
Ours	**10.77**	**32.90**	**46.25**
Ours w/ GT support	11.28	33.71	47.46
Ours w/ GT-region	16.63	37.61	54.86
Detailed ground truth labeling	67.85	77.53	85.78

Table 4.2: Results of 84-class semantic segmentation on both automatic region proposals and ground truth regions.

capability that will be analyzed in the ablation study Chapter 4.4.5.

To demonstrate the generalization capability to other dataset, we show qualitative results on the SUN-RGBD dataset [136] in Figure 4.8. Since SUN-RGBD dataset does not have detailed shape annotations to train on, we directly use our model trained on NYUd v2 dataset. Our method is able to obtain the accurate 3D parse of layouts and the main furniture like beds, shelves and desks. Errors mainly come from region proposal classification failure and missing small objects from the region proposal step, due to the domain difference between the two datasets.

84-class instance segmentation. We can infer region labels by projecting the 3D scene models to the 2D images. Though instance segmentation neglects the evaluation of 3D localization of shapes and layout, which is the main purpose of our method, it can be a reasonable evaluation of object localization and classification. We evaluate the 84-class instance segmentation (81 object classes and 3 layout classes: wall, ceiling, floor) on the inferred region labels of our scene composition result. The evaluation follows the protocol in RMRC [137] that computes the coverage of ground truth regions, weighted or unweighted by area, for each image. The average coverage across all images is reported in Table 4.1. We see improvements, compared with Guo *et al.*, due to both classification method (CNN vs. CCA) and region proposal method (MCG vs. Prim's).

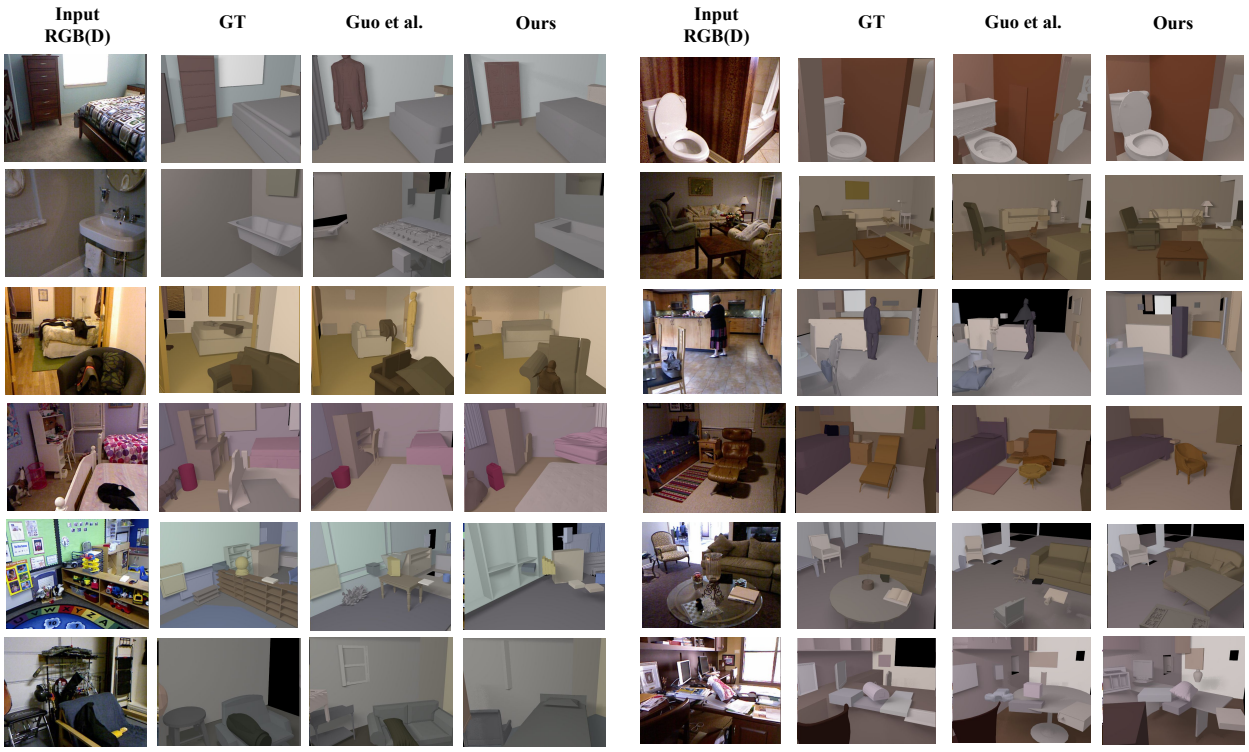


Figure 4.9: Qualitative results on scene composition given ground truth 2D labeling as region proposals. We randomly sample images from the top 25% (first two rows), medium 50% (row 3-4) and worst 25% (last two rows) based on 84-class semantic segmentation accuracy.

Also, it is interesting to see that using ground truth regions does not dramatically improve results, indicating that the region proposal and scene composition methods are highly effective.

84-class semantic segmentation. Semantic segmentation is evaluated by the percent of pixels that are correctly classified, either averaged across pixels ("avg pixel"), instances ("avg instance"), or classes ("avg class"), as reported in Table 4.2. The average across pixels tends to be highest, since the most common and largest objects are more likely to be correctly labeled. Again, our approach outperforms Guo *et al.*due to improved region proposals and classification. Support prediction also helps here. We also compute the 40-class semantic segmentation to compared with the state-of-the-art [138]: 18.3 average class / 33.8 average instance / 47.7 average pixel vs. their 34.0/49.5/65.4. Since our method projects 3D models of classified regions onto the images, it may not adhere to image boundaries as well as a direct semantic segmentation method and also may have more difficulty with small objects.

Layout estimation. In Table 4.3, we evaluate accuracy of labeling background surfaces into "left wall", "right wall", "front wall", "ceiling", and "floor". Ground truth is obtained by rendering the 3D annotation of layout surfaces, and our prediction is obtained by rendering our predicted

Method	Layout Pixel Error		
	overall	occluded	visible
NYU Parser [129]	34.6	50.0	5.2
[35]	10.9	14.0	**4.8**
Ours	**10.6**	**13.6**	**4.8**

Table 4.3: Pixel labeling error for layout surfaces with 5 categories (full dataset, 654 image).

Method	Layout Pixel Error		
	overall	occluded	visible
[139]	10.0	13.0	5.9
[35]	5.4	7.6	2.3
Ours	**5.1**	**7.2**	**2.0**

Table 4.4: Results of pixel labeling error with 5 categories (on intersection of test subsets in [129] and [139], 47 images).

layout surfaces. The labels of "openings" (*e.g.* , windows that are cut out of walls) are assigned based on the observed depth and surface normal. We compare with the RGBD region classifier of Silberman *et al*. [129] on the full test set. As expected, we outperform significantly on occluded surfaces (13.6% vs. 50.0% error) but also outperform on visible surfaces (4.8% vs. 5.2% error), which is due to the benefit of a structured scene model. Our method outperforms Guo *et al*.'s method on occluded surfaces, which means the layout estimation take benefits from the better object interpretation during the scene composition step. We also compare with Zhang *et al*. [139] who estimate box-like layout from RGBD images on the intersection of their test set with the standard test set (Table. 4.4. These images are easier than average, and our method outperforms substantially, cutting the error nearly in half (5.1% vs. 10.0%).

We evaluate layout depth prediction, the rendered depth of the room without foreground objects (Table 4.5). Error is the difference in depth from the ground truth layout annotation. On visible portions of layout surfaces, the error of our prediction is very close to that of the sensor, with the difference within the sensor noise range. On occluded surfaces, the sensor is inaccurate (because it measures the foreground depth, rather than that of the background surface), and the average depth error of our method is only 0.15 meters, which is quite good considering that the sensor noise is conservatively 0.03*depth.

Method	Layout Depth Error		
	overall	visible	occluded
Sensor	0.517	**0.059**	0.739
[35]	0.166	0.074	0.204
Ours	**0.150**	0.074	**0.181**

Table 4.5: Depth error for visible and occluded portions of layouts. Sensor error is the difference between the input depth image and annotated layout.

Occupancy and Free space Evaluation. We evaluate our scene prediction performance based on voxel prediction. The voxel representation has advantages of being computable from various volumetric representations, viewpoint-invariant, and usable for models constructed from multiple

Method	Occupancy, visible space			
	precision	recall	precision-ϵ	recall-ϵ
Bbox	0.423	0.463	0.669	0.734
Guo *et al*. [35]	**0.415**	0.346	**0.645**	0.581
Ours	0.376	**0.356**	0.621	**0.643**
Ours w/ GT-region	0.473	0.352	0.729	0.603

Table 4.6: Results of predicted object occupied voxel precision/recall, compared with fitting bounding boxes to ground truth regions (Bbox). All are calculated in the *visible* space.

Method	Occupancy, occluded space			
	precision	recall	precision-ϵ	recall-ϵ
Bbox	0.492	0.293	0.751	0.556
Guo *et al*. [35]	0.473	0.352	0.729	0.603
Ours	**0.483**	**0.400**	**0.739**	**0.709**
Ours w/ GT-region	0.552	0.421	0.812	0.680

Table 4.7: Results of predicted object occupied voxel precision/recall, compared with fitting bounding boxes to ground truth regions (Bbox). All are calculated in the *occluded* space.

Method	Free space, visible space	
	precision	recall
Sensor	1.000	0.988
Guo *et al*. [35]	0.998	0.995
Ours	0.998	0.995
Ours w/ GT-region	0.998	0.996

Table 4.8: Results of free space voxel estimation. Unoccupied voxel precision/recall using our method, given ground truth segmentation (GT-region) or only automatic region proposals. A baseline of free space inferred from depth point (Sensor) is compared. All are calculated in the *visible* space.

Method	Free space, occluded space	
	precision	recall
Sensor	–	0.000
Guo *et al*. [35]	0.732	0.606
Ours	0.732	**0.611**
Ours w/ GT-region	0.748	0.646

Table 4.9: Results of free space voxel estimation. Unoccupied voxel precision/recall using our method, given ground truth segmentation (GT-region) or only automatic region proposals. A baseline of free space inferred from depth point (Sensor) is compared. All are calculated in the *occluded* space.

views (as opposed to depth- or pixel-based evaluations). The scope of the evaluation is the space surrounded by annotated layout surfaces. Voxels that are out of view or behind solid ground truth 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 and layouts; free space is the complement of the set of occupied voxels.

Our voxel representation is constructed on a fine grid with $0.03m$ resolution to allow inspection of shape details of the 3D model objects we use. For detailed investigation, We evaluate voxel occupancy and freespace prediction performance in the visible space and the occluded space respectively. Table 4.6 and Table 4.7 present the voxel occupancy precision and recall in the two space; Table 4.8 and Table 4.9 present the voxel free precision and recall in the two space.

There is inherent annotation and sensor noise in our data, which is often much greater than $0.03m$. 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 perform evaluation

Method	Voxel occupancy				Semantic segmentation			Instance segmentation	
	precision	recall	precision-ϵ	recall-ϵ	avg class	avg instance	avg pixel	Mean coverage weighted	Mean coverage unweighted
DSS [46]	0.318	**0.585**	0.572	**0.667**	22.92	**29.52**	**41.45**	27.36	18.21
Ours	**0.381**	0.275	**0.642**	0.490	**24.72**	27.57	33.64	**33.75**	**25.57**

Table 4.10: Comparison with Deep Sliding Shapes (DSS) [46]. We evaluate voxel occupancy (our metric), semantic segmentation, and instance segmentation metric based on the predicted 19 object classes defined in DSS.

Method	Scene completion			Semantic scene completion											
	precision	recall	IoU	ceiling	floor	wall	window	chair	bed	sofa	table	tvs	furniture	objects	avg
SSCNet	57.0	**94.5**	**55.1**	15.1	**94.7**	**24.4**	0	**12.6**	32.1	**35**	**13**	**7.8**	**27.1**	10.1	**24.7**
Ours	**69.9**	63.0	49.4	**19.4**	68.2	21.0	**16.5**	10.7	**43.1**	22.2	0.7	4.3	23.0	**15.3**	22.8

Table 4.11: Comparison with Semantic Scene Completion (SSCNet) [47]. We follow the metric used by SSCNet and evaluate on the NYUd v2 test split with kinect depth map.

with a tolerance, proportional to the depth of the voxel, for which we use $\epsilon = 0.05 * depth$, based on the sensor resolution of Kinect. Specifically, an occupied voxel within ϵ of a ground truth voxel is considered to be correct (for precision) and to have recalled that ground truth voxel.

We compare with two simple baselines. For free space, the simple baseline is the *observed* free space from depth sensor. The free space from observed depth predicts 100% of the visible free space but recalls none of the free space that is occluded. Our approach performs similarly to the Guo *et al*.variation in visible space, and has better recall than Guo *et al*.in occluded space. For occupancy, our baseline generates bounding boxes based on ground truth segmentations with 10% outlier rejection. We outperform this baseline, whether using ground truth segmentations or automatic region proposals to generate the model. We also perform similarly to the Guo *et al*.variation in visible space, while largely outperform Guo *et al*.in occluded space. Also, note that precision is higher than recall, which means it is more common to miss objects than to generate false ones. It's also worth noting that except the bounding box baseline, performance in occluded space outperforms performance in visible space. This is because occluded space are often occupied by larger solid furniture, and the occupancy prediction becomes easier than estimating the fine detail of visible object surfaces.

4.4.3 Comparison with Deep Sliding Shape

We show in Table 4.10 the comparison between our approach and the DSS method. Our approach demonstrates better precision of voxel occupancy, average class semantic segmentation and both instance segmentation metrics, indicating a better shape representation based on CAD

model over bounding box. DSS shows better recall in object voxel occupancy, which is benefited from their 3D local search regime for each object in the whole scene. The better recall of predicted 3D objects also lead to their better performance in the average pixel accuracy and average instance accuracy of semantic segmentation.

We compare our performance on 3D object detection metrics in Table 4.12. Although 3D object detection is not our direct goal, our 3D detection on bed class is better than Sliding Shapes. We observe that our lower mAP for chair, sofa and table is due to three factors: 1) the failing of distinguishing between similar object classes: *e.g.* , we predict on desk shape which is in ground truth a table, and we confuse between sofa and chair (with arms like sofa shape); 2) 3D localization error, which is especially for toilet case; 3) our method is constrained to produce objects that do not overlap, even with classes outside the five main furniture.

Method	bed	toilet	sofa	table	chair
Sliding Shapes [45]	33.5	67.3	33.8	34.5	29
DSS [46]	**84.7**	**89.9**	**55.4**	**70.5**	**61.1**
Ours	36.9	36.1	14.12	4.81	3.1

Table 4.12: 3D Amodal object detection in NYUd v2 dataset. We evaluate on the mAP (%) as defined on [45]

4.4.4 Comparison with Semantic Scene Completion Network

We show in Table 4.11 the comparison between our approach and the SSCNet method. For scene completion, our approach outperforms SSCNet in precision, which benefits from our detailed shape modeling for both objects and layout. SSCNet shows better performance in recall, due to their voxel-level prediction from observed occupancy space. Our lower recall also leads to our slightly lower performance in voxel-level occupancy IoU than SSCNet. For semantic scene completion, our predictions on ceiling, window, bed and other object classes outperform SSCNet. Our average voxel-level IoU is less than 2% lower than SSCNet. Errors mainly come from objects like table with fewer observed depth points to fit object and shape classification confusion like tv (classified as other objects) and sofa (classified as bed or chair) class.

4.4.5 Ablation study

Region proposals. In Tables 4.1 and 4.2, we see the impact of changing the region proposal method from Guo *et al.*'s Prim's based algorithm to Gupta *et al.*'s [131] Multiscale Combinatorial Grouping (MCG). The "MCG + CCA" method is Guo *et al.*with only a substitution of the region proposal method and improves over "Guo *et al.*" in each case. We can also see that using ground

Method	avg class accuracy (%)			avg 3D IoU			avg surface distance (m)		
	top 1	top 2	top 3	top 1	top 2	top 3	top 1	top 2	top 3
Guo *et al*. [35]	11.97	16.36	19.65	0.134	0.177	0.200	0.033	0.029	0.027
Ours	**41.56**	**54.47**	**62.07**	**0.191**	**0.231**	**0.249**	**0.026**	**0.024**	**0.023**

Table 4.13: Quantitative evaluation for our retrieval method compared with Guo *et al*.'s method.

