# From Pixels to Scenes: Recovering 3D Geometry and Semantics for Indoor Environments

Yinda Zhang

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: Professor Thomas Funkhouser

November 2018

# Abstract

Understanding the 3D geometry and semantics of real environments is in critically high demand for many applications, such as autonomous driving, robotics, and augmented reality. However, it is extremely challenging due to imperfect and noisy measurements from real sensors, limited access to ground truth data, and cluttered scenes exhibiting heavy occlusions and intervening objects. To address these issues, this thesis introduces a series of works that produce a geometric and semantic understanding of the scene in both pixel-wise and holistic 3D representations. Starting from estimating a depth map, which is a fundamental task in many approaches for reconstructing the 3D geometry of the scene, we introduce a learning-based active stereo system that is trained in a self-supervised fashion and reduces the disparity error to 1/10th of other canonical stereo systems. To handle a more common case where only one input image is available for scene understanding, we create a high-quality synthetic dataset facilitating pre-training of data-driven approaches, and demonstrating that we can improve the surface normal estimation and improve raw depth measurements from commodity RGBD sensors. Lastly, we pursue holistic 3D scene understanding by estimating a 3D representation of the scene, in which objects and room layout are represented using 3D bounding box and planar surfaces respectively. We propose methods to produce such a representation from either a single color panorama or a depth image, leveraging scene context. On the whole, these proposed methods produce understanding of both 3D geometry and semantics from the most fine-grained pixel level to the holistic scene scale, building foundations that support future work in 3D scene understanding.

# Acknowledgements

At the very beginning, I would like to send my deepest gratitude to my thesis advisor Thomas Funkhouser. Working with him taught me a lot about critical and creative thinking, hard working, and integrity. His open mindedness, optimistic altitude, and excitement about research greatly encourage and inspire me to continuously learn and explore interesting topics. His generous and selfless advises on research, career, and life are truly invaluable to me.

I also want to thank Ping Tan and Jianxiong Xiao for building up the foundation of my professional skills and introducing me into the realm of computer vision and more specifically 3D scene understanding.

I am very fortunate to have great internship opportunities in Adobe, Microsoft, Matterport, and Google, and advised by very responsible mentors, including Ersin Yumer, Pushmeet Kohli, Matt Bell, Sean Fanello, and Shahram Izadi. Through daily working with them, I learned a lot about finding and solving problems from both the perspectives of academia and industrial.

I am also very lucky to work in Princeton Vision Lab and Graphics Group. I truly appreciate Adam Finkelstein and Szymon Rusinkiewicz for their valuable comments on coursework, presentations, and research projects. I enjoyed staying with brilliant students, especially Manolis Savva, Angel Chang, Tianqiang Liu, Pingmei Xu, Xinyi Fan, Maciej Halber, and Zeyu Jin. The collaboration, discussion, and personal friendship with them was a fruitful and indispensable experience in my Ph.D life.

I want to thank my thesis committee members, including Thomas Funkhouser, Szymon Rusinkiewicz, James Hays, Olga Russakovsky, and Ryan Adams, for their generous offering of time and valuable feedback on thesis writing and final oral presentation. I also thank Planner5d for large-scale virtual scenes, Matterport for large-scale real data, and authors of many other datasets including SUNCG, ScanNet, NYUv2, SUNRGBD. My projects would not be possible without their great efforts in collecting and sharing the data. My research projects and this thesis was funded partially by Princeton Fellowship, NSF, Google, and Intel.

Finally, I would like to dedicate this thesis to my parents for their forever understanding, support, and encouragement, and Renzhi Jing for endless love and staying on my side over a lot of deadlines.

*To my family*

*and*

*in memory of my dad.*

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Understanding the surrounding environment is a fundamental function of human vision. In daily life, humans process visual information from the surrounding environment and process it to produce higher-level semantic understanding of the environment [82, 244]. For example, when we enter a new hotel room, we understand the room layout and furniture arrangement together with their shape and functionality from a single glance. Similarly, when we drive, we recognize and interpret traffic signs even while moving at great speed. This capability of human scene understanding continuously provides us essential information, enables us to perform complex tasks, and is critical to our survival.

Granting computers the potential of scene understanding from visual inputs is an important goal of computer vision [209] and useful in many applications. For the outdoor environment, recognizing roads, traffic signs, moving pedestrians, and vehicles supports the control and decision-making of autonomous driving systems [118]; whereas, recognizing furniture, objects, as well as room layout in an indoor environment is critical for a robot to successfully explore the building and perform useful tasks [68]. Scene understanding also builds the foundation for augmented/virtual reality systems to generate and remove virtual contents in a consistent way relative to the the real scene [3, 182, 147, 4].

