Data-Driven 3D Scene Understanding

Shuran Song

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Thomas A. Funkhouser

November 2018

Abstract

Intelligent robots require advanced vision capabilities to perceive and interact with the real physical world. While computer vision has made great strides in recent years, its predominant paradigm still focuses on analyzing image pixels to infer two dimensional outputs (e.g. 2D bounding boxes, or labeled 2D pixels.), which remain far from sufficient for real-world robotics applications.

This dissertation presents the use of amodal 3D scene representations that enable intelligent systems to not only recognize what is seen (e.g. Am I looking at a chair?), but also predict contextual information about the complete 3D scene beyond visible surfaces (e.g. What could be behind the table? Where should I look to find an exit?).

More specifically, it presents a line of work that demonstrates the power of these representations: First it shows how 3D amodal scene representation can be used to improve the performance of a traditional tasks such as object detection. We present SlidingShapes and DeepSlidingShapes for the task of amodal 3D object detection, where the system is designed to fully exploit the advantage of 3D information provided by depth images. Second, we introduce the task of semantic scene completion and our approach SSCNet, whose goal is to produce a complete 3D voxel representation of volumetric occupancy and semantic labels for a scene from a single-view depth map observation. Third, we introduce the task of semantic-structure view extrapolation and our approach Im2Pano3D, which aims to predict the 3D structure and semantic labels for a full 360°panoramic view of an indoor scene when given only a partial observation. Finally, we present two large-scale datasets (SUN RGB-D and SUNCG) that enable the research on data-driven 3D scene understanding.

This dissertation demonstrates that leveraging a complete 3D scene representations not only significantly improves algorithm's performance for traditional computer vision tasks, but also paves the way for new scene understanding tasks that have previously been considered ill-posed given only 2D representations.

Acknowledgements

I would like to thank my advisor, Thomas Funkhouser, for being an excellent role model throughout my years in graduate school. I enjoyed our day-to-day discussions – from brainstorming high-level ideas to nailing down detailed implementation issues. More importantly, his work ethic, passion, patience, and wisdom continually demonstrates what it means to be a great advisor and researcher. I believe his principles will continue to positively influence and guide me throughout my life.

I would also like to thank my thesis committee members, Szymon Rusinkiewicz, Adam Finkelstein, Olga Russakovsky and Alberto Rodriguez for their feedback and comments, and for inspirating by example.

I was fortunate to receive lots of help throughout my PhD. It has been an honor to work along side with many great researchers including Andy Zeng, Manolis Savva, Angel Chang, Yinda Zhang, Fisher Yu, Jianxiong Xiao, Matthias Niener, Matthew Fisher, Zhirong Wu, Linguang Zhang, and Samuel P. Lichtenberg.

I was extremely lucky to be part of Princeton Computer Vision and Graphics Group. I thank all current and previous group members, especially Maciej Halber, Elena Balashova, Xinyi Fan, Zeyu Jin, Nora Willett, and Kyle Genova for their support and suggestions.

I would also like to thank all members of the MIT-Princeton Amazon Robotics Challenge team. Collaborating with them has been both an incredible and humbling learning experience.

My graduate studies at Princeton University and this thesis were generously supported by the Princeton Wallace Fellowship, the Siebel Scholar Fellowship, and the Facebook Fellowship.

Finally, I would like to thank my family for their continuous love and endless support.

To my parents.

Contents

Abstract iii

Acknowledgements iv

List of Tables ix

List of Figures xi

1 Introduction 2

1.1 Motivation 2

1.2 Challenges in 3D Scene Understanding 4

1.3 Dissertation Structure 6

2 Understanding Amodal 3D Objects 8

2.1 3D Amodal Object Detection 8

2.2 Related works 11

2.3 Sliding Shapes 12

2.4 Deep Sliding Shapes 14

2.4.1 Data Representation for 3D Deep Learning 15

2.4.2 Multi-scale 3D Region Proposal Network 16

2.4.3 Joint Amodal Object Recognition Network 22

2.4.4 Evaluation 25

2.5 Summary 30

3 Understanding Amodal 3D Scenes 33

3.1 Semantic Scene Completion 33

3.2 Related Work 36

3.3 Semantic Scene Completion Network 38

3.3.1 Volumetric Data Encoding 39

3.3.2 Network Architecture 40

3.4 Synthesizing training data 43

3.4.1 Synthetic depth map generation 44

3.4.2 Volumetric ground truth generation 45

3.5 Evaluation 46

3.5.1 Evaluation metric. 46

3.5.2 Experimental results 47

3.6 Summary 51

4 Understanding 3D Scenes Beyond the Field of View 54

4.1 Semantic-Structure View Extrapolation 54

4.2 Related Work 57

4.3 Im2Pano3D Network 59

4.3.1 Whole Room Panoramic Representation 59

4.3.2 Representing 3D Surfaces with Plane Equations 60

4.3.3 Im2Pano3D Network Architecture 62

4.3.4 Im2Pano3D Network Losses 63

4.4 Evaluation 65

4.4.1 Datasets 65

4.4.2 Baseline Methods 65

4.4.3 Experimental Results 69

4.5 Summary 73

5 Datasets and Benchmarks for 3D Scene Understanding 75

5.1 Real-world RGB-D dataset: SUN RGB-D 75

5.1.1 Related Work 77

5.1.2 Dataset Construction 78

5.1.3 Ground Truth Annotation 83

5.1.4 SUN RGB-D Dataset Statistics 85

5.1.5 Benchmark Design 85

5.2 Synthetic 3D Scene Dataset: SUNCG 94

5.2.1 Related Work 96

5.2.2 Dataset Construction 98

5.2.3 SUNCG Dataset Statistics 98

5.2.4 Tasks Supported by SUNCG 101

5.3 Summary 103

6 Future Directions 104

7 Conclusion 107

Bibliography 108

List of Tables

2.1 **Evaluation for Amodal 3D Object Proposal.** [All Anchors] shows the performance upper bound when using all anchors. 27

2.2 **Evaluation of proposal generation on SUN RGB-D test set.** . 27

2.3 **Control Experiments on NYUv2 Test Set.** Not working: box (too much variance), door (planar), monitor and tv (no depth). . . . 29

2.4 **Evaluation of 3D amodal object detection on SUN RGB-D test set.** 29

2.5 **Comparison on 3D Object Detection.** 31

3.1 Semantic scene completion results on the NYU with kinect depth map. 44

3.2 **Scene completion on the rendered NYU test set as [33]** . . . 47

3.3 **Ablation study on SUNCG testset.** First two row shows the evaluation on surface segmentation with and without joint training. The following rows show the evaluation on semantic scene completion task. D: 3D dilated convolution. M: multi-scale aggregation. 49

4.1 Comparing to baseline methods on Matterport3D. Row 2 to 5 shows baseline methods. Our models are named by their input output modalities (same as Tab.4.2) and training set (s: SUNCG, m: Matterport3D). Bold numbers indicate best performances in each group. 71

4.2 Ablation studies on SUNCG. Models are named by their input and output modalities. rgb: color, s: semantic segmentation, d: depth, p: plane distance, n: surface normal. A: adversarial loss, S: scene attribute loss. 71

4.3 **Camera configurations.** The table shows the average PoG, median surface and normal error for each configuration. Example inputs for each configuration can be found in Fig.4.10. For models [rgbpano] and [rgbpano+1], we evaluate on regions that do not have depth observation. For all other models, we evaluate on regions with no color and depth observation. 74

5.1 **Semantic segmentation.** We evaluate performance for 37 object categories. Here shows 8 selected ones: floor, ceiling, chair, table, bed, nightstand, books, and person. The mean accuracy is for all the 37 categories. A full table is in the supp. material. 88

5.2 **Evaluation of 2D object detection.** We evaluate on 19 popular object categories using Average Precision (AP): bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, garbage bin, lamp, monitor, night stand, pillow, sink, sofa, table, tv and toilet. 88

5.3 **Evaluation of total scene understanding.** With the objects detection result from Sliding Shape and RCNN and Manhattan Box for room layout estimation, we evaluate four ways to integrate object detection and room layout: (1) directly combine (2) constrain the object using the room. (3) adjust room base on the objects (4) adjust the room and objects together. 94

5.4 **Cross-sensor bias.** 94

List of Figures

2.1 **Training Data Generation:** We use a collection of CG models to train a 3D detector. For each CG model, we render it from hundreds of view angles to generate a pool of positive training data. For each rendering, we train an Exemplar-SVM model. And we ensemble all SVMs from all renderings of all CG chair models to build a 3D chair detector. 13

2.2 **Sliding Shape:** We extract 3D features of point cloud from depth rendering of Computer Graphics model to train a 3D classifier. And during testing time, we slide a window in 3D to evaluate the score for each window using an ensemble of Exemplar-SVMs. 15

2.3 **Visualization of TSDF Encoding.** We only visualize the TSDF values when close to the surface. Red indicates the voxel is in front of surfaces; and blue indicates the voxel is behind the surface. The resolution is 208×208×100 for the Region Proposal Network, and 30×30×30 for the Object Recognition Network. 16

2.4 **3D Amodal Region Proposal Network**: Taking a 3D volume from depth as input, our fully convolutional 3D network extracts 3D proposals at two scales with different receptive fields. 17

2.5 **Joint Object Recognition Network:** For each 3D proposal, we feed the 3D volume from depth to a 3D ConvNet, and feed the 2D color patch (2D projection of the 3D proposal) to a 2D ConvNet, to jointly learn object category and 3D box regression. 18

2.6 **List of All Anchors Types.** The subscripts show the width $\times$ depth $\times$ height in meters, followed by the number of orientations for this anchor after the colon. 19

2.7 **Examples for Detection Results.** For the proposal results, we show the heat map for the distribution of the top proposals (red is the area with more concentration), and a few top boxes after NMS. For the recognition results, our amodal 3D detection can estimate the full extent of 3D both vertically (*e.g.*bottom of a bed) and horizontally (*e.g.*full size sofa in the last row). 21

2.8 **Distributions of Heights for Amodal v.s Modal Boxes**. The modal bounding boxes for only the visible parts of objects have much larger variance in box sizes, due to occlusion, truncation, or missing depth. Representing objects with amodal box naturally enables more invariance for learning. 21

2.9 2D t-SNE embedding of the last layer features learned from the 3D ConvNet. Color encodes object category. 24

2.10 **Comparision with Sliding Shapes [155].** Our algorithm is able to better use shape, color and contextual information to handle more object categories, resolve the ambiguous cases, and detect objects with atypical size. 30

2.11 **Top True Positives.** 31

2.12 **Top False Positives.** (1)-(2) show detections with inaccurate locations. (3)-(6) show detections with wrong box size for the big bookshelf, L-shape sofa, bunk bed, and monitor. (7)-(10) show detections with wrong categories. 31

2.13 **Misses.** Reasons: heavy occlusion, outside field of view, atypical size object, or missing depth. 31

3.1 **Semantic scene completion.** [Left] Input single-view depth map and visible surface from the depth map; color is for visualization only. [Right] Semantic scene completion result: our model jointly predicts volumetric occupancy and object categories for each of the 3D voxels in the view frustum. Note that the entire volume occupied by the bed is predicted to have the bed category. 34

3.2 Given a single-view depth observation of a 3D scene the goal of our SSCNet is to predict both occupancy and object category for the voxels on the observed surface and occluded regions. 38

3.3 **Different encodings for surface (a).** The projective TSDF (b) is computed with respect to the camera and is therefore view-dependent. The accurate TSDF (c) has less view dependency but exhibits strong gradients in empty space along the occlusion boundary (circled in gray). In contrast, the flipped TSDF (d) has the strongest gradient near the surface. 40

3.4 **SSCNet: Semantic scene completion network.** Taking a single depth map as input, the network predicts occupancy and object labels for each voxel in the view frustum. The convolution parameters are shown as (number of filters, kernel size, stride, dilation). 41

3.5 **Comparison of receptive fields and voxel sizes between SSC-Net and prior work.** (a) Object-centric networks such as [180] and [111] scale objects into the same 3D voxel grid thus discarding physical size information. In (b)-(d), colored regions indicate the effective receptive field of a single neuron in the last layer of each 3D ConvNet. With the help of 3D dilated convolution SSCNet drastically increases its receptive field compared to other 3D ConvNet architectures [156, 191] thus capturing richer 3D contextual information. 42

3.6 **Synthesizing Training Data.** We collected a large-scale synthetic 3D scene dataset to train our network. For each of the 3D scenes, we select a set of camera positions and generate pairs of rendered depth images and volumetric ground truth as training examples. 44

3.7 **What 3D context does the network learn?** The first figure shows the input depth map (a desk) and the following figures show the predictions for other objects. Without observing any information for other objects the SSCNet is able to hallucinate their locations based on the observed object and the learned 3D context. 50

3.8 **Qualitative results.** We compare with scene completion results from Zheng *et al.* [199] and Firman *et al.* [33] (on a subset of the test set), and semantic scene completion results from Lin *et al.* [99] and Geiger and Wang [36]. Zheng *et al.*[33] tested on rendered depth, [99] and [36] tested on RGB-D frame from kinect, [33] and SSCNet tested on kinect depth. Overall, SSCNet gives more accurate voxel predictions such as the lamps and pillows in the first row, and the sofa in the third row. . 52

4.1 **Semantic-structure view extrapolation.** Given a partial observation of the room in the form of an RGB-D image, our Im2Pano3D predicts both 3D structure and semantics for a full panoramic view of the same scene. 55

4.2 **Whole room representation.** We use a sky-box-like multi-channel panorama to represent 3D scenes. The views are circularly connected, hence, observing the inner two views is equivalent to observing the outer two views of its shifted panorama. 60

4.3 **3D structure prediction with different encodings.** The plane equation encoding (B) is a better output representation than raw depth encoding (A); its regularization enables the network to predict higher quality geometry. 61

4.4 **Im2Pano3D network architecture.** the network uses a multi-stream autoencoder structure. A PN-layer is used to ensure consistency between normal and plane distance predictions. 62

4.5 Directly predicting 3D structure and semantics (b) (rgbpn2pns) provides a more accurate result than predicting the same information from generated color pixels (d) (inpaint). 66

4.6 **Human completion.** Left shows the input observations. Middle shows completion results from different users overlaid on the observations. Right shows ground truth and our prediction. 67

4.7 **Probability distribution of semantics.** The first row shows the average distribution of each semantic category over all training examples. The following rows show the predicted probability distribution of semantics from Im2Pano3D overlaid on top of the ground truth testing images. Red areas on the heat maps indicate higher probabilities. . . 69

4.8 **Qualitative Results.** For each example, we show semantic segmentations labeled using the highest predicted class probability for each pixel, and normal maps from 3D structure predictions. We also show reconstructed 3D point clouds (right column), colored by semantic labels, with bounding boxes around semantically connected components. More results in supplementary material. 70

4.9 **Experiments.** (a) shows mean IOU with respect to distance from observation. (b) shows accuracy of predictions in the unobserved regions while increasing input horizontal FoV from 5° to 350°. The error bar shows the error margin across test cases. 72

4.10 **Camera configurations.** For different camera configurations, Im2Pano3D provides a unified framework to efficiently filling in missing 3D structure and semantics of the unobserved scene. The observation coverage is shown in parentheses. 3D structure is represented with normals. The data for [RGB-D+motion] comes from NYUv2 [149]. More examples can be found in supplementary material. 73

5.1 **Comparison of RGB-D recognition benchmarks.** Apart from 2D annotation, our benchmark provided high quality 3D annotation for both objects and room layout. 77

5.2 **Comparison of the four RGB-D sensors.** The raw depth map from Intel RealSense is noisier and has more missing values. Asus Xtion and Kinect v1's depth map have observable quantization effect. Kinect v2 is more accurate to measure the details in depth, but it is more sensitive to reflection and dark color. Across different sensors our depth improvement algorithm manages to robustly improve the depth map quality. 79

5.3 **Data Capturing Process.** (a) RealSense attached to laptop, (b) Kinect v2 with battery, (c) Capturing setup for Kinect v2. 82

5.4 **Example images with annotation from SUN RGB-D dataset.** 83

5.5 **Statistics of semantic annotation in our dataset.** 85

5.6 **Confusion matrices for various scene recognition algorithms.** Each combination of features and classifiers is run on RGB, D and RGB-D. The numbers inside the parentheses are the average accuracy for classification. 87

5.7 **Example results for 3D object detection and orientation prediction.** We show the angle difference and IoU between predicted boxes (blue) and ground truth (red). 89

5.8 **Object orientation estimation.** Here we show the distribution of the orientation errors for all true positive detections. 90

5.9 **Example visualization** to compare the three 3D room layout estimation algorithms. 91

5.10 **Free space evaluation.** The free space is the gray area inside the room, outside any object bounding boxes, and within the effective minimal and maximal range [0.5m-5.5m]. For evaluation, we use IoU between the gray areas of the ground truth and the prediction as the criteria. 92

5.11 **Visualization of total scene understanding results.** 93

5.12 Real data (top) vs. synthetic data (bottom). For the real data, note the noise in normal map and the diminishing accuracy at object boundaries in the semantic labels. 96

5.13 **Scene structure statistics.** Distribution of number of rooms and number of floors in our SUNCG dataset. Our dataset contains a large variety of 3D indoor scenes such as small studios, multi-room apartments, and multi-floor houses. 99

5.14 **Distribution of physical sizes** (in meters^2) per room, floor, and house of the SUNCG dataset. 100

5.15 Distribution of different room types in the SUNCG dataset (left), and examples of rooms per room type (right). 100

5.16 **Distribution of object categories in the SUNCG dataset.** We have 84 object categories in total. Here we show the top 50 object categories with highest number of occurrences in our dataset. 101

5.17 **Object-Room Relationship.** On the left we show the distribution of number of objects in each room. On average there are more than 14 objects in each room. On the right, we show the object category distribution conditioned on different room type. Size of the square shows the frequency of a given object category appears in the certain room type. The frequency is normalized for each object category. As expected, object occurrences are tightly correlated with the room type. For example, kitchen counters have a very high chance to appear in kitchen, while chairs appear frequently in many room types. 102

ets us split up the document (and each include starts a new page):

Chapter 1

Introduction

1.1 Motivation

Imagine a domestic robot preparing to set a dining table. Which piece of visual information would it find to be more useful for the task? (info A) I see a table on the left and chairs on the right, or (info B) I see a table two meters away from me behind three chairs, the tabletop is one meter above the floor, and there is enough empty space on the table to place the dishes. While performing complex tasks such as preparing dining tables, autonomous robotic systems would typically benefit from a complete 3D visual understanding of the object in the scene, their locations and orientations, spatial relationships, and free space (i.e. info B). However, most computer vision algorithms will only produce information to the extent of object bounding boxes and image labels (i.e. info A).

This example highlights a fundamental limitation behind classic *2D image-centric* computer vision tasks: they are targeted at understanding 2D images, but not the 3D physical world behind them. Moreover, since images are only 2D partial representations of complete 3D scenes, they can exhibit extreme variations from minor changes to camera viewpoint, materials, lighting, and object arrangements, which continue to obscure state-of-the-art image recognition algorithms.

The goal of this dissertation is to develop computer vision algorithms that can understand the visual world in terms of both low-level 3D structure and high-level semantics. More importantly, the system should not only be able to recognize what it sees (e.g. Am I looking at a chair?), but also be able to reason contextual information related to its complete 3D environment - including regions beyond the visible surfaces in view (e.g. What could be behind the table? Where should I look to find an exit?).

Towards this goal, this thesis aims to develop **3D amodal scene representations** that can harness the power of both color and depth (RGB-Depth) scan data from 3D sensors such as the Microsoft Kinect, to bypass the aforementioned challenges, and directly provide useful 3D outputs for real-world applications. My research is uniquely defined by the following aspects:

- **3D.** Explore the direct use of 3D data representations and grounded in real-world physical metrics, as both input and output for computer vision algorithms, instead of reasoning over 2D image pixels to infer 3D information.

- **Amodal.** Leverage and provide complete 3D representations of scenes regardless of occlusions and image field of view (amodal), instead of only considering surfaces visible to the camera (modal).

- **Scene.** Make use of contextual information beyond single objects to extract rich information about the scene as a whole.

In contrast to image representations, 3D amodal scene representations provide a more faithful digital encoding of the world we live in, allowing us to store and manipulate all kinds of information (e.g., materials, semantics, affordances) in a way that is more direct and complete. Beyond computer vision, advances in 3D scene representations will also have a significant impact on many other domains, including graphics, robotics, mechanical engineering, and computational biology, where 3D data representations serve as a common foundation for data-driven analysis.

This research has been made possible by the recent availability of affordable 3D scanning devices (e.g., Microsoft Kinect), as well as the dramatic growth of online 3D model repositories (e.g., Trimble Warehouse). The combination of these two developments changes how we can approach visual scene understanding tasks, there has never been a more exciting time to explore the idea of how we can achieve robust 3D scene understanding for intelligent systems in real-world applications.

1.2 Challenges in 3D Scene Understanding

While the opportunity is exciting, data-driven 3D scene understanding also presents unique challenges. The following paragraphs summarize the major questions and challenges behind 3D scene understanding:

How should we represent a 3D scene? In traditional 2D computer vision, scene representations are often made up of 2D bounding boxes and/or 2D pixel labels. While it is possible to produce analogous representations for 3D scene understanding via 3D bounding boxes and/or 3D point cloud labels for visible surfaces, these representations may not be sufficient for many real-world applications. For example, to successfully grasp a 3D object, a robotic system may need to infer the complete 3D shape of the object rather than just its visible surfaces.

This dissertation studies a set of new tasks which aim to produce various 3D scene representations that are more useful for supporting real-world applications. These tasks include amodal 3D object detection, as explained in Chapter 2, semantic scene completion; as discussed in Chapter 3; and semantic-structure view extrapolation, as elaborated in Chapter 4.

How should we encode 3D information? Color images are naturally encoded as a 2D array of pixel color values. However, it is unclear how 3D data should be encoded. Representing 3D information using view-based depth images holds the advantage of being able to reuse pre-trained deep models from color image datasets (which are widely available). However, such models are often highly viewpoint-dependent and can lose valuable information encoded within the 3D spatial locality. On the other hand, 3D volumetric representations directly encode 3D geometry and preserve 3D spatial locality, but at the cost of exponential compute.

In this thesis we utilize several different 3D data encodings, and discuss their pros and cons under various task settings. For example, we use 3D point cloud based representations in Chapter 2, the 3D volumetric truncated signed distance function (TSDF) encoding and its variations in Chapters 2 and 3, as well as a 2D plane equation encoding in Chapter 4.

How should we design an algorithm that best utilizes 3D information? Many of the algorithms in this thesis have been inspired by recent progress in 2D computer vision. However, it is critical to continually re-think and re-design these algorithms so that they can better adapt to the unique aspects of 3D data and its applications. In this dissertation, we explore how we can design machine learning algorithms in ways that make them more effective for use with 3D information. For example, how can we make use of real-world physical metrics for object recognition? How can we handle data sparsity in 3D volumetric data representations during training? How can we learn useful long-range contextual priors of 3D scenes without significantly increasing the computation complexity?

How to obtain the training data. In contrast to the large, well-established datasets and benchmarks for 2D computer vision, existing datasets for 3D scene understanding (e.g. the NYU dataset [120]) are significantly smaller. The scarcity of available 3D data is a common bottleneck for research in the field of 3D scene understanding. As a reaction to the lack of freely available 3D data, this dissertation introduces two large-scale 3D scene understanding datasets: SUNG RGB-D for real-world RGB-D data (Section 5.1) and SUNCG (Section 5.2) for synthetic 3D scene data.

1.3 Dissertation Structure

This dissertation is divided into the following chapters:

Chapter 2: I present two approaches (SlidingShapes [155] and DeepSlidingShapes [156]) for accomplishing the task of amodal 3D object detection and show how we can make use of depth information to improve the performance of a traditional computer vision task.

Chapter 3: I introduce the task of semantic scene completion, a task for producing a complete 3D voxel representation of volumetric occupancy and semantic labels for a scene from a single-view depth map observation. I also show how to address this task by leveraging the coupled nature of 3D geometry and semantic information through a semantic scene completion network [157].

Chapter 4: I try to push the boundary even further by introducing the task of semantic-structure view extrapolation, which aims to predict the 3D structure and semantic labels for a full 360 °panoramic view of an indoor scene when given only a partial observation. Then I present our approach (Im2Pano3D) to address this task by leveraging strong contextual priors learned from large-scale indoor scenes [158].

Chapter 5: I present two large-scale datasets and benchmarks that we constructed (SUN RGB-D [154] and SUNCG [157]), and how they can support different research topics in the area of 3D scene understanding.

Chapter 2

Understanding Amodal 3D Objects

In this chapter, we focus on the task of amodal 3D object detection in RGB-D images, which aims to produce a 3D bounding box of an object in metric form at its full extent regardless of truncation or occlusion. We present two approaches for this task SlidingShapes [155] in Section 2.3 and its improved version DeepSlidingShapes [156] in Section 2.4. These 3D detector systems utilize the depth maps captured by consumer-level RGB-D sensor, and are designed to fully exploit the advantage of 3D information by directly reasoning on the 3D space.

2.1 3D Amodal Object Detection

Typical object detection predicts the category of an object along with a 2D bounding box on the image plane for the visible part of the object. While this type of result is useful for some tasks, such as object retrieval, it is rather unsatisfactory for doing any further reasoning grounded in the real 3D world. In this chapter, we focus on the task of amodal 3D object detection in RGB-D images, which aims to produce an object's 3D bounding box that gives real-world dimensions at the object's full extent, regardless of truncation or occlusion. This kind of recognition is much more useful, for instance, in the perception-manipulation loop for robotics applications. But adding a new dimension for prediction significantly enlarges the search space, and makes the task much more challenging. The arrival of reliable and affordable RGB-D sensors (*e.g.*, Microsoft Kinect) has given us an opportunity to revisit this critical task.

To utilize the depth information for object detection. Depth RCNN [48] takes a 2D approach: detect objects in the 2D image plane by treating depth as extra channels of a color image, then fit a 3D model to the points inside the 2D detected window by using ICP alignment. However direct converting 2D detection results to 3D does not work well (see Table 2.5 and [48]).

Instead, we proposed to formulate this problem directly in 3D in order to fully exploits the advantage of 3D provided by the depth. In Section 2.3 we first introduce **Sliding Shapes**, where the algorithm directly slide a 3D detection window in 3D space. By exploitng the 3D information in a data driven fashion, the algorithm is designed to overcome the major difficulties for recognition, namely the variations of texture, illumination, shape, viewpoint, clutter, occlusion, self occlusion and sensor noises.

However, the hand-crafted features limit how accurately Sliding Shapes can recognize objects and its multi-step algorithm makes it quite slow. To address these problems, in Section 2.4, we introduce **Deep Sliding Shapes**, a complete 3D formulation to learn object proposals and classifiers using 3D convolutional neural networks (ConvNets). In this work, We propose the first 3D Region Proposal Network (RPN) that takes a 3D volumetric scene as input and outputs 3D object proposals (Figure 2.4). It is designed to generate amodal proposals for whole objects at two different scales for objects with different sizes. We also propose the first joint Object Recognition Network (PRN) to use a 2D ConvNet to extract image features from color, and a 3D ConvNet to extract geometric features from depth (Figure 2.5). This network is also the first to regress 3D bounding boxes for objects directly from 3D proposals. Extensive experiments show that our 3D ConvNets can learn a more powerful representation for encoding geometric shapes (Table 2.5), than 2D representations (*e.g.*HHA in Depth-RCNN). Our algorithm is also much faster than Depth-RCNN and the the original Sliding Shapes, as it only requires a single forward pass of the ConvNets in GPU at test time.

Our design fully exploits the advantage of 3D. Therefore, our algorithm naturally benefits from the following five aspects: First, we can predict 3D bounding boxes without the extra step of fitting a model from extra CAD data. This elegantly simplifies the pipeline, accelerates the speed, and boosts the performance because the network can directly optimize for the final goal. Second, amodal proposal generation and recognition is very difficult in 2D, because of occlusion, limited field of view, and large size variation due to projection. But in 3D, because objects from the same category typically have similar physical sizes and the distraction from occluders falls outside the window, our 3D sliding-window proposal generation can support amodal detection naturally. Third, by representing shapes in 3D, our ConvNet can have a chance to learn meaningful 3D shape features in a better aligned space. Fourth, in the RPN, the receptive field is naturally represented in real world dimensions, which guides our architecture design. Finally, we can exploit simple 3D context priors by using the Manhattan world assumption to define bounding box orientations.

While the opportunity is encouraging, there are also several unique challenges for 3D object detection. First, a 3D volumetric representation requires much more memory and computation. To address this issue, we propose to separate the 3D Region Proposal Network with a low-res whole scene as input, and the Object Recognition Network with high-res input for each object. Second, 3D physical object bounding boxes vary more in size than 2D pixel-based bounding boxes (due to photography and dataset bias) [96]. To address this issue, we propose a multi-scale Region Proposal

Network that predicts proposals with different sizes using different receptive fields. Third, although the geometric shapes from depth are very useful, their signal is usually lower in frequency than the texture signal in color images. To address this issue, we propose a simple but principled way to jointly incorporate color information from the 2D image patch derived by projecting the 3D region proposal.

2.2 Related works

Our work has been inspired by research in object recognition of images, range scans, depth maps, RGB-D and CAD models, in the following section we will briefly review the works that are most relevant to this dissertation.

2D Object Detector in RGB Images Deep ConvNets have revolutionized 2D image-based object detection. RCNN [38], Fast RCNN [37], and Faster RCNN [129] are three iterations of the most successful state-of-the-art. Beyond predicting only the visible part of an object, [79] further extended RCNN to estimate the amodal box for the whole object. But their result is in 2D and only the height of the object is estimated, while we desire an amodal box in 3D.

2D Object Detector in RGB-D Images 2D object detection approaches for RGB-D images treat depth as extra channel(s) appended to the color images, using hand-crafted features [47], sparse coding [14, 15], or recursive neural networks [152]. Depth-RCNN [49, 48] is the first object detector using deep ConvNets on RGB-D images. They extend the RCNN framework [38] for color-based object detection by encoding the depth map as three extra channels (with Geocentric Encoding: Disparity, Height, and Angle) appended to the color images. [48] extended Depth-RCNN to produce 3D bounding boxes by aligning 3D CAD models to the recognition results. [52] further improved the result by cross model supervision transfer. For 3D CAD model classification, [164] and [144] took a view-based deep learning approach by rendering 3D shapes as 2D image(s). The main difference is that our algorithm operates fully in 3D, using 3D sliding windows and 3D features, which can handle occlusion and other problems naturally.

3D Classification: Classification-based approaches [12, 14, 13, 153, 11, 94, 93, 27, 192, 80, 143, 63, 40, 188, 201, 178] typically consider the whole object at the same time by extracting a holistic feature for the whole object and classifying the feature vector via a classifier. But the typical setting is to have the segmented object as the input (or even a solo 3D model with complete mesh), and classify an object into one of the fixed categories, which is a much easier task than object detection that needs to localize the object and tell a non-object window apart.

3D Feature Learning HMP3D [91] introduced a hierarchical sparse coding technique for unsupervised learning features from RGB-D images and 3D point cloud data. The feature is trained on a synthetic CAD dataset, and tested on scene labeling task in RGB-D video. In contrast, we desire a supervised way to learn 3D features using the deep learning techniques that are proven to be more effective for image-based feature learning.

3D Deep Learning 3D ShapeNets [180] introduced 3D deep learning for modeling 3D shapes, and demonstrated that powerful 3D features can be learned from a large amount of 3D data. Several recent works [111, 31, 187, 66] also extract deep learning features for retrieval and classification of CAD models. While these works are inspiring, none of them focuses on 3D object detection in RGB-D images.

2.3 Sliding Shapes

As the first attempt to address the amodal 3D object detection task we proposed Sliding Shapes [2], which slides a three-dimensional window over an RGB-D image to detect objects in that image. The **main idea** is to exploit the depth information in a data-driven fashion to overcome the major difficulties in object detection, namely the variations of texture, illumination, shape, viewpoint, self occlusion, clutter and occlusion.

The algorithm works as follows: for a given object category (e.g. chair), we use Computer Graphics (CG) CAD models from the Internet. We render each CG model from hundreds of viewpoints to obtain synthetic depth maps, as if they are viewed by a typical RGB-D sensor (Figure 2.1).For each rendering, a feature vector is extracted from the 3D point cloud corresponding to the rendered depth map to train an exemplar Support Vector Machine (SVM) [108] using negative data from a RGB-D data set [149] (Figure 2.2).. During testing or hard-negative mining, we slide a 3D detection window in the 3D space to match the exemplar shape and each window. Finally, to make use of the color information, we combine our depth-based object detector with bottom-up super pixel segmentation [50] to further improve the performance.