Method	avg per class	avg precision	avg over instance	picture	chair	cabinet	pillow	bottle	books	paper	table	box	window	door	sofa	bag	lamp	clothes
w/o support height	43.7± 0.3	37.7 ± 0.1	40.8± 0.3	57.5	46.1	39.5	66.5	**55.6**	30.0	40.7	36.7	10.6	61.4	54.9	63.0	14.9	64.6	25.3
w/ support height	**44.7± 0.3**	**39.7±0.1**	**42.7±0.2**	57.5	**53.1**	**44.35**	**69.0**	54.9	**33.5**	**43.5**	**39.5**	**14.1**	**62.4**	**57.8**	**65.9**	**15.1**	**65.9**	**26.9**

Table 4.14: **81-class classification accuracy on ground truth 2D regions in the test set.** We compare on: our classification network with/without estimating support height. We compute the average accuracy for each class, average precision based on the predicted probability and the accuracy averaged over instances. The classification networks are trained and evaluated 10 times and the means and standard deviations (reflecting variation due to randomness in learning) are reported. 15 common object class results are also listed. Bold numbers signify better performance.

truth regions improves performance over automatic region proposals, but the improvement is less than we had expected, which indicates that our region proposal and scene composition steps are effective.

Retrieval. We evaluate our candidate shape retrieval method compared with Guo *et al*.'s retrieval method, as shown in table 4.13. To isolate from the influence of errors in other steps like region proposals generation or scene composition, we assume that the ground truth regions are given (and the candidate selection in Chapter 4.3.2 for the CNN-based method is not performed). We evaluate top N retrieved class accuracy and top N retrieved shape similarity, based on the shape intersection over union (IoU) and the surface-to-surface distance [66]. In our experiment, we set $N \in \{1, 2, 3\}$. To isolate from rotation ambiguity in shape similarity measurement, we rotate each retrieved object to find the best shape similarity score with the ground truth 3D shape. Our CNN-based retrieval method outperforms the Guo *et al*.'s CCA method under all the evaluation criteria.

We can indirectly measure the effectiveness of our **scene composition** by comparing "Ours" to "GT-Region" (use the best fitting 3D object model and class for each ground truth 2D region). For semantic segmentation, occupancy (Table 4.6 and 4.7), and free space (Table 4.8 and 4.9, we see a relatively small improvement by using ground truth regions. Qualititatively as well, our automatic region proposal and scene composition produce similarly good results to those produced using ground truth regions. Although the region proposal method is far from perfect, our scene composition step appears to be effective at keeping the good region proposals and discarding the others.

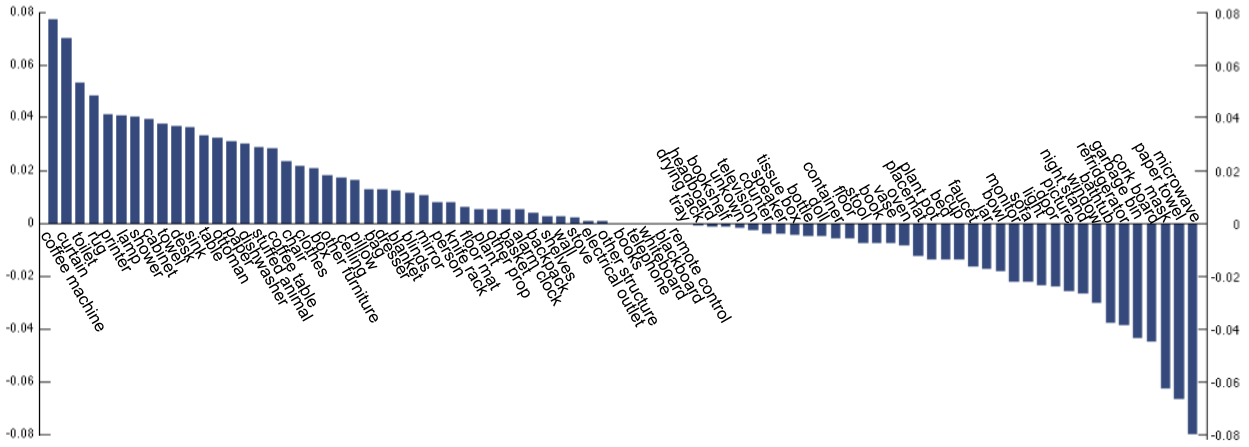


Figure 4.10: Semantic segmentation accuracy of each class w/ support minus accuracy w/o support accuracy with automatic region proposals

Support height prediction. We first examine the effect of predicting support height in region categorization. In table 4.14, we report the region classification accuracy of our classification network given or without the support prediction on the ground truth 2D regions in the test set. Overall, incorporating support height prediction improves the classification results. For certain classes, objects often appear on the same height (*e.g.* , chair, desk, rug) have better classification accuracy given object's height, while objects that can appear on several heights (*e.g.* , picture) do not have this benefit. Table 4.15 shows the per-instance classification accuracy under different occlusion ratios of the ground truth regions in the test set. The occlusion ratio is computed from the observed 2D region over the 2D projection of the ground truth 3D label. The improvement in the classification accuracy is larger for highly occluded area, which agrees with our hypothesis that estimating object's support height will help classify occluded regions.

Occlusion Ratio	< 0.5	> 0.5
w/ support height	**45.8**	**38.5**
w/o support height	44.2	35.7

Table 4.15: Classification accuracy for ground truth regions under different occlusion ratios in test set

As an ablation study of the effect of incorporating support inference in the overall scene composition performance, in Table 4.1 and Table 4.2, we report our result with or without support height prediction and using the ground truth support height as the upper bound for including support reasoning. As we can see, incorporating support improves the semantic segmentation performance by about 1%. Figure 4.10 shows the effect of applying support reasoning on semantic segmentation accuracy of each class. Classes that are often at a certain height: toilet, printer, lamp, etc.

will have benefits from support height information. Classes that might appear in several levels of height: mask, pixture, light won't gain benefits from this. Using ground truth support height generally leads to a larger benefit than estimated support height, as expected.

4.5 DISCUSSIONS

We discuss the performance of our framework and the contributions of each of the component. Our framework follows three steps: region proposal, classification and shape retrieval, and scene composition. Each step solves for a loosely separate sub-problem, but are all closely related to produce a compact result for the complete 3D scene parsing task. For in-depth investigation of each of the component, as in the ablation study in Chapter 4.4.5, we perform experiments and analyze the effectiveness of the design choices in each step. For the region proposal step, we see improvements on applying the state-of-the-art RGBD region proposal method by Gupta *et al*. [131] instead of our previous Prim's based method. We observe less difference in overall performance if we use ground truth regions in our framework, which indicates the downstream step's robustness to errors from region proposals. For the retrieval step, our use of CNNs to classify and retrieve similar shapes outperforms our previous simpler CCA-based retrieval approach. We validate our hypothesis that estimating support height of objects can lead to better classification to improve retrieval, especially for occluded objects. The heavy occlusion characteristic of indoor scene makes the region classification harder, and accumulates errors to the retrieval step. To resolves the retrieval errors, we incorporate the scene composition step with combinatorial optimization to select a subset of shape candidates that fits the observation and scene regularity like no 3D overlap between shapes. We combine greedy search and a hill-climbing method to efficiently solve the hard optimization problem which outperforms a variety of other attempted methods including general integer programming algorithms and relaxations. Note that possible improvements for the framework exists, as we will discuss in the following paragraphs.

Common errors made by our approach include splitting large objects into small ones, completely missing small objects, difficulty with highly occluded objects such as chairs. Based on the previous discussions, we see the major sources of error of our framework are: 1) region classification error and 2) shape fitting error. Errors mainly result from the heavy occlusion from objects in indoor scene. Fewer errors accumulate in the shape retrieval step, referring to the comparison between Table 4.14 and 4.13. The method does not seem to be very sensitive to the region proposal method, based on the comparison in Table 4.1, 4.2, 4.6 and 4.7. In fact, we were surprised to find that results from automatic region proposals are often comparable to those from ground truth regions. In part, this is because the later fitting and optimization steps are effective at discarding bad regions

and retaining good ones.

We see many interesting directions for future work: 1) modeling object context such as chairs tend to be near/under tables, relative pose among objects and the layouts; 2) modeling self-similarity, *e.g.* , most chairs within one room will look similar to each other; 3) incorporating semantic constraints. For example, SSCNet by Song *et al.* [47] employs neural network to perform semantic completion from single depth image. For further improvement, our system can incorporate the semantic completion from Song *et al.*to generate candidate region proposals and add semantic constraint for scene composition in Chapter 4.3.3.

4.6 CONCLUSION

In this chapter, we proposed an approach to predict a complete 3D scene model of individual objects and surfaces from a single RGBD image. Our representation encodes the layout of walls as 3D planes and all interpretable objects as 3D mesh models, parsing both the visible and occluded portion of the scene. 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 spatial consistency. We incorporate support inference to aid interpretation and propose a retrieval scheme that uses CNNs to classify regions and find objects with similar shapes. We demonstrate the performance of our method on our newly annotated NYUd v2 dataset with detailed 3D shapes. Compared with our earlier work (Guo *et al.* [35]), we demonstrate improvements due to better region proposals and use of CNNs to classify and retrieve similar shapes. We also show that our results from automatic region proposals are nearly as good as from ground truth regions, demonstrating the effectiveness of our scene composition and, more generally, of finding a scene hypothesis that is consistent in semantics and geometry. We also showed that estimating support height of objects can lead to better classification, especially for occluded objects.

CHAPTER 5: SILHOUETTE GROUNDED POINT CLOUD RECONSTRUCTION BEYOND OCCLUSION

One major challenge in 3D object reconstruction is to infer the complete shape geometry from partial foreground occlusion. In this chapter, we propose a method to reconstruct the complete 3D shape of an object from a single RGB image, with robustness to occlusion. Given the image and a silhouette of the visible region, our approach completes the silhouette of the occluded region and then generates a point cloud. We improve state-of-the-art for 3D shape prediction with a 2D re-projection loss from multiple synthetic views and a surface-based smoothing and refinement step. We further show improvements for reconstruction of non-occluded and partially occluded objects by providing the predicted complete silhouette as guidance. Experiments demonstrate the efficiency of our approach both quantitatively and qualitatively on a real scene dataset.

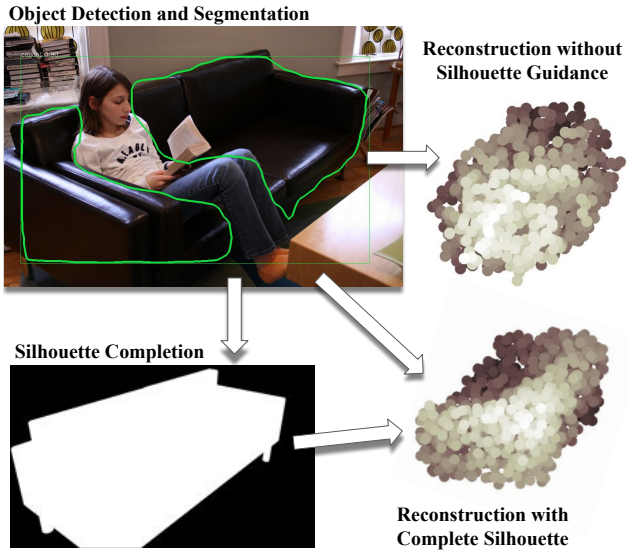


Figure 5.1: **Illustration.** The person sitting on the sofa blocks much of the sofa from view, causing errors in existing shape prediction methods. We propose improvements to shape prediction, including the prediction and completion of the object silhouette as an intermediate step, and demonstrate more accurate reconstruction of both occluded and non-occluded objects. Best viewed in color.

5.1 INTRODUCTION

3D reconstruction from 2D images has many applications in robotics and augmented reality. One major challenge is to infer the complete shape of a partially occluded object. Occlusion frequently occurs in natural scenes: *e.g.* we often see an image of a sofa occluded by a table in front and a dining table partially occluded by a vase on top. Even multi-view approaches [81, 82, 83] may fail to recover complete shape, since occlusions may block most views of the object. Single-view learning-based methods [85, 86, 87] have approached seeing beyond occlusion as a 2D semantic segmentation completion task, but complete 3D shape recovery adds the challenges of predicting 3D shape from a 2D image and being robust to the unknown existence and extent of

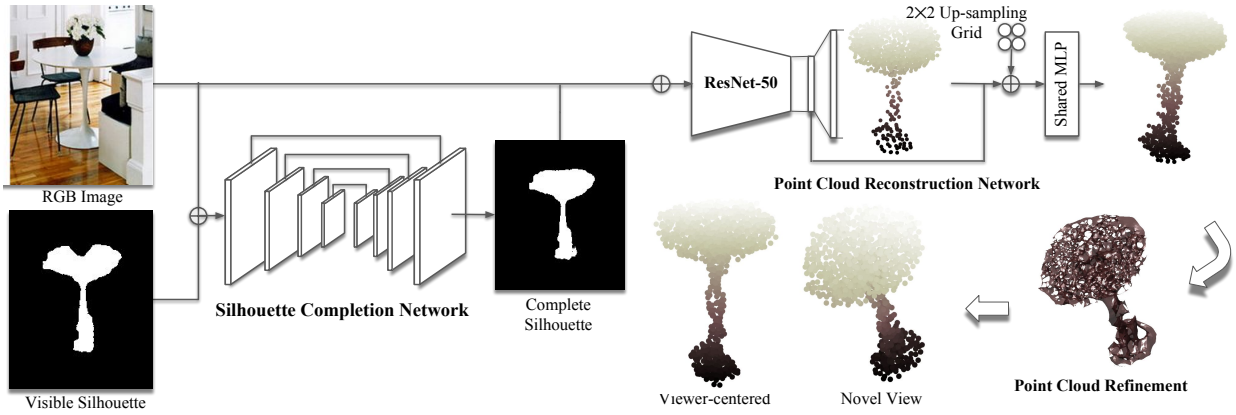


Figure 5.2: **Approach**. Our approach is composed of three steps. In the first step, the silhouette completion network takes an RGB image and the visible silhouette as input, and predict the object's complete silhouette beyond occlusion. In the second step, given the RGB image and the predicted complete silhouette, the reconstruction network predicts point clouds in viewer-centered coordinates. Finally, we perform a post refinement step to produce smooth and uniformly distributed point clouds. Best viewed in color.

an occluding region.

In this chapter, our goal is to reconstruct a complete 3D shape from a single RGB image, in a way that is robust to occlusions. We follow a data-driven approach, using convolution neural networks (CNNs) to encode shape-relevant features and decode them into an object point cloud. To simplify the shape prediction, we split the task into: (1) determining the visible region of the object; (2) predicting a completed silhouette (filling in any occluded regions); and (3) predicting the object 3D point cloud based on the silhouette and RGB image (Figure 5.1). We reconstruct the object in a viewer-centered manner, inferring both object shape and pose. We show that, provided with ground truth silhouettes, shape prediction achieves nearly the same performance for occluded objects as non-occluded objects. We obtain the visible portion of the silhouette using Mask-RCNN [127] and then predict the completed silhouette using an auto-encoder. Using the predicted silhouette as part of shape prediction also yields large improvements for both occluded and non-occluded objects, indicating that providing an explicit foreground/background separation for the object in RGB images is helpful.

Our reconstruction represents a 3D shape as a set of point clouds, which is flexible and easy to transform. Our method follows an encoder-decoder strategy, and we demonstrate performance gains using a 2D re-projection loss from multiple synthetic views and a surface-based post refinement step, achieving state-of-the-art. Our silhouette guidance approach is related to shape from silhouette [140, 141, 142], but our silhouette guidance is part of a learning approach rather than explicit constraint.

5.2 SILHOUETTE GROUNDED POINT CLOUD RECONSTRUCTION

Direct point set prediction from a single image is challenging due to the unknown camera viewpoint or object pose and large intraclass variations in shape. This requires a careful design choice on the network architecture. We aim to have an encoder that can capture object pose and shape from a single image, and a decoder that is flexible in producing unordered, dense point clouds.

In this section, we introduce our network architecture, the training scheme including a 2D reprojection loss on multiple synthetic views to improve performance (Chapter 5.2.1). We then introduce a post-refinement step via surface-based fitting (Chapter 5.2.2) to produce smooth and uniformly distributed point sets.

5.2.1 Point cloud reconstruction network

Our network architecture is illustrated in Figure 5.2. The network predicts 3D point clouds in viewer-centered coordinates. The encoder is based on ResNet-50 [127] to better capture object shape and pose feature. The decoder follows a coarse-to-fine multi-stage generation scheme in order to efficiently predict dense points with limited memory. Our decoder follows the design of PCN [143]. The coarse predictor predicts $N = 1024$ sparse points. The refinement branch produces $4N$ finer points, by learning a 2×2 up-sampling surface grid centered on each coarse point via local folding operation. We experimented with a higher up-sampling rate (*e.g.* 9, 16) as PCN but observed repetitive patterns across all surface patches, missing local shape details. We also experienced with a fully convolutional encoder instead and observed that performance drops. Note that our network is able to generate a denser prediction with another up-sampling branch on top, but we find that the current structure best balances accuracy and training/inference speed. Our approach does not require features from partial points like PCN, and we observe an on-par performance with PSG [89], a state-of-the-art method in point set generation, even without the refinement step we introduce in the next section (see experiments in Chapter 5.4.1).