Driven by this huge demand, scene understanding has drawn attention from both academia and industry, and has been studied for decades. However, current computational systems are far from comparable with human performance, mostly due to brittleness in uncontrolled real-world scenarios that exhibit the high complexity of the scene understanding task in three different aspects. First, scene understanding needs to deal with highly variant input data. Scenes are usually cluttered with multiple objects undergoing occlusions and interactions. Scenes also vary largely in scale and style, leading to different requirements depending on the specific domain involved in a task. For example, depth estimation for the indoor environment usually deals with distances of up to 10 meters, whereas distances of up to hundreds of meters are common in the outdoor environment, thus requiring the use of different techniques [127]. Second, beyond the data variance, scene understanding tasks themselves involve many analysis along many different aspects of the environment. Scenes exhibit information like 3D geometry, semantics, materials, physics, and potential human interactions, which require fundamentally different models. In practice, understanding multiple aspects might be necessary to function real applications. Assuming that a robot is asked to bring a cup of water from the kitchen, it needs to understand: a) the scene and object categories to find the kitchen and the cup, b) the 3D room and object geometry to navigate through the house and successfully grasp objects, and c) physics and material properties to safely fill the cup with water. Last but not least, even within each specific vision task, there often exist variant formulations which capture different scales of the understanding. These formulations usually require implementation with different data structures and are thus handled with different model designs. For example, 3D geometry can be represented as surface normal maps, depth maps, point clouds, or meshes [96], and semantics can be represented in category labels, bounding boxes, and segmentation masks [177]. Due to this high variance over the characteristics of data, tasks, and formulations, scene understanding is still an open research area, requiring much additional work.

In this thesis, we present a series of works to understand the 3D geometry and the semantics of indoor environments [295, 297, 294, 296, 293]. Compared to outdoor scenes, indoor scenes are small in scale but exhibit more complicated geometry and patterns of clutter, which are usually more challenging to process. Moreover, most human activities occur indoors, so understanding indoor scenes has direct impact on a large fraction of daily life activities. Two of the most fundamental tasks, knowing geometry and semantics are in high demand by many applications, such as robotics, intelligent home applications, AR/VR, and can potentially have high impact for other related academic communities.

We first studied scene understanding using a pixel-wise representation, which directly encodes the predicted information in 2D arrays. Such a representation is in direct pixel-level alignment with the widely used 2D image representation, and thus the vision tasks can be straightforwardly formulated into learning a mapping function from image to image. It can also make use of advanced machine learning techniques running on 2D regular grids, like convolutional neural networks [140]. Some typical vision tasks formulated in pixel-wise representation are semantic segmentation, stereo matching, and depth estimation from a single color image, which will all be addressed in this thesis.

We started with stereo matching, in which the depth is estimated through triangulation between two views obtained from a dual camera system. Traditional stereo matching suffers in texture-less regions where there are no discriminative features that can be found to establish correspondences [220]. One way to handle this is to have an IR projector casting random dot patterns into the scene as additional texture, and such a setting is typically called an active stereo system [193]. To achieve more accurate depth measurements, we proposed a deep learning-based solution which is fully self-supervised, avoiding the need for ground truth data [295]. We fixed the problems in traditional self-supervised training such that the model handles distant and occluded areas better. The self-supervised model produces precise disparity with sharper boundaries.

Since dual camera systems are not practical in many scenarios, single view-based understanding is a more practical problem statement addressing the low cost and widely available internet images datasets. As a result of the limited information from the single view, some problems are ill-posed, like depth from a single RGB image, and therefore require more advanced learning techniques which greatly rely on training data [107, 218]. However, obtaining precise and dense pixel-wise ground truth is very labor intensive and sometimes infeasible to scale up [270]. To address this problem, one way is to use synthetic data, which however may suffer from a domain gap between the rendered and real images. Attempting to fill in this gap, we proposed a large-scale synthetic dataset with 500K physically-based rendered images from 45K realistic 3D indoor scenes [297]. We studied the effects of rendering methods and scene lighting on training for three computer vision tasks: surface normal estimation, semantic segmentation, and object boundary detection. We then provided insights into best practices for training with synthetic data, with the key takeaway point being that more expensive but realistic rendering is worth the extra cost.

Single view scene understanding helps to provide a compromised and lower-quality prediction when more expensive sensor systems are unavailable or impractical, but it also helps to refine results from such more complex systems. As an example, we demonstrated that 3D surface normals learned from a single color image can greatly improve an aligned depth image [294]. This is useful for commodity-grade depth cameras which often fail to sense depth for highly specular, transparent, or distant surfaces. Even though other information can be estimated from the color image as well, like the absolute depth [51, 146], we found in practice that it is more stable to estimate normals and solve depth from them, which outperforms many straightforward end-to-end systems.