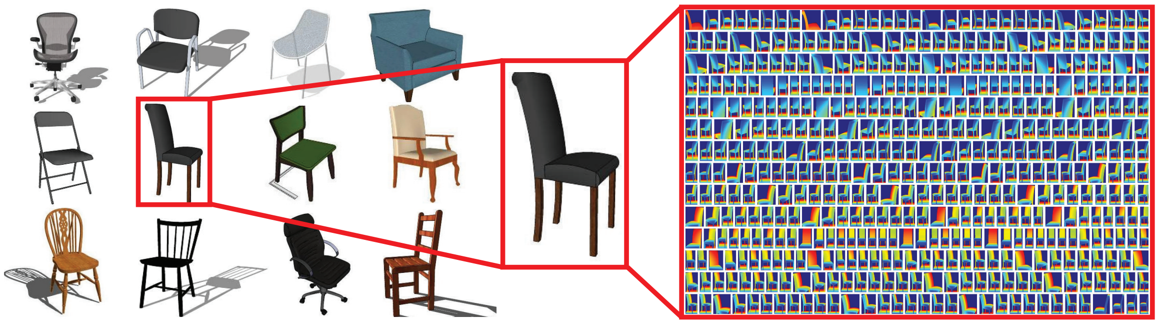


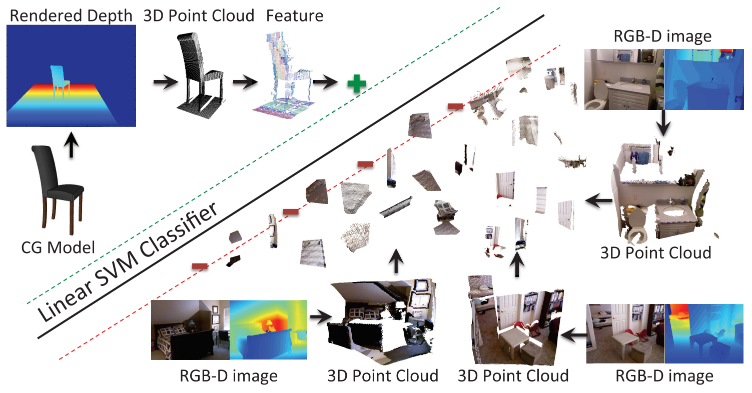
Figure 2.1: **Training Data Generation:** We use a collection of CG models to train a 3D detector. For each CG model, we render it from hundreds of view angles to generate a pool of positive training data. For each rendering, we train an Exemplar-SVM model. And we ensemble all SVMs from all renderings of all CG chair models to build a 3D chair detector.

Our design is based on several **key insights**: To handle *texture* and *illumination* variance, we use depth maps instead of RGB images, which is independent of appearance. To handle *shape* variance, we use a data-driven approach to leverage a collection of CG models that cover the space of shape variance in real world (the data set). We also add a small variance on the size of CG model to make the detector more robust. Furthermore, in contrast to direct mesh alignment [77], learning the SVM using both positive and negative data also increases the generalization of the detector. Because the CG models with complete mesh, to handle *viewpoint* variance, we can densely render different viewpoints of an object to cover all typical viewing angles. To handle *depth-sensor error and noise*, we use CG models to obtain perfect rendering and use it as positive training data. To bridge the *domain gap* between CG training data and RGB-D testing data, we render the depth map (but not color) as if the CG model is viewed from a typical RGB-D sensor. To handle *clutter* (e.g. a chair's seat under a table), we use 3D sliding window with a mask to indicate which parts should be considered during classification. To handle *inter-object occlusion*, we make use of the depth map to reason about the source of occlusion and regard the occluded area as missing data. To make use of *self-occlusion*, we render the CG model and compute the Truncated Signed Distance Function (TSDF) [121] as a feature.

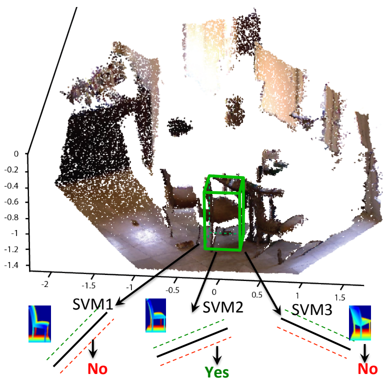
Since our *generic* object detector does not reply on any assumption about the background or requires a dominant supporting plane [167, 85], and its *single-view* nature doesn't require a (semi-)complete scan of an object [118], it can be used as a basic building block for general scene understanding tasks.

2.4 Deep Sliding Shapes

While Sliding Shapes algorithm demonstrate promising result that by exploiting depth information and reasoning in 3D. However, the hand-crafted features limit how accurately Sliding Shapes can recognize objects and its multi-step algorithm makes it quite slow. To address these problems, we proposed Deep Sliding Shapes [156] to use state-of-the-art deep learning techniques to learn powerful features from a large amount of 3D data.



(a) Training each 3D exemplar detector independently.



(b) Testing with exemplars.

Figure 2.2: **Sliding Shape:** We extract 3D features of point cloud from depth rendering of Computer Graphics model to train a 3D classifier. And during testing time, we slide a window in 3D to evaluate the score for each window using an ensemble of Exemplar-SVMs.

2.4.1 Data Representation for 3D Deep Learning

The first question that we need to answer for 3D deep learning is: how to encode a 3D space to present to the ConvNets? For color images, naturally the input is a 2D array of pixel color. For depth maps, Depth RCNN [48, 49] proposed to encode depth as a 2D color image with three channels. Although it has the advantage to reuse the pre-trained ConvNets for color images [52], we desire a way to encode the geometric shapes naturally in 3D, preserving spatial locality. Furthermore, compared to methods using hand-crafted 3D features [31, 187], we desire a representation that encodes the 3D geometry as raw as possible, and let ConvNets learn the most discriminative features from the raw data.

To encode a 3D space for recognition, we propose to adopt a directional Truncated Signed Distance Function (TSDF). Given a 3D space, we divide it into an equally spaced 3D voxel grid. The value in each voxel is defined to be the shortest distance between the voxel center and the surface from the input depth map. Figure 2.3 shows a few examples. To encode the direction of the surface point, instead of a single distance value, we propose a directional TSDF to store a three-dimensional vector

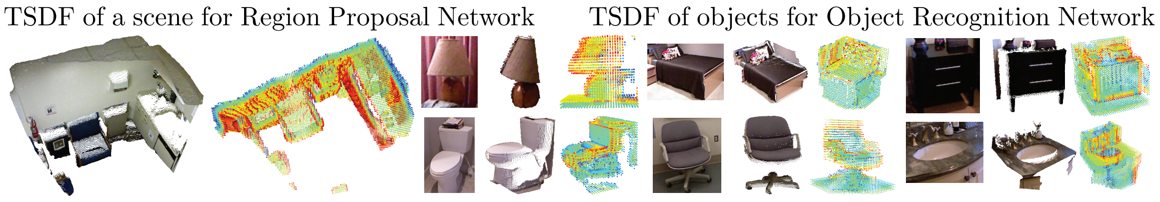

TSDF of a scene for Region Proposal Network
TSDF of objects for Object Recognition Network

Figure 2.3: **Visualization of TSDF Encoding.** We only visualize the TSDF values when close to the surface. Red indicates the voxel is in front of surfaces; and blue indicates the voxel is behind the surface. The resolution is 208×208×100 for the Region Proposal Network, and 30×30×30 for the Object Recognition Network.

$[dx, dy, dz]$ in each voxel to record the distance in three directions to the closest surface point. The value is clipped by 2δ, where δ is the grid size in each dimension. The sign of the value indicates whether the cell is in front of or behind the surface.

To further speed up the TSDF computation, as an approximation, we can also use projective TSDF instead of accurate TSDF where the nearest point is found only on the line of sight from the camera. The projective TSDF is faster to compute, but empirically worse in performance compared to the accurate TSDF for recognition (see Table 5.2). We also experiment with other encodings, and we find that the proposed directional TSDF outperforms all the other alternatives (see Table 5.2). Note that we can also encode colors in this 3D volumetric representation, by appending RGB values to each voxel [177].

2.4.2 Multi-scale 3D Region Proposal Network

Region proposal generation is a critical step in an object detection pipeline [38, 37, 129]. Instead of exhaustive search in the original Sliding Shapes, we desire a region proposal method in 3D to provide a small set of object agnostic candidates and to speed up the computation, while still utilizing the 3D information . But there are several unique challenges in 3D. First, because of an extra dimension, the possible locations for an object increases by 30 times [1]. This makes the region proposal step

[1]45 thousand windows per image in 2D [37] v.s. 1.4 million in 3D.

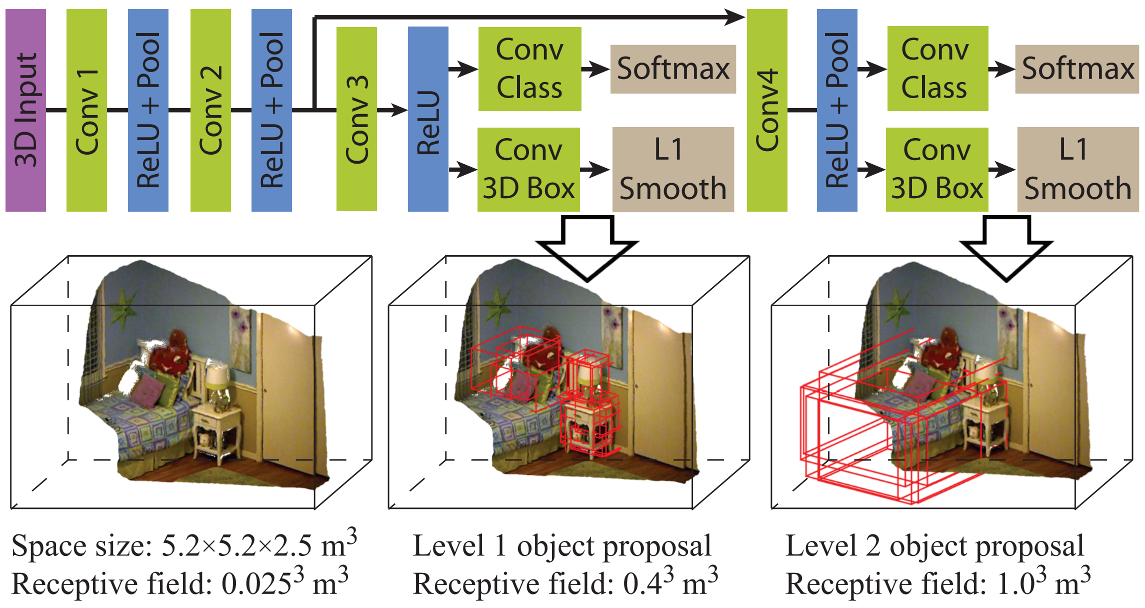


Figure 2.4: **3D Amodal Region Proposal Network**: Taking a 3D volume from depth as input, our fully convolutional 3D network extracts 3D proposals at two scales with different receptive fields.

much more important and challenging as it need to be more selective. Second, we are interested in amodal detection that aims to estimate the full 3D box that covers the object at its full extent. Hence an algorithm needs to infer the full box beyond the visible parts. Third, different object categories have very different object size in 3D. In 2D, a picture typically only focuses on the object of interest due to photography bias. Therefore, the pixel areas of object bounding boxes are all in a very limited range [129, 96]. For example, the pixel areas of a bed and a chair can be similar in picture while their 3D physical sizes are very different.

To address these challenges, we propose a multi-scale 3D Region Proposal Network (RPN) to learn 3D objectness using back-propagation (Figure 2.4). Our RPN takes a 3D scene as input and output a set of 3D amodal object bounding boxes with objectness scores. The network is designed to fully utilize the information from 3D physical world such as object size, physical size of the receptive field, and room orientation. Instead of a bottom-up segmentation based approach (*e.g.*[172]) that

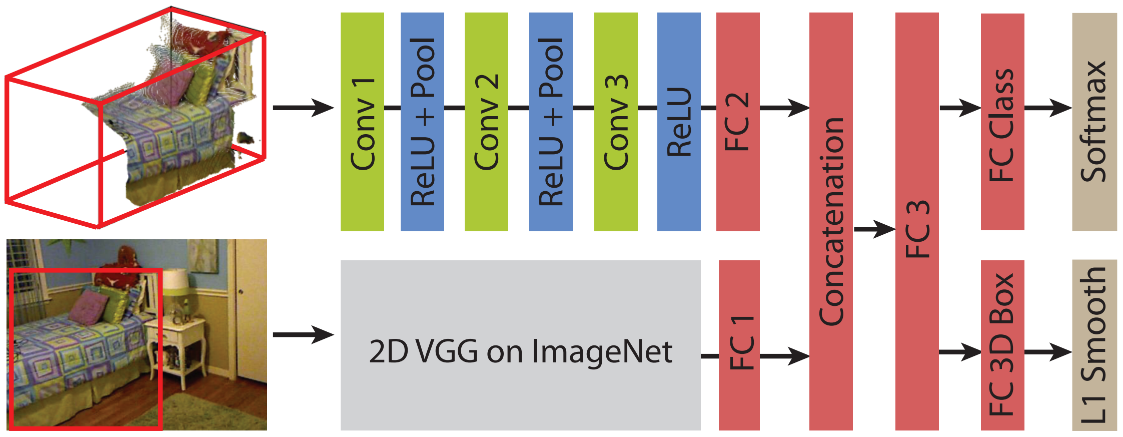


Figure 2.5: **Joint Object Recognition Network:** For each 3D proposal, we feed the 3D volume from depth to a 3D ConvNet, and feed the 2D color patch (2D projection of the 3D proposal) to a 2D ConvNet, to jointly learn object category and 3D box regression.

can only identify the visible part, our RPN looks at all the locations for the whole object, in a style similar to sliding windows, to generate amodal object proposals. To handle different object sizes, our RPN targets at two scales with two different sizes of receptive fields.

Range and resolution For any given 3D scene, we rotate it to align with gravity direction as our camera coordinate system. Based on the specs. for most RGB-D cameras, we target at the effective range of the 3D space $[-2.6, 2.6]$ meters horizontally, $[-1.5, 1]$ meters vertically, and $[0.4, 5.6]$ meters in depth. In this range we encoded the 3D scene by volumetric TSDF with grid size 0.025 meters, resulting in a $208 \times 208 \times 100$ volume as the input to the 3D RPN.

Orientation We desire a small set of proposals to cover all objects with different aspect ratios. Therefore, as a heuristic, we propose to use the major directions of the room for the orientations of all proposals. Under the Manhattan world assumption, we use RANSAC plane fitting to get the room orientations. This method can give us pretty accurate bounding box orientations for most object categories. For objects that do not follow the room orientations, such as chairs, their horizontal aspect ratios tend to be a square, and therefore the orientation doesn't matter much in terms of Intersection-Over-Union.

Anchor For each sliding window (*i.e.*convolution) location, the algorithm will predict N region proposals. Each of the proposal corresponds to one of the N anchor boxes. In our case, based on statistics of object sizes, we define a set of $N = 19$ anchors shown in Figure 2.6. For the anchors with non-square horizontal aspect ratios, we define another anchor with the same size but rotated 90 degrees.

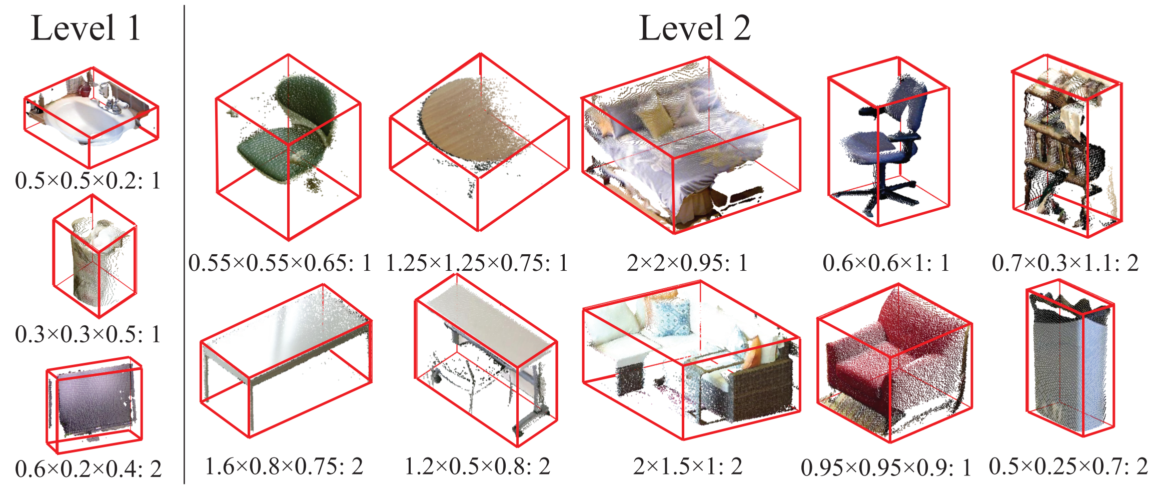


Figure 2.6: **List of All Anchors Types.** The subscripts show the width × depth × height in meters, followed by the number of orientations for this anchor after the colon.

Multi-scale RPN The physical sizes of anchor boxes vary a lot, from 0.3 meters (*e.g.*trash bin) to 2 meters (*e.g.*bed). If we use a single-scale RPN, the network would have to predict all the boxes using the same receptive fields. This means that the effective feature map will contain many distractions for small object proposals. To address this issue, we propose a multi-scale RPN to output proposals at small and big scales, the big one has a pooling layer to increase receptive field for bigger objects. We group the list of anchors into two levels based on their physical sizes, and use different branches of the network to predict them.

Fully 3D convolutional architecture To implement a 3D sliding window style search, we choose a fully 3D convolutional architecture. Figure 2.4 shows our network architecture. The stride for the last convolution layer to predict objectness score and bounding box regression is 1, which is 0.1 meter in 3D. The filter size is $2 \times 2 \times 2$ for Level 1 and $5 \times 5 \times 5$ for Level 2, which corresponds to 0.4 m^3 receptive field for Level 1 anchors and 1 m^3 for Level 2 anchors.

Empty box removal Given the range, resolution, and network architecture, the total number of anchors for any image is 1,387,646 ($19 \times 53 \times 53 \times 26$). On average, 92.2% of these anchor boxes are empty, with point density less than 0.005 points per cm^3. To avoid distraction, we automatically remove these anchors during training and testing.

Training sampling For the remaining anchors, we label them as positive if their 3D IOU scores with ground truth are larger than 0.35, and negative if their IOU are smaller than 0.15. In our implementation, each mini-batch contains two images. We randomly sample 256 anchors in each image with positive and negative ratio 1:1. If there are fewer than 128 positive samples we pad the mini-batch with negative samples from the same image. We select them by specifying the weights for each anchor in the final convolution layers. We also try to use all the positives and negatives with proper weighting, but the training cannot converge.

3D box regression We represent each 3D box by its center $[c_x, c_y, c_z]$ and the size of the box $[s_1, s_2, s_3]$ in three major directions of the box (the anchor orientation for anchors, and the human annotation for ground truth). To train the 3D box regressor, we will predict the difference of centers and sizes between an anchor box and its ground truth box. For simplicity, we do not do regression on the orientations. For each positive anchor and its corresponding ground truth, we represent the offset of box centers by their difference $[\Delta c_x, \Delta c_y, \Delta c_z]$ in the camera coordinate system. For the size difference, we first find the closest matching of major directions between the

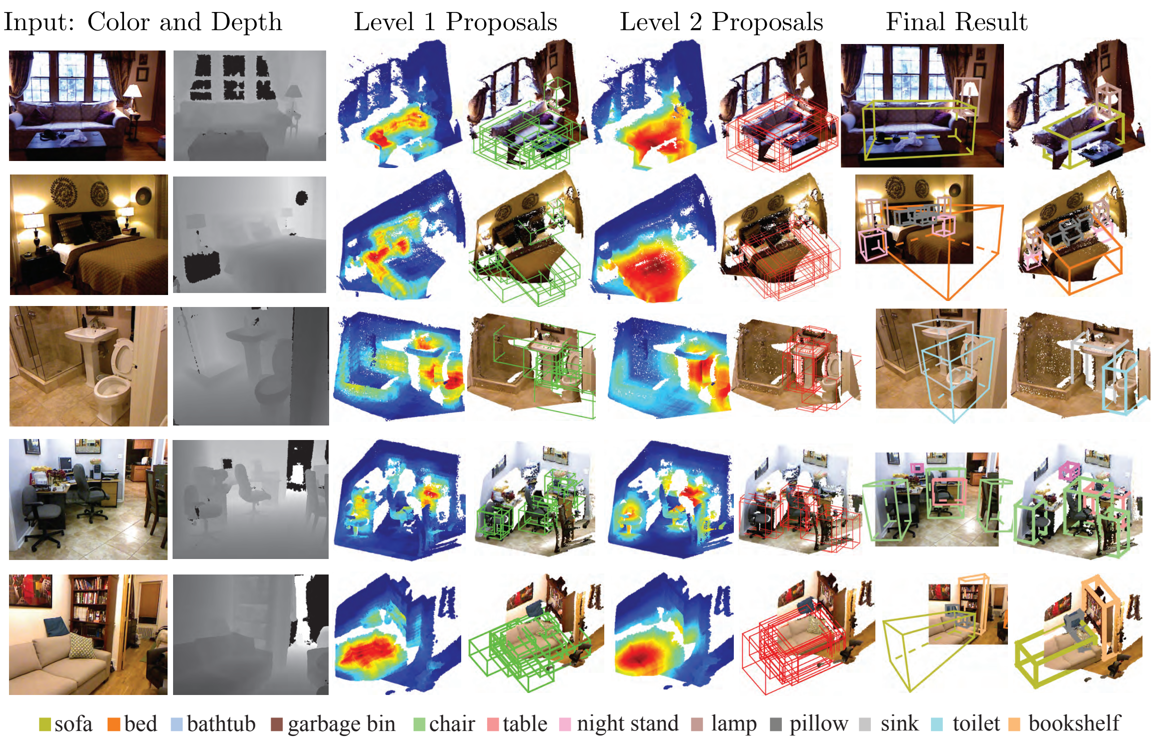


Figure 2.7: **Examples for Detection Results.** For the proposal results, we show the heat map for the distribution of the top proposals (red is the area with more concentration), and a few top boxes after NMS. For the recognition results, our amodal 3D detection can estimate the full extent of 3D both vertically (*e.g.*bottom of a bed) and horizontally (*e.g.*full size sofa in the last row).

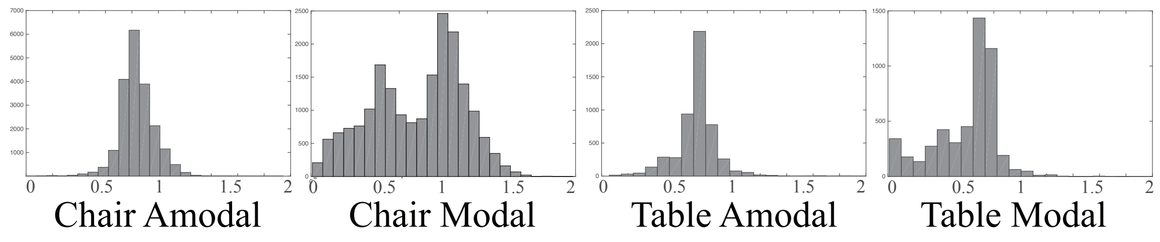


Figure 2.8: **Distributions of Heights for Amodal v.s Modal Boxes**. The modal bounding boxes for only the visible parts of objects have much larger variance in box sizes, due to occlusion, truncation, or missing depth. Representing objects with amodal box naturally enables more invariance for learning.

two boxes, and then calculate the offset of box size $[\Delta s_1, \Delta s_2, \Delta s_3]$ in each matched direction. Similarly to [129], we normalize the size difference by its anchor size. Our target for 3D box regression is a 6-element vector for each positive anchor $\mathbf{t} = [\Delta c_x, \Delta c_y, \Delta c_z, \Delta s_1, \Delta s_2, \Delta s_3]$.

Multi-task loss Following the multi-task loss in [37, 129], for each anchor, our loss function is defined as:

$$L(p, p^*, \mathbf{t}, \mathbf{t}^*) = L_{\text{cls}}(p, p^*) + \lambda p^* L_{\text{reg}}(\mathbf{t}, \mathbf{t}^*), \tag{2.1}$$

where the first term is for objectness score, and the second term is for the box regression. p is the predicted probability of this anchor being an object and p^* is the ground truth (1 if the anchor is positive, and 0 if the anchor is negative). L_{cls} is log loss over two categories (object v.s. non object). The second term formulates the 3D bounding box regression for the positive anchors (when $p^* = 1$). L_{reg} is smooth $\mathbf{L}_1$ loss used for 2D box regression by Fast-RCNN [37].

3D NMS The RPN network produces an objectness score for each of the non-empty proposal boxes (anchors offset by regression results). To remove redundant proposals, we apply 3D Non-Maximum Suppression (NMS) on these boxes with IOU threshold 0.35 in 3D, and only pick the top 2000 boxes to input to the object recognition network. These 2000 boxes are only 0.14% of all sliding windows, and it is one of the key factor that makes our algorithm much faster than the original Sliding Shapes [155].

2.4.3 Joint Amodal Object Recognition Network

Given the 3D proposal boxes, we feed the 3D space within each box to the Object Recognition Network (ORN). In this way, the final proposal feed to ORN could be the actual bounding box for the object, which allows the ORN to look at the full object to increase recognition performance, while still being computationally efficient. Furthermore, because our proposals are amodal boxes containing the whole objects at their full extent, the ORN can align objects in 3D meaningfully to be more invariant to occlusion or missing data for recognition.

3D object recognition network For each proposal box, we pad the proposal bounding box by 12.5% of the sizes in each direction to encode some contextual information. Then, we divide the space into a $30 \times 30 \times 30$ voxel grid and use TSDF

(Section 2.4.1) to encode the geometric shape of the object. The network architecture is shown in Figure 2.5. All the max pooling layers are 2^3 with stride 2. For the three convolution layers, the window sizes are 5^3, 3^3, and 3^3, all with stride 1. Between the fully connected layers are ReLU and dropout layers (dropout ratio 0.5). Figure 2.9 visualizes the 2D t-SNE embedding of 5,000 foreground volumes using their the last layer features learned from the 3D ConvNet. Color encodes object category.

2D object recognition network The 3D network only makes use of the depth map, but not the color. For certain object categories, color is a very discriminative feature, and existing ConvNets provide very powerful features for image-based recognition that could be useful. For each of the 3D proposal box, we project the 3D points inside the proposal box to 2D image plane, and get the 2D box that contains all these 2D point projections. We use the state-of-the-art VGGnet [150] pre-trained on ImageNet [136] (without fine-tuning) to extract color features from the image. We use a Region-of-Interest Pooling Layer from Fast RCNN [37] to uniformly sample 7×7 points from conv5_3 layer using the 2D window with one more fully connected layer to generate 4096-dimensional features as the feature from 2D images.

We also tried the alternative to encode color on 3D voxels, but it performs much worse than the pre-trained VGGnet (Table 5.2 [dxdydz+rgb] v.s. [dxdydz+img]). This might be because encoding color in 3D voxel grid significantly lowers the resolution compared to the original image, and hence high frequency signal in the image get lost. In addition, by using the pre-trained model of VGG, we are able to leverage the large amount of training data from ImageNet, and the well engineered network architecture.

2D and 3D joint recognition We construct a joint 2D and 3D network to make use of both color and depth. The feature from both 2D VGG Net and our 3D ORN (each has 4096 dimensions) are concatenated into one feature vector, and fed into

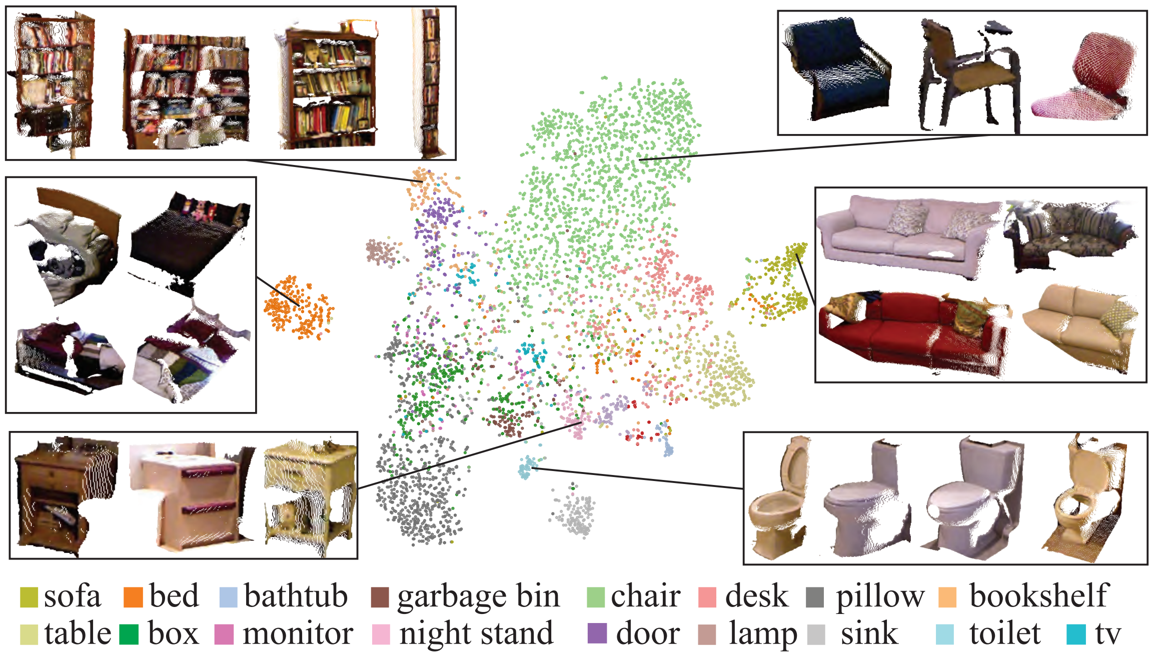


Figure 2.9: 2D t-SNE embedding of the last layer features learned from the 3D ConvNet. Color encodes object category.

a fully connected layer , which reduces the dimension to 1000. Another two fully connected layer take this feature as input and predict the object label and 3D box.

Multi-task loss Similarly to RPN, the loss function consists of a classification loss and a 3D box regression loss:

$$L(p, p^*, \mathbf{t}, \mathbf{t}^*) = L_{\text{cls}}(p, p^*) + \lambda'[p^* > 0]L_{\text{reg}}(\mathbf{t}, \mathbf{t}^*), \tag{2.2}$$

where the p is the predicted probability over 20 object categories (negative non-objects is labeled as class 0). For each mini-batch, we sample 384 examples from different images, with a positive to negative ratio of 1:3. For the box regression, each target offset $\mathbf{t}^*$ is normalized element-wise with the object category specific mean and standard deviation. During testing, we 0.1 asthe 3D NMS threshold. For box regressions, we directly use the results from the network.

Object size pruning When we use amodal bounding boxes to represent objects, the bounding box sizes provide useful information about the object categories. To make use of this information, for each of the detected box, we check the box size in each direction, aspect ratio of each pair of box edge. We then compare these numbers with the distribution collected from training examples of the same category. If any of these values falls outside 1st to 99th percentile of the distribution, which indicates this box has a very different size, we decrease its score by 2.

2.4.4 Evaluation

The training of RPN and ORN takes around 10 and 17 hours respectively on a NVIDIA K40 GPU. During testing, RPN takes 5.62s and ORN takes 13.93s per image, which is much faster than Depth RCNN (40s CPU + 30s GPU + expensive post alignment) and Sliding Shapes (25 mins $\times$ number of object categories). We implement our network architecture in Marvin [186], a deep learning framework that supports N-dimensionalconvolutional neural networks. For the VGG network [150], we use the weights from [52] without fine tuning.

We evaluate our 3D region proposal and object detection algorithm on the standard NYUv2 dataset [149] and SUN RGB-D [154] dataset. The amodal 3D bounding box are obtained from SUN RGB-D dataset. We modified the rotation matrix from SUN RGB-D dataset to eliminate the rotation on x,y plane and only contains camera tilt angle. Following the evaluation metric in [155], we assume the all predictions and ground truth boxes are aligned in the gravity direction. We use 3D volume intersection over union between ground truth and prediction boxes, and use 0.25 as the threshold to calculate the average recall for proposal generation and average precision for detection.

Object Proposal Evaluation

Evaluation of object proposal on NYU dataset is shown in Table 2.1. On the left, we show the average recall over different IOUs. On the right, we show the recall for each object category with IOU threshold 0.25, as well as the average best overlap ratio (ABO) across all ground truth boxes. Table 2.2 shows the evaluation on SUNRGB-D dataset.

Naïve 2D To 3D Our first baseline is to directly lift 2D object proposal to 3D. We take the 2D object proposals from [48]. For each of them, we get the 3D points inside the bounding box (without any background removal), remove those outside 2 percentiles along all three directions, and obtain a tight fitting box around these inlier points. Obviously this method cannot predict amodal bounding box when the object is occluded or truncated, since 3D points only exist for the visible part of an object.