Loss function. We consider the training loss in 3D space using the bi-directional Chamfer distance. Given predicted point clouds $\hat{p} \in \hat{P}$ and the ground truth $p \in P$, we have:

$$L_{rec} = d_{Chamfer}(P, \hat{P}) = \frac{1}{|P|} \sum_{p \in P} \min_{\hat{p} \in \hat{P}} \|p - \hat{p}\|_2^2 + \frac{1}{|\hat{P}|} \sum_{\hat{p} \in \hat{P}} \min_{p \in P} \|p - \hat{p}\|_2^2 \tag{5.1}$$

To further boost the performance, we propose a 2D reprojection loss on point sets as follow:

$$L_{proj} = d(Proj(P), Proj(\hat{P})) = \frac{1}{|P|} \sum_{p \in P} \min_{\hat{p} \in \hat{P}} \|K[R\ t]p - K[R\ t]\hat{p}\|_2^2 \tag{5.2}$$

Where $Proj(\cdot)$ is a 2D projection operation from 3D space, with 3D rotation R and translation t in world coordinates and a known camera intrinsic K. Since our reconstruction is viewer-centered, we can simply set $R = I, t = 0$ assuming projections on the image plane. Our 2D reprojection loss is a one-directional Chamfer distance; we only penalize the sum of minimum distance to projected predictions for each projected ground truth point. This is because the Chamfer distance on another direction tends to be redundant, when the predicted points are projected inside the ground truth region, resulting in zero gradient and cannot enforce the predictions to form the same 2D distributions as the ground truth. Although we project points instead of surfaces or voxel occupancy, producing non-continuous 2D segments, our 2D re-projection loss is computational efficient and shows promising improvements in experiment, and we find that fitting a surface to these points is effective.

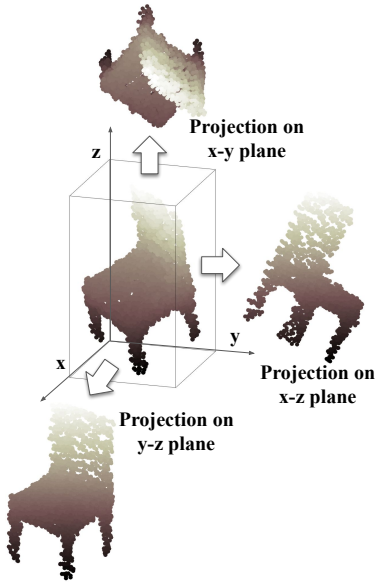


Figure 5.3: Illustration of our multi-view 2D reprojection loss.

We can extend Eq. 5.2 to project 3D points based on multiple synthetic views: *e.g.* projection on x, y or z plane in world coordinates, as illustrated in Figure 5.3. In this case we can simply change the rotation matrix R based on each configurations. We found that projections on 3 views (x-y,y-z,x-z plane) are sufficient to learn a better 3D shape prior. Our 2D reprojection loss does not require additional rendered 2D silhouette ground truth of known view points, which makes the training possible on the dataset where the 3D ground truth is available.

When being projected on the image plane (same as y-z plane in world coordinates), the ground truth is a subset of the ground truth silhouette. We thus use 2D points sampled from the ground truth segmentation mask $s \in S$ instead of projecting ground truth 3D points p:

$$L_{silhouette} = \frac{1}{|P|} \sum_{s \in S} \min_{\hat{p} \in \hat{P}} \|s - K[R\ t]\hat{p}\|_2^2 \tag{5.3}$$

The **overall loss function** is shown below:

$$L = w_{rec} L_{rec} + w_{silhouette} L_{silhouette} + w_{proj} L_{proj} \tag{5.4}$$

which is the weighted summation over the 3D Chamfer loss and the 2D reprojection losses. Here $L_{silhouette}$ is the 2D reprojection loss on the image plane, and L_{proj} is the projection on x, z planes. Note that our network produces point clouds of coarse and fine resolutions, but we only penalize on the finest output, which helps ease training and have better performance.

Implementation details. Our network gets as input a 224×224 image with pixel values normalized to $[0, 1]$. The bottleneck feature size is 1024. The coarse decoder consists two fc layers, with feature size of 1024 and 3072 and ReLU in between. We set the surface grid for point up-sampling to be zero-centered and side length of 0.1. We observed the importance of using pre-trained weights from ImageNet to initialize the ResNet encoder, and found that a stage-wise training scheme leads to faster convergence and easier training: first train to predict coarse point cloud, fix the trained layers, then train the up-sampling header, and finally train the whole network end-to-end. We use ADAM [134] to update network parameters with a learning rate of $1e^{-4}$ and $\epsilon = 1e^{-6}$ and batch size 32. We set $w_{rec} = 1$, $w_{silhouette} = 1e^{-8}$ and $w_{proj} = 1e^{-9}$ in Eq. 5.4 based on grid search in the validation set.

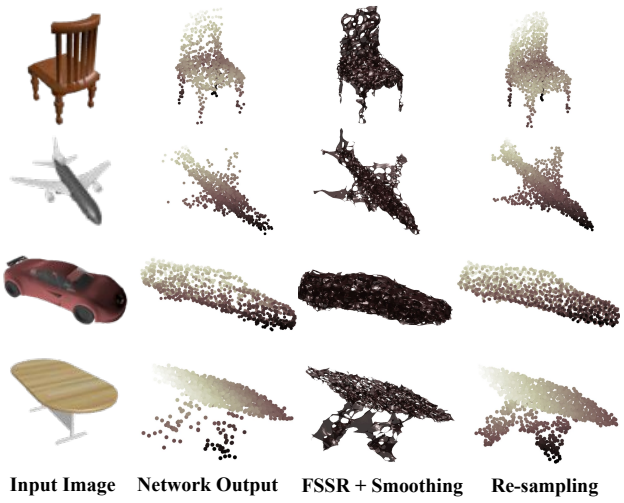


Figure 5.4: Surface-based point clouds refinement. We show from left to right: input RGB image, network prediction, FSSR surface fitting and smoothing and the re-sampled point clouds on the surfaces. Each sample consists the same number of points and we visualize each predicted shape in a novel view for better illustration. Our refinement step is able to produce smooth and uniformly distributed point sets. Best viewed in color.

Data augmentation. We augment the training samples by gamma correction with γ between 0.5-2. We re-scale image intensity with a minimum intensity ranges between 0-127 and a fixed maximum intensity of 255. We add color jittering to each RGB channel independently by multiplying a factor ranges in 0.8-1.2. Each augmentation parameter is uniformly and randomly sampled from the defined range.

5.2.2 Surface-based point clouds refinement

One important 3D shape property is the smooth and continuous shape surfaces, especially for thin structures like chair legs and light stands. To impose this property, we perform a post-refinement step (Figure 5.4), fit surfaces from dense points, smooth surfaces and uniformly re-sample points from the surfaces again. Our surface fitting method is based on Floating Scale Surface Reconstruction (FSSR) [144], which is a state-of-the-art surface-based reconstruction from dense point clouds. FSSR requires additional per-point normal factor and a scale value for implicit surface range. We obtain point-wise normal by plane fitting on 6 neareast neighbors, set per-point scale as the average distance to the two closest points. We found that this local surface fitting can help preserve shape details. A mesh cleaning step goes after FSSR to remove small and redundant patches. We also experimented with Poisson surface reconstruction [145]), but FSSR produces a better surface in our case.

Smoothing. Given the fitted surfaces, we apply implicit integration method [146] for smoothing. We use curvature flow as the smoothing operator with 5 iterations (efficient enough in experiment). We then uniformly sample point clouds based on Poisson disc sampling to get our optimized point clouds.

Our surface fitting, smoothing, and resampling enables production of evenly distributed points that can model thin structures and smooth surfaces, but the predicted shape may not be closed, preventing volumetric-based evaluation. Also, small and disconnected sections of points that model details may be lost, and points that are incorrectly joined into mesh surfaces may increase errors in some regions. Overall, though, this surface-based refinement step improves the performance (large gain in earth-mover's distance with a small cost to Chamfer distance).

5.3 RECONSTRUCTION OF OCCLUDED OBJECTS

So far, our proposed point cloud reconstruction approach does not consider foreground occlusions, such as a table in front of a sofa, or a person or pillows on the sofa. Standard approaches do not handle occlusions well since the model does not know whether or where an object is occluded. Starting with an RGB image and an initial silhouette of the visible region, we present a 2D silhouette completion approach to produce object's complete silhouette. We show that prediction based on the true completed silhouette greatly improves shape prediction and brings performance on occluded objects close to non-occluded; using predicted silhouettes also improves performance in general, with completed silhouettes slightly outperforming predicted silhouettes of the visible portion.

5.3.1 Silhouette completion network

We assume a detected object and its RGB image crop I and the segmentation of the visible region S_v. Those can be acquired by recent approaches like Mask-RCNN [127]. Our silhouette completion network (Figure 5.2) predicts the complete 2D silhouette S_f of the object based on S_v. The network follows an encoder-decoder strategy, gets as input the concatenation of I and S_v, and predicts S_f with the same resolution as S_v. The encoder is a modified ResNet-50 and the decoder consists 5 up-sampling layers, producing a single channel silhouette. Intuitively, when occlusion occurs, we want the network to complete silhouette, and when no occlusion occurs, we want the network to predict the original segmentation. We found that adding skip connections can obtain this property. We concatenate the feature after the $i-th$ conv layer of the encoder to the input feature layer of the $(6-i)-th$ decoder layer. We observed that adding skip links to all the conv layers helps ease training and produces better results.

Implementation details. For each input I and S_v, we resize them to 224×224 with the aspect ratio preserved and white pixel padded. The value of I is re-scaled to $[0, 1]$ and S_v is a binary map with 1 indicating the object. For the encoder, we remove the top fc layer and the average pooling layer of a pre-trained ResNet on ImageNet and get a bottleneck feature of 7×7. The decoder applies nearest neighbor up-sampling with a scale factor of 2, followed by a convolution layer (kernel size 3×3, stride 1) and ReLU. The decoder feature sizes are 2048, 1024, 512, 256, 64 and 1 respectively, with a Sigmoid operation on top. We train the network using binary cross entropy loss between the prediction and the ground truth complete silhouette. We optimize the network through ADAM [134] with a learning rate of $1e^{-4}$ and $\epsilon = 1e^{-6}$ and batch size 32. Our final prediction is a binary mask thresholded by 0.5. To account for the truncation of the full object due to the unknown extent of occluded region behind, we expand the bounding box around each object by 0.3 on each side.

Data augmentation. For each training sample, we perform random cropping on the input image. We crop with a uniformly and randomly sampled ratio ranges between $[0.2, 0.4]$ on each side of the input image. Other augmentations include left-right flipping with 50% probability and random rotation uniformly sampled within ± 5 degrees on the image plane. We also perform image gamma correction, intensity changes and color jittering as in Chapter 5.2.1.

5.3.2 Silhouette grounded reconstruction

Given the predicted complete silhouette S_f, we modify our point cloud reconstruction network to be robust to occlusion. We concatenate S_f as an additional input channel to the input RGB image I. We found this modification can effectively guide reconstruction for both partially occluded and

non-occluded objects. We show in experiments the importance of using silhouette comparing with the approach with no segmentation guidance at all. To ease training, we first train our network with the task of single image reconstruction without occlusion (as in Chapter 5.2.1), then fine-tune our trained network on synthetic images with occlusion as described below.

Synthetic occlusion dataset. Since there is no large-scale 3D dataset of rendered 2D object image with occlusion and the complete silhouette ground truth available, we propose to generate a synthetic occlusion dataset. Instead of off-line rendering samples which is time consuming and has limited variety of occlusion, we propose a "cut-and-paste" strategy to create random occlusion. Starting with a set of pre-rendered 2D images without occlusion and known ground truth silhouettes, for each input training sample, we cut out the object from another randomly selected image from the dataset, paste and overlay the object on the training sample. To be more specific, we paste around the region defined by $[(h_0 - h', w_0 - w'), (h_1 + h', w_1 + w')]$, where $[(h_0, w_0), (h_1, w_1)]$ denotes the top left and bottom right position of the bounding box around the training object image. h', w' are the height and width of the pasted new object segment considered as foreground occlusion.

We exclude samples with occlusion over 50% of the object region to ease training. We perform "cut-and-paste" with 50% probability in training and further add randomly sampled real-scene background with 50% probability. The training samples are then applied with the same data augmentation as in Chapter 5.2.1. We penalize the network on the ground truth complete 3D point clouds and the 2D re-projection loss assuming full shape 2D re-projection. We use the ground truth visible and complete silhouettes to train the network.

5.4 EXPERIMENT

We evaluate our approach on (1) single image 3D reconstruction; and (2) the robustness to occlusion with silhouette guidance, both quantitatively and qualitatively. Our network achieves the sate-of-the-art for point cloud prediction. We discuss in ablation study the performance gains with our proposed surface-based refinement step and the 2D re-projection loss. To evaluate the robustness to occlusion, we compare on the performance using occluded objects and non-occluded objects, analyze and discuss the effects with different configurations: with ground truth complete silhouettes, with our predicted complete silhouettes, with pedicted visible silhouettes and without silhouette guidance.

We implement our network using PyTorch, train and test on a single NVIDIA Titan X GPU with 12GB memory. A single forward pass of the network takes 12.7 ms. The point cloud refinement step is C++ based and takes less than 1s on a Linux machine with Intel Xeon 3.5G Hz in CPU

mode. Code and data will be available upon acceptance.

5.4.1 Single image 3D reconstruction

We evaluate our approach on ShapeNet [147], which is the standard dataset to compare with the state-of-the-art. Since ShapeNet objects are non-occluded, we train our network without silhouette guidance.

We train and test our network following the official train-test split defined by Choy *et al*. [64]. The dataset consists objects from 13 classes and each has 24 randomly selected views rendered in 2D. We carefully split out 10% shapes from each class in the train split to form the validation set. The viewer-centered point clouds ground truth is generated by Wang *et al*. [56].

We consider the following two standard metrics:

1. Chamfer distance (CD), defined in Eq. 5.1.

2. Earch mover's distance (EMD). defined as:

$$d_{EMD}(P, \hat{P}) = \min_{\phi: P \to \hat{P}} \frac{1}{|P|} \sum_{p \in P} \|p - \phi(p)\| \tag{5.5}$$

   where $\phi : P \to \hat{P}$ denotes a bi-jection that minimizes the average distance between corresponding points. Since ϕ is expensive in computation, we follow the approximation solution as in Fan *et al*. [89].

Figure 5.5: Sample point cloud reconstruction results on ShapeNet dataset. Each from left to right: input RGB image, reconstruction in viewer-centered coordinates and a novel view of the predicted 3D shape. Our approach is able to reconstruct the thin structures and to model smooth and continuous surfaces. Best viewed in color.

We show in Figure 5.5 sample qualitative results by our approach on ShapeNet. We report in Table 5.1 our quantitative comparison with the state-of-the-art viewer-centered reconstruction approaches. PSG [89] generates point clouds and Pixel2Mesh [56] produces meshes. We sample

Category	CD			EMD		
	PSG	Pixel2Mesh	Ours	PSG	Pixel2Mesh	Ours
plane	0.430	0.477	**0.386**	**0.396**	0.579	0.527
bench	0.629	0.624	**0.436**	1.113	0.965	**0.815**
cabinet	0.439	0.381	**0.373**	2.986	2.563	**2.147**
car	0.333	**0.268**	0.308	1.747	**1.297**	1.306
chair	0.645	0.610	**0.606**	1.946	1.399	**1.257**
monitor	0.722	0.755	**0.501**	1.891	1.536	**1.314**
lamp	1.193	1.295	**0.969**	1.222	1.314	**1.007**
speaker	0.756	0.739	**0.632**	3.490	2.951	**2.441**
firearm	**0.423**	0.453	0.463	**0.397**	0.667	0.572
couch	0.549	0.490	**0.439**	2.207	1.642	**1.536**
table	0.517	**0.498**	0.589	2.121	1.480	**1.340**
cellphone	0.438	0.421	**0.332**	1.019	0.724	**0.674**
watercraft	0.633	0.670	**0.478**	0.945	0.814	**0.730**
mean	0.593	0.591	**0.501**	1.653	1.380	**1.205**

Table 5.1: Viewer-centered single image shape reconstruction performance compared with the state-of-the-art on ShapeNet. We report both the Chamfer distance (left) and the Earch mover's distance (right). Our approach outperforms PSG [89] and Pixel2Mesh [56] on both the metrics.