Even though the pixel-wise representation is convenient for algorithm design and allows for predicting fine-grained information at the image resolution or even at sub-pixel accuracy, it does not provide complete 3D information since images are 2D projections of the real 3D world from a certain viewing direction. The pixel-wise representation is also frequently redundant and expensive to store in cases where only high-level representations are necessary for the task. For example a 3D plane can be represented with four parameters in a planar equation and capture millions of depth values in a depth map. Conceptually, a complete and compact representation for scene understanding would be desirable if it consists of high-level abstractions and resides in 3D space, preferably integrating multiple kinds of information. Such a representation is readier to use for applications like AR and robotics. Therefore, we next investigated how to produce 3D holistic representation for scene understanding. We picked a bounding box representation where the whole scene was composed of a set of 3D cuboids and planes [101] aligned in three mutually orthogonal directions, i.e. the Manhattan World Assumption [34]. Each 3D cuboid represents the location and rough size of a furniture object with a label representing its semantic category.ach plane represents a wall, ceiling, or floor. Even though cuboids and planes are unable to fully incorporate all possible complicated indoor environments at a fine-grained level, they are compact and enough to support applications that do not require precise interaction with the objects.

We first use this holistic bounding box representation to study the impact of contextual relationships between objects and rooms for scene understanding. Context had been well studied in many previous work (Sec. 4.1.2) and demonstrated to be useful for scene understanding, but its relative importance compared to local evidence was not well studied. Tempting to address this question, we proposed a system that produces bounding box predictions for whole room understanding from a single color full-view panorama [296]. Since a full-view panorama has a full field of view it can observe the greatest number of objects in a single image. Thus, using it as the input enables us to fully exploit the scene context. Our model first produces a set of hypotheses of the scene, and then picks the best according to both primitive local evidence and global scene context. An empirical study shows that the model can achieve comparable or even better performance when heavily relying on context compared to traditional object detection systems that mainly make use of local evidence. Combining the two further improves performance, which indicates that local evidence and scene context capture different aspects of information for scene understanding.

Although the context is important for scene understanding, it can be hard to learn since by nature it involves multiple objects and therefore requires reliable extraction of bottom-up information (e.g. object level) and may exhibit complex high-order relations. So, advanced machine learning techniques are necessary to learn complicated context, and we investigated how to learn context using a deep learning framework. To make the problem easier and focus on context, we choose to use a regular depth image as the input and design a deep learning model to produce the bounding box representation of the scene. Instead of learning a black-box model and hoping that the context is learned automatically, we impart the network architecture with insights such that context can be encoded explicitly [293]. To do so, we proposed to divide scenes into sub-areas, where each is associated with a certain functionality. The furniture layout is usually similar for each functional area, and thus a box-like representation, namely a *scene template*, can be learned by clustering beforehand to encode prior context information for the sub-area. The deep learning model then only needs to learn which sub-area should be used and how to make local adjustments of the scene template. By doing so, the network is fully aware of the prior context and can make more reliable and context-viable results compared to local object detectors.

The remainder of this thesis is organized as follows. Chapter 2 provides a general overview of the related previous work in scene understanding. Chapter 3 and 4 introduce a series of scene understanding projects using pixel-wise representation [295, 297, 294] and holistic 3D representations [296, 293] respectively. Chapter 5 concludes our work and makes discussion on remaining challenges and potential future work.

# Chapter 2

# Background

Scene understanding is one of the central research topics in computer vision. Due to its highly complex nature, numerous efforts have been made to address related vision tasks isolating different aspects into separate problem statements. The following sections briefly discuss some of these aspects in the domain of indoor environments.

## 2.1 Scene Understanding Tasks

There are two fundamental types of tasks for scene understanding: geometric understanding and semantic understanding.

### 2.1.1 Geometry

The purpose of geometric estimation is to capture the 3D shape of objects and environments [96]. In the scope of computer vision, one family of approaches rely on triangulation between multiple images, inspired by the fact that the human visual system relies on two eyes to measure distance. In a typical structure from motion setting, correspondences between two images are established, and then camera poses and 3D sparse points are solved by a global optimization, i.e. bundle adjustment [251].

Depending on the output representation, post process might be required to produce dense a point cloud or mesh model. Sharing the same idea with triangulation, the stereo system assumes calibrated known camera poses and focuses on building dense correspondences across images for high-resolution depth. Similar approaches have also been adapted to work in the infrared domain, with one of the images being a predefined pattern from a projector, i.e. structured light [75], which typically works more reliably and is a technique adopted widely by commercial depth sensors [115, 194].