3D Selective Search For 2D regoin proposal, Selective Search [172] is one of the most popular state-of-the-arts. It starts with a 2D segmentation and uses hierarchical grouping to obtain the object proposals at different scales. We study how well a similar method based on bottom-up grouping can work in 3D (3D SS). We first use plane fitting on the 3D point cloud to get an initial segmentation. For each big plane that covers more than 10% of the total image area, we use the RGB-D UCM segmentation from [49] (with threshold 0.2) to further split it. Starting with on this over-segmentation, we hierarchically group [172] different segmentation regions, with the following similarity measures:

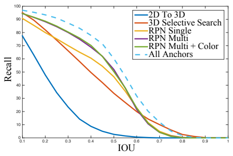
· $s_{\text{color}}(r_i, r_j)$ measures color similarity between region r_t and r_j using histogram intersection on RGB color histograms;

· $s_{\#\text{pixels}}(r_i, r_j) = 1 - \frac{\#\text{pixels}(r_i)+\#\text{pixels}(r_j)}{\#\text{pixels}(im)}$, where #pixels$(\cdot)$ is number of pixels in this region;

· $s_{\text{volume}}(r_i, r_j) = 1 - \frac{\text{volume}(r_i)+\text{volume}(r_j)}{\text{volume}(room)}$, where volume$(\cdot)$ is the volume of 3D bounding boxes of the points in this region;

· $s_{\text{fill}}(r_i, r_j) = 1 - \frac{\text{volume}(r_i)+\text{volume}(r_j)}{\text{volume}(ri \cup rj)}$ measures how well region r_i and r_j fit into each other to fill in gaps.

The final similarity measure is a weighted sum of these four terms. To diversify our



																				Recall	ABO	#Box
2D To 3D	41.7	53.5	37.9	22.0	26.9	46.2	42.2	11.8	47.3	33.9	41.8	12.5	45.8	20.7	49.4	55.8	54.1	15.2	50.0	34.4	0.210	2000
3D Selective Search	79.2	80.6	74.7	**66.0**	66.5	92.3	80.9	**53.9**	89.1	**89.8**	**83.6**	**45.8**	85.4	75.9	83.1	85.5	80.9	**69.7**	83.3	74.2	0.409	2000
RPN Single	87.5	98.7	70.1	15.6	95.0	100.0	93.0	20.6	94.5	49.2	49.1	12.5	100.0	34.2	81.8	94.9	93.3	57.6	96.7	75.2	0.425	2000
RPN Multi	100.0	**98.7**	**73.6**	42.6	94.7	**100.0**	92.5	21.6	**96.4**	78.0	69.1	37.5	**100.0**	75.2	97.4	97.1	96.4	66.7	100.0	84.4	0.460	2000
RPN Multi Color	**100.0**	98.1	72.4	42.6	**95.0**	**100.0**	**93.0**	19.6	**96.4**	79.7	76.4	37.5	**100.0**	**79.0**	**97.4**	**97.1**	**95.4**	57.6	**100.0**	**84.9**	**0.461**	2000
All Anchors	100.0	98.7	75.9	50.4	97.2	100.0	97.0	45.1	100.0	94.9	96.4	83.3	100.0	91.2	100.0	97.8	96.9	84.8	100.0	91.0	0.511	107674

Table 2.1: **Evaluation for Amodal 3D Object Proposal.** [All Anchors] shows the performance upper bound when using all anchors.

																				Recall	ABO	#Box
3D SS	78.8	87.2	72.8	72.2	65.5	86.1	75.1	65.0	70.0	87.1	67.5	53.1	68.1	82.8	86.8	84.4	85.0	69.2	94.0	72.0	0.394	2000
RPN	98.1	99.1	79.5	51.5	93.3	89.2	94.9	24.0	87.0	79.6	62.0	41.2	96.2	77.9	96.7	97.3	96.7	63.3	100.0	88.7	0.485	2000

Table 2.2: **Evaluation of proposal generation on SUN RGB-D test set.**

strategies, we run the grouping 5 times with different weights: $[1,0,0,0]$, $[0,1,0,0]$, $[0,0,1,0]$, $[0,0,0,1]$, $[1,1,1,1]$. For each of the grouped region, we will obtain two proposal boxes: one tight box and one box with height extended to the floor. We also use the room orientation as the box orientation. After that we will remove the redundant proposals with 3D IOU greater than 0.9 by arbitrary selection. Using both 3D and color, this very strong baseline achieves an average recall 74.2%. But it is slow because of its many steps, and the handcrafted segmentation and similarity might be difficult to tune.

Our 3D RPN Row 3 to 5 in Table 2.1 shows the performance of our 3D region proposal network. Row 3 shows the performance of single-scale RPN. Note that the recalls for small objects like lamp, pillow, garbage bin are very low. When one more scale is added, the performance for those small objects boosts significantly. Adding RGB color to the 3D TSDF encoding slightly improves the performance, and we use this as our final region proposal result. From the comparisons we can see that mostly planar objects (*e.g.*door) are easier to locate using segmentation-based selective search. Some categories (*e.g.*lamp) have a lower recall mostly because of lack of training examples. Table 5.2 shows the detection AP when using the same ORN architecture but different proposals (Row [3D SS: dxdydz] and Row [RPN: dxdydz]). We can see that the proposals provided by RPN helps to improve the detection performance by a large margin (mAP from 27.4 to 32.3).

Object Detection Evaluation

We conducted several control experiments to understand the importance of each component.

Feature encoding From Row [RPN: dxdydz] to Row [RPN: dxdydz+img] in Table 5.2, we compare different feature encodings and reach the following conclusions. (1) TSDF with directions encoded is better than single TSDF distance ([dxdydz] v.s. [tsdf dis]). (2) Accurate TSDF is better than projective TSDF ([dxdydz+img] v.s. [proj dxdydz+img]). (3) Directly encoding color on 3D voxels is not as good as using 2D image VGGnet ([dxdydz+rgb] v.s. [dxdydz+img]), probably because the latter one can preserve high frequency signal from images. (4) Adding HHA does not help, which indicates the depth information from HHA is already exploited by our 3D representation ([dxdydz+img+hha] v.s. [dxdydz+img]).

Does bounding box regression help? Previous works have shown that box regression can significantly improve 2D object detection [37]. For our task, although we have depth, there is more freedom on 3D localization, which makes regression harder. We turn the 3D box regression on ([3DSS dxdydz], [RPN dxdydz]) and off ([3DSS dxdydz no bbreg], [RPN dxdydz no bbreg]). Whether we use 3D Selective Search or RPN for proposal generation, the 3D box regression always helps significantly.

Does size pruning help? Compared with and without the post-processing ([dxdydz] v.s. [dxdydz no size]), we observe that for most categories, size pruning reduces false positives and improves the AP by the amount from 0.1 to 7.8, showing a consistent positive effect.

Is external training data necessary? Comparing to Sliding Shapes that uses extra CAD models, and Depth-RCNN that uses Image-Net for pre-training and CAD models for 3D fitting, our [depth only] 3D ConvNet does not require any external training data outside NYUv2 training set, and still outperforms the previous methods, which shows the power of 3D deep representation.

	algorithm																				mAP
3D SS	dxdydz no bbreg	43.3	55.0	16.2	23.1	3.4	10.4	17.1	30.7	10.9	35.4	20.3	41.2	47.2	25.2	43.9	1.9	1.6	0.1	9.9	23.0
	dxdydz	52.1	60.5	19.0	30.9	2.2	15.4	23.1	36.4	19.7	36.2	18.9	52.5	53.7	32.7	56.9	1.9	0.5	0.3	8.1	27.4
RPN	dxdydz no bbreg	51.4	74.8	7.1	51.5	15.5	22.8	24.9	11.4	12.5	39.6	15.4	43.4	58.0	40.7	61.6	0.2	0.0	1.5	2.8	28.2
	dxdydz no size	59.9	78.9	12.0	51.5	15.6	24.6	27.7	12.5	18.6	42.3	15.1	59.4	59.6	44.7	62.5	0.3	0.0	1.1	12.9	31.5
	dxdydz	59.0	80.7	12.0	59.3	15.7	25.5	28.6	12.6	18.6	42.5	15.3	59.5	59.9	45.3	64.8	0.3	0.0	1.4	13.0	32.3
	tsdf dis	61.2	78.6	10.3	61.1	2.7	23.8	21.1	25.9	12.1	34.8	13.9	49.5	61.2	45.6	70.8	0.3	0.0	0.1	1.7	30.2
	dxdydz+rgb	58.3	79.3	9.9	57.2	8.3	27.0	22.7	4.8	18.8	46.5	14.4	51.6	56.7	45.3	65.1	0.2	0.0	4.2	0.9	30.1
	proj dxdydz+img	58.4	81.4	20.6	53.4	1.3	32.2	36.5	18.3	17.5	40.8	19.2	51.0	58.7	47.9	71.4	0.5	0.2	0.3	1.8	32.2
	dxdydz+img+hha	55.9	83.0	18.8	63.0	17.0	33.4	43.0	33.8	16.5	54.7	22.6	53.5	58.0	49.7	75.0	2.6	0.0	1.6	6.2	36.2
	dxdydz+img	62.8	82.5	20.1	60.1	11.9	29.2	38.6	31.4	23.7	49.6	21.9	58.5	60.3	49.7	76.1	4.2	0.0	0.5	9.7	36.4

Table 2.3: **Control Experiments on NYUv2 Test Set.** Not working: box (too much variance), door (planar), monitor and tv (no depth).

																				mAP
Sliding Shapes	-	42.09	-		33.42	-	-	-	-	-	-	-	-	-	-	23.28	25.78	-	61.86	-
Deep Sliding Shapes	44.2	78.8	11.9	1.5	61.2	4.1	20.5	0.0	6.4	20.4	18.4	0.2	15.4	13.3	32.3	53.5	50.3	0.5	78.9	26.9

Table 2.4: **Evaluation of 3D amodal object detection on SUN RGB-D test set.**

Comparison to the state-of-the-arts

We evaluate our algorithm on the same test set as [48] (The intersection of the NYUv2 test set and Sliding Shapes test set for the five categories being studied under "3D all" setting). Table 2.5 shows the comparison with the two state-of-the-arts for amodal 3D detection: 3D Sliding Shapes [155] with hand-crafted features, and 2D Depth-RCNN [48] with ConvNets features. Our algorithm outperforms by large margins with or without colors. Different from Depth-RCNN that requires fitting a 3D CAD model as post-processing, our method outputs the 3D amodal bounding box directly, and it is much faster. Table 2.4 shows the amodal 3D object detection results on SUN RGB-D dataset compared with Sliding Shapes [155].

Figure 2.10 shows side-by-side comparisons to Sliding Shapes. First, the object proposal network and box regression provide more flexibility to detect objects with atypical sizes. For example, the small child's chairs and table in the last row are missed by Sliding Shapes but detected by Deep Sliding Shape. Second, color helps to distinguish objects with similar shapes (*e.g.*bed v.s. table). Third, the proposed algorithm can extend to many more object categories easily.

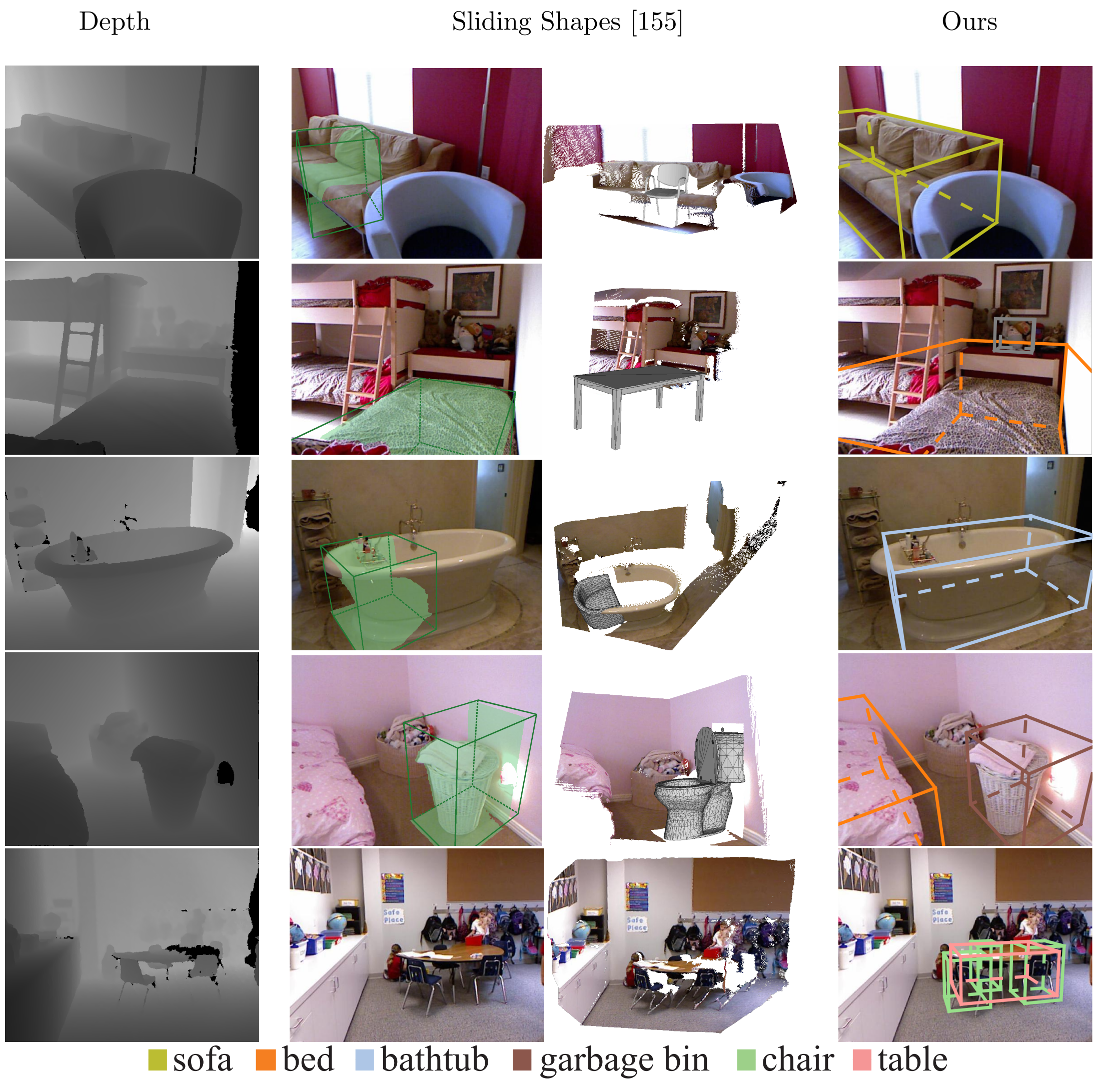


Figure 2.10: **Comparision with Sliding Shapes [155].** Our algorithm is able to better use shape, color and contextual information to handle more object categories, resolve the ambiguous cases, and detect objects with atypical size.

2.5 Summary

In this chapter we introduced the task of Amodal 3D object detection, which aims to produce a 3D bounding box of an object in metric form at its full extent regardless of occlusion and image truncation.

We present two approaches for this task SlidingShapes [155] and DeepSlidingShapes [156]. DeepSlidingShapes is a complete 3D formulation for learning object

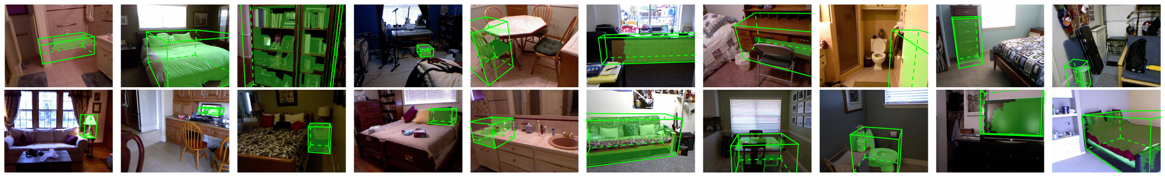
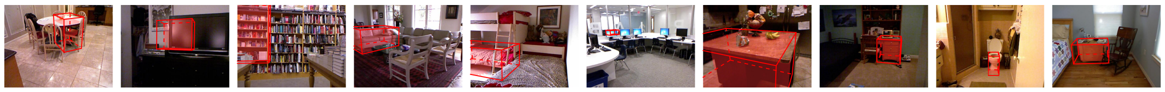
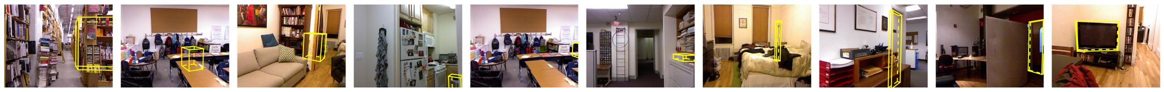


Figure 2.11: **Top True Positives.**



(1)chair (2)tv (3)bookshelf (4)sofa (5)bed (6)monitor (7)desk (8)night. (9)garbage. (10)box

Figure 2.12: **Top False Positives.** (1)-(2) show detections with inaccurate locations. (3)-(6) show detections with wrong box size for the big bookshelf, L-shape sofa, bunk bed, and monitor. (7)-(10) show detections with wrong categories.



bookshelf chair dresser garbage bin sofa box lamp door door tv

Figure 2.13: **Misses.** Reasons: heavy occlusion, outside field of view, atypical size object, or missing depth.

Algorithm	input	🛏	🪑	⊤	🛋	🚽	mAP
Sliding Shapes [155]	d	33.5	29	34.5	33.8	67.3	39.6
[48] on instance seg	d	71.0	18.2	30.4	49.6	63.4	46.5
[48] on instance seg	rgbd	74.7	18.6	28.6	50.3	69.7	48.4
[48] on estimated model	d	72.7	47.5	40.6	54.6	72.7	57.6
[48] on estimated model	rgbd	73.4	44.2	33.4	**57.2**	84.5	58.5
ours [depth only]	d	83.0	58.8	68.6	49.5	79.2	67.8
ours [depth + img]	rgbd	**84.7**	**61.1**	**70.5**	55.4	**89.9**	**72.3**

Table 2.5: **Comparison on 3D Object Detection.**

proposals and classifiers using 3D convolutional neural networks. It directly operates over 3D volumetric distance fields projected from input RGB-Depth scans. These distance fields serve as a 3D data representation that is defined with respect to physical metrics (e.g., in meters) and does not suffer from the scaling ambiguities caused by perspective projection. These aspects elegantly simplify the 3D object detection pipeline, accelerate run-time speeds, and boost overall detection performance.

While being useful for many applications, these object-centric recognition frameworks are characterized by two major limitations: 1) examining each object (or object proposal) in isolation causes the algorithms to ignore strong statistical relationships that exist between objects (contextual information) and 2) the resulting bounding box are often unable to express detailed geometry at the level of granularity required for applications like precise manipulation.

In the next chapters, we will describe how we are able to produce more detail scene description of the complete 3D environment beyond single objects through the task of "semantic scene completion".

Chapter 3

Understanding Amodal 3D Scenes

In this chapter we focus on the task of **semantic scene completion** – a task for producing a complete 3D voxel representation of volumetric occupancy and semantic labels for a scene from a single-view depth map observation. This task will enable us to produce much a more detailed representation of the complete environment instead of simple 3D bounding boxes described in the Chapter 2.

Prior work in this space has considered scene completion and semantic labeling of depth maps separately. However, we observe that these two problems are tightly intertwined. To leverage the coupled nature of these two tasks, we introduce the semantic scene completion network (**SSCNet**), an end-to-end 3D convolutional network that takes a single depth image as input and simultaneously outputs occupancy and semantic labels for all voxels in the camera view frustum. Our experiments demonstrated that the joint model outperforms methods addressing each task in isolation and outperforms alternative approaches on the semantic scene completion task.

3.1 Semantic Scene Completion

We live in a 3D world where empty and occupied space is determined by the physical presence of objects. To successfully navigate within and interact with the world, we rely on an understanding of both the 3D geometry and the semantics of the environment. Similarly, for a robot, the ability to infer complete 3D shape from partial observations is necessary for low-level tasks such as grasping and obstacle avoidance [173], while the ability to infer the semantic meaning of objects in the scene enables high-level tasks such as retrieval of objects.

With this motivation, our goal is to have a model that predicts both volumetric occupancy (i.e., scene completion) and object category (i.e., scene labeling) from a single depth image of a 3D scene — we refer to this task as **semantic scene completion** (Figure 3.1). Prior work is limited to address only part of this problem as shown in Figure 3.2: RGB-D segmentation approaches consider only visible surfaces without the full 3D shape [47, 130], while shape completion approaches consider only geometry without semantics [33] or a single object out of context [169, 180].

Our key observation is that the occupancy patterns of the environment and the semantic labels of the objects are tightly intertwined. Therefore, the two problems of predicting voxel occupancy and identifying object semantics are strongly coupled. In other words, knowing the identity of an object helps us predict what areas of the scene it is likely to occupy without direct observation (e.g., seeing the top of a chair

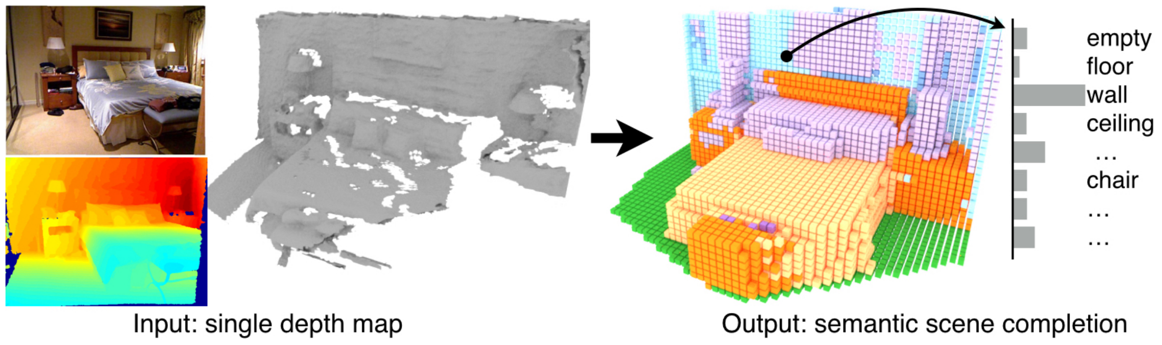


Figure 3.1: **Semantic scene completion.** [Left] Input single-view depth map and visible surface from the depth map; color is for visualization only. [Right] Semantic scene completion result: our model jointly predicts volumetric occupancy and object categories for each of the 3D voxels in the view frustum. Note that the entire volume occupied by the bed is predicted to have the bed category.

behind a table and inferring the presence of a seat and legs). Likewise, having an accurate occupancy pattern for an object helps us recognize its semantic class.

To leverage the coupled nature of the two tasks we jointly train a deep neural network using supervision targeted at both tasks. Given a single-view depth map as input, our semantic scene completion network (**SSCNet**) produces one of N+1 labels for all voxels in the view frustum. Each voxel is labeled as occupied by one of N object categories or free space. Most critically, this prediction extends beyond the projected surface implied by the depth map, thus providing occupancy information for the entire scene.

To achieve this goal there are several issues that must be addressed. First, how do we effectively capture contextual information from 3D volumetric data, where the signal is sparse and lacks high-frequency detail? Second, since existing RGB-D datasets only provide annotations on visible surfaces, how do we obtain training data with complete volumetric annotations at scene level?

To address the first issue, we design a 3D dilation-based context module that effectively expands our network's receptive field to model the contextual information. We find that a big receptive field is crucial for the task. As demonstrated in Figure 3.2, looking at the small region of a chair in isolation, it is hard to recognize and complete the chair. However, if we consider the context due to surrounding objects, such as the table and floor, the problem is much easier.

To address the second issue, we construct SUNCG, a large-scale synthetic 3D scene dataset with more than 45622 indoor environments designed by people. All the 3D scenes are composed of individually labeled 3D object meshes, from which we can compute 3D scene volumes with dense object labels though voxelization.

Our experiments with these solutions demonstrate that a method that jointly predicts volumetric occupancy and object semantic can outperform methods addressing each task in isolation. Both the 3D context model learned by our network and the large-scale synthetic training data help to improve performance significantly.

Our main contribution is to formulate an end-to-end 3D ConvNet model (SSCNet) for the joint task of volumetric scene completion and semantic labeling. In support of that goal, we design a dilation-based 3D context module that enables efficient context learning with large receptive fields. To provide the training data for our network, we introduce SUNCG, a manually created large-scale dataset of synthetic 3D scenes with dense occupancy and semantic annotations.

3.2 Related Work

In this section, we review the related work on RGB-D segmentation, 3D shape completion, and voxel space semantic labeling.

RGB-D semantic segmentation. Many prior works focus on RGB-D image segmentation [47, 130, 149, 91]. However, these methods focus on obtaining semantic labels for only the observed pixels without considering the full shape of the object, and hence cannot directly perform scene completion or predict labels beyond the visible surface.

Shape completion. Other prior works focus on single object shape completion [173, 133, 180, 169]. To apply those methods to scenes, additional segmentation or object masks would be required. For scene completion, when the missing regions are relatively small, methods using plane fitting [116] or object symmetry [84, 110] can be applied to fill in holes. However, these methods heavily rely on the regularity of the geometry and often fail when the missing regions are big. Firman *et al.*[33] show promising completion results on scenes. However, their approach is based purely on geometry without semantics, and thus it produces less accurate results when the scene structure becomes complex.

3D model fitting. One possible approach to obtain the complete geometry and semantic labels for a scene is to retrieve and fit instance-level 3D mesh models to the observed depth map [48, 155, 36, 90, 119, 98, 86]. However, the prediction quality of this type of approach is limited by the quality and variety of 3D models available for retrieval. Naturally, observed objects that cannot be explained by the available models tend to be missed. Or, if the 3D model library is large enough to include all observations, then a difficult retrieval and alignment problem must be solved. Alternatively, it is possible to use 3D primitives such as bounding boxes to approximate the 3D geometry of objects [76, 99, 156]. However, the bounding box approximation limits the geometric detail of the output predictions.

Voxel Space Reasoning. Another line of work completes and labels 3D scenes, but with separate modules for feature extraction and context modeling. Zheng *et al.* [199] predict the unobserved voxels by physical reasoning. Kim *et al.* [83] train a Voxel-CRF model from labeled floor plans to optimize the semantic labeling and reconstruction for indoor scenes. Hane *et al.* [56] and Blaha *et al.*[10] use joint optimization for multi-view reconstruction and segmentation for outdoor scenes. However, this line of work uses predefined features and separates the feature learning from the context modeling, and it is expensive for CRF-based models to encode long-range contextual information. In contrast, our model is able to jointly learn the low-level feature representation and high-level contextual information end-to-end from large-scale 3D scene data, directly modeling long-range contextual cues through a big receptive field.

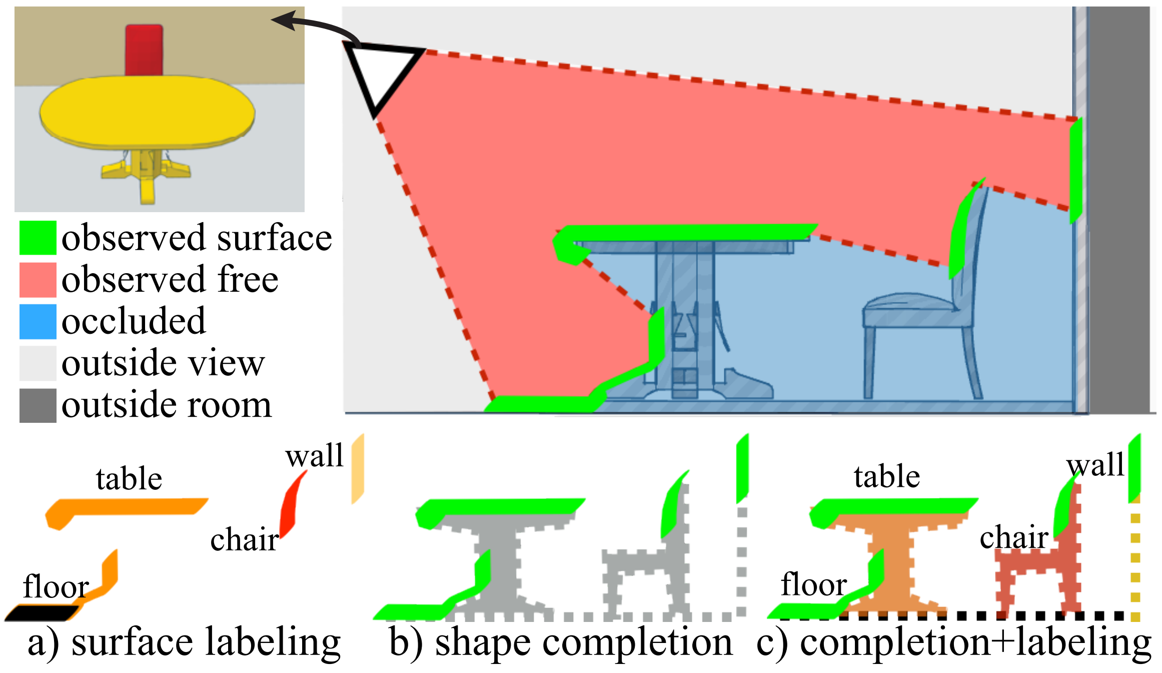


Figure 3.2: Given a single-view depth observation of a 3D scene the goal of our SSCNet is to predict both occupancy and object category for the voxels on the observed surface and occluded regions.

3.3 Semantic Scene Completion Network

Given a single-view depth map observation of a 3D scene, the goal of our semantic scene completion network is to map the voxels in the view frustum to one of the class labels $C = \{c_0, ...c_{N+1}\}$, where N is number of object classes and c_0 represents empty voxels. During training, we render depth maps from virtual viewpoints of our synthetic 3D scenes and voxelize the full 3D scenes with object labels as ground truth. During testing, the observation depth images come from a RGB-D camera.

Figure 3.4 shows an overview of our processing pipeline. We take a single depth map as input and encode it as a 3D volume. This 3D volume is then fed into a 3D convolutional network, which extracts and aggregates both local geometric and contextual information. The network produces the probability distribution of voxel occupancy and object categories for all voxels inside the camera view frustum.

The following subsections describe the core issues addressed in the design of the system: the data encoding (Section 3.3.1), network architecture (Section 3.3.2) and training data generation (Section 3.4).

3.3.1 Volumetric Data Encoding

The first issue we need to address is how to encode the observed depth as input to the network. For the semantic scene completion task, the ideal encoding should directly represent the 2D observation into the same 3D physical space as the output in a way that is invariant to the viewpoint projection, and provide a meaningful signal for the network to learn geometry and scene representation. To this end, we adopt Truncated Signed Distance Function (TSDF) to encode the 3D space, where every voxel stores the distance value d to its closest surface and the sign of the value indicates whether the voxel is in free space or in occluded space. To better suit our task, we make the following modifications to the standard TSDF.

Eliminate view dependency. Most RGB-D reconstruction pipelines speed up the TSDF computation by using the projective TSDF which finds the closest surface points only in the line of sight of the camera [70]. This projective TSDF is fast to compute but is inherently view-dependent. Instead, we choose to compute the distance to the closest point anywhere on the full observed surface.

Eliminate strong gradients in empty space. Another issue with TSDF is that strong gradients occur in the empty space along the occlusion boundary between $\pm d_{\text{max}}$. It is possible to eliminate this gradient by removing the sign. However, the sign is important for completion task since it indicates the occluded regions of the scene that need to be completed. To solve this problem we flip the TSDF value d as follows: $d_{\text{flipped}} = \text{sign}(d)(d_{\text{max}} - d)$. This flipped TSDF has the strong gradient on the surface, providing a more meaningful signal for the network to learn better

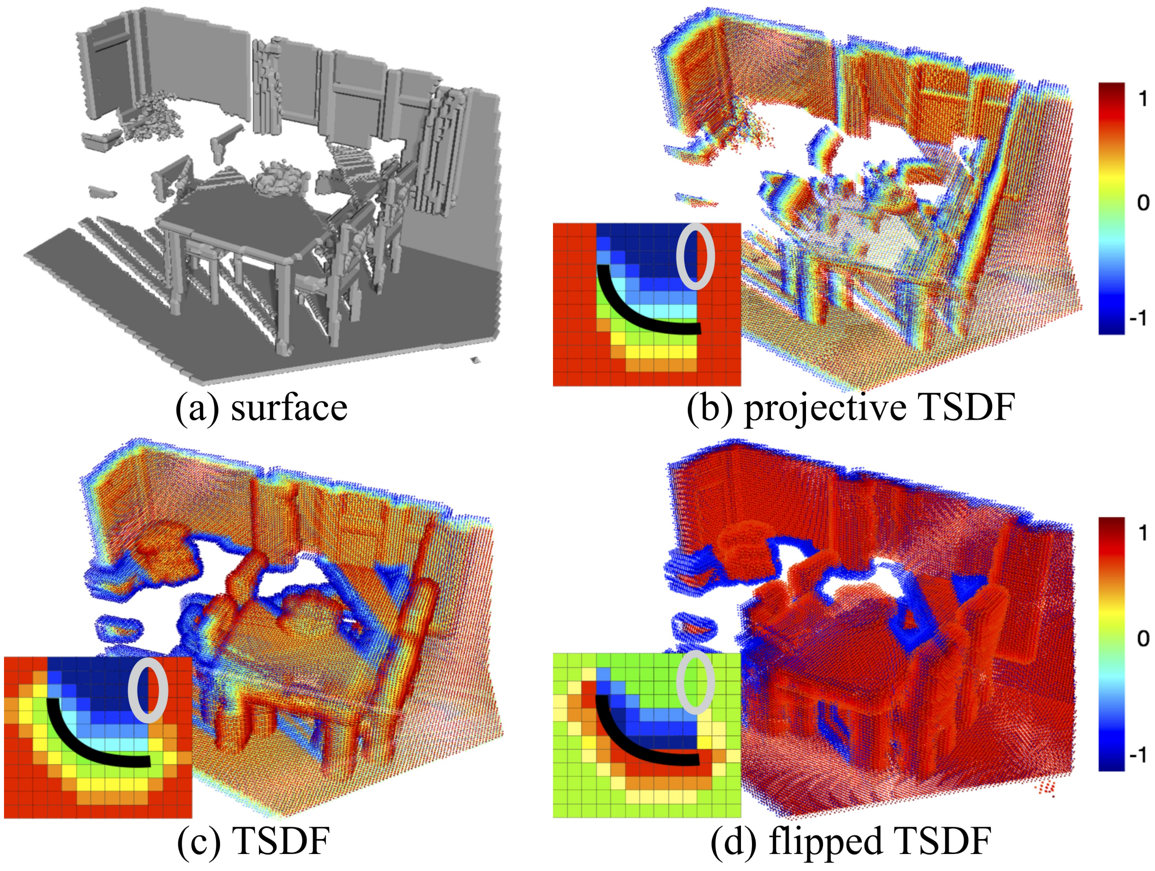


Figure 3.3: **Different encodings for surface (a).** The projective TSDF (b) is computed with respect to the camera and is therefore view-dependent. The accurate TSDF (c) has less view dependency but exhibits strong gradients in empty space along the occlusion boundary (circled in gray). In contrast, the flipped TSDF (d) has the strongest gradient near the surface.

geometric features. The different encoding is visualized in Figure 3.3, and Table 3.3 shows its impact on performance.