2466 points as in Pixel2Mesh for evaluation. Our approach out-performs the two methods on both metrics.

Ablation study. We show in Table 5.2 our performance on ShapeNet with different configurations: without both surface-based refinement step and 2D reprojection loss, without refinement step only, without 2D reprojection loss only and our full approach. We observe the improvement with 2D reprojection loss on both CD and EMD. The improvement with 2D loss is less significant when we have the surface-based refinement step, showing the refinement step mitigates some problems with point cloud quality that are otherwise remedied by training with the reprojection loss. The refinement step increases Chamfer distance, mainly due to the smoothed out small and sparse point pieces that model the thin and complex shape structure like railings or handles (Figure 5.5, 1st row, 2nd column), and the enhanced error point predictions by connecting sparse point sets (Figure 5.5, 3rd row, 2nd column). It's also worth noting that, even without the post-refinement step, our approach achieves better CD than PSG and a slightly worse EMD, proving the superiority of our network architecture.

Comparison with object-centered approach. We show in Table 5.3 our comparison with the state-of-the-art object-centered point cloud reconstruction approach: 3D-LMNet [90] on ShapeNet. We follow the evaluation procedure as in 3D-LMNet, sample 1024 points and re-scale our prediction (and ground truth) to be zero-centered and unit length of 1, and perform ICP to fit to the ground truth. Although our approach targets on a harder task of both shape prediction and view

Method	CD	EMD
Ours w/o surface refine & w/o 2D proj loss	0.398	1.784
Ours w/o surface refine	**0.389**	1.660
Ours w/o 2D proj loss	0.502	1.220
Ours w/ silhouette	0.512	1.214
Ours Full	0.501	**1.205**

Table 5.2: Ablation study. We evaluate our performance on ShapeNet for both Chamfer distance and Earth mover's distance with different configurations: without both surface refinement and 2D reprojection loss, without surface refinement only, without 2D reprojection loss only and our full approach. Our full approach seeks for a balanced performance under both two metrics.

Method	CD	EMD
3D-LMNet	**5.40**	7.00
Ours	5.54	**5.93**

Table 5.3: Comparison with the state-of-the-art object-centered point cloud reconstruction approach on ShapeNet, reported in both Chamfer distance and Earth mover's distance. Although our method is trained on a harder viewer-centered prediction, our method achieves much better EMD and slightly worse CD.

point estimation, we achieve a much better EMD and only a slightly worse CD. Note that the reported CD and EMD are of different scales compared with that reported in the comparison with Pixel2Mesh, this is because 3D-LMNet takes a squared value of each CD distance.

Silhouette guidance. Although all of the objects are fully visible with a plain background in ShapeNet, we evaluate the impact of the silhouette guidance on ShapeNet object shape prediction. Since ShapeNet images [64] use white background and have no foreground occlusion, we obtain silhouettes for both training and testing with image intensity < 1 (given image value ranges between $[0, 1]$). We show in Table 5.2 the performance with silhouette guidance. The overall performance drops slightly. This is because input images with white background provide enough evidence for the network to capture object's shape feature, and additional silhouette guidance will confuse the network due to the ambiguity of similar silhouettes from quite different shapes.

5.4.2 Reconstruction of occluded objects

We show: (1) our silhouette completion network performance compared with the state-of-the-art; and (2) the reconstruction robustness to occlusions with silhouette guidance.

Silhouette completion. Since there's no real dataset with 2D ground truth of complete segmentation beyond occlusion, we train our network on the synthetic DYCE dataset introduced by

Method	Full	Visible	Occluded
SeGAN [86]	76.4	63.9	27.6
Ours (ResNet-18)	82.8	82.9	33.9
Ours (ResNet-50)	**84.3**	**83.4**	**36.2**

Table 5.4: Object silhouette completion performance compared with the state-of-the-art SeGAN [86] on DYCE dataset. We report the 2D IoU compared with ground truth visible, occluded and full region. We evaluate our approach with two configurations: ResNet-18 based encoder (same as SeGAN) and ResNet-50 based encoder. Both settings outperform SeGAN on all metrics.

Ehsani *et al*. [86]. The dataset contains segmentation mask for both visible and occluded region in photo-realistic indoor scenes. We tune the network parameters using the official train-val split and test our performance on the test set.

We show in Table 5.4 the comparison with the state-of-the-art silhouette completion approach SeGAN [86] on DYCE. We evaluate the 2D IoU between the predicted complete silhouette and the ground truth. For detailed comparison, we consider the IoU on the visible potion, the invisible portion and the full silhouette. Both SeGAN and our approach use ground truth visible 2D silhouette as input for training and testing. We report the performance of our approach with ResNet-18 and ResNet-50 encoder respectively. SeGAN uses ResNet-18 encoder, our approach with ResNet-18 outperforms SeGAN, due to our skip connection design to preserve the visible region and the up-sampling decoder to predict better region beyond occlusion. Since using ResNet-50 as encoder shows a better performance, we use the network with ResNet-50 encoder to generate complete silhouettes for the downstream reconstruction task.

To evaluate performance on real scenes, we directly test our trained approach on the challenging Pix3D dataset
[115]. Pix3D contains real images of objects (either occluded or non-occluded) with aligned 3D shape ground truth and a label of occluded or not. We test on the categories that co-occur in both Pix3D and ShapeNet: sofa, chair and table. We use Mask-RCNN to detect and obtain the visible silhouette of the object in each image and perform completion upon the detections. We obtain valid detections with correctly detected object class and a bounding box IoU > 0.5 compared with the ground truth. Table 5.5 reports our performance on three object classes. We use the projected object mask as the ground truth complete silhouette. Our approach achieves a better performance than the Mask-RCNN baseline (detects visible region). Training with ground truth visible silhouettes can have better performance than training with detected silhouettes from Mask-RCNN. This is because it's hard to correct much if the input visible silhouettes are error predicted. For the sofa class the visible silhouettes are good and our approach has larger margin of improvements,

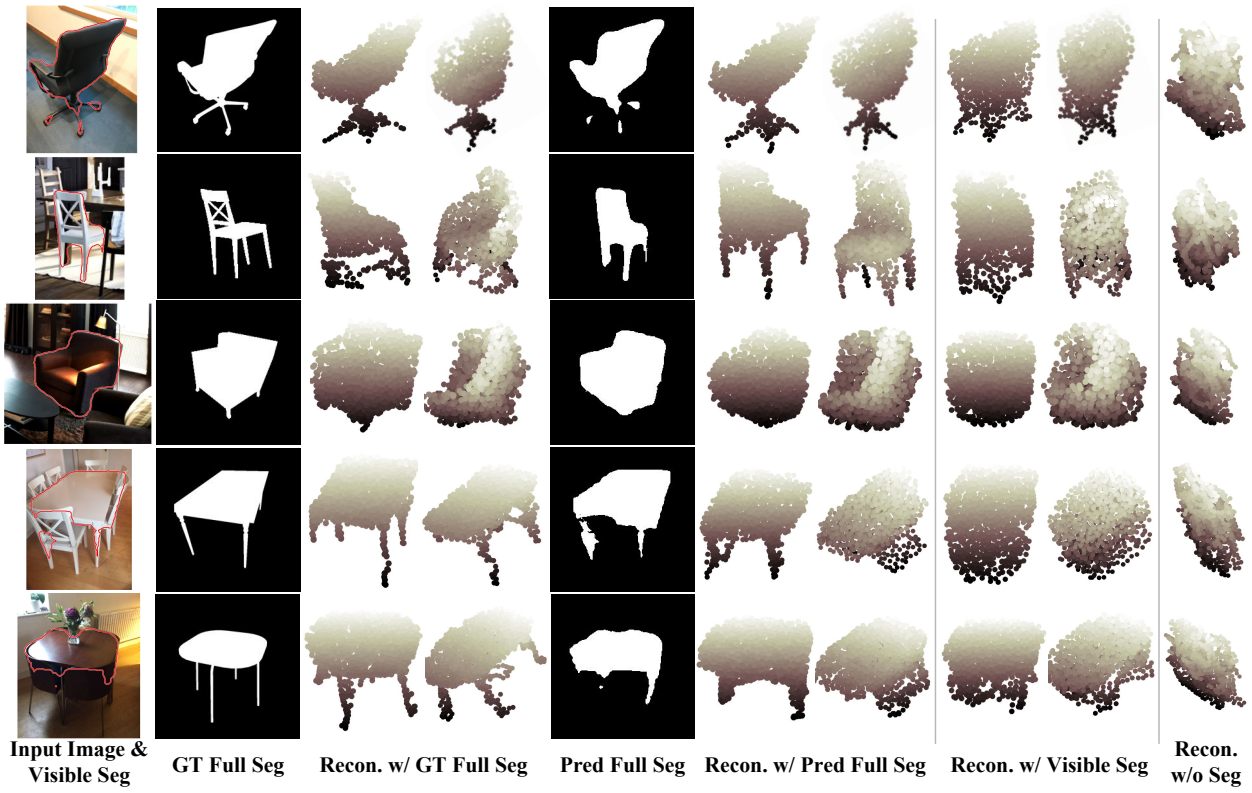


Figure 5.6: Qualitative performance on Pix3D dataset. We show in each row from left to right: input RGB image with predicted visible silhouette by Mask-RCNN (contoured with red), ground truth full silhouette, reconstruction guided by ground truth silhouette in two views (viewer-centered and a novel view), our predicted complete silhouette, reconstruction guided by predicted complete silhouette, reconstruction guided by visible silhouette from Mask-RCNN and reconstruction without silhouette guidance. The first row shows an non-occluded object, and other rows show occluded objects. In row two, The error prediction with ground truth silhouette is due to the railings on the chair back that confuses the network that sees it as a lounge chair. Overall, guided by ground truth silhouettes, our approach shows performance that is nearly as good for occluded as for non-occluded objects. Using the predicted silhouettes shows large improvements for both occluded and non-occluded objects. Using the visible silhouettes will fail on predicting occluded portion since it does not know the 3D shape behind. Without silhouette guidance the network meets the most difficulties. Best viewed in color.

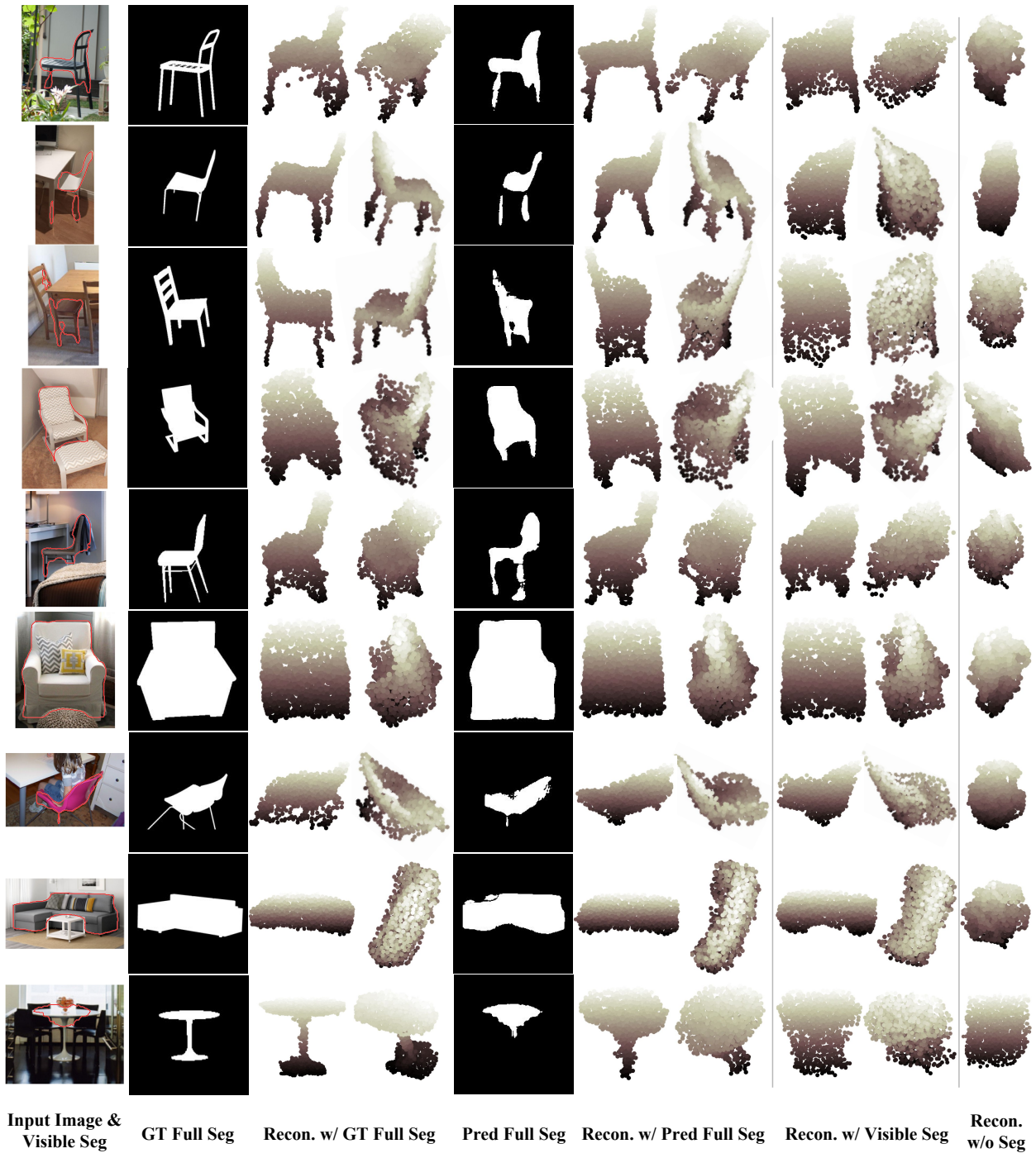


Figure 5.7: More qualitative results of occluded objects from Pix3D dataset. We show in each row from left to right: input RGB image with predicted visible silhouette by Mask-RCNN (contoured with red), ground truth full silhouette, reconstruction guided by ground truth silhouette in two views (viewer-centered and a novel view), our predicted complete silhouette, reconstruction guided by predicted complete silhouette, reconstruction guided by visible silhouette from Mask-RCNN and reconstruction without silhouette guidance. Best viewed in color.

Method	Occluded			Non-occluded		
	sofa	chair	table	sofa	chair	table
Mask-RCNN	84.34	59.05	58.32	91.99	**69.96**	64.90
Ours train w/ pred vis seg	85.24	56.95	58.33	91.24	68.55	**65.83**
Ours train w/ GT vis seg	**87.58**	**59.61**	**58.34**	**92.02**	69.88	64.46

Table 5.5: Object silhouette completion performance on Pix3D dataset. We report 2D IoU on the ground truth full silhouette of occluded and non-occluded objects. We compare with the Mask-RCNN baseline (detects visible region). We report our approach with two different training settings: trained with predicted visible silhouette from Mask-RCNN as input and trained with ground truth visible silhouette as input.

Method	CD			EMD		
	sofa	chair	table	sofa	chair	table
Ours w/o seg	15.54	17.96	24.35	16.63	16.51	22.56
Ours w/ pred vis seg	9.15	13.20	**17.96**	9.29	13.40	**17.81**
Ours w/ pred full seg	**9.07**	**13.13**	18.36	**9.26**	**13.20**	**17.81**
Ours w/ gt full seg	8.27	10.16	11.36	8.11	10.48	11.44

Table 5.6: Reconstruction of *occluded* objects on Pix3D dataset. We compare our approach with four different configurations: prediction without silhouette guidance, prediction with predicted visible silhouette from Mask-RCNN, prediction with our predicted full silhouette and prediction with ground truth complete silhouette.

and can even refine the silhouettes of non-occluded objects. Table class has the worse performed input visible silhouettes and meets the lowest improvement margin. The input visible silhouettes' quality also impose effects on the downstream reconstruction task.

Robustness to occlusion. To train the network, we construct a synthetic occlusion dataset (Chapter 5.3.2) based on ShapeNet and add real background from LSUN dataset. We then test our approach on Pix3D dataset.