3.3.2 Network Architecture

The network architecture of SSCNet is shown in Figure 3.4. Taking a high-resolution 3D volume as input, the network first uses several 3D convolution layers to learn a local geometry representation. We use convolution layers with stride and pooling layers to reduce the resolution to one-fourth of the original input. Then, we use a dilation-based 3D context module to capture higher-level inter-object contextual

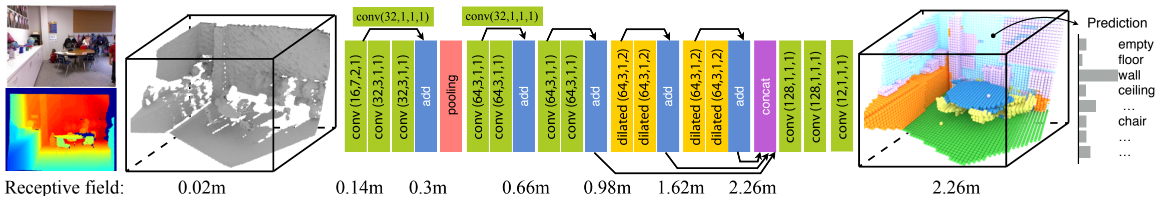


Figure 3.4: **SSCNet: Semantic scene completion network.** Taking a single depth map as input, the network predicts occupancy and object labels for each voxel in the view frustum. The convolution parameters are shown as (number of filters, kernel size, stride, dilation).

information. After that, the network responses from different scales are concatenated and fed into two more convolution layers to aggregate information from multiple scales. At the end, a voxel-wise softmax layer is used to predict the final voxel label. Several shortcut connections are added for better gradient propagation. In implementing this architecture, we made the following design decisions:

Input volume generation. Given a 3D scene, we rotate it to align with gravity and room orientation based on Manhattan assumption. The dimensions of the 3D space we consider are 4.8 m horizontally, 2.88 m vertically, and 4.8 m in depth. We encode the 3D scene into a flipped TSDF with grid size 0.02 m, truncation value 0.24 m, resulting in a $240 \times 144 \times 240$ volume as the network input.

Capturing 3D context with a big receptive field. Context can provide valuable information for understanding the scene, as demonstrated by much prior work in image segmentation [189]. In the 3D domain, context is more useful due to a lack of high-frequency signals compared to image textures. For example, tabletops, beds, and floors are all geometrically similar to flat horizontal surfaces, so it is hard to distinguish them given only local geometry. However, the relative positions of objects in the scene are a powerful discriminatory signal. To learn this contextual information, we need to make sure our network has a big enough receptive field. To this end, we extend the dilated convolution presented by Yu and Koltun [189] to 3D. Dilated convolution

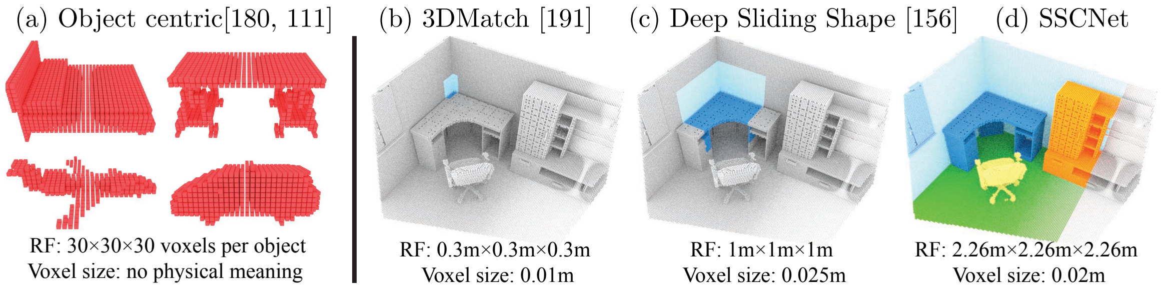


Figure 3.5: **Comparison of receptive fields and voxel sizes between SSCNet and prior work.** (a) Object-centric networks such as [180] and [111] scale objects into the same 3D voxel grid thus discarding physical size information. In (b)-(d), colored regions indicate the effective receptive field of a single neuron in the last layer of each 3D ConvNet. With the help of 3D dilated convolution SSCNet drastically increases its receptive field compared to other 3D ConvNet architectures [156, 191] thus capturing richer 3D contextual information.

extends normal convolution by adding a step size when the convolution extracts values from the input before convolving with the kernel. Thus we can exponentially expand the receptive field without a loss of resolution or coverage, while still using the same number of parameters. Figure 3.5 compares the receptive field size of SSCNet with 3D ConvNet architectures from prior work.

Multi-scale context aggregation. Different object categories have very different physical 3D sizes. This implies that the network will need to capture information at different scales in order to recognize objects reliably. For example, we need more local information to recognize smaller objects like TVs, while we need more global information to recognize bigger objects like beds. In order to aggregate information at different scales we add a layer that concatenates the network responses with a different receptive field. We then feed this combined feature map into two $1 \times 1 \times 1$ convolution layers, which allows us to propagate information across responses from different scales.

Data balancing. Due to the sparsity of 3D data, the ratio of empty vs. occupied voxels is around 9:1. To deal with this imbalanced data distribution, we sample the training so that each mini-batch has a balanced set of empty and occupied examples. For each training volume containing N occupied voxels, we randomly sample $2N$ empty voxels from occluded regions for training. Voxels in free space, outside the field of view, or outside the room are ignored.

Loss: voxel-wise softmax. The loss function of the network is the sum of voxel-wise softmax loss $L(p, y) = \sum_{i,j,k} w_{ijk} L_{\text{sm}}(p_{ijk}, y_{ijk})$, where L_{sm} is softmax loss, y_{ijk} is the ground truth label, p_{ijk} is the predicted probability of the voxel at coordinates (i, j, k) over the $N + 1$ classes, where N is the number of object categories and empty voxels are labeled as class 0. The weight w_{ijk} is equal to zero or one based on the sampling algorithm described above.

Training protocol. We implement our network architecture in Caffe [74]. Pre-training SSCNet on the SUNCG training set takes around a week on a Tesla K40 GPU, and fine-tuning on the NYU dataset takes 30 hours. During training, each mini-batch contains one 3D view volume, requiring 11 GB of GPU memory. To obtain more stable gradient estimates, we accumulate gradients over four iterations and update the weights once afterward.

3.4 Synthesizing training data

One of the main obstacles of training deep networks for scene-level dense 3D predictions is the lack of large annotated datasets with dense object semantic annotations at the voxel level. Existing RGB-D datasets with surface reconstructions are subject to occlusions or partial observations, and cannot provide the volumetric occupancy and semantic labels for the entire space at the voxel level. To obtain volumetric occupancy ground truth Firman *et al.* [33] collect a tabletop dataset with reconstructed RGB-D video using KinectFusion [70]. However, this data does not provide semantic labels,

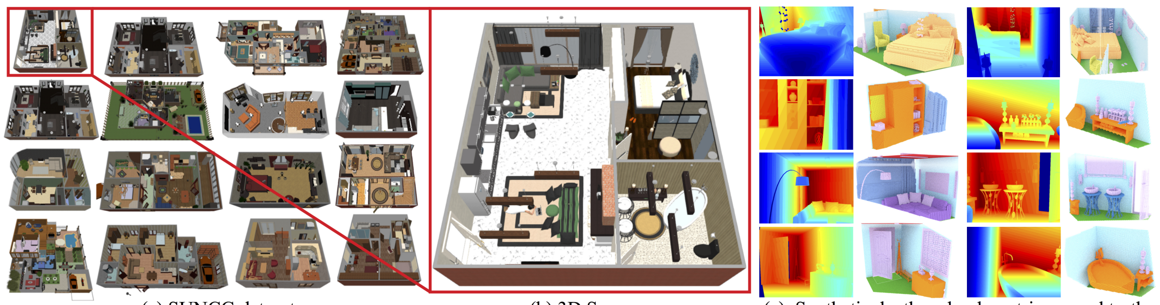

(a) SUNCG dataset (b) 3D Scene (c) Synthetic depth and volumetric ground truth

Figure 3.6: **Synthesizing Training Data.** We collected a large-scale synthetic 3D scene dataset to train our network. For each of the 3D scenes, we select a set of camera positions and generate pairs of rendered depth images and volumetric ground truth as training examples.

	scene completion			semantic scene completion											
method (train)	prec.	recall	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	avg.
Lin *et al.*(NYU) [99]	58.5	49.9	36.4	0	11.7	13.3	**14.1**	9.4	29	24	6.0	7.0	16.2	1.1	12.0
Geiger and Wang (NYU) [36]	65.7	58	44.4	10.2	62.5	19.1	5.8	8.5	40.6	27.7	7.0	6.0	22.6	5.9	19.6
SSCNet (NYU)	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
SSCNet (SUNCG)	55.6	91.9	53.2	5.8	81.8	19.6	5.4	12.9	34.4	26	13.6	6.1	9.4	7.4	20.2
SSCNet (SUNCG+NYU)	**59.3**	92.9	**56.6**	**15.1**	94.6	**24.7**	10.8	**17.3**	**53.2**	**45.9**	**15.9**	**13.9**	**31.1**	**12.6**	**30.5**

Table 3.1: Semantic scene completion results on the NYU with kinect depth map.

and only contains simple tabletop scenarios. To address these issues we constructed and make use of a new large-scale synthetic 3D scene dataset -SUNCG, from which we obtain a large amount of training data with synthetically rendered depth images and volumetric ground truth. Details about SUNCG dataset can be found in Section 5.2. Here we focus on describing how we use this dataset to generate training for the semantic scene completion task, which includes synthetic depth map generation and volumetric ground truth generation.

3.4.1 Synthetic depth map generation

To generate synthetic depth maps that mimic a typical image capturing process, we use a set of simple heuristics to pick camera viewpoints. Given a 3D scene, we start with a uniform grid of locations spaced at 1 m intervals on the floor and not occupied by objects. We then choose camera poses based on the distribution of the

NYU-Depth v2 dataset.[1] Then, we render the depth map using the intrinsic and resolution of the Kinect. After that, we use a set of simple heuristics to exclude bad viewpoints. Specifically, a rendered view is considered valid if it satisfies the following three criteria: a) valid depth area (depth values in range of $1\,\text{m}$ to $8\,\text{m}$) larger than 70% of image area, b) there are more than two object categories apart from wall, ceiling, floor, and c) object area apart from wall, ceiling, floor is larger than 30% of image area. To reduce data redundancy, we pick at most five images from each room. In total, we generate 130 K valid views for training our SSCNet.

3.4.2 Volumetric ground truth generation

Since the 3D scenes in the SUNCG dataset consist of a limited number of object instances, we speed up the voxelization process by first voxelizing each individual object in the library and then transforming the labels based on each scene configuration and viewpoint. Specifically, we first voxelize each object to a $128 \times 128 \times 128$ voxel grid. We set the voxel size s so that the largest dimension of the object is a tight fit to the object bounding box. Thus, s varies between objects due to the difference in object dimensions. We use the binvox [114] voxelizer which accounts for both surface and interior voxels by using a space carving approach.

Given a camera view, we define a $240 \times 144 \times 240$ voxel grid in world coordinates, with scene voxel size equals to $2\,\text{cm}$. Then for each object in the scene, we transform the object voxel grid by translating, rotating and scaling by the object's transformation. We then iterate over each voxel in the scene voxel grid that is inside the transformed object bounding box and calculate the distance to the nearest neighbor object voxel. If the distance is smaller than the object voxel size s, this scene voxel will be labeled with this object category. Similarly, we label all voxels in the scene that belong to walls, floors, and ceilings by treating them as planes with

[1]The camera height is sampled from a Gaussian distribution with $\mu = 1.5\,\text{m}$ and $\sigma = 0.1\,\text{m}$. The camera tilt angle is sampled from a Gaussian distribution with $\mu = -10°$ and $\sigma = 5°$.

a thickness equal to one scene voxel size. All remaining voxels are marked as empty space, therefore providing a fully labeled voxel grid for the camera view.

3.5 Evaluation

In this section, we evaluate our proposed methods with a comparison to alternative approaches and an ablation study to better understand the proposed model. We evaluate our algorithm on both real and synthetic datasets.

For the real data, we use the NYU dataset [149], which contains 1449 depth maps captured from Kinect. We obtain the ground truth by voxelizing the 3D mesh annotations from Guo *et al.* [45], mapping object categories based on Handa *et al.* [54]. The annotations consist of 33 object meshes in 7 categories, other categories approximated using 3D boxes or planes. In some cases, the mesh annotation is not perfectly aligned with depth due to labeling error and the limited set of meshes. To deal with this misalignment, Firman at el. [33] propose to use rendered depth map from the annotation for testing. However, by rendering the overly simplified meshes, geometric detail is lost especially in cases where objects are represented as a box. Therefore, we test with both rendered depth maps and the originals.

For synthetic data, we created a test set from SUNCG which has objects with detailed geometry, and for which the depth map and ground truth volumes are perfectly aligned. The SUNCG test set are rendered from 184 scenes that are not in the training set.

3.5.1 Evaluation metric.

As our evaluation metric, we use the voxel-level intersection over union (IoU) of predicted voxel labels compared to ground truth labels. For the semantic scene completion task, we evaluate the IoU of each object classes on both the observed and

method	training	prec.	recall	IoU
Zheng *et al.* [199]	NYU	60.1	46.7	34.6
Firman *et al.* [33]	NYU	66.5	69.7	50.8
SSCNet completion	NYU	66.3	96.9	64.8
SSCNet joint	NYU	75.0	92.3	70.3
SSCNet joint	SUNCG+NYU	**75.0**	**96.0**	**73.0**

Table 3.2: **Scene completion on the rendered NYU test set as [33]**

occluded voxels. For the scene completion task, we treat all non-empty object class as one category and evaluate IoU of the binary predictions on occluded voxels. Following Firman *et al.* [33], we do not evaluate on voxels outside the view or the room.

3.5.2 Experimental results

Table 3.2 and Table 3.1 summarize the quantitative results and Figure 3.8 shows qualitative comparisons.

Comparison to alternative approaches. In Figure 3.1 we compare on the semantic scene completion task with approaches from Lin *et al.* [99] and Geiger and Wang [36]. Both these algorithms take an RGB-D frame as input and produce object labels in the 3D scene. Lin *et al.*use 3D bounding boxes and planes to approximate all objects. Geiger and Wang retrieve and fit 3D mesh models to the observed depth map at test time. The mesh model library used for retrieval is a superset of the models used for ground truth annotations. Therefore, they can achieve perfect alignments by finding the exact mesh model in a small database. In contrast, our algorithm is based on only depth and does not use additional mesh model at test time. Despite this data disparity, our network produces more accurate voxel-level predictions (30.5% vs. 19.6%). An example of the difference is shown in the third row of Figure 3.8: both Lin *et al.*and Geiger and Wang's approaches confuse the sofa as a bed while our network correctly recognizes the sofa. Moreover, since our method does not require the model fitting step it is much faster at 7s compared to 127s per image [36].

Does recognizing objects help scene completion? Previous work has shown scene completion is possible without semantic understanding. We examine to what extent the supervision of object semantics benefits the scene completion task. To do this, we trained a model predicting the occupancy of each voxel by doing binary classification on each voxel ("empty" vs. "occupied"). We compare the performance of models trained with occupancy and multi-class labeling (see Figure 3.2 [completion] vs. [joint]). We also compare with Formal *et al.* [33] and Zheng *et al.* [199] which both predict binary voxel occupancy based on a single depth map without a semantic understanding of the scene. We use the re-implementation of Zheng *et al.*'s approach from Firman *et al.*, which only provides the completion result. We evaluate on the rendered NYU benchmark with the same test images used by Firman at al. (randomly picked 200 images from the full test set). While Firman *et al.*produces good results for many cases, their approach fails when the scene becomes complex. For instance, their algorithm fails to complete half of the bed in the first row of Figure 3.8, and also fails to complete the chairs in the fifth row. In contrast, SSCNet is able to better complete the geometry by leveraging the semantics of the 3D context. This result validates the idea that it is beneficial to understand object semantics in order to achieve better scene completion.

Does scene completion help in recognizing objects? To answer this question, we trained a model with a loss only accounting for semantic labels evaluated on the visible surface and compared with the model trained jointly with labeling and completion (see Table 3.3 [no completion] vs. [joint]). Even when only evaluating on the visible surface, the model trained with the added supervision of the scene completion task outperforms the model trained only on surface labeling (54.2% vs.

		scene completion			semantic scene completion											
method	encoding	prec.	recall	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	avg.
no completion	flipped	-	-	-	97.2	95.5	61.9	24.6	30.1	55.3	58.9	48.7	14.8	42.1	34.5	51.2
joint	flipped	-	-	-	97.7	94.5	66.4	30.0	36.9	60.2	62.5	56.3	12.1	46.7	33.0	54.2
no balancing	flipped	73.1	95.8	70.8	**96.4**	85.3	52.1	25.8	16.5	47.1	45.7	28.1	15.3	37.1	19.8	42.7
Basic	flipped	73.4	95.0	70.7	94.6	83.8	47.0	24.0	15.1	38.2	37.2	26.0	0.0	34.8	17.3	38.0
Basic+D	flipped	72.2	96.2	70.4	94.7	**85.9**	47.5	**29.2**	21.1	50.9	50.7	29.0	**21.3**	37.2	20.1	44.3
Basic+D+M	proj	72.0	92.3	67.9	91.6	80.9	45.1	14.6	10.2	39.4	29.8	19.8	0.0	27.4	14.3	33.9
Basic+D+M	tsdf	74.8	94.0	71.4	95.8	84.4	45.1	17.5	15.2	28.2	37.2	25.6	0.0	28.2	21.9	36.3
Basic+D+M	flipped	**76.3**	**95.2**	**73.5**	96.3	84.9	**56.8**	28.2	**21.3**	**56.0**	**52.7**	**33.7**	10.9	**44.3**	**25.4**	**46.4**

Table 3.3: **Ablation study on SUNCG testset.** First two row shows the evaluation on surface segmentation with and without joint training. The following rows show the evaluation on semantic scene completion task. D: 3D dilated convolution. M: multi-scale aggregation.

51.2%). This demonstrates that understanding complete object geometry and the 3D context is beneficial for recognizing objects.

Does synthetic data help? To investigate the effect of using synthetic training data, we compared models trained only with NYU and models pre-trained on SUNCG and then fine-tuned on NYU (see Table 3.2,3.1 NYU vs. NYU+SUNCG). We see a performance gain by using additional synthetic data especially for the semantic scene completion task having a 10.3% improvement in IoU.

Does a bigger receptive field help? In Table 3.3, the networks labeled [Basic] and [Basic+D] have the same number of parameter, while in [Basic+D] three convolution layers are replaced by dilated convolution, increasing the receptive field from 1.16 m to 2.26 m. Increasing the receptive field gives the network a opportunity to capture richer contextual information and significantly improve the network performance from 38.0% to 44.3%. To visualize the contextual information learned by the network, we perform the following experiment: given a depth map of a single object we predict labels for all unobserved voxels. Figure 3.7 shows the input depth and the predictions. Even without observing depth information for other objects SSCNet hallucinates plausible contextual object based on the observed object.

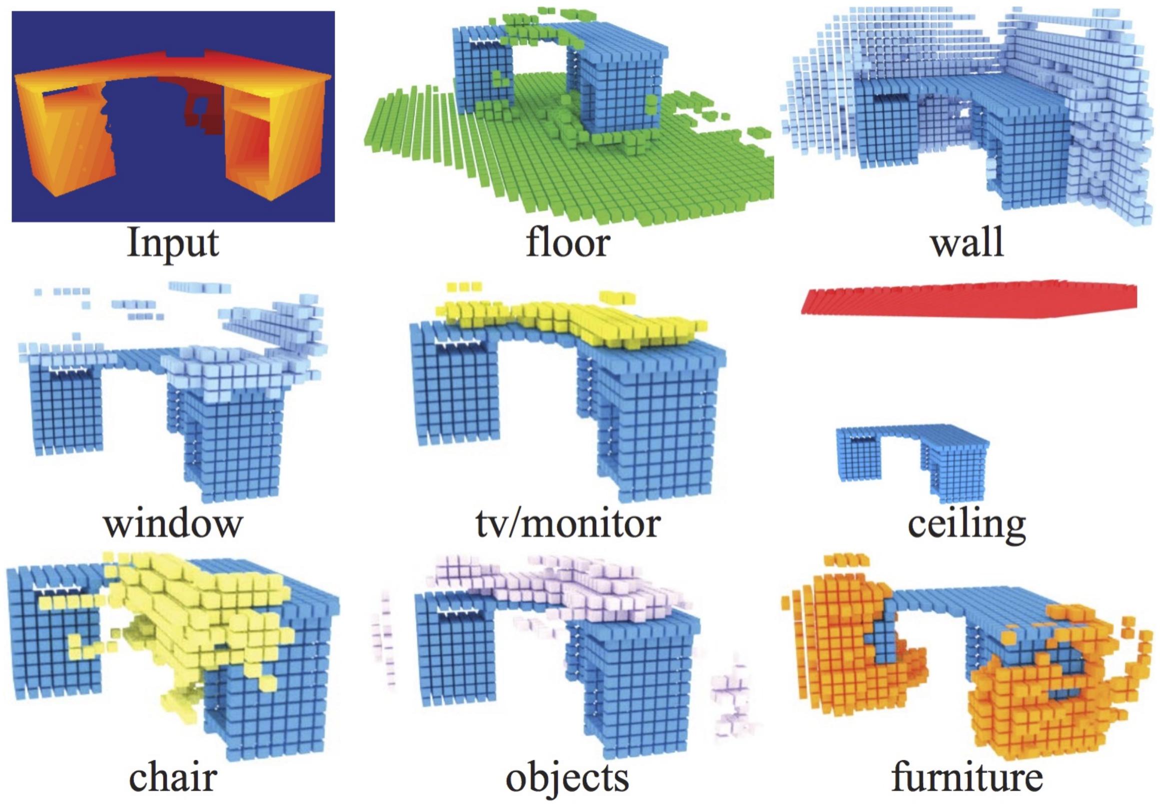


Figure 3.7: **What 3D context does the network learn?** The first figure shows the input depth map (a desk) and the following figures show the predictions for other objects. Without observing any information for other objects the SSCNet is able to hallucinate their locations based on the observed object and the learned 3D context.

Does multi-scale aggregation help? Comparing the network performance with and without the aggregation layer (see Table 3.3 [Basic+D] vs. [Basic+D+M]), we observe that the model with aggregation yields higher IoU for both the scene completion and semantic scene completion tasks by 3.1% and 2.1% respectively.

Do different encodings matter? The last three rows in Table 3.3 compare different volumetric encodings: projective TSDF [proj.], accurate TSDF [tsdf], and flipped TSDF [flipped]. We observe that removing the view dependency by using the accurate TSDF gives a 2.4% improvement in IoU. Making the gradients concentrated on the surface with the flipped TSDF leads to a 10.1% improvement.

Is data balancing necessary? To balance the empty and occupied voxel examples, we proposed to sample the empty voxels during training. In Table 3.3, [no balancing] shows the performance when we remove the sampling process during training, where we see a drop in IoU from 46.4% to 42.7%.

Limitations. Firstly, we do not use any color information, so objects missing depth such as "windows" are hard to handle. This also leads to confusion between objects with similar geometry or functionality. For example, in the second row of Figure 3.8 the network predicts the desk as the broader furniture category. Secondly, due to the GPU memory constraints, our network output resolution is lower than that of input volume. This results in less detailed geometry and missing small objects, such as the missed objects on the desk of the second row in Figure 3.8.

3.6 Summary

In this chapter, we introduced **semantic scene completion**, a task for producing a complete 3D voxel representation of volumetric occupancy and semantic labels for a scene from a single-view depth map observation.

To address this task, we proposed SSCNet, a 3D ConvNet for the semantic scene completion task of jointly predicting volumetric occupancy and semantic labels for full 3D scenes. To train our network, we constructed SUNCG – a manually created large-scale dataset of synthetic3D scenes with dense volumetric annotations. The experiment results demonstrate that our joint model outperforms methods addressing either component task in isolation, and by leveraging the 3D contextual information and the synthetic training data, we significantly outperform alternative approaches on the semantic scene completion task.

While SSCNet is powerful for extracting full 3D amodal scene representations from sparse 3D scans, the algorithm is only able to describe the scene within the

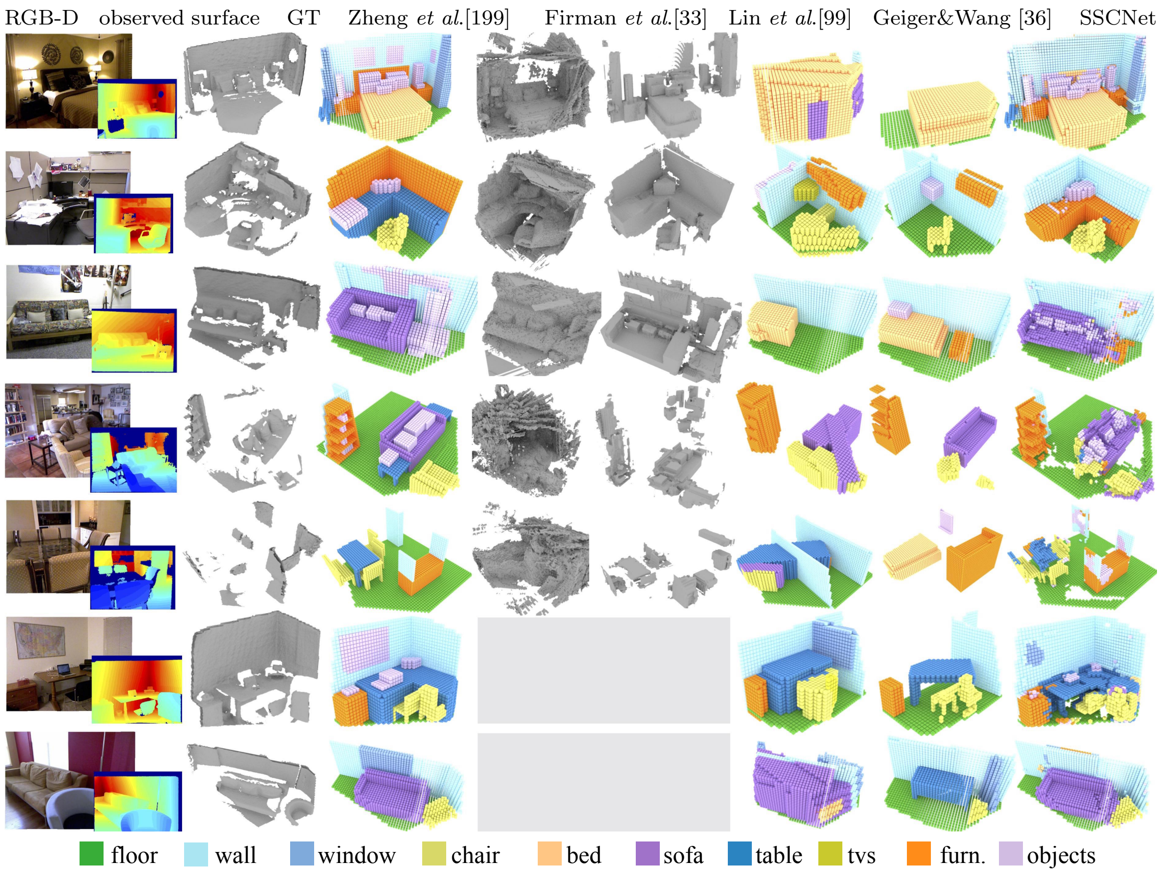


Figure 3.8: **Qualitative results.** We compare with scene completion results from Zheng *et al.* [199] and Firman *et al.* [33] (on a subset of the test set), and semantic scene completion results from Lin *et al.* [99] and Geiger and Wang [36]. Zheng *et al.*[33] tested on rendered depth, [99] and [36] tested on RGB-D frame from kinect, [33] and SSCNet tested on kinect depth. Overall, SSCNet gives more accurate voxel predictions such as the lamps and pillows in the first row, and the sofa in the third row.

camera field of view (FoV). However, cameras mounted on most robots and wearable devices have restricted FoVs. Hence, extrapolating useful 3D information beyond the FoV plays an important role in applications such as scene parsing [195], goal-driven navigation, and next-best-view approximation, where estimating the complete 3D environment naturally improves preemptive planning for these systems.

In the next chapter, we will explore the task of inferring the information of the 3D environment beyond camera field of view by leveraging the strong contextual information presented in the typical indoor environments – semantic-structure view extrapolation.

Chapter 4

Understanding 3D Scenes Beyond the Field of View

Extrapolating useful information outside the camera's field of view (FOV) is important for robotic tasks such as goal-driven navigation, where a full representation of the environment can improve high-level planning. In this chapter, we focus on semantic-structure view extrapolation – a task to infer the complete surrounding environment in the form of 3D structure and semantic information, given a partial observation. To address this task, we present Im2Pano3D, a convolutional neural network that generates a dense prediction of 3D structure and a probability distribution of semantic labels for a full $360°$ panoramic view of an indoor scene when given only a partial observation ($\leq 50\%$) in the form of an RGB-D image.

4.1 Semantic-Structure View Extrapolation

People possess an incredible ability to infer contextual information from a single image [67]. Whether it is by using prior experience or by leveraging visual cues [6, 107], people are adept at reasoning about what may lie beyond the field of view and make use of that information for building a coherent perception of the world [65].

Similarly, in robotics and computer vision, extrapolating useful information outside a camera's field of view (FOV) plays an important role for many applications, such as goal-driven navigation[202, 16] or next-best-view approximation [73], where a global representation of the environment can improve preemptive planning for intelligent systems.

However, prior work in view extrapolation typically only predicts the color pixels beyond the image boundaries [124, 197, 142]. While inspiring, these methods do not predict 3D structure or semantics, and hence cannot be used directly for high-level reasoning tasks in robotics applications.

In this chapter, we explore the task of directly extrapolating 3D structure and semantics for a full panoramic view of a scene when given a view covering 50% or less as input. We refer to this task as **semantic-structure view extrapolation**. Our method, Im2Pano3D, takes in a partial view of an indoor scene (e.g., a few RGB-D images) and uses a convolutional neural network to generate dense predictions for 3D structure and a probability distribution of semantic labels for a full 360° panoramic view of that same scene.

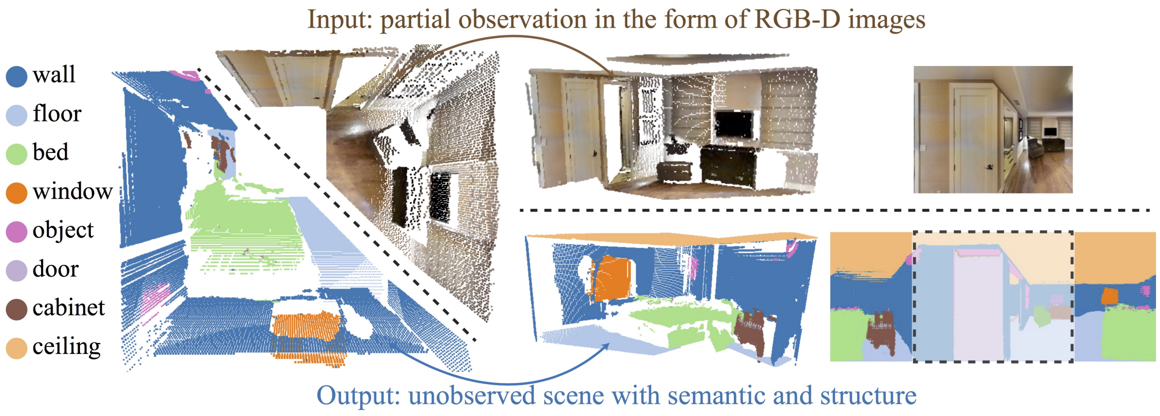

Input: partial observation in the form of RGB-D images
wall
floor
bed
window
object
door
cabinet
ceiling
Output: unobserved scene with semantic and structure

Figure 4.1: **Semantic-structure view extrapolation.** Given a partial observation of the room in the form of an RGB-D image, our Im2Pano3D predicts both 3D structure and semantics for a full panoramic view of the same scene.

This is a very challenging task. However, by learning the statistics of many typical room layouts, we can train a data-driven model to leverage contextual cues to predict what is beyond the field of view for typical indoor environments. For example, as shown in Fig.4.1, given half of a bedroom (180 °horizontal field of view), the system can predict the 3D structure and semantics for the other half. This requires it not only to extend the partially observed room structures (walls, floor, ceiling, etc.), but also to predict the existence and locations of objects that are not directly observed in the input (bed, window and cabinet) using statistical properties learned from data.

Semantic-structure view extrapolation poses three main challenges, which we address with corresponding key ideas shaping our approach to the task:

- How to leverage strong contextual priors for indoor environments.
- How to represent the 3D structure in a way that is good not only for recognition but also for reconstruction.
- How to design meaningful loss functions when the possible solution is not unique – a small change to object locations may still result in a valid solution.

To leverage strong *contextual priors* for indoor environments, we represent 3D scenes in a single panorama image with channels encoding 3D structure and semantics. We train our model over a large-scale synthetic (SUNCG [157]) and real-world indoor scenes (Matterport3D [20]) encoded in this representation to learn the contextual prior.

To leverage strong *geometric priors* for indoor environments, we represent the 3D structure for each pixel with a 3D plane equation, rather than raw depth value at each pixel. By doing so, we take advantage of the fact that indoor environments are often comprised largely of planar surfaces. Since all pixels on the same planar surface have the same plane equation, the 3D structure is piecewise constant in a typical scene, which makes dense predictions of plane equations more robust than alternative representations.

To provide *meaningful supervision* for the network to cover the large solution space, we make use of multiple loss functions that account for both pixel level accuracy (pixel-wise reconstruction loss) and global context consistency (adversarial loss, and scene attribute loss).

The primary contribution of our paper is to propose the task of semantic-structure view extrapolation and present Im2Pano3D, a unified framework able to produce a complete room structure and semantic labeling when given a partial observation of a scene. This unified framework is able to handle different camera configurations and input modalities. The experimental results show that direct prediction of the 3D structure and semantics for the unobserved scene provides a more accurate result than alternative methods. Both the plane equation encoding and the context model learned from multi-level supervision with large scale indoor scenes help to improve prediction quality.

4.2 Related Work

The general scene understanding problem focuses on understanding what is present in an image, including scene classification [97, 182], semantic segmentation [104], depth and normal estimation [35, 176], etc. In this section, we review prior work on these tasks beyond the visible scene.

Texture synthesis and image inpainting. Texture synthesis methods can be used for image hole filling and image extrapolation [25, 62]. For example, Barnes et al. [8] fills holes by cloning structures from similar patches. Pathak et. al [124] train an autoencoder network. These methods can achieve very impressive inpainting results for holes in color images. However, it is challenging for them to predict image content far outside the field of view, since they don't explicitly model structure and semantics.

Stitching images from the Internet. Methods have also been proposed to extrapolate images drastically beyond the field of view using collections of Internet images. For example, Shan et al. [142] produce "uncropped images" by stitching together collections of images captured in the same scene. Hays and Efros [57] fill large holes by copying content from similar images in a large collection. While these methods produce impressive results, they only work for scenes where collections are available with many images from nearby viewpoints.

User-guided view extrapolation. FrameBreak [197] performs dramatic view extrapolation. However, it uses a "guide image" provided by a person to constrain the image synthesis process. The guide image is chosen from a collection of panorama images, aligned with the input image, and then used to guide a patch-based texture synthesis algorithm. In this work, we aim to produce an image extrapolation framework that can be used for any common indoor environment without human intervention.

Predicting 3D structure in occluded regions. Recently there have been many works addressing the problem of shape completion for individual objects [173, 133, 180] or scenes [174, 157, 33]. Given a partial observation of an object or scene, the task is to complete the shape of object in the occluded regions within the field of view. Unlike these methods, Im2Pano3D needs to predict the 3D structure outside the field of view, where there is no direct observation, which makes the problem much harder.

Predicting semantic concepts beyond the visible scene. Khosla *et al.*[82] propose a framework to predict the locations of semantic concepts outside the visible scene, e.g., answering questions like "where can I find a restaurant" given a street-view image without direct sight of any restaurant. Although related, their work focuses on outdoor street view scenes and provides only high-level sparse semantic predictions. In contrast, we produce dense pixel-wise predictions for both 3D structure and semantics for pixels outside the observed view for indoor scenes.

4.3 Im2Pano3D Network

We formulate the semantic-structure view extrapolation problem as an image inpainting task by representing both the input observation and output prediction as multi-channel panoramic images. The goal of Im2Pano3D is to predict the 3D structure and semantics for all missing regions in the input panorama. For the semantic prediction, instead of representing it as a discrete category, we model it as a probability distribution over all semantic categories as shown in Fig.4.7, which explicitly models the prediction uncertainty.

4.3.1 Whole Room Panoramic Representation

Traditional view synthesis works [141, 140] represent observations and new views using a set of disjoint images with their camera parameters. However this requires the network to handle arbitrary numbers of input views, infer spatial relationships between them, and reason about how scene elements cross image boundaries.

In contrast, we propose to represent the 3D scene using a single panorama where each pixel is labeled with multiple channels of information (color, 3D structure, and semantic) or marked as unobserved. This data representation allows the network to learn a consistent whole-room context model by describing both the observed and unobserved parts of the entire scene from a single viewpoint. It is particularly efficient for deep learning because the observations and predictions are resampled in a regular 2D parameterization suitable for convolution. Meanwhile, it can naturally support different input camera configurations through reprojection (see Fig.4.10). Given an observation of a 3D scene reconstructed from registered RGB-D images, we pick a virtual camera center and render the mesh onto four perspective image planes in a sky-box like fashion (see Fig.4.2). Each image plane has a 90° horizontal FoV and a 116° vertical FoV with a image size 256×160. Virtual camera centers are chosen depending on the dataset: for the Matterport3D dataset, we use tripod locations; for the SUNCG dataset, we randomly select locations in empty space; for short RGB-D videos, we use the median of all camera centers.

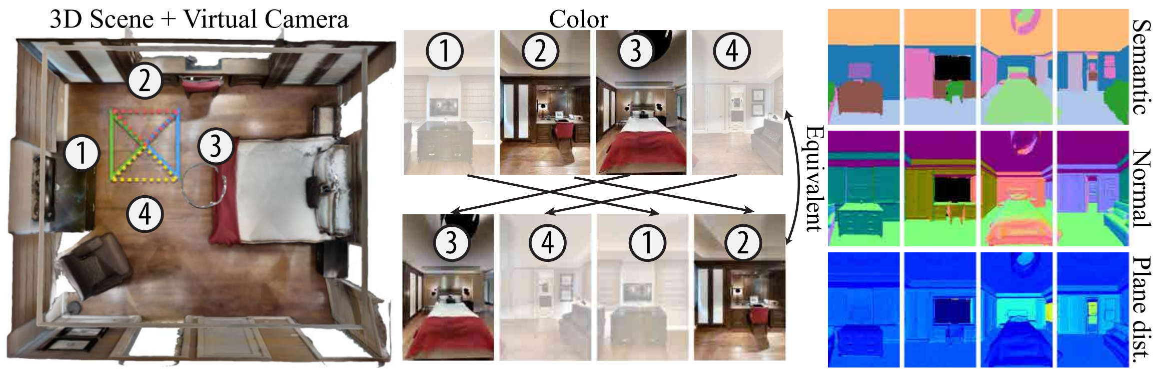


Figure 4.2: **Whole room representation.** We use a sky-box-like multi-channel panorama to represent 3D scenes. The views are circularly connected, hence, observing the inner two views is equivalent to observing the outer two views of its shifted panorama.

4.3.2 Representing 3D Surfaces with Plane Equations

While deep networks have been shown to perform well for predicting color pixels and semantic labels, they continue to struggle at predicting high-quality 3D structure. Current methods for direct regressing raw depth values produce blurred results [168, 95, 26], partly due to the viewpoint-dependent nature of depth maps and the large

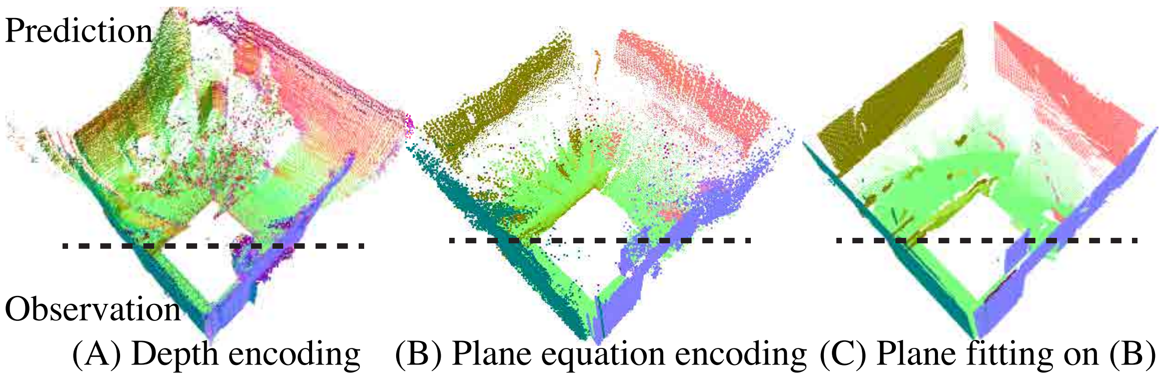


Figure 4.3: **3D structure prediction with different encodings.** The plane equation encoding (B) is a better output representation than raw depth encoding (A); its regularization enables the network to predict higher quality geometry.

value variance of depth values even for nearby pixels on the same 3D plane. Surface normal predictions are generally higher quality; however, solving depth from normals is under-constrained and sensitive to noise. Other more complicated encodings, such as HHA [49], are designed for recognition, but cannot be used directly to recover the 3D structure.

In response to these issues, we propose to represent 3D surfaces with their plane equations: surface normal n and plane distance p to the virtual camera origin. We expect this representation to be easier to predict in indoor environments composed of large planar surfaces because all pixels on the same planar surface share the same plane equation – i.e., the representation is mostly piecewise constant. Moreover, the 3D location of each pixel can be solved trivially from its plane equation by intersection with a camera ray.

Our network is trained to optimize the predicted plane equations. We find this representation of 3D structure to be more effective than raw depth values. Fig.4.3 shows the qualitative comparison. We also have a post-processing step to further improve visual quality of the predicted geometry using plane-fitting on the predicted parameters (this step is not included in our quantitative evaluations).

4.3.3 Im2Pano3D Network Architecture

Our network architecture follows an encoder-decoder structure (Fig.4.4), where the encoder produces a latent vector from an input panorama with missing regions, and the decoder uses that latent vector to produce an output panorama where the missing regions are filled. In this section, we discuss the key features of our network architecture.

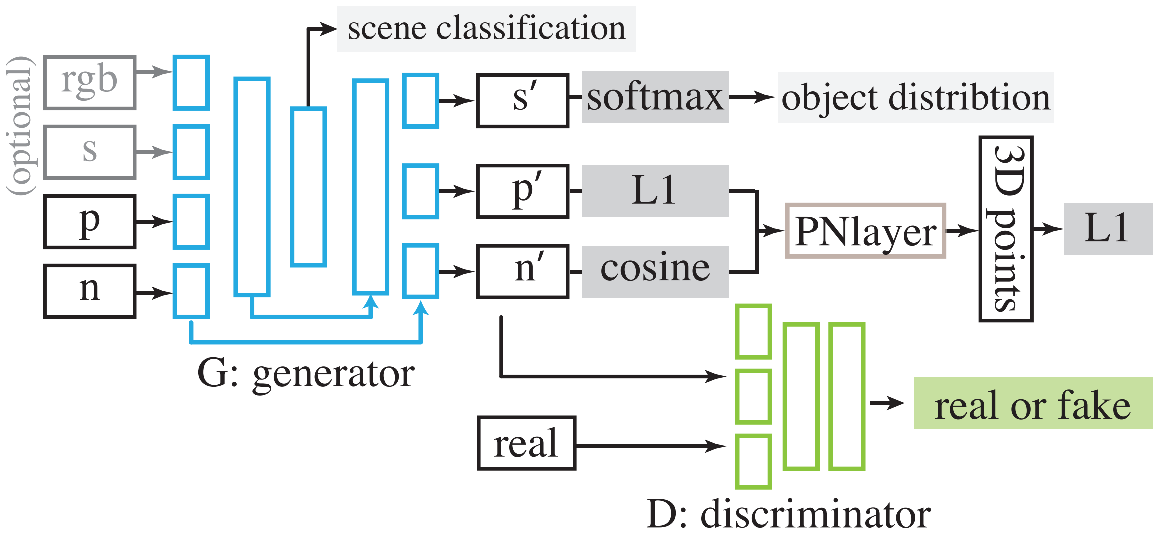


Figure 4.4: **Im2Pano3D network architecture.** the network uses a multi-stream autoencoder structure. A PN-layer is used to ensure consistency between normal and plane distance predictions.

Multi-stream network. Since our panoramic data representation consists of multiple channels (*e.g.*color, normal, plane distance to the origin, and probability distribution of semantics), we structured our network to process each channel with disjoint streams before merging into and after splitting from the middle layers. In the encoder, each stream is made up of three convolutional layers. The features produced from each stream are merged together by concatenation across channels and then passed through six joint convolutions layers to produce the latent vector. Mirroring this structure, the decoder passes the latent vector through six joint convolutions layers before splitting into multiple streams. This multi-stream structure provides the network a balance of learning both channel-specific parameters within each stream, and joint information through shared layers.

Reconstructing 3D surfaces with PN-Layer Although our network architecture predicts the parameters of the plane equation as separate channels (surface normals n and plane distances p), there is no explicit supervision to enforce the consistency between these two outputs. As a result, we find that with only the individual supervision, the 3D surfaces reconstructed from the predicted parameters tend to be noisy. To address this issue, we designed an additional layer in the network (called the PN-Layer) which takes the normal and plane distances as input, and uses the plane equation to produce a dense map of 3D point locations (x, y, z) for each pixel based on its respectively predicted n, p, and pixel location. This layer is fully differentiable, and therefore an additional regression loss can be added on the predicted 3D point locations in order to enforce the consistency between the surface normal and plane distance predictions.

4.3.4 Im2Pano3D Network Losses

When predicting the scene content for the unobserved regions, the plausible solution might not be unique. For example, a valid prediction with slight changes to its locations could still represent an valid solution. To provide the supervision that reflects this flexibility, we use multiple losses to capture three levels of information: pixel-wise accuracy, mid-level contextual consistency using Patch-GAN (adversarial) loss [69], and global scene consistency measured by scene category and object distributions. The final loss for each channel is a weighted sum of the three level losses:

Pixel-wise reconstruction loss. As part of network supervision, we backpropagate gradients based on the pixel-level reconstruction loss between the prediction and the ground truth panoramas. The loss differs for each output channel. We use softmax loss for semantic segmentation s, cosine loss for normal n, and $L1$ loss for plane distance p and final 3D point locations (x, y, z).

Adversarial loss. Following the recent success of generative adversarial networks, we model supervision for generating high-frequency structures in the output panoramas by using a discriminator network [41] adapt from PatchGAN [69]. Similar to the generator, the discriminator network processes each channel with disjoint streams before merging features into shared layers. For the real semantic examples, we converted them into a probabilistic distribution over C classes of size $H \times W \times C$ before feeding them into the discriminator. We adopt the method proposed by Luc *et al.*[106]: For each pixel i, given its ground-truth label l, we set the probability for that pixel and that label to be $y_{il} = max(\gamma, s(x)_{il})$, where $s(x)_{il}$ is the corresponding prediction from netwotk, and $\gamma = 0.8$. For all other classes we set $y_i c = s(x)_i c(1 - y_{il})/(1 - s(x)_{il})$, so that the label probabilities in y sum to one for each pixel.

Scene attribute loss. We add additional supervision to the network in order to regularize high level scene attributes such as scene category and overall object distributions. To make the network aware of different scene categories, we added two fully connected layers that predict the room category (over 8 scene categories) of the input panorama from its latent code generated by the encoder. We backpropagate gradients directly through the encoder from the softmax classification loss on the scene category predictions. Furthermore, we added another auxiliary network that computes the pixel-level distribution of different object classes from its semantic prediction, and backpropagates gradients from comparing this distribution to the ground truth distribution through an L_1 loss. Our ablation studies in Sec.4.4 demonstrate that these additional losses help to improve the semantic predictions, especially for small objects.

4.4 Evaluation

In this section, we present a set of experiments to evaluate Im2Pano3D. We not only investigate how well it predicts semantics and structure for unseen parts of a scene, but also study the impact of each algorithmic components through ablation studies. In most of our experiments, we consider the case where the input observation has a 180° horizontal and 116° vertical FoV, resulting in 50% partial observation (Fig.4.8). In later experiments, we demonstrate our approach on other camera configurations. All evaluations are performed on unobserved regions only.

4.4.1 Datasets

For our experiments, we use both synthetic (SUNCG [157]) and real (Matterport3D [20]) datasets. The former is used for pre-training and ablation studies. The latter is used for final evaluation on real data.

- **SUNCG [157]:** This dataset contains synthetically rendered panoramic images with color, depth and semantic of synthetic 3D indoor rooms. In total, we use 58,866 panoramas for training, and 480 for testing.
- **Matterport3D [20]:** This dataset contains real-word RGB-D panoramas captured with a tripod-mounted Matterport camera. We use color, depth and semantics provided by the dataset, but re-rendered them to form our panoramic representation (Sec. 4.3.1). In total, we use 5,315 panoramas for training, and 480 for testing.

4.4.2 Baseline Methods

To our knowledge, there is currently no prior work that performs this task exactly. To provide baselines for comparison, we consider the following extensions to prior work:

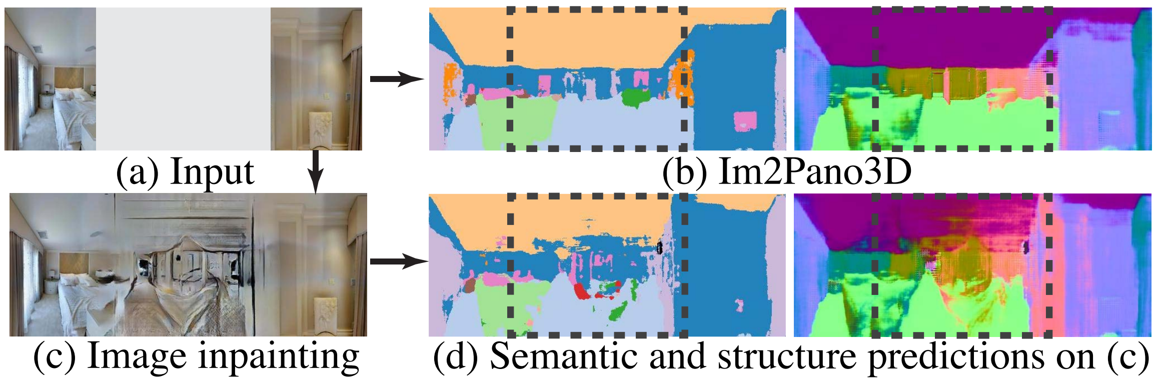


Figure 4.5: Directly predicting 3D structure and semantics (b) (rgbpn2pns) provides a more accurate result than predicting the same information from generated color pixels (d) (inpaint).

- **Average distribution (avg)** computes a per pixel average of all images within the training set.
- **Average distribution by scene category (avg-type)** computes a per pixel average of all training images within the scene category. The prediction is chosen by the testing images' ground truth scene categories.
- **Nearest neighbor (nn)** retrieves the nearest neighbor image based on ImageNet features, and uses its semantic segmentation and depth map as the prediction.
- **Image inpainting (inpaint)** uses the context encoder of [69] to directly predict the color pixels in the unobserved regions, followed by a segmentation and plane equation estimation network with the same architecture as Im2Pano3D. Fig.4.5 shows an example result.
- **Human completion (human)** asks people to complete the 3D scene using a 3D design tool [1], where users can define room layouts and furniture arrangements. Fig.4.6 shows a few example completions, and Tab.4.1 shows the average performance across four users.

Tab.4.2 and 4.1 summarize the quantitative results. Models are labeled by their input and output modality acronyms; rgb: color, s: semantics, d: depth, p: plane dis-

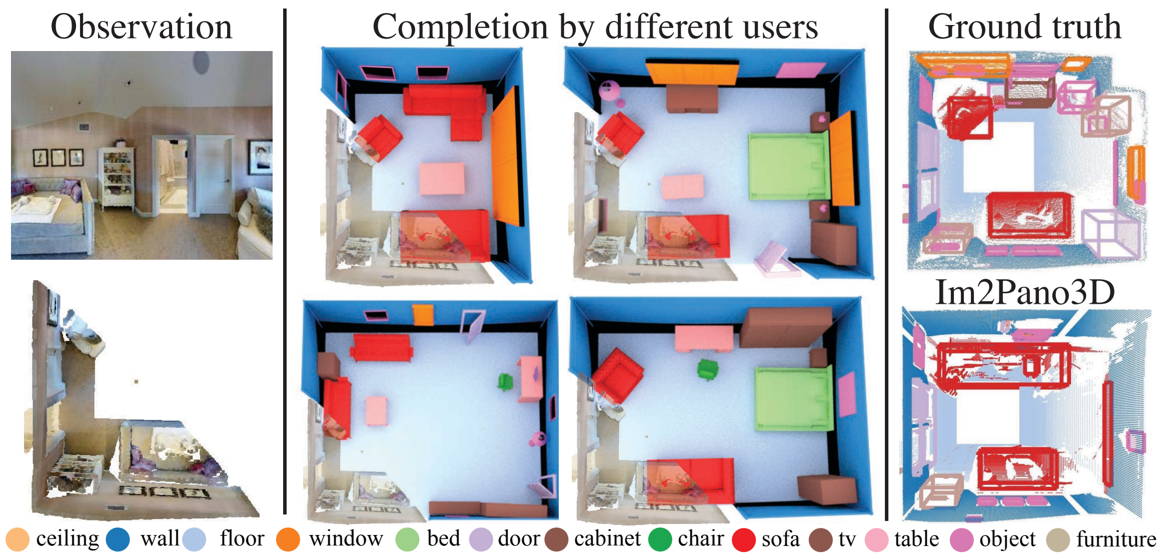


Figure 4.6: **Human completion.** Left shows the input observations. Middle shows completion results from different users overlaid on the observations. Right shows ground truth and our prediction.

tance, n: surface normal. For example, model [d2d] takes in a depth map as input and predicts the raw depth values of the unobserved regions. To evaluate the algorithm's performance independent of segmentation accuracy over the observed regions, for the [pns2pns] models, we assume ground truth segmentation for the observed region as input.

Evaluation Metrics. We measure the quality of the predicted 3D geometric structure with the following metrics:

- **Normal angle:** the mean and median angles (in degrees) between prediction and the ground truth, and the percentage of pixels with error less than three thresholds (11.25°, 22.5°, 30°).
- **Surface distance:** the mean and median L2 distances (in meters) between final predicted 3D point locations and the ground truth, and the percentage of pixels with error less than three thresholds (0.2m, 9.5m, 1m).

We measure the quality of the predicted semantic with the following metrics:

- **Probability over ground truth (PoG):** the pixelwise probability prediction of the ground truth labels averaged within each class then averaged across categories.
- **Class existence (exist):** the F1 score of object class existence predictions averaged across all classes (where existence defined as $\geq$ 400 pixels).
- **Class size (size):** the pixel size difference between ground truth and predictions divided by the ground truth size. Evaluated on the object categories with correct existence predictions only.
- **Earth Mover's Distance (EMD):** the average Earth Mover's Distance [135] between the predicted and ground truth 3D points for the categories with correct existence prediction. The weight of each 3D point is assigned with its predicted probability. The probability is normalized to sum up to one for each category. We use k-center clustering (k=50) to cluster the 3D points before calculating the EMD.
- **IoU:** the intersection over union of the most likely predicted pixel label, averaged across all classes.
- **Accuracy (acc):** the percentage of correctly predicted pixels across all pixels.
- **Inception score (incept.):** the scene classification score on the generated semantic map using an off-the-shelf image classification network (ResNet50[58]) trained on ground truth semantic maps, similar to the FCN scores that are normally used to measure the generated image quality [138].

The first four metrics of semantic evaluation are newly introduced for this task. Unlike most semantic segmentation tasks, where predictions are made for pixels directly observed with a camera, our task is to predict semantics for large regions of unobserved pixels. For this task, predicting the existence and size of unseen objects is already very difficult and useful for many applications, and thus we include the existence and size metrics, which are invariant to precise object locations. We also introduce metrics

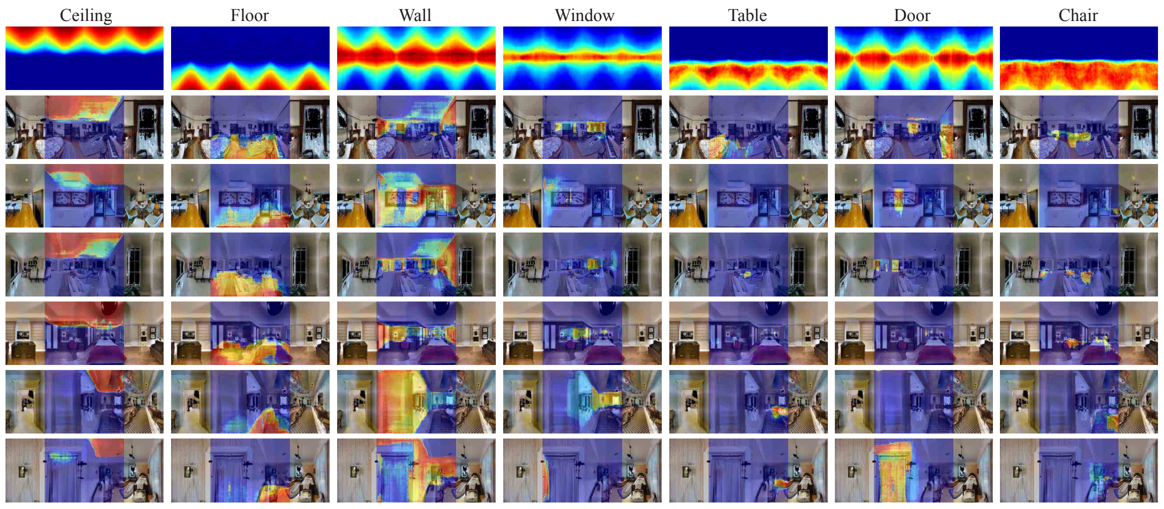


Figure 4.7: **Probability distribution of semantics.** The first row shows the average distribution of each semantic category over all training examples. The following rows show the predicted probability distribution of semantics from Im2Pano3D overlaid on top of the ground truth testing images. Red areas on the heat maps indicate higher probabilities.

based on the predicted probability distribution (PoG and EMD), which account for soft errors in position. We use PoG to rank algorithms in our comparisons.

4.4.3 Experimental Results

Tab.4.2 and 4.1 summarize the quantitative results and Fig.4.8 shows qualitative results. More results and visualizations can be found in the supplementary material.

Comparing to Baseline Methods. Comparing our model [rgbpn2pns (s+m)] to all baseline methods (Tab.4.1 row 2-5), our proposed model produces better predictions in terms of both semantics and 3D structure. In particular, compared to the two-step process of predicting semantic labels over predicted color images in the unobserved regions [inpaint], directly predicting semantic labels in a one-step process can generate a more accurate result (+13% in PoG and -0.24m in surface distance). Fig.4.5 shows a qualitative comparison.

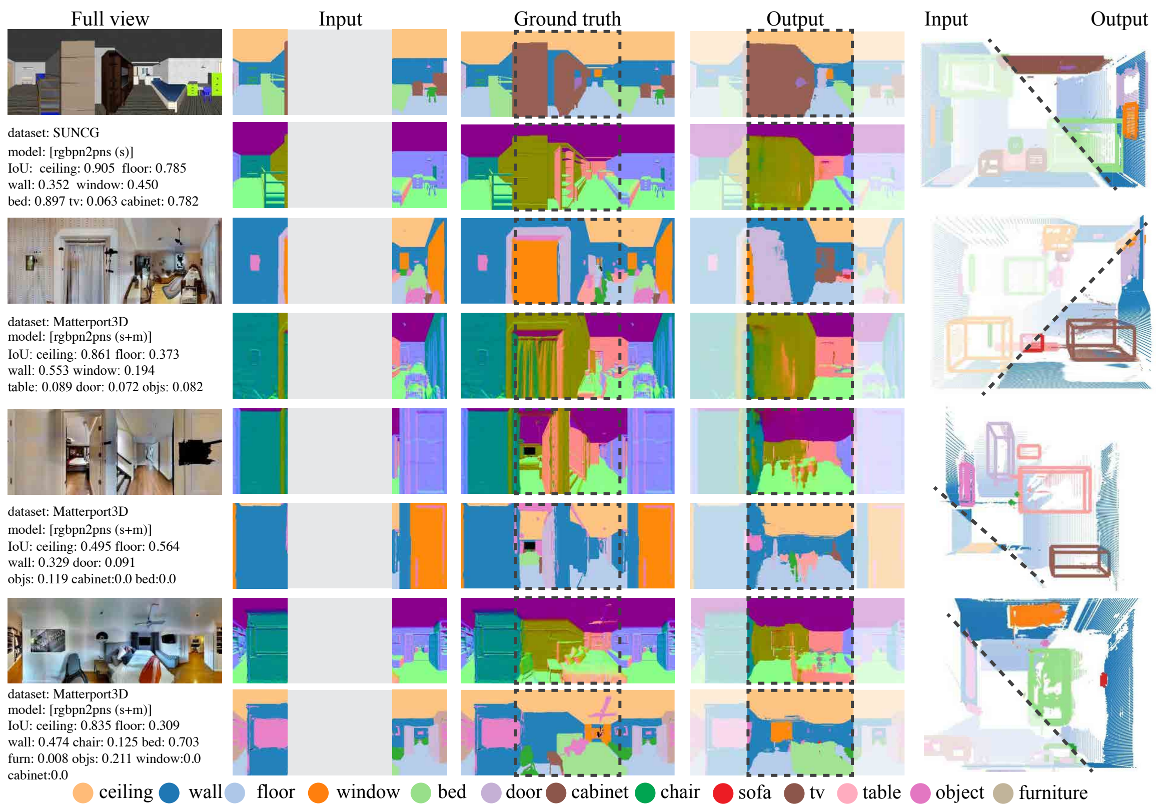

Full view
Input
Ground truth
Output
Input
Output
dataset: SUNCG
model: [rgbpn2pns (s)]
IoU: ceiling: 0.905 floor: 0.785
wall: 0.352 window: 0.450
bed: 0.897 tv: 0.063 cabinet: 0.782
dataset: Matterport3D
model: [rgbpn2pns (s+m)]
IoU: ceiling: 0.861 floor: 0.373
wall: 0.553 window: 0.194
table: 0.089 door: 0.072 objs: 0.082
dataset: Matterport3D
model: [rgbpn2pns (s+m)]
IoU: ceiling: 0.495 floor: 0.564
wall: 0.329 door: 0.091
objs: 0.119 cabinet:0.0 bed:0.0
dataset: Matterport3D
model: [rgbpn2pns (s+m)]
IoU: ceiling: 0.835 floor: 0.309
wall: 0.474 chair: 0.125 bed: 0.703
furn: 0.008 objs: 0.211 window:0.0
cabinet:0.0
● ceiling ● wall ● floor ● window ● bed ● door ● cabinet ● chair ● sofa ● tv ● table ● object ● furniture

Figure 4.8: **Qualitative Results.** For each example, we show semantic segmentations labeled using the highest predicted class probability for each pixel, and normal maps from 3D structure predictions. We also show reconstructed 3D point clouds (right column), colored by semantic labels, with bounding boxes around semantically connected components. More results in supplementary material.