For evaluation, we use the ground truth point cloud provided by Mandikal *et al*. [90]. Each ground truth shape has uniformly sampled 1024 points. Since Pix3D evaluates on object-centered reconstruction, while our approach is viewer-centered, we rotate each ground truth shape to have the viewer-centered orientation w.r.t. camera. For evaluation we re-scale both ground truth and our prediction to be zero-centered and unit-length, then perform ICP with translation only. In this case our reported performance can also reflect the performance of view-point estimation. We follow the officially provided evaluation metrics of Chamfer Distance and Earth mover's distance by Sun *et al*. [115].

We show in Figure 5.6 and Figure 5.7 our qualitative performance. Table 5.6 and Table 5.7 summarizes the quantitative performance on three object classes for occluded objects and non-

Method	CD			EMD		
	sofa	chair	table	sofa	chair	table
Ours w/o seg	12.62	16.00	20.66	13.19	15.44	19.93
Ours w/ pred vis seg	**8.75**	11.35	**15.56**	**8.67**	11.83	15.75
Ours w/ pred full seg	**8.75**	**10.91**	15.72	8.81	**11.34**	**15.66**
Ours w/ gt full seg	8.24	9.21	10.50	8.28	9.66	11.43

Table 5.7: Reconstruction of *non-occluded* objects on Pix3D. We compare our approach with four different configurations: prediction without silhouette guidance, prediction with predicted visible silhouette from Mask-RCNN, prediction with our predicted full silhouette and prediction with ground truth complete silhouette.

occluded objects respectively. We analyze the performance with four configurations: with ground truth complete silhouette, with our predicted complete silhouette, with predicted visible silhouette from Mask-RCNN and without silhouette guidance. We train the network having predicted visible silhouette as input with the ground truth visible silhouette generated from ShapeNet. We train the network having no silhouette guidance with RGB images as input only. All configurations use the same training settings and training losses. Using ground truth complete silhouette can make the network be robust to occlusion. Without silhouette guidance, the network meets a lot difficulties because it does not know whether or where an object is occluded and what to reconstruct. With the guidance of predicted complete silhouette, our approach is able to narrow down the performance gap between occluded and non-occluded objects, and slightly outperforms the approach with predicted visible silhouettes.

5.5 CONCLUSION

In this chapter, we propose a method to reconstruct the complete 3D shape of an object from a single RGB image, with robustness to occlusion. Our point cloud reconstruction approach achieves the state-of-the-art with the major improvement by the surface-based refinement step. We show that, when provided with input ground truth silhouettes, the shape prediction performance is nearly as good for occluded as for non-occluded objects. Using the predicted silhouette also yields large improvements for both occluded and non-occluded objects, indicating that providing an explicit foreground/background separation for the object in RGB images is helpful.

CHAPTER 6: GENERATING SHAPE PRIMITIVES WITH RECURRENT NEURAL NETWORKS

In this chapter, we explore the structural and abstract representation of 3D objects to support applications like robot grasping and manipulation. The success of various applications including robotics, digital content creation, and visualization demand a structured and abstract representation of the 3D world from limited sensor data. Inspired by the nature of human perception of 3D shapes as a collection of simple parts, we explore such an abstract shape representation based on primitives. Given a single depth image of an object, we present **3D-PRNN**, a generative recurrent neural network that synthesizes multiple plausible shapes composed of a set of primitives. Our generative model encodes symmetry characteristics of common man-made objects, preserves long-range structural coherence, and describes objects of varying complexity with a compact representation. We also propose a method based on Gaussian Fields to generate a large scale dataset of primitive-based shape representations to train our network. We evaluate our approach on a wide range of examples and show that it outperforms nearest-neighbor based shape retrieval methods and is on-par with voxel-based generative models while using a significantly reduced parameter space.

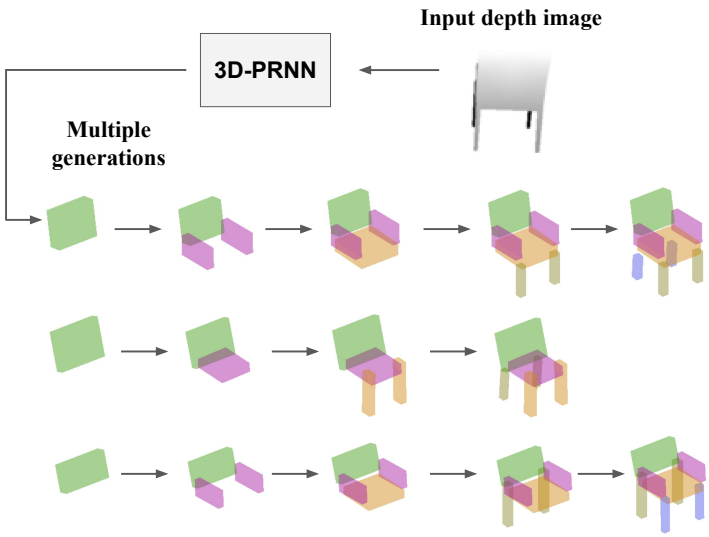


Figure 6.1: A step-by-step primitive-based shape generation by **3D-PRNN**. As an illustration, given single depth image, we sequentially predicts sets of primitives that form the shape. Each time we randomly sample one primitive from a set and generate the next set of primitives conditioning on the current sample. Better view in color.

6.1 INTRODUCTION

Many robotics and graphics applications require 3D interpretations of sensory data. For example, picking up a cup, moving a chair, predicting whether a stack of blocks will fall, or looking for keys on a messy desk all rely on at least a vague idea of object position, shape, contact and

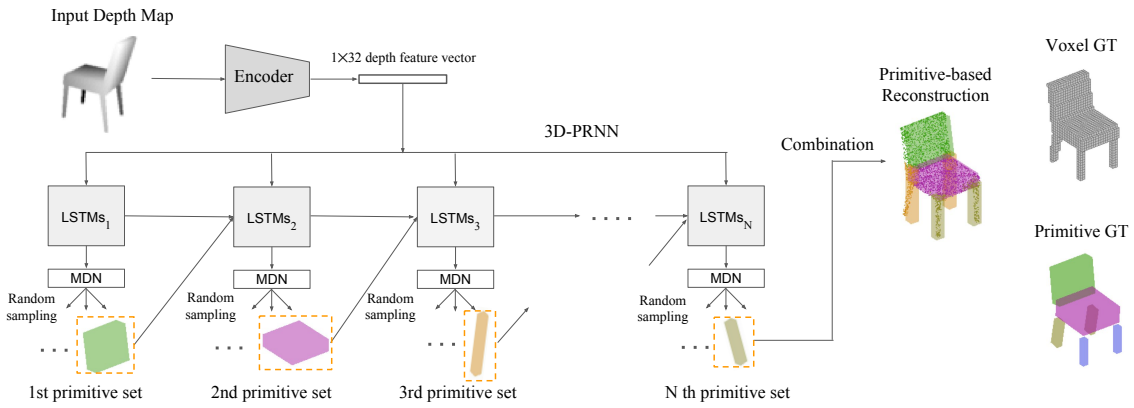


Figure 6.2: **3D-PRNN overview**. We illustrate the method on the task of single depth shape completion. The network starts from encoding the input depth image into a feature vector, which is then sent to the "recurrent generator" consisting stacks of Long Short-Term Memory (LSTM) and a Mixture Density Network (MDN). At each time step, the network predicts a set of primitives conditioned on both the depth feature and the previously sampled single primitive. The final reconstruction result and ground truth are shown on the right.

connectedness. A major challenge is how to represent 3D object geometry in a way that (1) can be predicted from noisy or partial observations; and (2) is useful for reasoning about contact, support, extent, and so on. Recent efforts often focus on voxelized volumetric representations (*e.g.* , [61, 67, 62, 64]). Instead, we propose to represent objects with 3D primitives (oriented 3D rectangles, i.e. cuboids). Compared with voxels, the primitives are much more *compact*, for example 45-D for 5 primitives parameterized by scale-rotation-translation vs 32,256-D for a 32x32x32 voxel grid. Also, primitives are *holistic* — representing an object with a few parts greatly simplifies reasoning about stability, connectedness, and other important properties. Primitive-based 3D object representations have long been popular in psychology (*e.g.* , "geons" by Biederman [94]) and interactive graphics (*e.g.* , "Teddy" [96]), but they are less commonly employed in modern computer vision due to the challenges of learning and predicting models that consist of an arbitrary number of parameterized components.

Our goal is to learn 3D primitive representations of objects from unannotated 3D meshes. We follow an encoder-decoder strategy, inspired by recent work [116, 117], using a recursive neural network (RNN) to encode an implicit shape representation and then sequentially generate primitives to approximate the shape as shown in Figure 6.1. One challenge in training such a primitive generation network is acquiring ground truth data for primitive-based shape representations. To address this challenge, we propose an efficient method based on Gaussian Fields and energy minimization [148] to iteratively parse shapes into primitive components. We optimize a differentiable loss function using robust techniques (L-BFGS[123]). We use this (unsupervised) optimization

process to create the primitive ground truth, solving for a set of primitives that approximates each 3D mesh in a collection. The RNN can then be trained to generate new primitive-based shapes that are representative of an object class' distribution or to complete an object's shape given a partial observation such as a depth image or point cloud.

To model shapes, we propose 3D-PRNN, an RNN-based generative model that predicts context-sensitive sequences of primitives in object-centric coordinates, as shown in Figure 3.2. To predict shape from depth, the network is trained jointly with the single depth image and a sequence of primitives configurations (shape, translation and rotation) that form the complete shape. During testing, the network gets input of a depth map and sequentially predicts primitives (ending with a stop signal) to reconstruct the shape. Our generative RNN architecture is based on a Long Short-Term Memory (LSTM) and a Mixture Density Network (MDN).

We evaluate our proposed generative model by comparing with baselines and the state-of-the-art methods. We show that, even though our method has less degrees of freedom in representation, it achieves comparable accuracy with voxel based reconstruction methods. We also show that encoding further symmetry and rotation axis constraints in our network significantly boosts performance.

6.2 FITTING PRIMITIVES FROM POINT CLOUDS

One challenge in training our 3D-PRNN primitive generation network is the lack of large scale ground truth primitive based shape reconstruction data. We propose an efficient method to generate such data. Given a point cloud representation of a shape, our approach finds the most plausible primitives to fit in a sequential manner, *e.g.* , given a table, the algorithm might identify the primitive that fits to the top surface first and then the legs successively. We use rectangular cuboids as primitives which provide a plausible abstraction for most man-made objects. Our method proposes a fast parsing solution to decompose shapes with varying complexity into a set of such primitives.

6.2.1 Primitive Fitness Energy

We formulate the successive fitting of primitives as an energy minimization scheme. While primitive fitting at each step resembles the method of Iterative Closest Point (ICP) [149], we have additional challenges. ICP ensures accurate registration when provided with a good initialization, but in our case we have no prior knowledge about the number and the rough shape of the primitives. Moreover, we need to solve the more challenging partial matching problem since each primitive matches only part of the shape, which we do not know in advance.

We represent the shape of each primitive with scale parameters $S = [s_x, s_y, s_z]$, which denotes the scale of a unit cube along three orthogonal axes. The position and orientation of the primitive are represented by translation, $T = [t_x, t_y, t_z]$, and Euler angles, $\theta = [\theta_x, \theta_y, \theta_z]$, respectively. Thus the primitive is parameterized by $x = [s_x, s_y, s_z, t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]$. Furthermore, we assume a fixed sampling of the unit cube into a set of points, $P = \{p_m\}_{m=1,\dots,M}$. Given a point cloud representation of a shape, $Q = \{q_n\}_{n=1,\dots,N}$, our goal is to find the set of primitives $X = \{x_t\}_{t=1,2,3,\dots}$ that best fit the shape. We employ the idea of Gaussian Force Fields [148] and Truncated Signed Distance Function (TSDF) [150] to formulate the following continuously differentiable energy function which is convex in a large neighborhood of the parameters:

$$E_p = -\sum_{m,n} V_p \min\left(\exp\left(-\frac{\|R(\theta)Sp_m + T - q_n\|^2}{\sigma^2}\right), \xi\right) \tag{6.1}$$

where $R(\theta)$ is the rotation matrix, ξ is the truncation parameter ($\xi = 0.9$ in our experiments) and V_p denotes the volumetric-wise sampling ratio that is calculated as the volume of primitive P over its number of sampled points M. V_p helps avoid local minimum that results in a too small or too large primitive. Our formulation represents the error as a smooth sum of Gaussian kernels, where far away point pairs are penalized less to account for partial matching.

The energy function given in Eq. 6.1 is sensitive to the parameter σ. A larger σ will encourage fitting of large primitives while allowing larger distances between matched point pairs. In order to prefer tighter fitting primitives, we introduce the concept of *negative shape*, Q^-, which is represented as a set of points sampled in the non-occupied space inside the bounding box of a shape. We update our energy function

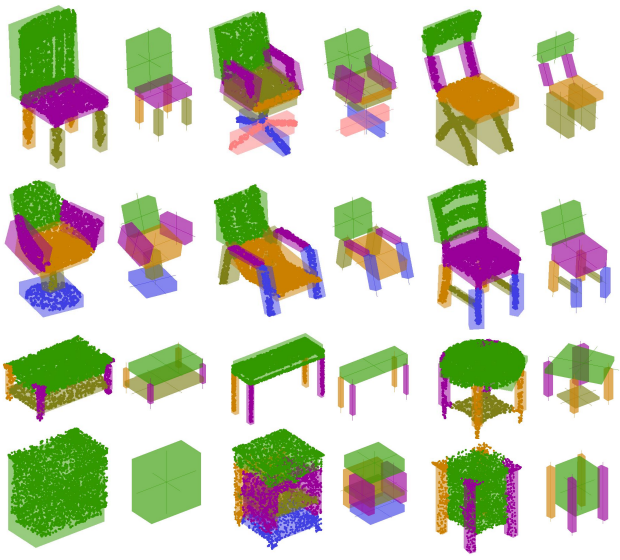


Figure 6.3: **Sample primitive fitting result**. We show our primitive fitting results on chairs, tables and nightstands. We overlay our fitted primitives on the sampled 3D point clouds of each shape.

as:

$$E_w = E_P^+ - \alpha E_P^-, \tag{6.2}$$

where E_P^+ is the fitting energy between the shape and the primitive and E_P^- is the fitting energy between the negative shape and the primitive. Given point samples, both E_P^+ and E_P^- are computed as in Eq. 6.1. α denotes the relative weighting of these two terms and is defined as $\alpha = \max\big(\min(10, |Q|/|Q^-| \times 5), 0.1\big)$.

6.2.2 Optimization

Given the energy formulation described in the previous section, we perform primitive fitting in a sequential manner. During each iteration, we randomly initialize 10 primitives, optimize Eq. 6.2 for each of these primitives and add the best fitting primitive to our primitive collection. We then remove the points in Q that are fit by the selected primitive and iterate. We stop once all the points in Q are fit by a primitive. We optimize Eq. 6.2 in an iterative manner. We first fix θ and solve for S and T, we then fix S and T and solve for θ. In our experiments this optimization converges in 5 iterations and we use the L-BFGS toolbox [151] at each optimization step. We summarize this process with the pseudo-code given in Alg. 6.2.

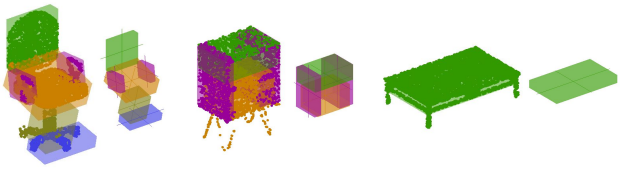


Figure 6.4: **Failure cases**. Main causes are : too complex shape details to be represented by primitive blocks (left), The smoothing property of Gaussian force fields is not good at describing small hollow shape (middle), small cluster of point clouds are easily missed through our randomized search scheme (middel and right).

Simplification with symmetry. We utilize the symmetry characteristics of man-made shapes to further speed up the primitive parsing procedure. We use axis-aligned 3D objects where symmetric objects have a common global symmetry plane. We compare the geometry on the two sides of this plane to decide whether an object is symmetric or not. Once we obtain a primitive that lies on one side of the symmetry plane, we automatically generate the symmetric primitive on the other side of the plane.

Refinement. At each step, we fit primitives with a relatively larger Gaussian field ($\sigma = 2$ in Eq. 6.1) for fast convergence and easier optimization. We then refine the fitting with a finer energy space ($\sigma = 0.5$) to match the primitive to the detailed shape of the object. While our random

Algorithm 6.2 Primitive fitting

Given shape point clouds Q and empty primitive set X;
$\beta = 0.97|Q|$, t = 0;
while $|Q| < \beta$ or $i <$maxPrimNum **do**
    $E_{best} =$ Inf;
    for $i = 1$:maxRandNum **do**
        $\theta = [0, 0, 0]$, random initialize S, T, $j = 0$;
        while $\delta < 0.01$ or $j <$maxIter **do**
            fix θ, solve $S, T \rightarrow S^*, T^*$ by Eq .6.1;
            fix S^*, T^*, update $\theta \rightarrow \theta^*$ by Eq .6.1, calculate $E_w(S^*, T^*, \theta^*)$ by Eq .6.1;
            if $E_w < E_{best}$ **then**
                $E_{best} = E_w$, $x_{best} = [S^*, T^*, \theta^*]$;
            $\delta = \|[S, T, \theta] - [S^*, T^*, \theta^*]\|^2$, $S = S^*$, $T = T_p^*$, $k = k + 1$;
    $x_t = x_{best}$, add x_t to X, $t = t + 1$; Remove fitted points from Q and add to non-occupied space Q^-
return X

search scheme enables a fast parsing method, errors may accumulate in the final set of primitives. To avoid such problems, we perform a post-refinement step. We refine the parameters of a single primitive x_t while fixing the other parameters. We use the parameters of x_t obtained from the initial fitting as initialization. We define the energy terms in Eq. 6.2 with respect to the points that are fit by x_t and the points that are not fit by any primitive yet. We note that this sequential refinement is similar to back propagation used to train neural networks. In our experiments, we perform the refinement each time we fit 3 new primitives.

6.3 3D-PRNN: 3D PRIMITIVE RECURRENT NEURAL NETWORKS

Generating primitive-based 3D shapes is a challenging task due to the complex multi-modal distribution of shapes and the unconstrained number of primitives required to model such complex shapes. We propose 3D-PRNN, a generative recurrent neural network to accomplish this task. 3D-PRNN can be trained to generate novel shapes both randomly and by conditioning on partial shape observations such as a single depth map.

6.3.1 Network architecture

An overview of the 3D-PRNN network is illustrated in Figure 3.2. The network gets as input a single depth image and sequentially predicts primitives to form a 3D shape. For each primitive, the network predicts its shape (height, length, width), position (i.e. translation), and orientation (i.e.

rotation). Additionally, at each step, a binary *end of generation* signal is predicted which indicates no more primitive should be generated.

Depth map encoder. Each input depth map, I, is first resized to be 64×64 in dimension with values in the range $[0, 1]$ (we set the value of background regions to 0). I is passed to an encoder which consists of stacks of convolutional and LeakyRelu [152] layers as shown in Figure 6.5 (a): the first layer has 32 kernels of size 7×7 and stride 2, with a LeakyRelu layer of 0.1 slope in the negative part. The second layer consists of 64 kernels of size 5×5 (stride 2), followed by the same setting of LeakyRelu and a max pooling layer. The third layer has 128 kernels of size 3×3 (stride 2) followed by LeakyRelu and max pooling. The next two fully-connected layers has neurons of 64 and 32. The output 1×32 feature vector d is then sent to the recurrent generator to predict a sequence of primitives.

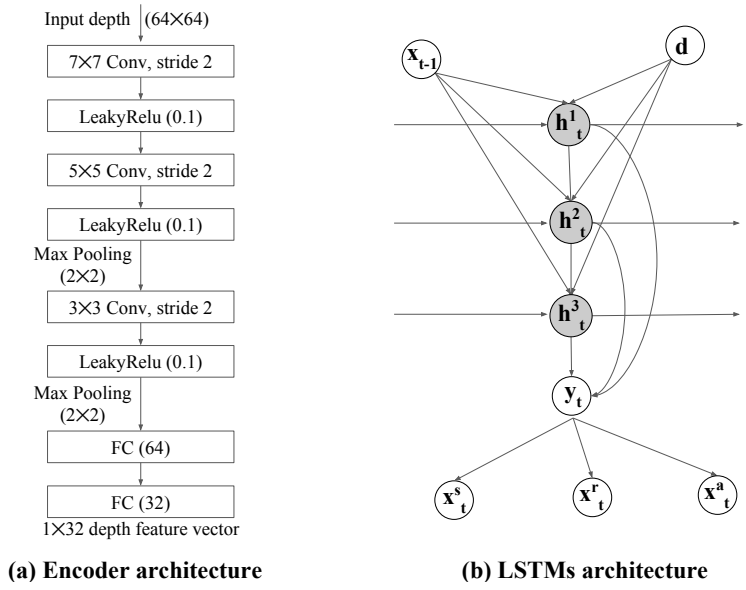


Figure 6.5: Detailed architectures of (a) the depth map encoder and (b) the primitive recurrent generator unit in **3D-PRNN**. See the architecture descriptions in Section 4.1.

Recurrent generator. We apply the Long Short-Term Memory (LSTM) unit inside the recurrent generator, which is shown to be better at alleviating the vanishing or exploding gradient problems [153] when training RNNs. The architectural design is shown in Figure 6.5 (b). The prediction unit consists of L layers of recurrently connected hidden layers (we set $L = 3$, which is found to be sufficient to model the complex primitive distributions) that encode both the depth feature d and the previously predicted primitive x_{t-1} and then computes the output vector, y_t. y_t is used to parametrize a predictive distribution $Pr(x_t|y_t)$ over the next possible primitive x_t. The hidden layer activations are computed by iterating over the following equations in the range $t = [1, T]$

and $l = [2, L]$:

$$z_t^l = W_x^l x_{t-1} + W_h^l h_{t-1}^l + W_c^l h_t^{l-1} + W_d^l d \tag{6.3}$$

$$[i_t^l, f_t^l, o_t^l] = \sigma(z_t^l) \tag{6.4}$$

$$g_t^l = \tanh(z_t^l) \tag{6.5}$$

$$c_t^l = f_t^l c_{t-1}^l + i_t^l g_t^l \tag{6.6}$$

$$h_t^l = o_t^l \tanh(c_t^l) \tag{6.7}$$

where z_t^l capsules the input features in the l-th layer (when $l = 1$, there is no hidden value propagated from the previous layers and thus $z_t^1 = W_x^1 x_t + W_h^1 h_{t-1}^1 + W_c^1 d$), h_t and c_t denote the hidden and cell states, whereas $W_x^l, W_h^l, W_c^l, W_d^l$ denote the linear weight matrix (we omit the bias term for brevity), i_t, f_t, o_t, g_t are respectively the input, forget, output, and context gates, which have the same dimension as the hidden states (size of 400). σ is the logistic sigmoid function and $\tanh$ is the hyperbolic tangent function.

At each time step t, the distribution of the next primitive is predicted as $y_t = W_y[h_t^1, h_t^2, ..., h_t^L]$, where we perform a linear transformation on the concatenation of all the hidden values. This concatenation is similar in spirit to using *skip connections* [154, 127], which is shown to help training and mitigate the vanishing gradient problem. In a similar fashion, we also pass the depth feature d to all the hidden layers. We will explain latter how the primitive configuration x_t is sampled from a distribution predicted from y_t.

We predict parameters of one axis per time conditioned on the previous axis. We model this joint distribution of parameter on each axis $x_i^s = [s_i, t_i]$ (where i indicates one of the 3 axes of space) as a mixture of Gaussians conditioned on previous axis with K mixture components:

$$\left(\{\pi_t^k, \mu_t^k, \sigma_t^k, \rho_t^k\}_{k=1}^K, e_t\right) = f(y_t), \tag{6.8}$$

where π_t, μ_t, σ_t and ρ_t are the weight, mean, standard deviation, and correlation of each mixture component respectively, predicted from a fully connected layer $f(y_t)$. Note that e_i is the binary stopping sign indicating whether the current primitive is the final one and it helps with predicting a variable-length sequence of primitives. In our experiments we set $K = 20$. We randomly sample a single instance $x_i^s = [s_i, t_i, e_i] \in \mathbb{R} \times \mathbb{R} \times \{0, 1\}$ drawn from the distribution $f(y_t)$. The sequence x_t represents the parameters in the following order: $x_x^s \rightarrow x_y^s \rightarrow x_z^s$ for the shape translation configuration on x, y, z axis of the first primitive and the stopping sign.

This is essentially a mixture density network (MDN) [155] on top of the LSTM output and its

loss is defined:

$$L_s(x) = \sum_{t=1}^{T} -\log\Big(\sum_k \pi_t^k N(x_{t+1}|\mu_t^k, \sigma_t^k, \rho_t^k)\Big) \\ -\mathbb{I}\big((x_{t+1})_3 = 1\big)\log e_t - \mathbb{I}\big((x_{t+1})_3 \neq 1\big)\log(1-e_t) \tag{6.9}$$

The MDN is trained by maximizing the log likelihood of ground truth primitive parameters in each time step, where we calculate gradients explicitly for backpropagation as shown by Graves [116]. We found this stepwise supervised training works well and avoids sequential sampling used in [93, 156].

Geometric constraints. Another challenge in predicting primitive-based shape is to model rotation, given that the rotation axis is sensitive to slight change in rotation values under Euler angles. We found that by jointly predicting the rotation axis x^a and the rotation value x^r, both the rotation prediction performs better and the overall primitive distribution modeling get alleviated as shown in Figure 6.6, quantitative experiments are in Chapter 6.4.3. The rotation axis (x^a) is predicted by a three-layered fully connected network $g(y_t)$ with size 64,32 and 1 and sigmoid function as shown in fig. 6.5. The rotation value (x^r) is predicted by a separate three-layered fully connected network $g^*(y_t)$ with size 64, 32 and 1 and a $Tanh(\cdot)$ function.

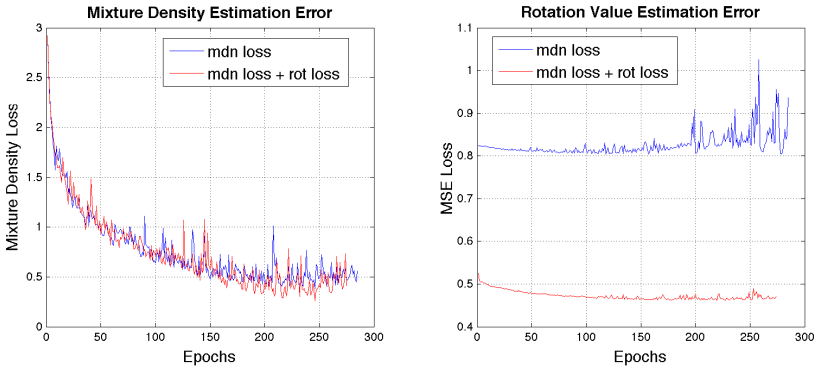


Figure 6.6: **Training performance comparison on validation set of synthetic depth map from ModelNet.** Both the mixture density loss and the rotation MSE loss are averaged by sequence length. The rotation values are normalized and values can have ranges around 13, compared with the < 1 MSE loss. Our mixture density estimation and rotation value estimation performs better by enforcing loss on predicting rotation axis.

6.3.2 Loss function

The overall sequence loss of our network is:

$$L(x) = L_s(x^s) + L_r(x^r) + L_a(x^a), \tag{6.10}$$

$$L_r(x) = \sum_t \|x_t^r - \hat{x_t^r}\|^2 \tag{6.11}$$

$$L_a(x) = \sum_t \|x_t^a - \hat{x_t^a}\|^2 \tag{6.12}$$

$L_s(x)$ is defined in Eq. 6.9. $L_r(x)$ is a mean square loss between predicted, x_t^r, and target, $\hat{x_t^r}$, rotation. $L_a(x)$ is the mean square loss between the predicted, x_t^a, and ground truth, $\hat{x_t^a}$, rotation axis.

6.4 EXPERIMENTS AND DISCUSSIONS

We show quantitative results on automatic shape synthesis. We quantitatively evaluate our 3D-PRNN in two tests: 1) 3D reconstruction on synthetic depth maps and 2) using real depth maps as input.

We train our 3D-PRNN on ModelNet [67] categories: 889 chairs, 392 tables and 200 night-stands. We employ the provided another 100 testing samples from each class for evaluation. We train a single network with all shapes classes jointly. In all experiments, to avoid overfitting, we hold out 15% of the training samples, which are then used to choose the number of training epochs. We then retrain the network using the entire training set. Since a single network is trained to encode all three classes, when predicting shape from depth images, for example, there is an implicit class prediction as well.

6.4.1 Implementation

We implement 3D-PRNN network using Torch. We train our network on primitive-based shape configurations generated as described in Chapter 6.2. The parameters of each primitive (i.e. shape, translation and rotation) are normalized to have zero mean and standard deviation. We observe that the order of the primitives generated by the method described in Chapter 6.2 involves too much randomness that makes training hard. Instead, we pre-sort the primitives based on the height of each shape center in a decreasing fashion. This simple sorting strategy significantly boosts the training performance. Additionally, our network is trained only on one side of the symmetric shapes to shorten the sequence length and speed up the training process. To train with the gen-

erative mechanism, we use simple random sampling technique. We use ADAM [157] to update network parameters with a learning rate of 0.001, $\alpha = 0.95$, and $\epsilon = e^{-6}$. We train the network with batch size 380 and 50 on the synthetic data and on the real data respectively.

At test time, the network takes a single depth map and sequentially generates primitives until a stop sign is predicted. To initialize the first RNN feature x, we perform a nearest neighbor query based on the encoded feature of the depth map to retrieve the most similar shape in the training set and use the configuration on its first primitive.

Sampling. Note that an unexpected sample that is far from the mean of the distribution will cause accumulated error to the following predictions, during testing each time we sample from the first two most possible mixture component, in training we still perform random sampling on all mixture components. This strategy improves stability of our network performance in synthetic data case. In the real data case, we found that applying random sampling among all mixture components during both training and testing time can produce successive reasonable shapes. This is due to the fact that the ground truth shapes in real data are of simple structures that is easier to model by the network.

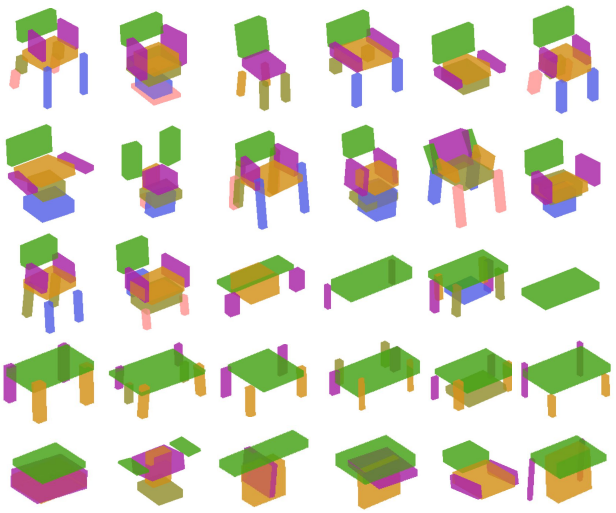


Figure 6.7: **Shape synthesis result**. We show various random sampled shapes by our 3D-PRNN. The network is trained and tested without context input. The coloring indicates the prediction order. Better view in color.

6.4.2 Shape synthesis

3D-PRNN can be trained to generate new primitive-based shapes. Figure 6.7 shows our randomly generated shapes synthesized from all three shape classes. We initialize the first RNN feature x with a random sampled primitive configuration from the training set. Since the first feature corresponds to "width", "translation in x-axis", and "rotation on x-axis" of the primitive, formally this initialization process is defined as drawing a sample from a discrete uniform distribution of these parameters where the discrete samples are constructed from the training examples. The figure shows that 3D-PRNN can learn to generate representative samples from multiple classes and sometimes creates hybrids from multiple classes.

	chair	table	night stand
GT prim	0.473	0.533	0.657
NN Baseline	**0.269**	0.220	0.256
Wu *et al*. [67] (mean)	0.253	0.250	**0.295**
3D-PRNN	0.245	0.188	0.204
3D-PRNN + rot loss	0.238	0.**263**	0.266

Table 6.1: 3D IoU evaluation in synthetic depth map in ModelNet. We explore two settings of 3D-PRNN w/ or w/o rotation axis constrains, and compare it with ground truth primitive and the nearest neighbor baseline. We also compare with the Wu *et al*. [67] deep network voxel generation method.

	chair	table	night stand
GT prim	0.049	0.044	0.044
NN baseline	0.075	0.089	0.100
Wu *et al*. [67] (mean)	**0.045**	**0.035**	**0.057**
3D-PRNN	0.074	0.080	0.104
3D-PRNN + rot loss	0.074	0.078	0.092

Table 6.2: Surface-to-surface distance evaluation in synthetic depth map in ModelNet. We explore two settings of 3D-PRNN w/ or w/o rotation axis constrains, and compare it with ground truth primitive and the nearest neighbor baseline.