Does synthetic data help? Comparing our models [pns] and [rgbpn2pns] trained with and without the SUNCG dataset and testing on the Matterport3D dataset, we observe that pre-training on SUNCG significantly improves the model's performance, 9% and 4% improvement in PoG respectively. In particular, when the input is a segmentation map instead of a color image [pns2pns], the model trained only on SUNCG can even achieve better performance than the model trained on Matterport3D alone (+1.3% in PoG and -0.08m in surface distance). This result demonstrates that training on synthetic data is critical for this task, as it enables the network to learn a rich whole-room contextual prior from a large variety of indoor scenes, which is extremely expensive to obtain with real data.

models		semantics							3D surface (m)					normals (°)				
type	train	PoG↑	exist↑	size↓	emd↓	IoU↑	acc. ↑	incept. ↑	mean	med. ↓	0.2	0.5	1%↑	mean	med. ↓	11.25	22.5	30%↑
human	-	0.303	0.650	1.474	0.943	0.203	0.522	-	0.661	0.449	29.1	57.7	78.7	49.9	17.4	51.2	58.2	60.8
avg all	m	0.131	0.228	1.574	2.007	0.098	0.498	-	0.925	0.685	12.6	37.8	67.9	46.2	41.8	3.1	17.5	31.4
avg type	m	0.155	0.260	1.265	2.089	0.107	0.508	-	0.905	0.668	13.8	39.6	69.6	45.8	40.4	4.5	20.7	34.0
nn	m	0.126	0.531	1.901	2.820	0.078	0.302	-	1.286	0.898	15.8	33.6	56.4	65.1	58.1	23.8	31.2	34.9
inpaint	s+m	0.145	0.488	1.407	1.984	0.082	0.347	0.183	0.867	0.591	19.3	46.3	72.3	59.5	50.4	23.3	32.8	37.9
rgbpn2pns	s	0.185	0.56	1.589	1.729	0.129	0.378	0.233	0.609	0.365	32.3	63.4	82.5	47.2	20.8	43.6	54.7	59.4
rgbpn2pns	m	0.245	0.542	**0.933**	1.535	0.174	**0.566**	0.394	0.603	0.361	37.4	63.7	82.1	**39.1**	22.4	34.9	52.6	60.4
rgbpn2pns	s+m	**0.275**	**0.616**	0.936	**1.487**	**0.208**	**0.566**	**0.402**	**0.524**	**0.280**	**43.6**	**69.5**	**85.5**	43.6	**19.0**	**42.9**	**57.2**	**62.8**
pns2pns	s	0.317	0.658	0.858	1.507	0.256	0.603	0.365	0.581	0.367	32.3	65.0	84.4	44.1	15.5	52.1	61.8	65.4
pns2pns	m	0.304	0.618	**0.854**	1.526	0.243	0.61	0.406	0.610	0.373	32.6	63.4	83.2	42.3	20.0	37.9	57.3	63.6
pns2pns	s+m	**0.355**	**0.665**	0.881	**1.425**	**0.282**	**0.623**	**0.427**	**0.563**	**0.321**	**38.5**	**67.6**	**84.6**	**41.2**	**19.7**	**40.3**	**56.9**	**63.2**

Table 4.1: Comparing to baseline methods on Matterport3D. Row 2 to 5 shows baseline methods. Our models are named by their input output modalities (same as Tab.4.2) and training set (s: SUNCG, m: Matterport3D). Bold numbers indicate best performances in each group.

models	semantics							3D surface (m)					normals (°)				
type+loss	PoG↑	exist↑	size↓	emd↓	IoU↑	acc. ↑	incept.↑	mean	med.↓	0.2	0.5	1%↑	mean	med.↓	11.25	22.5	30%↑
pn2pn+A	-	-	-	-	-	-	-	**0.320**	**0.119**	**67.6**	81.4	91	38.5	5.5	70.3	74.5	76
d2d+A	-	-	-	-	-	-	-	0.353	0.148	63.1	79.6	90.1	59.0	41.2	12.7	29.3	38.9
rgbpn2pns+A+S	0.376	0.688	0.702	1.204	0.321	0.721	0.446	0.306	0.124	67.1	**82.4**	**92.1**	36.0	4.6	72.5	**76.5**	**78.2**
pns2pns+S	0.379	0.613	**0.653**	**1.184**	0.313	**0.728**	0.375	0.416	0.227	51.8	74.3	88.9	**32.5**	7.6	62.3	72.2	76.0
pns2pns+A	0.370	0.681	0.750	1.269	0.318	0.719	0.452	0.343	0.15	63.3	80.4	90.9	37.7	**4.4**	72.2	76.0	77.4
pns2pns+A+S	0.382	**0.710**	0.754	1.204	**0.330**	0.716	**0.463**	0.339	0.151	64.0	80.8	91.1	36.9	4.6	**73.0**	76.4	77.8

Table 4.2: Ablation studies on SUNCG. Models are named by their input and output modalities. rgb: color, s: semantic segmentation, d: depth, p: plane distance, n: surface normal. A: adversarial loss, S: scene attribute loss.

Do different surface encodings matter? Comparing the model using raw depth values [d2d] to the model using the plane equation encoding [pn2pn] (Tab.4.2 and Fig.4.3), we can see that the plane equation encoding provides a strong regularization allowing the network to predict higher quality 3D geometry with lower surface distance and normal error, 0.03m and 21° less respectively.

What are the effects of different losses? Comparing the model trained with adversarial loss [pns2pns+S+A] and without [pns2pns+S] in Tab.4.2, we can see that the adversarial loss improves the prediction accuracy for small objects, which is reflected in higher IoU (+2%). Meanwhile the adversarial loss reduces recall for objects with big pixel area, which is reflected in lower total pixel accuracy (-1.2%). Similarly, the scene attribute loss also improves IoU (+2%), with a small compromise on total pixel accuracy (-0.3%).

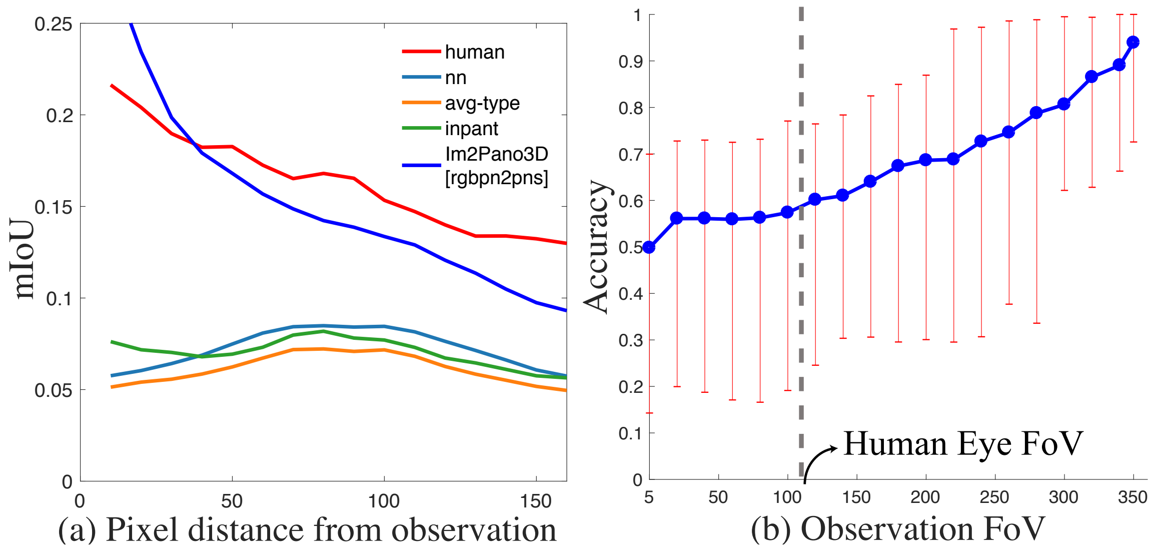


Figure 4.9: **Experiments.** (a) shows mean IOU with respect to distance from observation. (b) shows accuracy of predictions in the unobserved regions while increasing input horizontal FoV from 5° to 350°. The error bar shows the error margin across test cases.

How is accuracy influenced by distance to observation? Fig.4.9 (a) shows the average IoU with respect to its distance to the nearest observed pixel. As expected, the performance for Im2Pano3D decreases for pixels that are further from the input observation. However, the performance is still much higher than other baselines when the region is far from the observation or completely behind the camera, yet still not as high as human performance.

How is accuracy influenced by input FoV? To investigate how the input FoV affects the prediction accuracy, we do the following experiment: we keep the vertical FoV of the input image at 116° while steadily increasing the horizontal FoV from 5° to 350°, and ask the network to predict the structure and semantics for the full panorama. Fig.4.9 (b) shows prediction accuracy in the unobserved regions with respect to input FoV, which shows that the prediction accuracy improves as the input FoV increases.

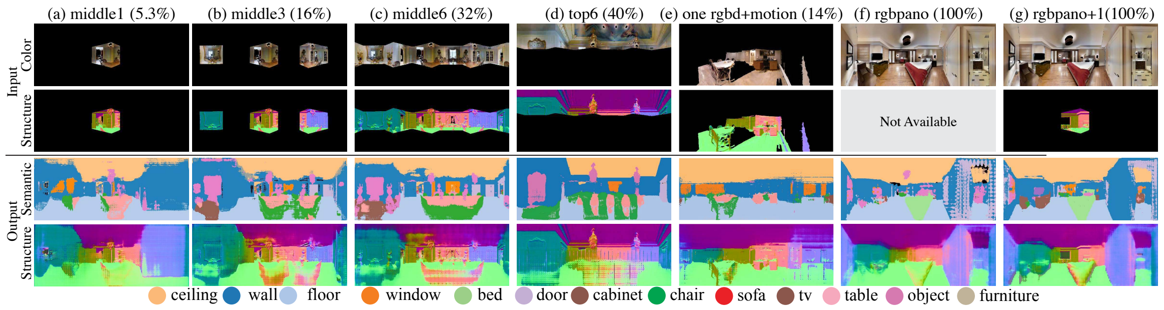


Figure 4.10: **Camera configurations.** For different camera configurations, Im2Pano3D provides a unified framework to efficiently filling in missing 3D structure and semantics of the unobserved scene. The observation coverage is shown in parentheses. 3D structure is represented with normals. The data for [RGB-D+motion] comes from NYUv2 [149]. More examples can be found in supplementary material.

Generalizing to different camera configurations. In most of our evaluations, we consider the case where the input observation has a 180° horizontal FoV. However, in real robotic applications, systems may be equipped with different types of cameras resulting in different observation FoV patterns. Here we demonstrate how Im2Pano3D can generalize to other cases. The camera configurations we consider includes: single or multiple registered RGB-D cameras such as Matterport cameras (Fig.4.10 (a-d)), single RGB-D camera capturing a short video sequence (e), color-only panoramic camera (f), and color panoramic cameras paired with a single depth camera (g). To improve the ability of the network to generalize to different input observation patterns, we use a random view mask during training. Tab.4.3 shows the qualitative evaluation. For all of these camera configurations, Im2Pano3D provides a unified framework that effectively fills in the missing 3D structure and semantic information of the unobserved scene.

4.5 Summary

In this chapter, we introduce the task of semantic-structure view extrapolation and present Im2Pano3D, a unified framework to produce a complete room structure and

	middle1	middle3	top6	bottom6	middle6	rgbpano	rgbpano+1
camera obs.(%)	5.3	16.7	40.4	40.1	32.7	100	100
PoG	0.188	0.304	0.269	0.286	0.392	**0.393**	0.425
normal	29.0	13.4	14.3	14.0	**8.8**	11.3	9.5
surface	0.454	0.238	0.237	0.322	**0.148**	0.290	0.250

Table 4.3: **Camera configurations.** The table shows the average PoG, median surface and normal error for each configuration. Example inputs for each configuration can be found in Fig.4.10. For models [rgbpano] and [rgbpano+1], we evaluate on regions that do not have depth observation. For all other models, we evaluate on regions with no color and depth observation.

semantic estimation conditioned on a partial observation of the scene. Experiments demonstrate that the direct prediction of structure and semantics for the unobserved scene provides more accurate results than alternative approaches. However, while Im2Pano3D explores the possibilities of whole-room contextual reasoning for 3D scene understanding, the proposed system is still far from perfect. Possible future directions may include: explicitly modeling semantics at the instance-level as opposed to category-level, and exploring alternative data representations that consider occluded regions.

Chapter 5

Datasets and Benchmarks for 3D Scene Understanding

Data-driven 3D scene understanding is still a young field. Compared to many research areas in 2D computer vision, where there are many well-established datasets and benchmarks, the existing datasets for 3D datasets such as NYU dataset [120] are much smaller. We believe the limited scale of 3D scene understanding dataset is now becoming the critical common bottleneck in advancing research to the next level. Besides causing overfitting of the algorithm during evaluation, they cannot support training data-hungry algorithms. Therefore, one of the major contribution of this dissertation is to build up infrastructure for its community, which includes creating several major datasets and benchmarks in the area. In this chapter, I will introduce a real-world RGB-D dataset, SUN RGB-D [154]; and a synthetic 3D scene dataset, SUNCG [157].

5.1 Real-world RGB-D dataset: SUN RGB-D

The recent arrival of affordable depth sensors in consumer markets enabled us to acquire reliable depth maps at a very low cost, stimulating breakthroughs in several vision tasks, such as body pose recognition [145, 147], intrinsic image estimation [9], 3D modeling [70] and SfM reconstruction [185]. RGB-D sensors have also enabled rapid progress for scene understanding (e.g. [50, 47, 130, 100, 75, 43, 76, 120]). However, while we can crawl color images from the Internet easily, it is not possible to obtain large-scale RGB-D data online. Consequently, the existing RGB-D recognition benchmarks, such as NYU Depth v2 [120], are an order-of-magnitude smaller than modern recognition datasets for color images (e.g. PASCAL VOC [29]). Although these small datasets successfully bootstrapped initial progress in RGB-D scene understanding in the past few years, the size limit is now becoming the critical common bottleneck in advancing research to the next level. Besides causing overfitting of the algorithm during evaluation, they cannot support training data-hungry algorithms that are currently the state-of-the-art in color-based recognition (e.g. [39, 89]). If a large-scale RGB-D dataset were available, we could borrow the same success to the RGB-D domain as well. Furthermore, although the RGB-D images in these datasets contain depth maps, the annotation and evaluation metrics are mostly in 2D image domain, but not directly in 3D (Figure 5.1). Scene understanding is much more useful in the real 3D space for most applications. We desire to reason about scenes and evaluate algorithms in 3D.

To this end, we introduced SUN RGB-D, a dataset containing 10,335 RGB-D images with dense annotations in both 2D and 3D, for both objects and rooms. Based on this dataset, we focus on six important recognition tasks towards total scene understanding, which recognizes objects, room layouts and scene categories. For each task, we propose metrics in 3D and evaluate baseline algorithms derived from the state-of-the-art. Since there are several popular RGB-D sensors available, each with different size and power consumption, we constructed our dataset using four different kinds of sensors to study how well the algorithms generalize across sensors. By constructing a PASCAL-scale dataset and defining a benchmark with

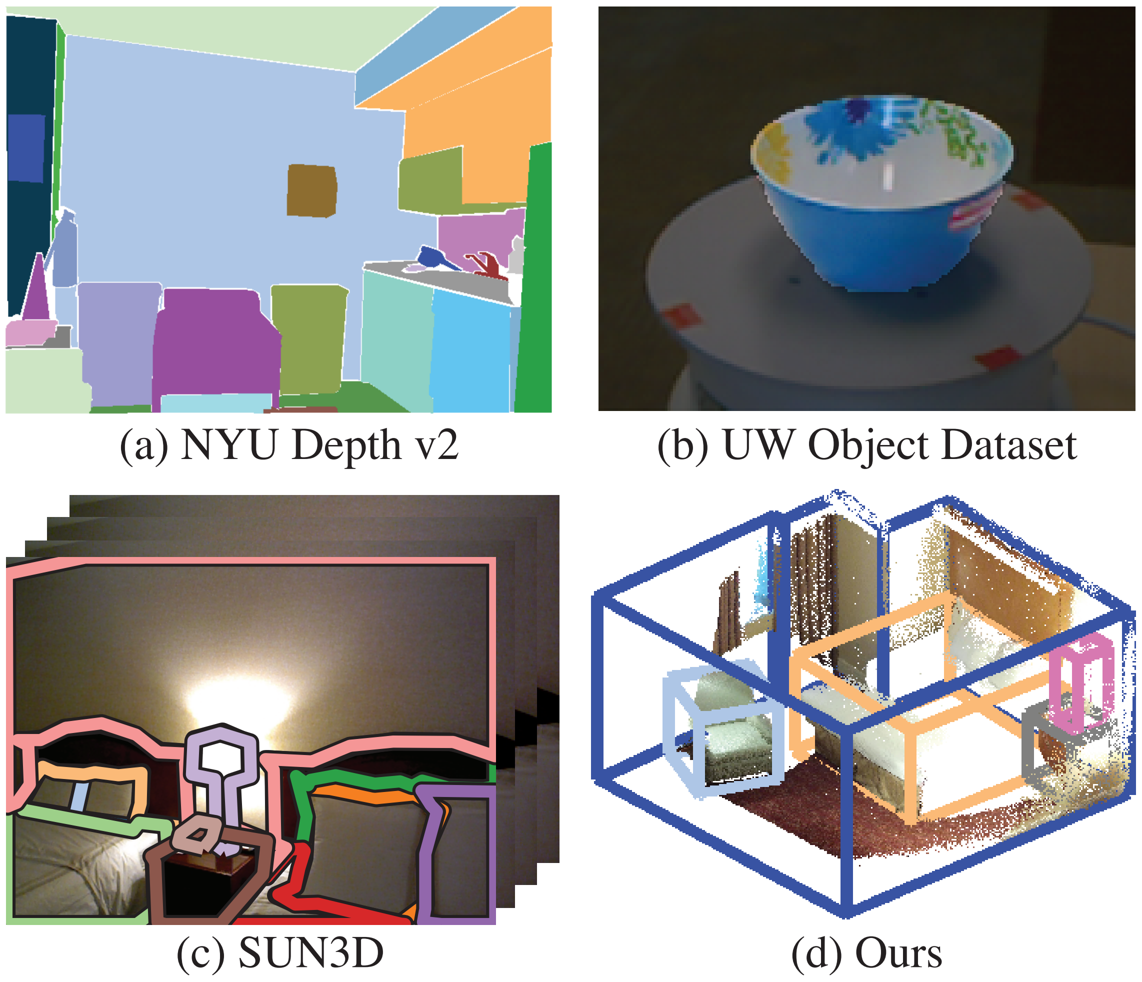


Figure 5.1: **Comparison of RGB-D recognition benchmarks.** Apart from 2D annotation, our benchmark provided high quality 3D annotation for both objects and room layout.

3D evaluation metrics, we hope to lay the foundation for advancing RGB-D scene understanding in the coming years.

5.1.1 Related Work

There are many existing RGB-D datasets available [132, 113, 2, 64, 109, 148, 120, 110, 72, 163, 146, 127, 112, 42, 53, 190, 88, 166, 7, 103, 30, 160, 105, 162, 161, 115, 34, 81, 28, 122, 68, 19]. Figure 5.1 shows some of them. Here we will briefly describe and compare to several most relevant ones[1].

There are datasets [151, 92] that capture objects on a turntable instead of real-world scenes. For natural indoor scene datasets, NYU Depth v2 [120] is probably the most popular one. They labeled 1,449 selected frames from short RGB-D videos using 2D semantic segmentation on the image domain. [44] annotates each object by aligning a CAD model with the 3D point cloud. However, the 3D annotation is quite noisy, and in our benchmark we reuse the 2D segmentation but recreate the 3D annotation by ourselves. Although this dataset is very good, the size is still small compared to other modern recognition datasets, such as PASCAL VOC [29] or ImageNet [136]. B3DO [71] is another dataset with 2D bounding box annotations on the RGB-D images. But its size is smaller than NYU and it has many images with an unrealistic scene layouts (e.g. snapshot of a computer mouse on the floor). The Cornell RGBD dataset [3, 87] contains 52 indoors scenes with per-point annotations on the stitched point clouds. SUN3D [185] contains 415 RGB-D video sequence with 2D polygon annotation on some keyframes. Although they stitched the point cloud in 3D, the annotation is still purely in the 2D image domain, and there are only 8 annotated sequences.

5.1.2 Dataset Construction

The goal of our dataset construction is to obtain an image dataset captured by various RGB-D sensors at a similar scale as the PASCAL VOC object detection benchmark. To improve the depth map quality, we take short videos and use multiple frames to obtain a refined depth map. For each image, we annotate the objects with both 2D polygons and 3D bounding boxes and the room layout with 3D polygons.

[1] A full list with brief descriptions is available at `http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/`.

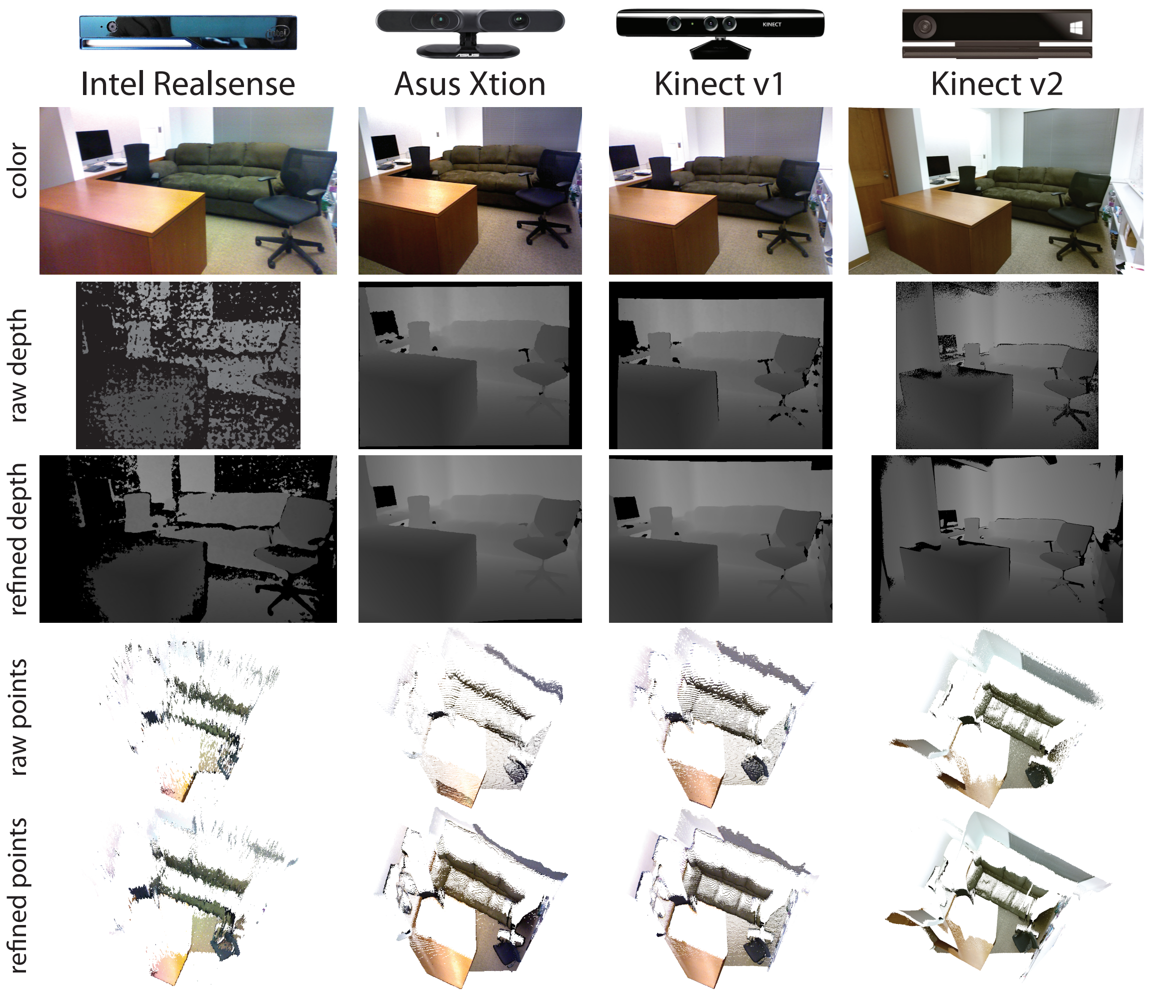


Figure 5.2: **Comparison of the four RGB-D sensors.** The raw depth map from Intel RealSense is noisier and has more missing values. Asus Xtion and Kinect v1's depth map have observable quantization effect. Kinect v2 is more accurate to measure the details in depth, but it is more sensitive to reflection and dark color. Across different sensors our depth improvement algorithm manages to robustly improve the depth map quality.

Sensors

Since there are several popular sensors available, with different size and power consumption, we construct our dataset using four kinds – Intel RealSense 3D Camera for tablets, Asus Xtion LIVE PRO for laptops, and Microsoft Kinect versions 1 and 2 for desktop. Figure 5.2 shows the example color and depth images captured.

Intel RealSense is a lightweight, low power consuming depth sensor designed for tablets. It will soon reach consumers; we obtained two pre-release samples from Intel. It projects an IR pattern to the environment and uses stereo matching to obtain the depth map. For outdoor environments, it can switch automatically to stereo matching without IR pattern; however, we visually inspect the 3D point cloud and believe the depth map quality is too low for use inaccurate object recognition for outdoors. We thus only use this sensor to capture indoor scenes. Figure 5.2 shows its raw depth is worse than that of other RGB-D sensors, and the effective range for reliable depth is shorter (depth gets very noisy around 3.5 meters). But this type of lightweight sensor can be embedded in portable devices and be deployed at a massive scale in consumer markets, so it is important to study algorithm performance with it.

Asus Xtion and Kinect v1 use a near-IR light pattern. Asus Xtion is much lighter and powered by USB only, with worse color image quality than Kinect v1's. However, Kinect v1 requires an extra power source. The raw depth maps from both sensors have an observable quantization effect.

Kinect v2 is based on time-of-flight and also consumes significant power. The raw depth map captured is more accurate, with high fidelity to measure the detailed depth difference, but fails more frequently for black objects and slightly reflective surfaces. The hardware supports long-distance depth range, but the official Kinect for Windows SDK cuts the depth off at 4.5 meters and applies some filtering that tends to lose object details. Therefore, we wrote our own driver and decoded the raw depth in GPU (Kinect v2 requires software depth decoding) to capture real-time video without depth cutoffs or additional filtering.

Sensor calibration

For RGB-D sensors, we must calibrate the camera intrinsic parameters and the transformation between the depth and color cameras. For Intel RealSense, we use the default factory parameters. For Asus Xtion, we rely on the default parameters returned by OpenNI library without modeling radial distortion. For Kinect v2, the radial distortion is very strong. So we calibrate all cameras with standard calibration toolbox [17]. We calibrate the depth cameras by computing the parameters with the IR image which is the same with the depth camera. To see the checkerboard without overexposure on IR, we cover the emitter with a piece of paper. We use the stereo calibration function to calibrate the transformation between the depth (IR) and the color cameras.

Depth map improvement

The depth maps from these cameras are not perfect, due to measurement noise, view angle to the regularly reflective surface, and occlusion boundary. Because all the RGB-D sensors operate as a video camera, we can use nearby frames to improve the depth map, providing redundant data to denoise and fill in missing depth.

We propose a robust algorithm for depth map integration from multiple RGB-D frames. For each nearby frame in a time window, we project the points to 3D, get the triangulated mesh from nearby points, and estimate the 3D rotation and translation between this frame and the target frame for depth improvement. Using this estimated transformation, we render the depth map of the mesh from the target frame camera. After we obtain aligned and warped depth maps, we integrate them to get a robust estimation. For each pixel location, we compute the median depth and 25% and 75% percentiles. If the raw target depth is missing or outside the $25\% - 75\%$ range and the median is computed from at least 10 warped depth maps, we use the median depth value. Otherwise, we keep the original value to avoid over-smoothing. Examples are shown in Figure 5.2. Our depth map improvement algorithm, compared to [185] which uses a 3D voxel-based TSDF representation, requires much less memory and runs faster at equal resolution, enabling much high-resolution integration.

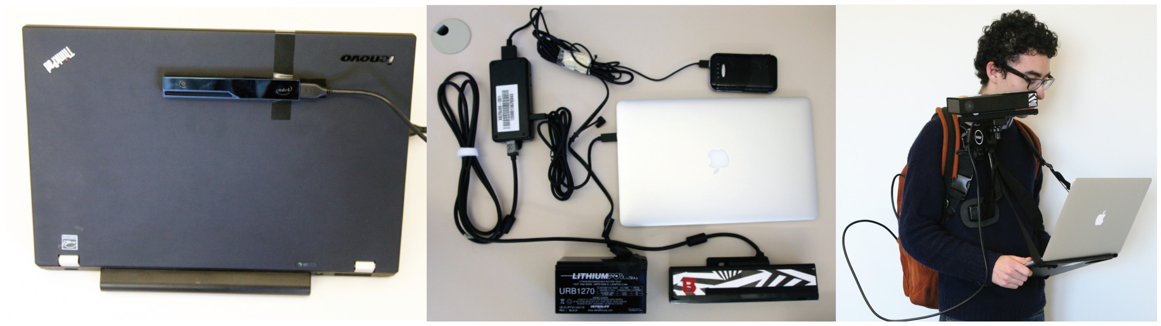


Figure 5.3: **Data Capturing Process.** (a) RealSense attached to laptop, (b) Kinect v2 with battery, (c) Capturing setup for Kinect v2.

Robust estimation of an accurate 3D transformation between a nearby frame and target frame is critical for this algorithm. To do this, we first use SIFT to obtain point-to-point correspondences between the two color images, obtain the 3D coordinates for the SIFT keypoints from the raw depth map, and then estimate the rigid 3D rotation and translation between these two sparse 3D SIFT clouds using RANSAC with three points. To obtain a more accurate estimation, we would like to use the full depth map to do dense alignment with ICP, but depending on the 3D structure, ICP can have severe drifting. Therefore, we first use the estimation from SIFT+RANSAC to initialize the transformation for ICP, and calculate the percentage of points for ICP matching. Using the initialization and percentage threshold, we run point-plane ICP until convergence, then check the 3D distances with the original SIFT keypoint inliers from RANSAC. If the distances significantly increase, it means ICP makes the result drift away from the truth; we will use the original RANSAC estimation without ICP. Otherwise, we use the ICP result.

Data Acquisition

To construct a dataset at the PASCAL VOC scale, we capture a significant amount of new data by ourselves and combine some existing RGB-D datasets. We capture 3,784 images using Kinect v2 and 1,159 images using Intel RealSense. We included

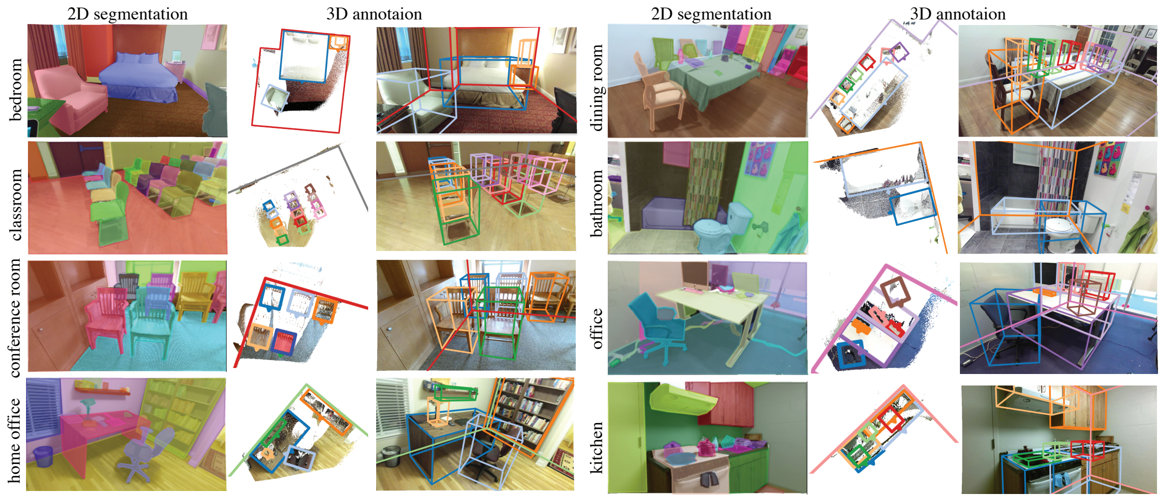


Figure 5.4: **Example images with annotation from SUN RGB-D dataset.**

the 1,449 images from the NYU Depth V2 [120], and also manually selected 554 realistic scene images from the Berkeley B3DO Dataset [71], both captured by Kinect v1. We manually selected 3,389 distinguished frames without significant motion blur from the SUN3D videos [185] captured by Asus Xtion. In total, we obtain 10,335 RGB-D images.

As shown in Figure 5.3, we attach an Intel RealSense to a laptop and carry it around to capture data. For Kinect v2 we use a mobile laptop harness and camera stabilizer. Because Kinect v2 consumes a significant amount of power, we use a 12V car battery and a 5V smartphone battery to power the sensor and the adaptor circuit. The RGB-D sensors only work well for indoors. And we focus on universities, houses, and furniture stores in North America and Asia. Some example images are shown in Figure 5.4.

5.1.3 Ground Truth Annotation

For each RGB-D image, we obtain LabelMe-style 2D polygon annotations, 3D bounding box annotations for objects, and 3D polygon annotations for room layouts. To ensure annotation quality and consistency, we obtain our own ground truth labels for images from other datasets; the only exception is NYU, whose 2D segmentation we use as our 2D annotation.

For 2D polygon annotation, we developed a LabelMe-style [137] tool for Amazon Mechanical Turk. To ensure high label quality, we add automatic evaluation in the tool. To finish the HIT, each image must have at least 6 objects labeled; the union of all object polygons must cover at least 80% of the total image. To prevent workers from cheating by covering everything with big polygons, the union of the small polygons (area $<$ 30% of the image) must cover at least 30% of the total image area. Finally, the authors visually inspect the labeling result and manually correct the layer ordering when necessary. Low-quality labelings are sent back for relabeling. We paid \$0.10 per image; some images required multiple labeling iterations to meet our quality standards.

For 3D annotation, the point clouds are first rotated to align with the gravity direction using an automatic algorithm. We estimate the normal direction for each 3D point with the 25 closest 3D points. Then we accumulate a histogram on a 3D half-sphere and pick the maximal count from it to obtain the first axis. For the second axis, we pick the maximal count from the directions orthogonal to the first axis. In this way, we obtain the rotation matrix to rotate the point cloud to align with the gravity direction. We manually adjust the rotation when the algorithm fails.

We design a web-based annotation tool and hire oDesk workers to annotate objects and room layouts in 3D. For objects, the tool requires drawing a rectangle on the top view with an orientation arrow, and adjusting the top and bottom to inflate it to 3D. For room layouts, the tool allows arbitrary polygon on the top view to describe the complex structure of the room (Figure 5.4). Our tool also shows the projection of the 3D boxes to the image in real time, to provide intuitive feedback during annotation. We hired 18 oDesk workers and trained them over Skype. The average hourly rate is \$3.90, and they spent 2,051 hours in total. Finally, all labeling results are thoroughly

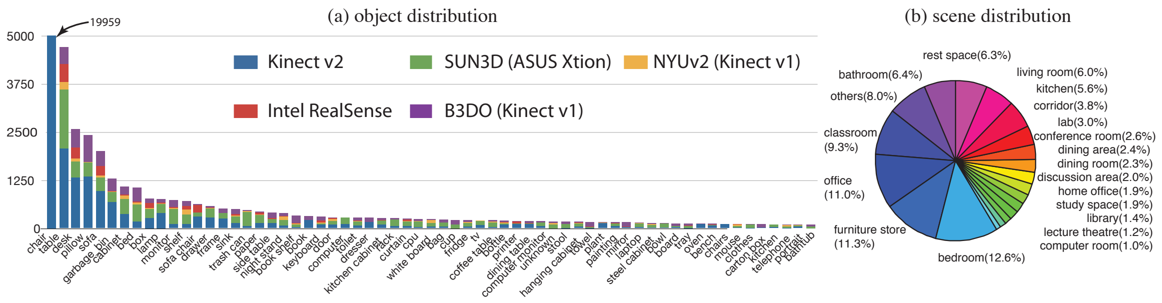


Figure 5.5: **Statistics of semantic annotation in our dataset.**

checked and corrected by the authors. For scene categories, we manually classify the images into basic-level scene categories.

5.1.4 SUN RGB-D Dataset Statistics

For the 10,335 RGB-D images, we have 146,617 2D polygons and 64,595 3D bounding boxes (with accurate orientations for objects) annotated. Therefore, there are 14.2 objects in each image on average. In total, there are 47 scene categories and about 800 object categories. Figure 5.5 shows the statistics for the semantic annotation of the major object and scene categories.

5.1.5 Benchmark Design

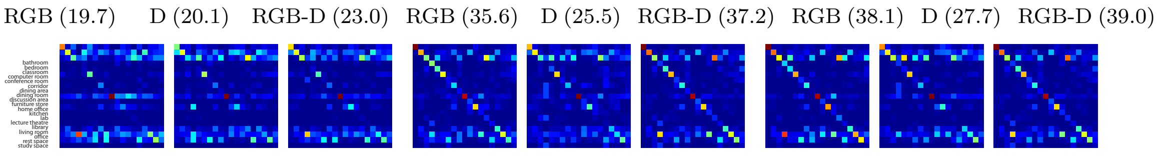
To evaluate the whole scene understanding pipeline, we select six tasks, including both popular existing tasks and new but important tasks, both single-object based tasks and scene tasks, as well as a final total scene understanding task that integrates everything.

We choose some state-of-the-art algorithms to evaluate each task. For the tasks without existing algorithm or implementation, we adapt popular algorithms from other tasks. For each task, whenever possible, we try to evaluate algorithms using color, depth, as well as RGB-D images to study the relative importance of color and depth, and gauge to what extent the information from both is complimentary. Various evaluation results show that we can apply standard techniques designed for color (e.g. handcraft features, deep learning features, detector, sift flow label transfer) to depth domain and it can achieve comparable performance for various tasks. In most of cases, when we combining these two source of information, the performance get improved.

For evaluation, we carefully split the data into training and testing set, ensuring each sensor has around half for training and half for testing, Since some images are captured from the same building or house with similar furniture styles, to ensure fairness, we carefully split the training and testing sets by making sure that those images from the same building either all go into the training set or the testing set and do not spread across both sets. For data from NYU Depth v2 [120], we use the original split.

Scene Categorization Scene categorization is a very popular task for scene understanding [183]. In this task, we are given an RGB-D image, classify the image into one of the predefined scene categories, and use the standard average categorization accuracy for evaluation.

We use the 19 scene categories with more than 80 images. We choose GIST [123] with a RBF kernel one-vs-all SVM as the baseline. We also choose the state-of-the-art Places-CNN [200] scene feature, which achieves the best performance in color-based scene classification on the SUN database [183]. This feature is learned using a Deep Convolutional Neural Net (AlexNet [89]) with 2.5 million scene images [200]. We use both linear SVM and RBF kernel SVM with this CNN feature. Also, empirical experiments [51] suggest that both traditional image features and deep learning features for color image can be used to extract powerful features for depth maps as well. Therefore, we also compute the GIST and Places-CNN on the depth images. We also evaluate the concatenation of depth and color features. The depth image is encoded as HHA image as in [51] before extract the feature. Figure 5.6 reports the accuracy for these experiments. We can see that the deep learning features indeed perform much better, and the combination of color and depth features also helps.



(a) GIST[123]+kernel SVM (b) Places-CNN+Linear SVM (c) Places-CNN+kernel SVM

Figure 5.6: **Confusion matrices for various scene recognition algorithms.** Each combination of features and classifiers is run on RGB, D and RGB-D. The numbers inside the parentheses are the average accuracy for classification.

Semantic Segmentation Semantic segmentation in the 2D image domain is currently the most popular task for RGB-D scene understanding. In this task, the algorithm outputs a semantic label for each pixel in the RGB-D image. We use the standard average accuracy across object categories for evaluation.

We run the state-of-the-art algorithm for semantic segmentation [130] on our benchmark and report the result on Table 5.1. Since our dataset is quite large, we expect non-parametric label transfer to work well. We first use Places-CNN features [200] to find the nearest neighbor and directly copy its segmentation as the result. We surprisingly found that this simple method performs quite well, especially for big objects (e.g. floor, bed). We then adapt the SIFT-flow algorithm [102, 101], on both color and depth to estimation flow. But it only slightly improves performance.

Object Detection Object detection is another important step for scene understanding. We evaluate both 2D and 3D approaches by extending the standard evaluation criteria for 2D object detection to 3D. Assuming the box aligns with the gravity direction, we use the 3D intersection over union of the predicted and ground truth boxes for 3D evaluation. With 0.25 as the threshold, we calculate the precision-recall curve.

For 2D object detection, we evaluate four state-of-the-art algorithms for object detection: DPM [32], Exemplar SVM [108], RGB-D RCNN [51], and Sliding Shapes

									mean
RGB NN	45.03	27.89	16.89	18.51	21.77	1.06	4.07	0	8.32
Depth NN	42.6	9.65	21.51	12.47	6.44	2.55	0.6	0.3	5.32
RGB-D NN	45.78	35.75	19.86	19.29	23.3	1.66	6.09	0.7	8.97
RGB [102]	47.22	39.14	17.21	20.43	21.53	1.49	5.94	0	9.33
Depth [102]	43.83	13.9	22.31	12.88	6.3	1.49	0.45	0.25	5.98
RGB-D [102]	48.25	49.18	20.8	20.92	23.61	1.83	8.73	0.77	10.05
RGB-D [130]	78.64	84.51	33.15	34.25	42.52	25.01	35.74	35.71	36.33

Table 5.1: **Semantic segmentation.** We evaluate performance for 37 object categories. Here shows 8 selected ones: floor, ceiling, chair, table, bed, nightstand, books, and person. The mean accuracy is for all the 37 categories. A full table is in the supp. material.

[155]. For DPM and Exemplar SVM, we use the depth as another image channel and concatenate HOG computed from that and from color images. To evaluate the first three 2D algorithms, we use 2D IoU with a threshold of 0.5 and the results are reported in Table 5.2. The 2D ground truth box is obtained by projecting the points inside the 3D ground truth box back to 2D and finding a tight box that encompasses these 2D points.

For 3D detection, please refer to Chapter 2 for more detailed experiment and evaluation on 3D object detection.

RGBD																				mAP
ESVM	7.3	12.9	7.4	0.0	12.4	0.02	0.8	0.5	1.8	6.0	6.1	0.4	6.0	1.6	6.1	14.0	11.8	0.8	14.8	5.8
DPM	34.2	54.7	14.4	0.45	29.3	0.8	4.7	0.4	1.8	13.2	23.3	11.9	23.3	9.3	15.5	21.6	24.0	8.7	23.7	16.6
RCNN[51]	**49.6**	**75.9**	**34.9**	**5.7**	**41.2**	**8.1**	**16.5**	**4.2**	**31.3**	**46.8**	**21.9**	**10.7**	**37.2**	**16.5**	**41.9**	**42.2**	**43.02**	**32.9**	**69.8**	**35.2**

Table 5.2: **Evaluation of 2D object detection.** We evaluate on 19 popular object categories using Average Precision (AP): bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, garbage bin, lamp, monitor, night stand, pillow, sink, sofa, table, tv and toilet.

Object Orientation Besides predicting the object location and category, another important vision task is to estimate its pose. For example, knowing the orientation of a chair is critical to sit on it properly. Because we assume that an object bounding box is aligned with gravity, there is only one degree of freedom in estimating the yaw angle for orientation.

We evaluate two exemplar-based approaches: Exemplar SVM [108] and Sliding Shapes [155].The prediction is evaluated by the angle difference between the prediction and the ground truth. We transfer the orientations from the training exemplars to the predicted bounding boxes. Some categories (e.g. roundtable) do not have well-defined orientations and are not included for evaluation. Figure 5.7 shows example results, and Figure 5.8 shows the distribution of prediction error.

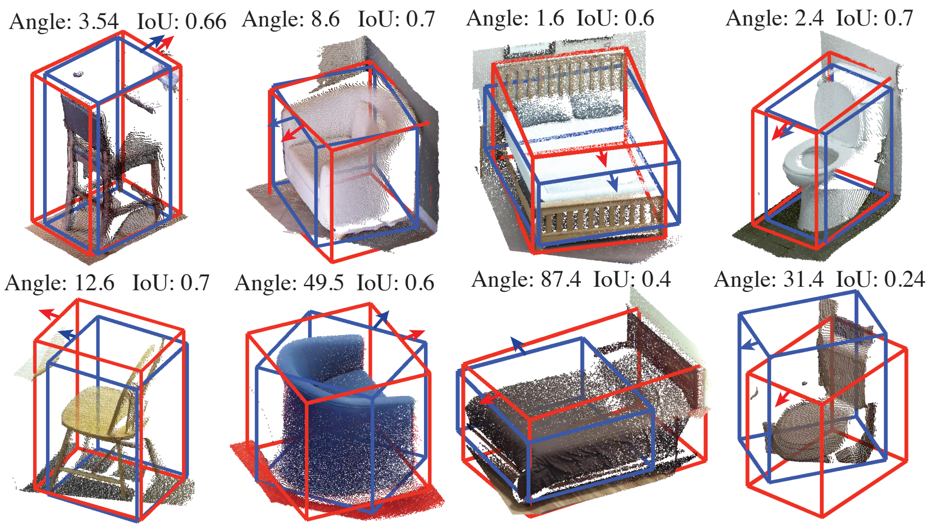


Figure 5.7: **Example results for 3D object detection and orientation prediction.** We show the angle difference and IoU between predicted boxes (blue) and ground truth (red).

Room Layout Estimation The spatial layout of the entire space of the scene allows more precise reasoning about free space (e.g., where can I walk?) and improved object reasoning. It is a popular but challenging task for color-based scene understanding (e.g. [59, 60, 61]). With the extra depth information in the RGB-D image, this task is considered to be much more feasible [193]. We evaluate the room layout estimation in 3D by calculating the Intersection over Union (IoU) between the free space from the ground truth and the free space predicted by the algorithm output.

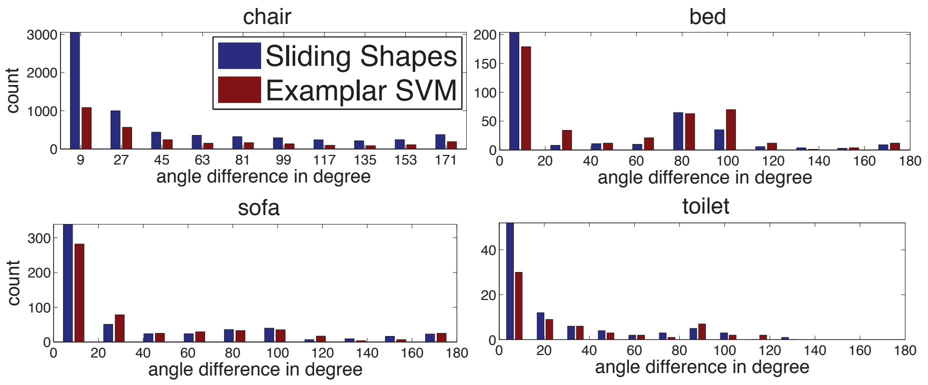


Figure 5.8: **Object orientation estimation.** Here we show the distribution of the orientation errors for all true positive detections.

As shown in Figure 5.10, the free space is defined as the space that satisfies four conditions: 1) within the camera field of view, 2) within the effective range, 3) within the room, and 4) outside any object bounding box (for room layout estimation, we assume empty rooms without objects). In terms of implementation, we define a voxel grid of $0.1 \times 0.1 \times 0.1$ meter^3 over the space and choose the voxels that are inside the field of view of the camera and fall between 0.5 and 5.5 meters from the camera, which is an effective range for most RGB-D sensors. For each of these effective voxels, given a room layout 3D polygon, we check whether the voxel is inside. In this way, we can compute the intersection and the union by counting 3D voxels.

This evaluation metric directly measures the free space prediction accuracy. However, we care only about the space within a 5.5-meter range; if a room is too big, all effective voxels will be in the ground truth room. If an algorithm predicts a huge room beyond 5.5 meters, then the IoU will be equal to one, which introduces bias: algorithms will favor a huge room. To address this issue, we only evaluate algorithms on the rooms with a reasonable size (not too big), since none of the RGB-D sensors, can see very far either. If the percentage of effective 3D voxels in the ground truth room is bigger than 95%, we discard the room in our evaluation.

Although there exists an algorithm for this task [193], we could not find an open source implementation. Therefore, we design three baselines: the simplest baseline

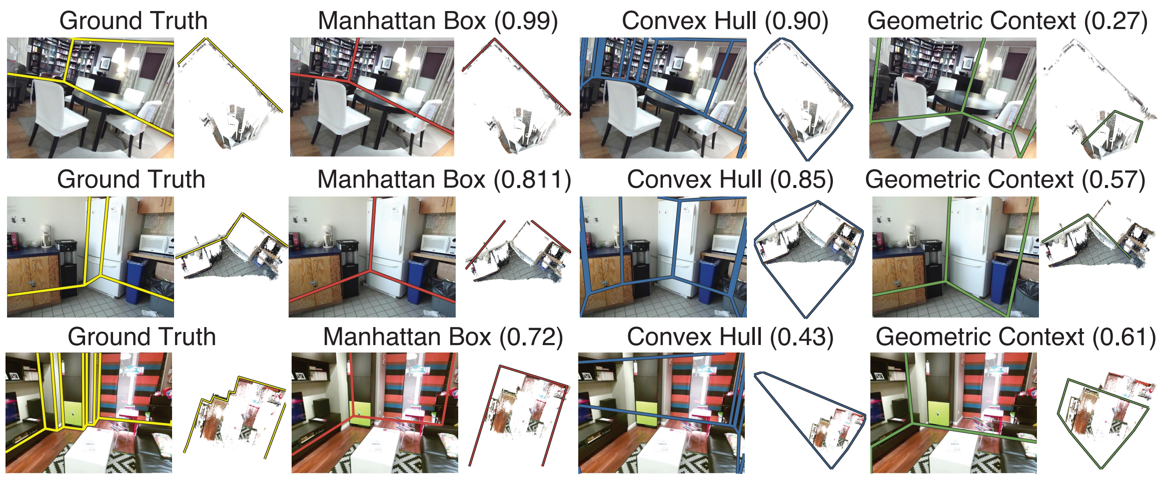


Figure 5.9: **Example visualization** to compare the three 3D room layout estimation algorithms.

(named Convex Hull) computes the floor and ceiling heights by taking the 0.1 and 99.9 percentiles of the 3D points along the gravity direction, and computes the convex hull of the point projection onto the floor plane to estimate the walls. Our stronger baseline (named Manhattan Box) uses plane fitting to estimate a 3D rectangular room box. We first estimate the three principal directions of the point cloud based on the histogram of normal directions (see Section 5.1.3). We then segment the point cloud based on the normal orientation and look for the planes with the furthest distance from the center to form a box for the room layout. To compare with the color-based approach, we run Geometric Context [59] on the color image to estimate the room layout in 2D. We then use the camera tilt angle from gravity direction estimation and the focal length from the sensor to reconstruct the layout in 3D with single-view geometry, using the estimated floor height to scale the 3D layout properly. Figure 5.9 shows examples of the results of these algorithms. Average IoU for Geometric Context is 0.442, Convex Hull is 0.713, and Manhattan Box is 0.734 performs best.

Total Scene Understanding The final task for our scene understanding benchmark is to estimate the whole scene including objects and room layout in 3D [100]. This task is also referred to "Basic Level Scene Understanding" in [184]. We propose

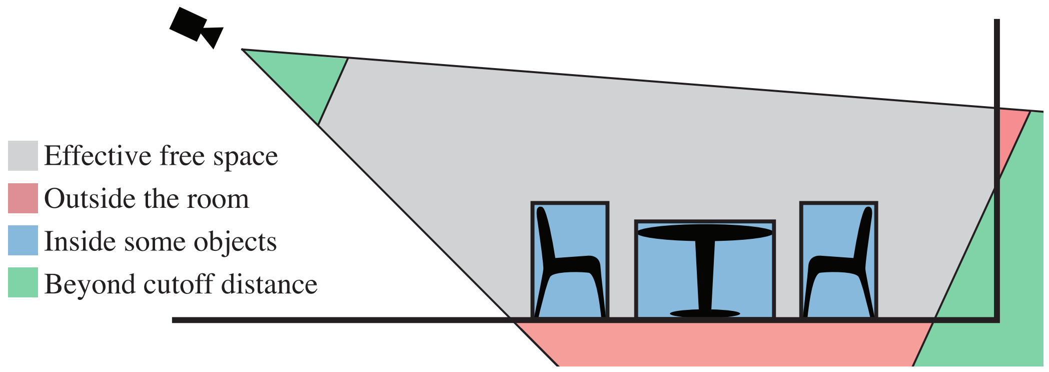


Figure 5.10: **Free space evaluation.** The free space is the gray area inside the room, outside any object bounding boxes, and within the effective minimal and maximal range [0.5m-5.5m]. For evaluation, we use IoU between the gray areas of the ground truth and the prediction as the criteria.

this benchmark task as the final goal to integrate both object detection and room layout estimation to obtain a total scene understanding, recognizing and localizing all the objects and the room structure.

We evaluate the result by comparing the ground truth objects and the predicted objects. To match the prediction with ground truth, we compute the IoU between all pairs of predicted boxes and ground truth boxes, and we sort the IoU scores in a descending order. We choose each available pair with the largest IoU and mark the two boxes as unavailable. We repeat this process until the IoU is lower than a threshold τ ($\tau = 0.25$ in this case). For each matched pair between ground truth and prediction, we compare their object label in order to know whether it is a correct prediction or not. Let $|\mathcal{G}|$ be the number of ground truth boxes, $|\mathcal{P}|$ be the number of prediction boxes, $|\mathcal{M}|$ be the number of matched pairs with IoU$> \tau$, and $|\mathcal{C}|$ be the number of matched pairs with a correct label. We evaluate the algorithms by computing three numbers: $R_r = |\mathcal{C}| \,/\, |\mathcal{G}|$ to measure the recall of recognition for both semantics and geometry, $R_g = |\mathcal{M}|/|\mathcal{G}|$ to measure the geometric prediction recall, and $P_g = |\mathcal{M}|/|\mathcal{P}|$ to measure the geometric prediction precision. We also evaluate the free space by using a similar scheme as for room layout: counting the visible 3D voxels for the free space, i.e. inside the room polygon but outside any object

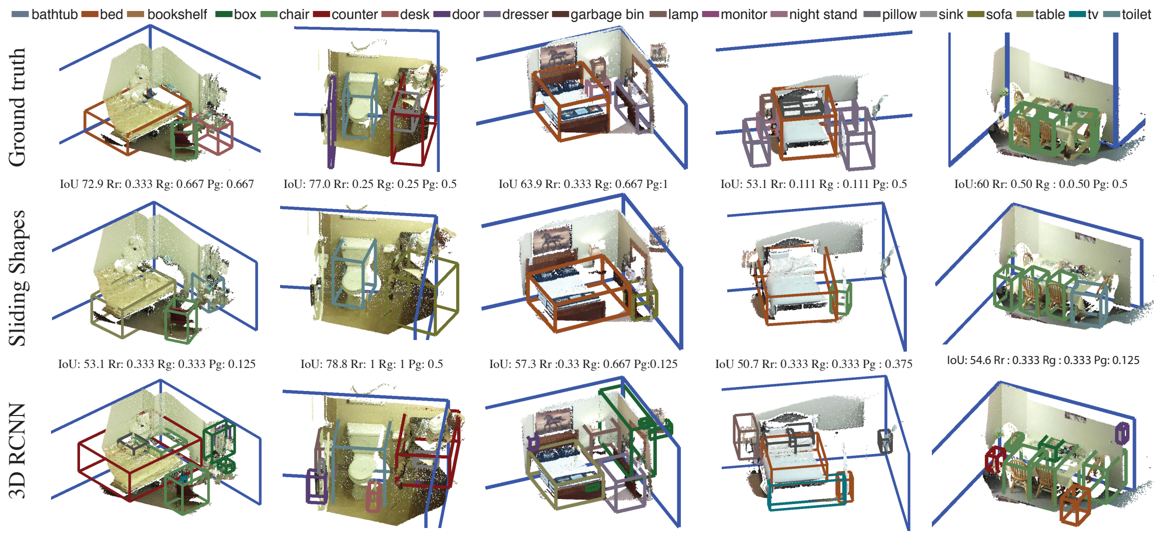

bathtub
bed
bookshelf
box
chair
counter
desk
door
dresser
garbage bin
lamp
monitor
night stand
pillow
sink
sofa
table
tv
toilet
Ground truth
Sliding Shapes
3D RCNN
IoU 72.9 Rr: 0.333 Rg: 0.667 Pg: 0.667
IoU: 77.0 Rr: 0.25 Rg: 0.25 Pg: 0.5
IoU 63.9 Rr: 0.333 Rg: 0.667 Pg:1
IoU: 53.1 Rr: 0.111 Rg : 0.111 Pg: 0.5
IoU:60 Rr: 0.50 Rg : 0.0.50 Pg: 0.5
IoU: 53.1 Rr: 0.333 Rg: 0.333 Pg: 0.125
IoU: 78.8 Rr: 1 Rg: 1 Pg: 0.5
IoU: 57.3 Rr :0.33 Rg: 0.667 Pg:0.125
IoU 50.7 Rr: 0.333 Rg: 0.333 Pg : 0.375
IoU: 54.6 Rr : 0.333 Rg : 0.333 Pg: 0.125

Figure 5.11: **Visualization of total scene understanding results.**

bounding box. Again, we compute the IoU between the free space of ground truth and prediction.

We use RGB-D RCNN and Sliding Shapes for object detection and combine them with Manhattan Box for room layout estimation. We do non-maximum suppression across object categories. For RGB-D RCNN, we estimate the 3D bounding boxes of objects from the 2D detection results. To get the 3D box we first project the points inside the 2D window to 3D. Along each major direction of the room we build a histogram of the point count. Starting from the median of the histogram, we set the box boundary at the first discontinuous location. We also set a threshold of detection confidence and maximum number of objects in a room to further reduce the number of detections. With the objects and room layout in hand we propose four simple ways to integrate them: (1) directly combines them; (2) remove the object detections that fall outside the estimated room layout; (3) adjust room to encompass 90 % the objects; (4) adjust the room according to the majority of objects and remove the out-of-room objects. Figure 5.11 and Table 5.3 show the results.

	RGB-D RCNN				Sliding Shapes			
	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
P_g	21.5	21.7	21.4	22.3	33.2	37.7	33.2	**37.8**
R_g	38.2	39.4	**40.8**	39.0	32.5	32.4	32.5	32.3
R_r	21.5	**32.6**	20.4	21.4	23.7	23.7	23.7	23.7
IoU	59.5	60.5	59.5	59.8	65.1	65.8	65.2	**66.0**

Table 5.3: **Evaluation of total scene understanding.** With the objects detection result from Sliding Shape and RCNN and Manhattan Box for room layout estimation, we evaluate four ways to integrate object detection and room layout: (1) directly combine (2) constrain the object using the room. (3) adjust room base on the objects (4) adjust the room and objects together.

Cross sensor Because real data likely come from different sensors, it is essential that an algorithm can generalize across them. Similar to dataset bias [170], we study sensor bias for different RGB-D sensors. We conduct an experiment to train a DPM object detector using data captured by one sensor and test on data captured by another to evaluate the cross-sensor generality. To separate out the dataset biases, we do this experiment on a subset of our data, where an Xtion and a Kinect v2 are mounted on a rig with large overlapping views of the same places. From the result in Table 5.4, we can see that sensor bias does exist. Both color and depth based algorithms exhibit some performance drop. We hope this benchmark can stimulate the development of RGB-D algorithms with better sensor generalization ability.

	Train / Test	Kinect v2			Xtion			Percent drop (%)		
		rgb	d	rgbd	rgb	d	rgbd	rgb	d	rgbd
chair	Kinect v2	18.07	22.15	24.46	18.93	22.28	24.77	-4.76	-0.60	-1.28
	Xtion	12.28	16.80	15.31	15.86	13.71	23.76	29.22	-18.39	55.23
table	Kinect v2	15.45	30.54	29.53	16.34	8.74	18.69	-5.78	71.38	36.70
	Xtion	8.13	24.39	28.38	14.95	18.33	24.30	45.64	-33.05	-16.79

Table 5.4: **Cross-sensor bias.**

5.2 Synthetic 3D Scene Dataset: SUNCG

While the real-world 3D scene datasets like SUNRGB-D are useful for training and validating algorithm, the high cost of the real-world data collection still limits the scale of the data that we can obtain. Moreover, current datasets lack pixel level accuracy due to sensor noise or labeling error (Fig. 5.12).

This has led to utilizing synthetic 3D data for training our data-driven 3D scene understanding model. We introduced SUNCG, a large-scale synthetic 3D scene dataset with more than 45622 indoor environments designed by people. All the3D scenes are composed of individually labeled 3D object meshes. Using such realistic indoor 3D environments enable us to create 2D images for training data-driven models. This dataset provides the following unique properties that open up new research opportunities:

- Realistic context settings that enable us to study contextual information beyond a single object.
- Complete 3D environment, which enables us to study problem like scene completion that predicts information beyond the visible 3D surfaces.
- Full control over the 3D scenes, which enables us to systematically manipulate objects or the lighting condition in the scene to generate more variety of data.
- Full control of camera viewpoints, which is not limited to the camera viewpoints during data capture.
- Precise and accurate label. Since we have access to the source 3D models, we can generate dense per-pixel or per-voxel labeled training data with little cost.

Because of these unique properties, dataset can support a wide range of applications that includes not only traditional computer vision tasks but also new tasks that were not possible before.

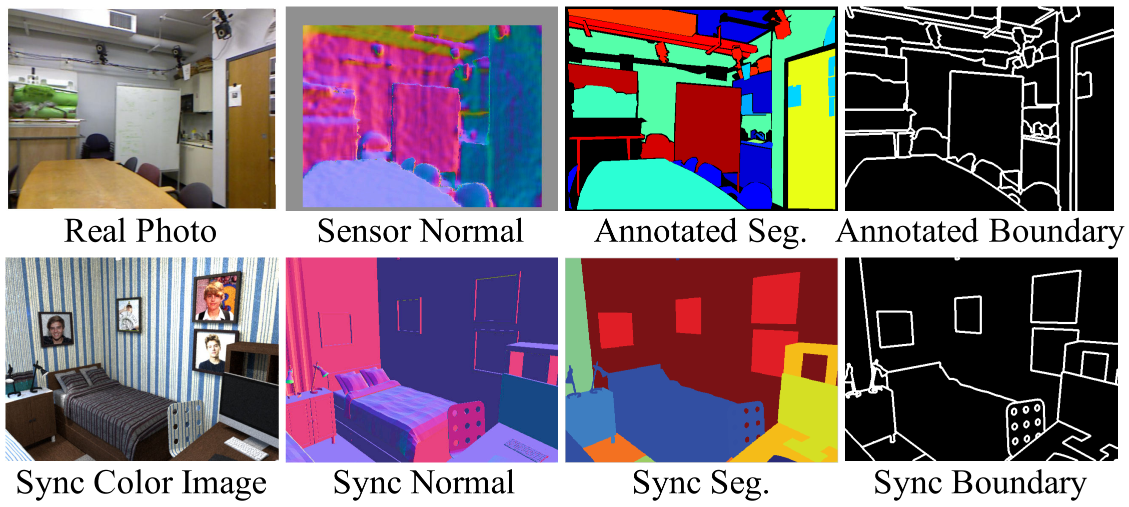


Figure 5.12: Real data (top) vs. synthetic data (bottom). For the real data, note the noise in normal map and the diminishing accuracy at object boundaries in the semantic labels.

5.2.1 Related Work

Using synthetic data to increase the data density and diversity for deep neural network training has shown promising results. To date, synthetic data have been utilized to generate training data for predicting object pose [165, 117, 46], optical flow [24], semantic segmentation [53, 55, 194, 131], and investigating object features [5, 78].

Su *et al.* [165] used individual objects rendered in front of arbitrary backgrounds with prescribed angles relative to the camera to generate data for learning to predict object pose. Similarly, Dosovitskiy *et al.* [24] used individual objects rendered with arbitrary motion to generate synthetic motion data for learning to predict optical flow. Both works used unrealistic OpenGL rendering with fixed lights, where physically based effects such as shadows, reflections were not taken into account. Movshovitz *et al.* [117] used environment map lighting and showed that it benefits pose estimation. However, since individual objects are rendered in front of arbitrary 2D backgrounds, the data generated for these approaches lack correct 3D illumination effects due to their surroundings such as shadows and reflections from nearby objects with different materials. Moreover, they also lack a realistic context for the object under consideration.

Handa *et al.* [53, 55] introduced a laboriously created 3D scene dataset and demonstrated the usage of semantic segmentation training. However, their data consisted of rooms on the order of tens, which has significantly limited variation in context compared to our dataset with 45K realistic house layouts. Moreover, their dataset has no RGB images due to lack of colors and surface materials in their scene descriptions, hence they were only able to generate depth channels. Zhang *et al.* [194] proposed to replace objects in depth images with 3D models from ShapeNet [21]. However, there is no guarantee whether replacements will be oriented correctly with respect to surrounding objects or be stylistically in context. In contrast, we take advantage of a large repository of indoor scenes created by human, which guarantees the data diversity, quality, and contextual relevance.

Xiang *et al.* [181] introduced a 3D object-2D image database, where 3D objects are manually aligned to 2D images. The image provides context, however, the 3D data contains only the object without room structures, it is not possible to extract per-pixel ground truth for the full scene. The dataset is also limited by the number of images provided (90K). In contrast, we can provide as many (rendered image, per-pixel ground truth) pairs as one wants.

Recently, Richter *et al.* [131] demonstrated collecting synthetic data from a realistic game engine by intercepting the communication between game and the graphics hardware. They showed that the data collected can be used for semantic segmentation task. Their method ensures as much context as there is in the game (Although it is limited to only outdoor context, similar to the SYNTHIA [134] dataset). However they largely reduced the human labor in annotation by tracing geometric entities across frames, the ground truth (i.e. per-pixel semantic label) collection process is not completely automated and error-prone due to the human interaction: even though they track geometry through frames and propagate most of the labels, a person needs to label new objects emerging in the recorded synthetic video. Moreover, it is not trivial to alter camera view, light positions, and intensity, or rendering method due to lack of access to low-level constructs in the scene. On the other hand, our data and label generation process is automated, and we have full control over how the scene is lit and rendered.