6.4.3 Shape reconstruction from single depth view

Synthetic data. We project synthetic depth maps from training meshes. For both training and testing, we perform rejection-sampling on a unit sphere for 5 views, bounded within 20 degrees of the equator. The complete 3D shape is then predicted using a single depth map as input to 3D-PRNN. Our model can generate a sampling of complete shapes that match the input depth, as well as the most likely configuration, determined as the mean of the Gaussian from the most probable mixture. We report 3D intersection over union (IoU) and surface-to-surface distance [66] of the most likely predicted shape to the ground truth mesh. To compute IoU, the ground truth mesh is voxelized to 30 x 30 x 30 resolution, and IoU is calculated based on whether the voxel centers fall inside the predicted primitives or not. Surface-to-surface distance is computed using 5,000 points sampled on the primitive and ground truth surfaces, and the distance is normalized by the diameter of a sphere tightly fit to the ground truth mesh (*e.g.* , 0.05 is 5% of object maximum dimension).

We show qualitative results in Figure 6.8. Tables 6.1 and 6.2 show our quantitative results. "GT prim" is the ground truth primitive representation generated by our parsing optimization method during training. This serves as an upper bound on performance by our method, corresponding to how well the primitive model can fit the true meshes. "NN Baseline" is the nearest neighbor retrieval of shape in training set based on the embedded depth feature from our network. By enforcing rotation axis constraints ("3D-PRNN + rot loss"), our 3D-PRNN achieves better performance, which conforms with the learning curve as shown in Figure 6.6. Though both nearest neighbor and 3D-PRNN are based on the trained encoding, 3D-PRNN outperforms NN Baseline for table and nightstand, likely because it is able to generate a greater diversity of shapes from limited training data. We compare with the voxel-based reconstruction of Wu *et al*. [67], training

Figure 6.8: **Sample reconstruction from synthetic data from ShapeNet**. We show the input depth map, with the most probable shape reconstruction from 3D-PRNN, and three successive random sampling results, compared with our ground truth primitive representation. Better view in color.

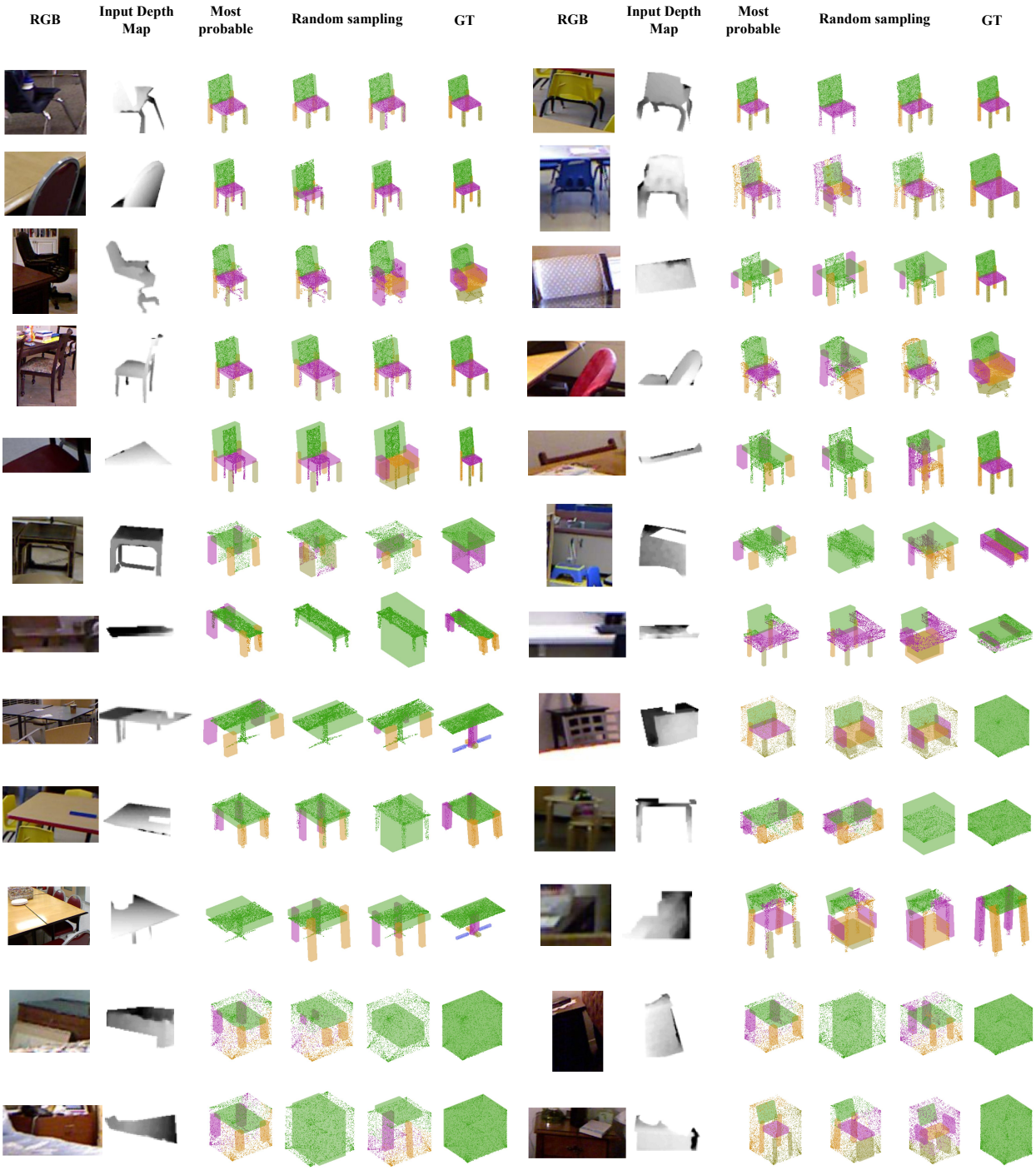


Figure 6.9: **Sample reconstruction from real depth map in NYUdv2**. We show the input depth map, with the most probable shape reconstruction from 3D-PRNN, and two successive random sampling results, compared with our ground truth primitive representation. Better view in color

class	chair	table	night stand
GT prim	0.543	0.435	0.892
NN baseline +ft	**0.171**	**0.078**	**0.286**
NN baseline	0.145	0.076	0.262
3D-PRNN +ft	0.158	0.075	0.081
3D-PRNN	0.138	0.052	0.086

Table 6.3: 3D IoU evaluation in real depth map in NYUd v2. We explore two settings of 3D-PRNN w/ (+ft) or w/o fine-tuning, and compare it with ground truth primitive and the nearest neighbor baseline.

class	chair	table	night stand
GT prim	0.037	0.048	0.020
NN baseline+ft	0.118	0.176	0.162
NN baseline	**0.101**	**0.164**	**0.160**
3D-PRNN+ft	0.112	0.168	0.192
3D-PRNN	0.110	0.181	0.194

Table 6.4: Surface-to-surface distance evaluation in real depth map in NYUd v2. We explore two settings of 3D-PRNN w/ (+ft) or w/o fine-tuning, and compare it with ground truth primitive and the nearest neighbor baseline.

and testing their method on the same data using publicly available code. Since Wu *et al*.generate randomized results, we measure the average result over ten runs. Our method performs similarly to Wu *et al*. [67] on the IoU measure. Wu *et al*.performs better on surface distance, which is less sensitive to alignment but more sensitive to details in structures. The performance of our ground truth primitives confirms that much of our reduced performance in surface distance is due to using a coarser abstraction (which though not preserving surface detail has other benefits, as discussed in introduction).

Real data (NYU Depth V2). We also test our model on NYU Depth V2 dataset [129] which is much harder than synthetic due to limited training data and the fact that depth images of objects are in lower resolution, noisy, and often occluded conditions. We employ the ground truth data labelled by Guo and Hoiem [158], where 30 models are manually selected to represent 6 categories of common furniture: chair, table, desk, bed, bookshelf and sofa. We fine-tune our network that was trained on synthetic data using the training set of NYU Depth V2. We report results on test set based on the same evaluation metric as the synthetic test shown in Table 6.3 and 6.4. Since nightstand is less common in the training set and often occluded depth regions may be similar to those for tables, the network often predicts primitives in the shapes of tables or chairs for nightstands, resulting in worse performance for that class. Sample qualitative results are shown in Figure 6.9.

6.5 CONCLUSION

In this chapter, we present 3D-PRNN, a generative recurrent neural network that uses recurring primitive based abstractions for shape synthesis. 3D-PRNN models complex shapes with a low parametric model, which advantages such as being capable of modeling shapes with fewer

training examples available, and a large intra- and inter-class variance. Evaluations on synthetic and real depth map reconstruction tasks show that results comparable to higher degree of freedom representations can be achieved with our method. Future explorations include allowing various primitive configurations beyond cuboids (i.e. cylinders or spheres), encoding explicit joints and spatial relationship between primitives.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 SUMMARY

In this thesis, we focus on 3D scene and object parsing from a single image. Our goal is to have an expressive 3D parse that is able to support applications such as robotics and navigation. To achieve this goal, we proposed approaches at three different levels of details. We start from the layout level to a more complex scene level and finally to the detailed object level. At layout level, in Chapter 3, we estimate the 3D room layout from a single RGB image. We propose LayoutNet, an approach that predicts room layout from a single image that generalizes across panoramas and perspective images, cuboid layouts and more general layouts (*e.g.* , "L"-shape room). We further improve upon LayoutNet by using a better image feature encoder, refined training details and a gradient ascent based post refinement step (named as LayoutNet v2). Next, in Chapter 4, at scene level, we make use of an additional depth image to explore a data-driven approach that recovers the complete 3D parse of all layouts and objects in the scene. Our approach generates sets of potential shapes, transfers and aligns associated 3D models while encouraging fit to observations and spatial consistency. At object level, in Chapter 5, we propose to recover the complete 3D geometry of an object with robustness to possible partial foreground occlusions, by completing the silhouette of the occluded region and then estimating the complete shape. Finally, in Chapter 6, we represent each shape as a 3D composite of a set of primitives, recurrently parsing each shape into primitives from a single depth view. We demonstrate the efficacy of each proposed approach with extensive experiments both quantitatively and qualitatively on public datasets.

7.2 FUTURE WORK

Our proposed approaches only make a small step toward an ideal 3D scene and object parsing. That is, much remains to be done: (1) for precise robot gesturing, a more detailed 3D object representation beyond oriented cuboid is needed; (2) for a free indoor navigation, a complete recovery of the 3D scene from multiple view is preferred; (3) for precise object localization, reasoning like object support and contact is needed to make up missing reconstruction due to occlusions or limited field of view; (4) to better assist humans, we need to learn to infer human-object and human-scene interactions. We discuss in details the above four directions of future work as follows.

Detailed shape modeling via structured representations. Besides the abstract primitive based 3D shape representation, many recent approaches try to build up finer structured shape representations like superquadrics [159] or implicit surfaces [160]. However, those templates are still

too abstract to preserve detailed shape properties. On the other hand, various approaches seek to represent 3D surfaces as the continuous decision boundary of a deep neural network classifier [161, 162], while treating a shape as a compact object rather than a composite of meaningful parts. Therefore, one future direction of our research is to investigate a part-based shape representation that is precise in surface modeling and can reasonably depict connectedness between parts.

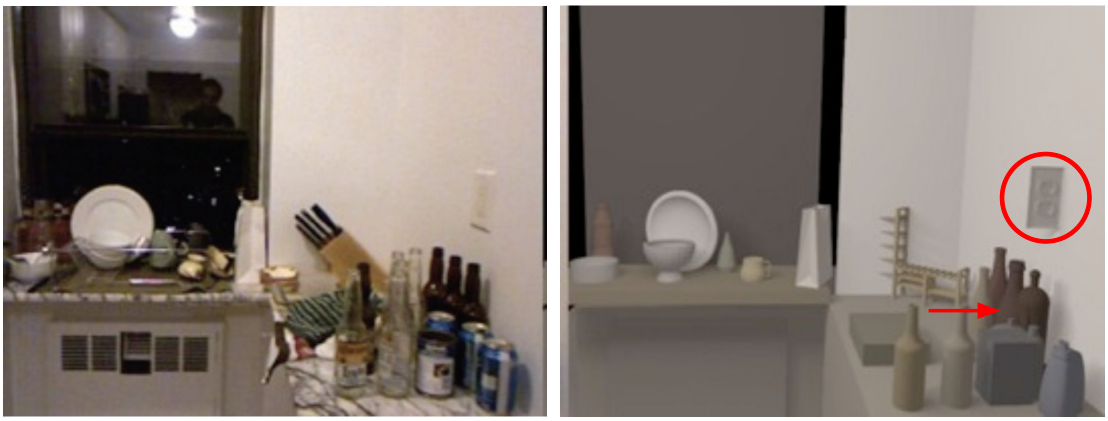


Figure 7.1: Illustrations of how occlusions (red arrow) and sensor errors (red circle) affect object localization and pose estimation. Left is a single RGB image, right is a complete 3D scene parsing of all objects and layouts, given a single RGBD image and the ground truth 2D semantic segmentation. However, we still have a wrong pose estimation of the outlet on the right wall due to missing depth values, and the wine bottles on the counter are hard to be precisely localized due to foreground occlusions.

Support reasoning for better 3D scene parsing. Object localization and pose estimation often meets obstacles due to occlusions and sensor error. As in Fig 7.1, the outlet on the wall has a wrong pose estimation due to the missing depth values from a Kinect sensor. The bottom extent of the wine bottles (red arrow) are unknown due to foreground occlusions. In this case, if we could incorporate reasoning like object support and contact, we can have a better 3D parse: the outlet should be supported by the right wall, and the bottom of the wine bottles should touch the counter top to be physically stable.

From single-view to multi-view 3D scene and object parsing. Information from a single image is limited: humans tend to walk around the scene to understand the free space around and to better localize a certain object. This requires a complete 3D parse from multiple view points. One compelling direction is to incorporate single-view 3D parsing into multi-view reconstruction approaches like SLAM [163] or SFM [164]. Recently, Phalak *et al*. [165] has shown promising results by fusing single image reconstructions from multiple perspective images to estimate a complete 3D floor plan. While challenges still exist: *e.g.* , how to sort out reliable reconstructions from

each image and discard unknown error predictions.

Modeling human-object and human-scene interactions. Another important aspect of 3D scene and object parsing is to interpret the interactions with humans: how can we move and interact with the cup on the desk? Where can we place the cup so that it's convenient for humans to pick it up? Moreover, we want to know if the inferred human-object and human-scene interactions can further improve the interpretations of both humans (*e.g.* , pose and shape) and the scene (more accurate 3D reconstruction). The modeling of the interactions is hard, due to the non-rigid property of human shapes, the large variations of poses interacted with the scene and the complexity of the 3D scene itself due to clutters of objects. One feasible solution is to use parametric models to represent both humans and the scene. The interactions can then be formulated as a function with constraints: *e.g.* , the fingers should surround the handle of the cup.

7.3 CONCLUSION

In this dissertation, we have taken a step toward 3D scene and object parsing from a single image. We hope that our investigations can inspire researchers to develop more advanced 3D scene and object parsing approaches, in the context of better 3D reconstruction beyond occlusion and sensor limitation, and moreover, perceiving human interactions with the scene.

In conclusion, we put forward the hypothesis and prove in experiments, that a data-driven approach is able to directly produce a complete 3D recovery from 2D partial observations. Moreover, we show that by imposing constraints of 3D patterns and priors into the learned model (e.g., layout surfaces are flat and orthogonal to each other, support height can infer the full extent of an occluded object, 2D complete silhouettes can guide reconstructions beyond partial foreground occlusions, and shapes can be decomposed into sets of primitives), we are able to obtain more accurate reconstruction of the scene and a structural representation of the object.

REFERENCES

[1] A. Criminisi, M. Kemp, and A. Zisserman, "Bringing pictorial space to life: computer techniques for the analysis of paintings," *Digital art history: A subject in transition*, vol. 1, p. 5, 2005.

[2] J. Delgado-Galvan, A. Navarro-Ramirez, J. Nunez-Varela, C. Puente-Montejano, and F. Martinez-Perez, "Vision-based humanoid robot navigation in a featureless environment," in *Mexican Conference on Pattern Recognition*. Springer, 2015, pp. 169–178.

[3] W. Yan, "Indoor vision-based robot navigation: a neural probabilistic approach," 2016.

[4] D. Riafio et al., "Object detection methods for robot grasping: Experimental assessment and tuning," in *Artificial Intelligence Research and Development: Proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence*, vol. 248. IOS Press, 2012, p. 123.

[5] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *ICCV*, vol. 2. IEEE, 1999, pp. 941–947.

[6] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2051–2059.

[7] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv:1702.01105*, 2017.

[8] C. Zou, R. Guo, Z. Li, and D. Hoiem, "Complete 3d scene parsing from an rgbd image," *International Journal of Computer Vision*, vol. 127, no. 2, pp. 143–162, 2019.

[9] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, "3d-prnn: Generating shape primitives with recurrent neural networks," in *Proc. ICCV*, vol. 2, 2017.

[10] L. G. Roberts, "Machine perception of 3-D solids," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[11] V. Hedau, D. Hoiem, and D. Forsyth, "Thinking inside the box: Using appearance models and context based on room geometry," *ECCV*, pp. 224–237, 2010.

[12] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Efficient structured prediction for 3d indoor scene understanding," in *CVPR*, 2012, pp. 2815–2822.

[13] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun, "Box in the box: Joint 3d layout and object reasoning from single images," in *ICCV*, 2013, pp. 353–360.

[14] J. Zhang, C. Kan, A. G. Schwing, and R. Urtasun, "Estimating the 3d layout of indoor scenes and its clutter from depth sensors," in *ICCV*, 2013, pp. 1273–1280.

[15] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *ICCV*, 2009.

[16] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2807–2814.

[17] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *CVPR*, vol. 2. IEEE, 2006, pp. 2418–2428.

[18] D. C. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *CVPR*. IEEE, 2009, pp. 2136–2143.

[19] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in *ICCV*, vol. 1. IEEE, 2005, pp. 654–661.

[20] A. G. Schwing and R. Urtasun, "Efficient exact inference for 3d indoor scene understanding," in *ECCV*, 2012, pp. 299–313.

[21] S. Ramalingam, J. K. Pillai, A. Jain, and Y. Taguchi, "Manhattan junction catalogue for spatial reasoning of indoor scenes," in *CVPR*, 2013, pp. 3065–3072.

[22] D. Lee, A. Gupta, M. Hebert, and T. Kanade, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," in *NIPS*, 2010, pp. 1288–1296.

[23] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard, "Bayesian geometric modeling of indoor scenes," in *CVPR*, 2012, pp. 2719–2726.

[24] L. Del Pero, J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard, "Understanding bayesian rooms using composite 3d object models," in *CVPR*, 2013, pp. 153–160.

[25] Y. Zhao and S.-C. Zhu, "Scene parsing by integrating function, geometry and appearance models," in *CVPR*, 2013, pp. 3119–3126.

[26] S. Dasgupta, K. Fang, K. Chen, and S. Savarese, "Delay: Robust spatial layout estimation for cluttered indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 616–624.

[27] H. Izadinia, Q. Shan, and S. M. Seitz, "IM2CAD," in *CVPR*, 2017.

[28] A. Mallya and S. Lazebnik, "Learning informative edge maps for indoor scene layout prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 936–944.

[29] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich, "Roomnet: End-to-end room layout estimation," *arXiv:1703.06241*, 2017.

[30] Y. Ren, C. Chen, S. Li, and C. J. Kuo, "A coarse-to-fine indoor layout estimation (CFILE) method," *arXiv:1607.00598*, 2016.

[31] Y. Zhang, S. Song, P. Tan, and J. Xiao, "Panocontext: A whole-room 3d context model for panoramic scene understanding," in *European Conference on Computer Vision*. Springer, 2014, pp. 668–686.

[32] H. Yang and H. Zhang, "Efficient 3d room shape recovery from a single panorama," in *CVPR*, 2016.

[33] J. Xu, B. Stenger, T. Kerola, and T. Tung, "Pano2CAD: Room layout from a single panorama image," *WACV*, pp. 354–362, 2017.

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

[35] R. Guo, C. Zou, and D. Hoiem, "Predicting complete 3d models of indoor scenes," *arXiv preprint arXiv:1504.02437*, 2015.

[36] C. Liu, P. Kohli, and Y. Furukawa, "Layered scene decomposition via the occlusion-crf," in *CVPR*, 2016, pp. 165–173.

[37] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *ICCV*, 2009.

[38] R. Cabral and Y. Furukawa, "Piecewise planar and compact floorplan reconstruction from images," in *CVPR*, 2014, pp. 628–635.

[39] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," in *ECCV*, 2012.

[40] C. Liu, A. G. Schwing, K. Kundu, R. Urtasun, and S. Fidler, "Rent3d: Floor-plan priors for monocular layout estimation," in *CVPR*, 2015, pp. 3413–3421.

[41] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3d geometric phrases," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 33–40.

[42] D. Lin, S. Fidler, and R. Urtasun, "Holistic scene understanding for 3d object detection with rgbd cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1417–1424.

[43] S. Huang, S. Qi, Y. Xiao, Y. Zhu, Y. N. Wu, and S.-C. Zhu, "Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation," in *Advances in Neural Information Processing Systems*, 2018, pp. 207–218.

[44] Z. Ren and E. B. Sudderth, "Three-dimensional object detection and layout prediction using clouds of oriented gradients," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1525–1533.

[45] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *European conference on computer vision*. Springer, 2014, pp. 634–651.

[46] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[47] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[48] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu, "Holistic 3d scene parsing and reconstruction from a single rgb image," *arXiv preprint arXiv:1808.02201*, 2018.

[49] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3–15, 2008.

[50] S. Walk, K. Schindler, and B. Schiele, "Disparity statistics for pedestrian detection: Combining appearance, motion and stereo," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 182–195.

[51] D. Lin, S. Fidler, and R. Urtasun, "Holistic scene understanding for 3d object detection with rgbd cameras," in *ICCV*, 2013.

[52] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Aligning 3d models to rgb-d images of cluttered scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4731–4740.

[53] Z. Ren and E. B. Sudderth, "3d object detection with latent support surfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 937–946.

[54] Z. Deng, S. Todorovic, and L. Jan Latecki, "Semantic segmentation of rgbd images with mutex constraints," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1733–1741.

[55] R. Guo and D. Hoiem, "Support surface prediction in indoor scenes," in *ICCV*, 2013.

[56] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–67.

[57] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "Single image 3d interpreter network," in *ECCV*, 2016, pp. 365–382.

[58] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive o-cnn: a patch-based deep representation of 3d shapes," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 217.

[59] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224.

[60] D. Shin, C. C. Fowlkes, and D. Hoiem, "Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3061–3069.

[61] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *NIPS*, 2016, pp. 1696–1704.

[62] R. Girdhar, D. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, 2016.

[63] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *IEEE CVPR*, 2014, pp. 3762–3769.

[64] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*, 2016, pp. 628–644.

[65] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, "Dense 3d object reconstruction from a single depth view," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[66] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem, "Completing 3d object shape from one depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2484–2493.

[67] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[68] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5431–5440.

[69] S. Fowler, H. Kim, and A. Hilton, "Towards complete scene reconstruction from single-view depth and human motion," in *Proceedings of the 28th British Machine Vision Conference (BMVC 2017)*, 2017.

[70] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation," *International Journal of Computer Vision*, vol. 112, no. 2, pp. 133–149, 2015.

[71] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2d-3d alignment via surface normal prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5965–5974.

[72] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing ikea objects: Fine pose estimation," in *IEEE CVPR*, 2013, pp. 2992–2999.

[73] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh, "3d object manipulation in a single photograph using stock 3d models," *ACM TOG*, vol. 33, no. 4, p. 127, 2014.

[74] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1966–1974.

[75] C. Kong, C.-H. Lin, and S. Lucey, "Using locally corresponding cad models for dense 3d reconstructions from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4857–4865.

[76] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5 d sketches," in *Advances in neural information processing systems*, 2017, pp. 540–550.

[77] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: Extracting editable objects from a single photo," *ACM TOG*, vol. 32, no. 6, p. 195, 2013.

[78] C. Kurz, X. Wu, M. Wand, T. Thormählen, P. Kohli, and H.-P. Seidel, "Symmetry-aware template deformation and fitting," in *CGF*, vol. 33, no. 6, 2014, pp. 205–219.

[79] V. Vaish, M. Levoy, R. Szeliski, C. L. Zitnick, and S. B. Kang, "Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2331–2338.

[80] P. Favaro and S. Soatto, "Seeing beyond occlusions (and other marvels of a finite lens aperture)," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2. IEEE, 2003, pp. II–II.

[81] A. O. Ulusoy, M. J. Black, and A. Geiger, "Semantic multi-view stereo: Jointly estimating objects and voxels," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4531–4540.

[82] L. Guan, J.-S. Franco, and M. Pollefeys, "Multi-view occlusion reasoning for probabilistic silhouette-based dynamic scene reconstruction," *International journal of computer vision*, vol. 90, no. 3, pp. 283–303, 2010.

[83] S. B. Kang, R. Szeliski, and J. Chai, "Handling occlusions in dense multi-view stereo," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[84] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 600–608.

[85] R. Guo and D. Hoiem, "Labeling complete surfaces in scene understanding," *International Journal of Computer Vision*, vol. 112, no. 2, pp. 172–187, 2015.

[86] K. Ehsani, R. Mottaghi, and A. Farhadi, "Segan: Segmenting and generating the invisible," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6144–6153.

[87] Y. Zhu, Y. Tian, D. Metaxas, and P. Dollár, "Semantic amodal segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1464–1472.

[88] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[89] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.

[90] P. Mandikal, K. L. Navaneet, M. Agarwal, and R. V. Babu, "3D-LMNet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[91] L. Jiang, S. Shi, X. Qi, and J. Jia, "Gal: Geometric adversarial loss for single-view 3d-object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 802–816.

[92] X. Zhang, Z. Zhang, C. Zhang, J. Tenenbaum, B. Freeman, and J. Wu, "Learning to reconstruct shapes from unseen classes," in *Advances in Neural Information Processing Systems*, 2018, pp. 2257–2268.

[93] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," *arXiv preprint arXiv:1612.00404*, 2016.

[94] I. Biederman, "Recognition-by-components: a theory of human image understanding." *Psychological review*, vol. 94, no. 2, p. 115, 1987.

[95] S. J. Dickinson, A. Rosenfeld, and A. P. Pentland, "Primitive-based shape modeling and recognition," in *Visual Form*, 1992, pp. 213–229.

[96] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *ACM SIGGRAPH '99*, pp. 409–416.

[97] R. Schnabel, P. Degener, and R. Klein, "Completion and reconstruction with primitive shapes," in *CGF*, vol. 28, no. 2, 2009, pp. 503–512.

[98] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra, "Globfit: Consistently fitting primitives by discovering global relations," in *ACM TOG*, vol. 30, no. 4, 2011, p. 52.

[99] R. Schnabel, "Efficient point-cloud processing with primitive shapes." Ph.D. dissertation, University of Bonn, 2010.

[100] A.-L. Chauve, P. Labatut, and J.-P. Pons, "Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data," in *IEEE CVPR*, 2010, pp. 1261–1268.

[101] F. Lafarge and P. Alliez, "Surface reconstruction through point set structuring," in *Computer Graphics Forum*, vol. 32, 2013, pp. 225–234.

[102] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool, "Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels," in *IEEE CVPR*, 2014, pp. 469–476.

[103] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," in *ACM TOG*, vol. 27, no. 5, 2008, p. 159.

[104] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

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

[106] D. Banica and C. Sminchisescu, "Cpmc-3d-o2p: Semantic segmentation of rgb-d images using cpmc and second order pooling," *CoRR*, 2013.

[107] J. Carreira and C. Sminchisescu, "Cpmc: Automatic object segmentation using constrained parametric min-cuts," *PAMI*, vol. 34, no. 7, pp. 1312–1328, 2012.

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

[109] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prim's algorithm," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2536–2543.

[110] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels," in *ECCV*, 2010.

[111] K. Karsch, C. Liu, and S. B. Kang, "Depth extraction from video using non-parametric sampling," in *ECCV*, 2012.

[112] R. Guo and D. Hoiem, "Beyond the line of sight: Labeling the underlying surfaces," in *ECCV*, 2012.

[113] K. Yamaguchi, M. H. Kiapour, and T. L. Berg, "Paper doll parsing: Retrieving similar styles to parse clothing items," in *ICCV*, 2013.

[114] S. Satkin and M. Hebert, "3dnn: Viewpoint invariant 3d geometry matching for scene understanding," in *ICCV*, 2013.

[115] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2974–2983.

[116] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[117] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, 2016. [Online]. Available: http://jmlr.org/proceedings/papers/v48/oord16.html pp. 1747–1756.

[118] G. Randall, J. Jakubowicz, R. G. von Gioi, and J.-M. Morel, "Lsd: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 32, pp. 722–732, 2008.

[119] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241.

[120] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[121] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.

[122] D. Farin, W. Effelsberg et al., "Floor-plan reconstruction from panoramic images," in *ACM Multimedia*. ACM, 2007, pp. 823–826.

[123] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[124] C. Fernandez-Labrador, J. M. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. J. Guerrero, "Corners for layout: End-to-end layout recovery from 360 images," *arXiv preprint arXiv:1903.08094*, 2019.

[125] S.-T. Yang, F.-E. Wang, C.-H. Peng, P. Wonka, M. Sun, and H.-K. Chu, "Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama," *arXiv preprint arXiv:1811.11977*, 2018.

[126] C. Sun, C.-W. Hsiao, M. Sun, and H.-T. Chen, "Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation," *arXiv preprint arXiv:1901.03861*, 2019.

[127] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[128] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[129] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[130] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[131] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[132] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Computer vision–ECCV 2014*. Springer, 2014, pp. 297–312.

[133] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2011, pp. 247–256.

[134] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[135] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik, "A multi-view embedding space for internet images, tags, and their semantics," *IJCV*, 2013.

[136] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite." in *CVPR*, vol. 5, 2015, p. 6.

[137] R. Urtasun, R. Fergus, D. Hoiem, A. Torralba, A. Geiger, P. Lenz, N. Silberman, J. Xiao, and S. Fidler, "Reconstruction meets recognition challenge," 2013.

[138] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[139] J. Zhang, C. Kan, A. G. Schwing, and R. Urtasun, "Estimating the 3d layout of indoor scenes and its clutter from depth sensors," in *ICCV*, 2013.

[140] K. Karsch, Z. Liao, J. Rock, J. T. Barron, and D. Hoiem, "Boundary cues for 3d object shape recovery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2163–2170.

[141] J. T. Barron and J. Malik, "Color constancy, intrinsic images, and shape estimation," in *European Conference on Computer Vision*. Springer, 2012, pp. 57–70.

[142] J. J. Koenderink, "What does the occluding contour tell us about solid shape?" *Perception*, vol. 13, no. 3, pp. 321–330, 1984.

[143] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737.

[144] S. Fuhrmann and M. Goesele, "Floating scale surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 4, p. 46, 2014.

[145] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[146] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. Citeseer, 1999, pp. 317–324.

[147] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[148] F. Boughorbel, M. Mercimek, A. Koschan, and M. Abidi, "A new method for the registration of three-dimensional point-sets: The gaussian fields framework," *Image and Vision Computing*, vol. 28, no. 1, pp. 124–137, 2010.

[149] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[150] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE ISMAR*, 2011, pp. 127–136.

[151] M. Schmidt, "minfunc: unconstrained differentiable multivariate optimization in matlab," *URL http://www.di.ens.fr/mschmidt/Software/minFunc.html*, 2012.

[152] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, vol. 30, no. 1, 2013.

[153] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." *ICML*, vol. 28, pp. 1310–1318, 2013.

[154] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *NIPS*, 2015, pp. 2377–2385.

[155] C. M. Bishop, "Mixture density networks," 1994.

[156] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton et al., "Attend, infer, repeat: Fast scene understanding with generative models," in *Advances in Neural Information Processing Systems*, 2016, pp. 3225–3233.

[157] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[158] R. Guo and D. Hoiem, "Support surface prediction in indoor scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2144–2151.

[159] D. Paschalidou, A. O. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3d shape parsing beyond cuboids," *arXiv preprint arXiv:1904.09970*, 2019.

[160] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," *arXiv preprint arXiv:1904.06447*, 2019.

[161] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[162] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," *arXiv preprint arXiv:1901.05103*, 2019.

[163] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[164] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. Reynolds, "structure-from-motionphotogrammetry: A low-cost, effective tool for geoscience applications," *Geomorphology*, vol. 179, pp. 300–314, 2012.

[165] A. Phalak, Z. Chen, D. Yi, K. Gupta, V. Badrinarayanan, and A. Rabinovich, "Deepperimeter: Indoor boundary estimation from posed monocular sequences," *arXiv preprint arXiv:1904.11595*, 2019.