Moreover, most of the work only use rendered image pairs of the dataset as training data, while the 3D aspect of such data has not been fully utilized.

5.2.2 Dataset Construction

Our SUNCG dataset contains $45,622$ different scenes with realistic room and furniture layouts that are manually created through the Planner5D platform [126]. Planner5D is an online interior design interface that allows users to create multi-floor room layouts, add furniture from an object library, and arrange them in the rooms. After removing duplicated and empty scenes, we ensured the quality of the data with a pure Mechanical Turk cleaning task. During the task, we showed a set of top view renderings of each floor and asked workers to vote whether this is a valid apartment floor. We collected three votes for each floor and consider a floor valid when it has at least two positive votes. In the end, we have $49,884$ valid floors, containing $404,058$ rooms and $5,697,217$ object instances from 2644 unique object meshes covering 84 categories. We manually labeled all the objects in the library to assign category labels. Figure 3.6 shows example scenes from the resulting SUNCG dataset.

5.2.3 SUNCG Dataset Statistics

In this section, we present several statistics related to our SUNCG dataset. We start by providing the basic statistics of scene structure and physical size for 3D scenes in our dataset and then move on to talk about higher-level statistics regarding object categories, room types, and object-room relationships.

Scene Structure Statistics Figure 5.13 illustrates the distribution of the number of rooms and number of floors per scene in the SUNCG dataset. The 3D scenes in our dataset range from single room studio to multi-floor houses. The average and the median number of rooms per-house was found to be 8.9 and 7 respectively. The average and the median number of floors per-house was found to be 1.3 and 1 respectively.

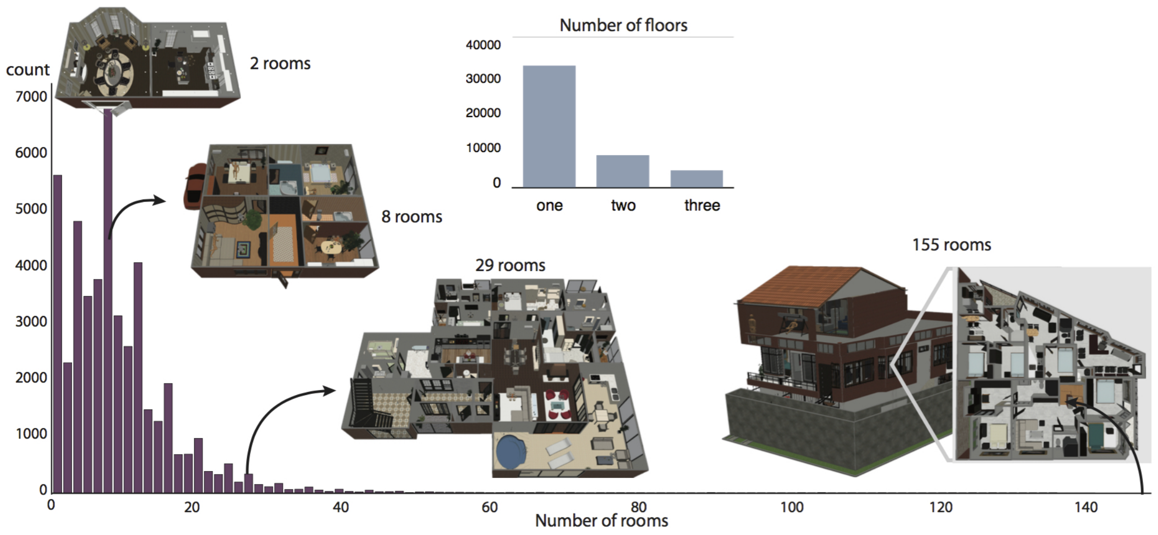


Figure 5.13: **Scene structure statistics.** Distribution of number of rooms and number of floors in our SUNCG dataset. Our dataset contains a large variety of 3D indoor scenes such as small studios, multi-room apartments, and multi-floor houses.

Physical Size Statistics All object meshes and 3D scenes in the SUNCG dataset are measured in real-world spatial dimensions (units are in meters). Figure 5.14 shows statistics related to physical size over three levels: rooms, floors, and houses.

Room Type Statistics Figure 5.15 shows the room type distribution and several example rooms per type from our dataset. In total, we have 24 room types that

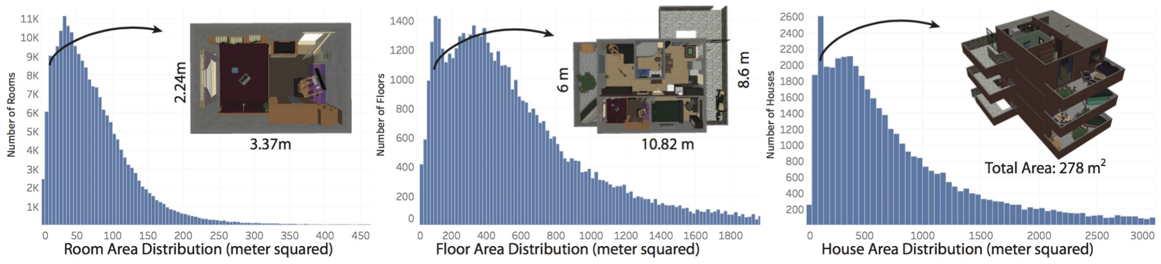


Figure 5.14: **Distribution of physical sizes** (in meters^2) per room, floor, and house of the SUNCG dataset.

are labeled by the user during creation. These labels include: living room, kitchen, bedroom, child room, dining room, bathroom, toilet, hall, hallway, office, guest room, wardrobe, room, lobby, storage, boiler room, balcony, loggia, terrace, entryway, passenger elevator, freight elevator, aeration, and garage. The four most common room types in our dataset are the bedroom, living room, kitchen and toilet, which agrees with the distribution in real-world living spaces.

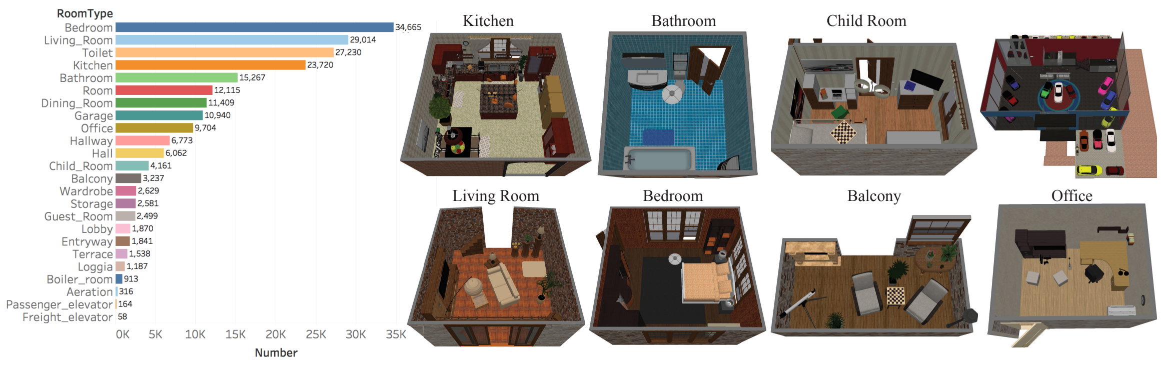


Figure 5.15: Distribution of different room types in the SUNCG dataset (left), and examples of rooms per room type (right).

Object Category Statistics Figure 5.16 shows overall object category occurrence in the SUNCG dataset. Object models from the object library contain a diverse set of common furniture and objects for common living spaces. Furthermore, during the creation of the 3D scenes, users have the flexibility to reshape, resize, and re-apply texture to objects to better fit the room style, which further improves the dataset diversity.

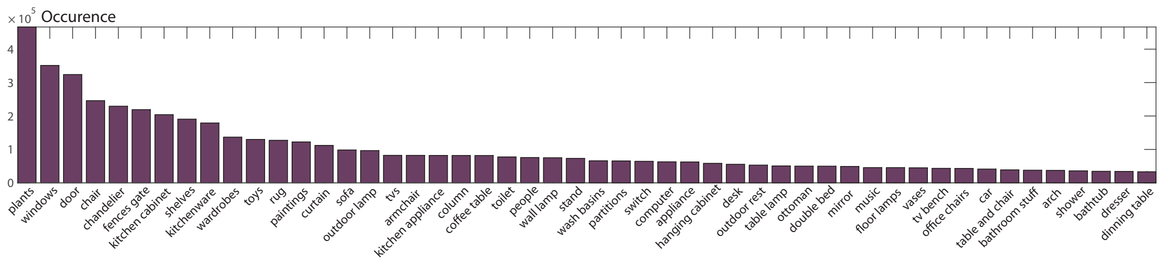


Figure 5.16: **Distribution of object categories in the SUNCG dataset.** We have 84 object categories in total. Here we show the top 50 object categories with highest number of occurrences in our dataset.

Object-Room Relationships With complete object and room type annotations, we can further study the object-room relationships in our dataset. Figure 5.17 (a) shows the distribution of the objects per room. Figure 5.17 (b) shows the distribution of object categories conditioned on different room types. On average there are more than 14 objects in each room. The occurrence and arrangements of these objects in rooms provide rich contextual information that we can learn from.

5.2.4 Tasks Supported by SUNCG

Because of the richness of the dataset, SUNCG has been used for a wide range of applications that includes not only traditional computer vision tasks but also new tasks which was not possible before. The rest of this section discusses some example tasks that are supported and enabled by SUNCG dataset, ranging from computer vision and computer graphics to robotics.

Computer Vision Tasks. In computer vision, SUNCG has been used for generating image data and their per-pixel ground truth annotation though rendering. For example, studies have used SUNCG to train models for normal prediction, semantic

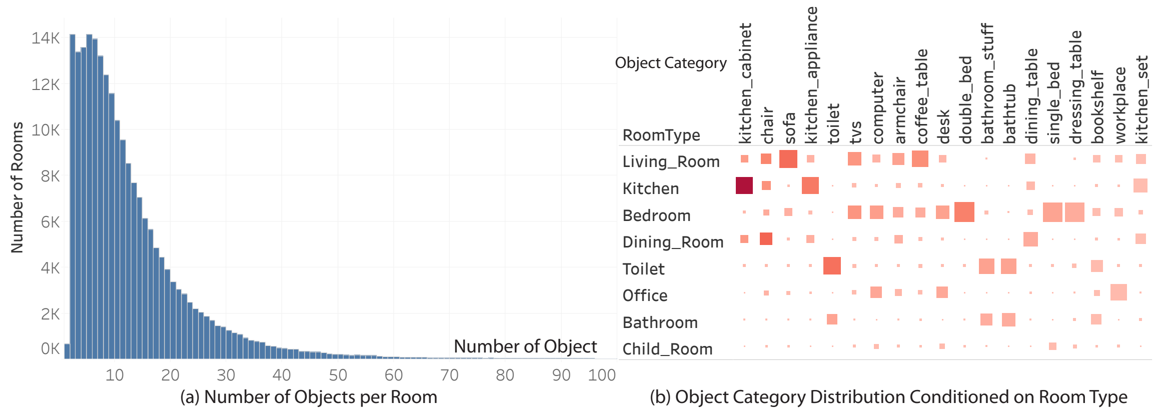


Figure 5.17: **Object-Room Relationship.** On the left we show the distribution of number of objects in each room. On average there are more than 14 objects in each room. On the right, we show the object category distribution conditioned on different room type. Size of the square shows the frequency of a given object category appears in the certain room type. The frequency is normalized for each object category. As expected, object occurrences are tightly correlated with the room type. For example, kitchen counters have a very high chance to appear in kitchen, while chairs appear frequently in many room types.

segmentation and object edge detection [196, 125]. Beyond traditional computer vision task, SUNCG also enables new tasks such as semantic scene completion [157, 22] (Chapter 3) by generating per-voxel ground truth and semantic-structure view extrapolation (Chapter 4) by generating full-view panorama of the environment, and 3D scene parsing [171] by providing the complete 3D scene information on the object's shape, pose and scene layout.

SUNCG can also be used to study problems in the intersection of computer vision and natural language processing. For example, Abhishek et al. [23] used SUNCG to train an embodied for the task of visual question answering, and Anderson et al. [4] used it to learn visually-grounded navigation instructions.

Computer Graphics Tasks. In computer graphics, the vast variety of realistic indoor environment configuration provided by SUNCG naturally served as a useful data source for the task of indoor scene synthesis. For example, Jian et al. [128] used SUNCG to learn a human-centric stochastic grammar for 3D scene generation. On the other hand, Wang et al. [175] proposed to use SUNCG to learn Deep convolutional priors for indoor scene synthesis. Similarly, Zhang et al. [198] learned a hybrid representation from SUNCG for scene generation.

Robotics Tasks. Apart from providing training data for image understanding and scene synthesis, SUNCG can also be used as simulation environments for training robotics task, such as navigation, object interaction, and high-level task planning. The popular simulation environments that uses SUNCG dataset includes, MINOS simulator [139], HOME simulator [18], and House3D simulator [179]. The realistic, complex and diverse scene configurations provided by SUNCG make it easier for the robotic agent to learn from different complex scenarios and adapt to real-world settings.

5.3 Summary

In this chapter, we introduced a real-world RGB-D dataset SUN RGB-D and a synthetic 3D scene dataset SUNCG. Apart from constructing these large-scale datasets, we also propose a set of 3D metrics and evaluate algorithms for all major tasks towards total 3D scene understanding. Beyond computer vision, these datasets have also been widely used as training data and simulation environments for robotics applications such as manipulation, navigation, and reinforcement learning. We hope that our benchmarks will help enable progress for data-driven 3D scene understanding in the coming years.

Chapter 6

Future Directions

Most computer vision algorithms are built to enable intelligent systems to understand the physical world. Yet, these algorithms continue to assume the system's role to be that of a passive observer - without the ability to interact with its environment. This assumption becomes a fundamental limitation for applications in robotics, where systems are intrinsically built to actively engage with the environment that it perceives.

The future research direction will include developing a new class of self-improving perception algorithms built on 3D amodal scene representations that not only enable the active exploration, but also make use of the interactions to achieve better perception. In the following section, we will discuss several possible future directions that are worth exploring in the field of data-driven 3D scene understanding.

Learning Interaction Strategies to Facilitate 3D Scene Understanding. Methods of active perception should expand beyond isolating objects from clutter to improve recognition, and next-best-view techniques [180] (e.g. selecting the next best viewing angle that best reduces recognition uncertainty). Inspired by the extent of humans' abilities that actively explore environments to retrieve information (i.e. flipping over a book to see its title), it is interesting to investigate how learning can be used to endow intelligent systems with the ability to autonomously acquire more complex behaviors in order to facilitate the perception task. Rather than heuristically instructing the system which strategy to use (e.g. our ARC approach), the robot should be able to autonomously select its own strategy when it deems possible and necessary. Interaction strategies might include: controlling the system's own lighting to reduce recognition ambiguity, intelligently manipulating a pile of objects to determine the identities of occluded objects underneath, and autonomously navigating through complex and unstable environments to improve the mapping of obscured regions. This research ties closely with applications that involve complex and uncommon scenarios such as emergency response robotics, which may require the system to determine it own strategies on the spot without any pre-defined instructions from an operator.

Leveraging Interactions to Optimize 3D Scene Data Collection. Apart from using interaction strategies to facilitate perception during test time, it is also possible to leverage interactions for collecting massive amounts of training data for perception. In 3DMatch [191], I make use of large-scale unlabeled video motion data to self-supervise the training of a powerful geometric descriptor. To take this idea one step further, instead of just using a simple mobile robot to wander around while collecting data, I plan to develop algorithms that leverage precise movement and manipulation in order to collect the most *effective* training data.

More specifically, given a task, an existing training dataset, and a machine learning model, the system's goal should be to autonomously collect new training images of the environment that can "best benefit" the learning algorithm. This process may involve exploring and capturing images of the scene from unfamiliar perspectives, collecting more data for uncommon scenarios (e.g. under rainy or snowy weather for outdoor applications), or interacting with the scene (e.g. opening/closing the curtains, turning on/off the TV) to create variations of the scene that can improve the robustness of the learning-based algorithms. I believe this interactive learning schema is especially important for the robustness of data-driven perception systems operating in diverse real-world environments, where no single dataset can cover all aspects of the scene.

Building Persistent 3D Scene Representations Across Time and Devices. As the number of smart devices, vehicles, and robots equipped with 3D scanners increases exponentially, it is important to explore how these systems could collectively contribute to a shared, persistent, and consistent 3D scene representation of the world that continually updates over time. With this representation, an agent making repeat visits to the same environment can leverage information about the static parts of scene learned from previous visits and focus on understanding the dynamic parts of the scene; likewise a mobile agent could optimize its own navigation routes by accounting for the information observed from separate cameras embedded in the environment.

As an initial step in this direction, in "Robot in a room" [159] I built a system featuring a mobile household robot in a single agent scenario. The system effectively tracks the changes that occur over time in a scene even without direct observation of the changes as they occur. The system then leverages the tracking to efficiently re-localize itself in the dynamic environment. Moving forward, it will be interesting to expand this idea into a distributed mutli-agent system, where all agents collectively contribute to a global understanding of 3D scenes via updates over time. These updates can be efficiently aggregated into a unified 3D amodal representation, which can be easily shared and utilized by all agents for complex tasks. As part of this direction, one of the possible long term goals is to enable a world-scale RGB-Depth device networks (e.g. robots, wearable devices, smart phones) to distributively capture data, understand their environments, and contribute their learned models to a global persistent 3D scene representation, to collectively improve their perceptual understanding of the world as a whole.

Chapter 7

Conclusion

The goal of this dissertation is to build computer systems that can see and understand our physical world in a way that they are able to safely interact with this world and assist us in our daily lives. We believe that this requires machines to understand the complete 3D scenes around them, including the 3D geometry, semantics, and contextual information, which is a task far beyond just labeling 2D pixels.

This dissertation work demonstrates that leveraging these 3D representations in data-driven models not only significantly outperforms analogous algorithms using image representations, but also paves the way for new scene understanding tasks (e.g. 3D scene completion) that have previously been considered ill-posed given only 2D representations.

Moving forward, I think it is also important to rethink the role of computer vision systems. Rather than treating them as passive observers, we should consider them as active explorers of our 3D world, and make use of this to improve their understanding of the environment continuously. By reconnecting interaction with perception, we will be able to create more powerful and intelligent AI systems.

Bibliography

[1] Scene toolkit: https://github.com/smartscenes/stk.

[2] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *ECCV*. 2012.

[3] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, 2012.

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

[5] Mathieu Aubry and Bryan C Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2875–2883, 2015.

[6] Moshe Bar. Visual objects in context. *Nature reviews. Neuroscience*, 5(8):617, 2004.

[7] B. I. Barbosa, M. Cristani, A. Del Bue, L. Bazzani, and V. Murino. Re-identification with rgb-d sensors. In *First International Workshop on Re-Identification*, 2012.

[8] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24–1, 2009.

[9] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013.

[10] Maroš Bláha, Christoph Vogel, Audrey Richard, Jan D Wegner, Thomas Pock, and Konrad Schindler. Large-scale semantic 3d reconstruction: an adaptive multi-resolution model for multi-class volumetric labeling.

[11] Manuel Blum, Jost Tobias Springenberg, Jan Wulfing, and Martin Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *ICRA*, 2012.

[12] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *CVPR*, 2011.

[13] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *IROS*, 2011.

[14] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for RGB-D based object recognition. In *ISER*, 2013.

[15] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Learning hierarchical sparse features for RGB-(D) object recognition. *IJRR*, 2014.

[16] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[17] Jean-Yves Bouguet. Camera calibration toolbox for matlab. 2004.

[18] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. Home: A household multimodal environment. *arXiv preprint arXiv:1711.11017*, 2017.

[19] Cristian Sminchisescu Catalin Ionescu, Fuxin Li. Latent structured models for human pose estimation. In *ICCV*, 2011.

[20] Angel X. Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017.

[21] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[22] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, Thomas Brox, et al. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE, 2015.

[25] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[26] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[27] Tarek El-Gaaly and Marwan Torki. Rgbd object pose recognition using local-global multi-kernel regression. In *ICPR*, 2012.

[28] Nesli Erdogmus and Sébastien Marcel. Spoofing in 2d face recognition with 3d masks and anti-spoofing with kinect. In *BTAS*, 2013.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[30] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *IJCV*, 2013.

[31] Yi Fang, Jin Xie, Guoxian Dai, Meng Wang, Fan Zhu, Tiantian Xu, and Edward Wong. 3D deep shape descriptor. In *CVPR*, 2015.

[32] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010.

[33] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *CVPR*, 2016.

[34] Simon Fothergill, Helena M. Mentis, Pushmeet Kohli, and Sebastian Nowozin. Instructing people for training gestural interactive systems. In *CHI*, 2012.

[35] Ravi Garg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[36] Andreas Geiger and Chaohui Wang. Joint 3D object and layout inference from a single RGB-D image. In *German Conference on Pattern Recognition (GCPR)*, 2015.

[37] Ross Girshick. Fast R-CNN. *ICCV*, 2015.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[39] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[40] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *ICCV*, 2009.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[42] David Gossow, David Weikersdorfer, and Michael Beetz. Distinctive texture features from perspective-invariant keypoints. In *ICPR*, 2012.

[43] Ruiqi Guo and Derek Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013.

[44] Ruiqi Guo and Derek Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013.

[45] Ruiqi Guo, Chuhang Zou, and Derek Hoiem. Predicting complete 3D models of indoor scenes. *ICCV*, 2015.

[46] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740, 2015.

[47] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013.

[48] Saurabh Gupta, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015.

[49] Saurabh Gupta, Ross Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014.

[50] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *Computer Vision–ECCV 2014*, pages 345–360. Springer, 2014.

[51] Saurabh Gupta, Ross Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014.

[52] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. *arXiv*, 2015.

[53] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014.

[54] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. *CoRR*, abs/1511.07041, 2015.

[55] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015.

[56] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3D scene reconstruction and class segmentation. In *CVPR*, pages 97–104. IEEE Computer Society, 2013.

[57] James Hays and Alexei A Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[59] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[60] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*. 2010.

[61] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering free space of indoor scenes from a single image. In *CVPR*, 2012.

[62] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[63] Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3d object recognition from range images using local feature histograms. In *CVPR*, 2001.

[64] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*. 2013.

[65] JE Hochberg. Perception (2nd edn), 1978.

[66] Haibin Huang, Evangelos Kalogerakis, and Benjamin Marlin. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. *Computer Graphics Forum*, 2015.

[67] Helene Intraub and Michael Richardson. Wide-angle memories of close-up scenes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(2):179, 1989.

[68] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *PAMI*, 2014.

[69] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[70] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

[71] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshop on Consumer Depth Cameras for Computer Vision*, 2011.

[72] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. 2013.

[73] Dinesh Jayaraman and Kristen Grauman. Learning to look around. *arXiv preprint arXiv:1709.00507*, 2017.

[74] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[75] Zhaoyin Jia, Andy Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d-based reasoning with blocks, support, and stability. In *CVPR*, 2013.

[76] Hao Jiang and Jianxiong Xiao. A linear approach to matching cuboids in RGBD images. In *CVPR*, 2013.

[77] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *PAMI*, 1999.

[78] Biliana Kaneva, Antonio Torralba, and William T Freeman. Evaluation of image features using a photorealistic virtual world. In *2011 International Conference on Computer Vision*, pages 2282–2289. IEEE, 2011.

[79] Abhishek Kar, Shubham Tulsiani, João Carreira, and Jitendra Malik. Amodal completion and size constancy in natural scenes. In *ICCV*, 2015.

[80] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *ICRA*, 2013.

[81] Michal Kepski and Bogdan Kwolek. Fall detection using ceiling-mounted 3d depth camera.

[82] Aditya Khosla, Byoungkwon An, Joseph J. Lim, and Antonio Torralba. Looking beyond the visible scene. In *CVPR*, Ohio, USA, June 2014.

[83] Byungsoo Kim, Pushmeet Kohli, and Silvio Savarese. 3D scene understanding by Voxel-CRF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1425–1432, 2013.

[84] Young Min Kim, Niloy Mitra, Dongming Yan, and Leonidas Guibas. Acquisition of 3D indoor environments with variability and repetition. *ACM Trans. Gr*, 2012.

[85] Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *TOG*, 2012.

[86] Young Min Kim, Niloy J. Mitra, Dongming Yan, and Leonidas Guibas. Acquiring 3D indoor environments with variability and repetition. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 31(6), 2012.

[87] Hema S Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *Advances in neural information processing systems*, pages 244–252, 2011.

[88] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *ijrr*, 2013.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[90] K. Lai and D. Fox. Object recognition in 3D point clouds using web data and domain adaptation. In *IJRR*, 2010.

[91] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3D scene labeling. In *ICRA*. IEEE, 2014.

[92] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011.

[93] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A scalable tree-based approach for joint object and pose recognition. In *AAAI*, 2011.

[94] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *ICRA*, 2011.

[95] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[96] Karel Lenc and Andrea Vedaldi. R-CNN minus R. *bmvc*, 2015.

[97] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pages 1378–1386, 2010.

[98] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, 2015.

[99] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *ICCV*, 2013.

[100] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *ICCV*, 2013.

[101] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR*, 2009.

[102] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *PAMI*, 2011.

[103] Li Liu and Ling Shao. Learning discriminative representations from rgb-d video data. In *IJCAI*, 2013.

[104] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[105] Matthias Luber, Luciano Spinello, and Kai Oliver Arras. People tracking in rgb-d data with on-line boosted target models. In *IROS*, 2011.

[106] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann Lecun. Predicting Deeper into the Future of Semantic Segmentation. In *ICCV 2017*, 2017.

[107] Keith Lyle and Marcia Johnson. Importing perceived features into false memories. *Memory*, 14(2):197–213, 2006.

[108] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[109] Julian Mason, Bhaskara Marthi, and Ronald Parr. Object disappearance for object discovery. In *IROS*, 2012.

[110] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*. Wiley Online Library, 2014.

[111] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015.

[112] Stephan Meister, Shahram Izadi, Pushmeet Kohli, Martin Hämmerle, Carsten Rother, and Daniel Kondermann. When can we use kinectfusion for ground truth acquisition? In *Proc. Workshop on Color-Depth Camera Fusion in Robotics*, 2012.

[113] Ajmal Mian, Mohammed Bennamoun, and R Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *IJCV*, 2010.

[114] Patrick Min. Binvox http://www.patrickmin.com/binvox/.

[115] Rui Min, Neslihan Kose, and J-L Dugelay. Kinectfacedb: A kinect database for face recognition.

[116] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: Rebuilding man-made scenes with regular arrangements of planes. *TOG*, 34(4):103, 2015.

[117] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? *arXiv preprint arXiv:1603.08152*, 2016.

[118] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *TOG*, 2012.

[119] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 31(6), 2012.

[120] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[121] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[122] Bingbing Ni, Gang Wang, and Pierre Moulin. Rgbd-hudaact: A color-depth video database for human daily activity recognition. 2011.

[123] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[124] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[125] Benjamin Planche, Ziyan Wu, Kai Ma, Shanhui Sun, Stefan Kluckner, Oliver Lehmann, Terrence Chen, Andreas Hutter, Sergey Zakharov, Harald Kosch, et al. Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition. In *3D Vision (3DV), 2017 International Conference on*, pages 1–10. IEEE, 2017.

[126] Planner5D. https://planner5d.com/.

[127] François Pomerleau, Stéphane Magnenat, Francis Colas, Ming Liu, and Roland Siegwart. Tracking a depth camera: Parameter exploration for fast icp. In *IROS*, 2011.

[128] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5899–5908, 2018.

[129] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[130] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. RGB-(D) scene labeling: Features and algorithms. In *CVPR*, 2012.

[131] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[132] Andreas Richtsfeld, Thomas Morwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *IROS*, 2012.

[133] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3D object shape from one depth image. In *CVPR*, 2015.

[134] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[135] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[136] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2014.

[137] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2008.

[138] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[139] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017.

[140] Steven M Seitz and Charles R Dyer. Physically-valid view synthesis by image interpolation. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 18–25. IEEE, 1995.

[141] Steven M Seitz and Charles R Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30. ACM, 1996.

[142] Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M Seitz. Photo uncrop. In *European Conference on Computer Vision*, pages 16–31. Springer, 2014.

[143] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *TOG*, 2012.

[144] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. DeepPano: Deep panoramic representation for 3-D shape recognition. *Signal Processing Letters*, 2015.

[145] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, et al. Efficient human pose estimation from single depth images. *PAMI*, 2013.

[146] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013.

[147] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 2013.

[148] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011.

[149] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[150] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[151] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2014.

[152] Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3D object classification. In *NIPS*. 2012.

[153] Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*. 2012.

[154] Shuran Song, Samuel Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015.

[155] Shuran Song and Jianxiong Xiao. Sliding Shapes for 3D object detection in depth images. In *ECCV*, 2014.

[156] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3D object detection in rgb-d images. In *CVPR*, 2016.

[157] Shuran Song, Fisher Yu, Andy Zeng, Angel Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[158] Shuran Song, Andy Zeng, Angel X. Chang, Manolis Savva, Silvio Savarese, and Thomas Funkhouser. Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. In *CVPR*, 2018.

[159] Shuran Song, Linguang Zhang, and Jianxiong Xiao. Robot In a Room: Toward perfect object recognition in closed environments. In *arXiv*, 2015.

[160] Luciano Spinello and Kai Oliver Arras. People detection in rgb-d data. In *IROS*, 2011.

[161] S. Stein and S. J. McKenna. User-adaptive models for recognizing food preparation activities. 2013.

[162] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013.

[163] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012.

[164] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.

[165] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[166] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from rgbd images. *Plan, Activity, and Intent Recognition*, 2011.

[167] Jie Tang, Stephen Miller, Arjun Singh, and Pieter Abbeel. A textured object recognition pipeline for color and depth image data. In *ICRA*, 2012.

[168] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. *arXiv preprint arXiv:1704.03489*, 2017.

[169] Duc Thanh Nguyen, Binh-Son Hua, Khoi Tran, Quang-Hieu Pham, and Sai-Kit Yeung. A field model for repairing 3D shapes. In *CVPR*, June 2016.

[170] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[171] Shubham Tulsiani, Saurabh Gupta, David Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 302–310, 2018.

[172] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[173] Jacob Varley, Chad DeChant, Adam Richardson, Avinash Nair, Joaqun Ruales, and Peter Allen. Shape completion enabled robotic grasping. *arXiv*, 2016.

[174] Kentaro Wada, Kei Okada, and Masayuki Inaba. Fully convolutional object depth prediction for 3d segmentation from 2.5 d input.

[175] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):70, 2018.

[176] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.

[177] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *ICRA*, 2013.

[178] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3d object classification. In *ROBIO*, 2011.

[179] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[180] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[181] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference on Computer Vision*, pages 160–176. Springer, 2016.

[182] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[183] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[184] Jianxiong Xiao, James Hays, Bryan C. Russell, Genevieve Patterson, Krista Ehinger, Antonio Torralba, and Aude Oliva. Basic level scene understanding: Categories, attributes and structures. *Frontiers in Psychology*, 4(506), 2013.

[185] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013.

[186] Jianxiong Xiao, Shuran Song, Daniel Suo, and Fisher Yu. Marvin: A minimalist GPU-only N-dimensional ConvNet framework. 2016. Accessed: 2015-11-10.

[187] Jin Xie, Yi Fang, Fan Zhu, and Edward Wong. DeepShape: Deep learned shape descriptor for 3D shape matching and retrieval. In *CVPR*, 2015.

[188] Xuehan Xiong, Daniel Munoz, J. Andrew (Drew) Bagnell, and Martial Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *ICRA*, 2011.

[189] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[190] Bernhard Zeisl, Kevin Koser, and Marc Pollefeys. Automatic registration of rgb-d scans via salient directions. In *ICCV*, 2013.

[191] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.

[192] Haopeng Zhang, Tarek El-Gaaly, Ahmed Elgammal, and Zhiguo Jiang. Joint object and pose recognition using homeomorphic manifold analysis. In *AAAI*, 2013.

[193] Jian Zhang, Chen Kan, Alexander G Schwing, and Raquel Urtasun. Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In *ICCV*, 2013.

[194] Yinda Zhang, Mingru Bai, Pushmeet Kohli, Shahram Izadi, and Jianxiong Xiao. Deep-context: Context-encoding neural pathways for 3d holistic scene understanding. *CoRR*, 2016.

[195] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. PanoContext: A whole-room 3D context model for panoramic scene understanding. In *ECCV*, 2014.

[196] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017.

[197] Yinda Zhang, Jianxiong Xiao, James Hays, and Ping Tan. Framebreak: Dramatic image extrapolation by guided shift-maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1171–1178, 2013.

[198] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *arXiv preprint arXiv:1808.02084*, 2018.

[199] Bo Zheng, Yibiao Zhao, Joey C. Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *CVPR*, pages 3127–3134. IEEE Computer Society, 2013.

[200] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.

[201] Xiaolong Zhu, Huijing Zhao, Yiming Liu, Yipu Zhao, and Hongbin Zha. Segmentation and classification of range image from an intelligent vehicle in urban environment. In *IROS*, 2010.

[202] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017.