Alternatively, 3D geometry can be estimated by data-driven approaches, where mapping functions between the input and the 3D geometry, like depth or surface normals, are learned. In the simplest case, geometry can be estimated from a single image even though it is ill-posed in theory. For a long time, the performance of data-driven approaches was far from comparable to the triangulation-based methods as it heavily relied on the capacity of machine learning models, the availability of sufficient training data, and required making assumptions about the input [107, 218]. Thanks to the recently emerging deep learning technologies, data-driven approaches have been significantly improved and can now handle challenging cases better than traditional approaches [51, 146]. Meanwhile, large-scale RGBD datasets have been proposed to serve in training data-hungry algorithms.he problems of missing and error-prone depth estimates is however still an unsolved problem [24, 35, 229].

### 2.1.2 Semantics

On the other hand, semantic understanding focuses on retrieving information about object category [249], scene type [282], and room layout [100], etc. Semantic understanding involves a series of tasks including scene classification, object detection, and semantic segmentation. Compared to algorithms that work on individual objects, scene understanding algorithms are more challenging but also benefit from the fact that scenes consist of multiple objects interacting with each other [159, 17, 15, 6, 302].

These interactions create challenging patterns such as occlusions, deformations, and inter-relations between objects, which imply that the computer vision algorithm must handle partial observation, additional variation, and noise. On the way to solve these problems, external knowledge, such as scene context, physics, and semantic knowledge bases, which are external to the image data can be leveraged. In terms of methodology, these external knowledge sources are often considered as top-down constraints, which are applied on results from bottom-up approaches [265, 91].

Specifically, scene context has drawn a lot of attention from the research community over decades to measure the relationships between objects [159, 249]. It has been formulated into co-occurrence rates [145, 63], pairwise relative distributions [67], tree hierarchies [31], or general graphs [265, 91]. Ideally, context is not limited in these formulations and may appear as higher-order relationships among multiple objects in an arbitrary format, which can be extremely hard to encode by handcrafted methods. Recently, data-driven approaches also provide new avenues for learning context. Neural networks are capable of learning context automatically due to a large receptive field [166]. However, simple expansion of receptive fields is impractical or ineffective in many scenarios, therefore further model design is required.

## 2.2 Representation

Representation is the core of research in scene understanding and highly variant depending on the application. For 3D geometry estimation, the 3D shape can be represented as a point cloud, depth map, surface normal map, 3D volume, or mesh. Triangulation-based methods (e.g., structure from motion and SLAM [247, 48]) tend to generate sparse point clouds as a first stage since correspondences can only be reliably built between a few feature points. Dense correspondences can be further created through optimization to produce dense measurements of geometry, such as depth and surface normal maps from stereo systems [220, 264]. Volumetric representations encode the concept of occupancy but can be extremely memory expensive when the resolution is high, and they also lack surface details [36, 232]. Comparatively, the mesh representation is lightweight and encodes surface detail but requires extra effort to create through handcrafted post-processing or special design of data-driven approaches [260].

Depending on the task, there are also many representations for semantic scene understanding . Well defined in benchmarks, classification uses a label set for categories. Detection uses bounding boxes for object instance indication. Semantic segmentation uses pixel-wise representation for fine-grained details [40, 158, 52]. Recently, the emerging 3D scene understanding work further extends these representations into 3D space, such as the 3D bounding box for detection [293, 206] and 3D per-voxel/point labels for segmentation [36, 232]. Beyond vision tasks transferred from the object level, representations can be extended to support more complete scene-specific problems. For example, room layout, i.e., walls, floor, and ceiling, can be represented as planar surfaces and placed with object 3D bounding boxes in the same world coordinate [206, 293].

## 2.3 Data

Early researches about scene understanding mainly focused on model design for small datasets [62, 61, 139], which overlooks the complexity of real-world data. Although small-scale semantic ground truth can be annotated in a well-controlled environment by a few researchers, obtaining accurate 3D geometry is extremely hard even with a lot of efforts and expensive scanning devices. To bypass this problem, synthetic data has been used, where the geometric ground truth can be obtained easily and accurately from virtual environments [220] . However, the input side often suffers from the domain shift as the rendered images are inevitably visually different from real photos such that what has been learned in the synthetic domain may not easily transfer to the real domain.

Recently, data started to draw more attention due to the emergence of powerful data-driven methods equipped with deep learning [40, 140]. Large-scale real geometry can be collected using commodity-level RGBD sensors but is still far from perfect and requires some post-processing to remove noise and failure cases [35, 24]. At the same time, the scale of synthetic data is also significantly larger than before with the help of the virtual indoor 3D shapes and scenes available on the internet [232, 266, 25]. For obtaining semantics, crowd-sourcing platforms enable large-scale human annotation but require effort to verify the annotation quality of inexperienced workers online [1]. Further increasing the scale of datasets is expensive and often involves impractical amounts of human labor, thus requiring algorithms that are directly involved in the labeling process, such as active learning [282].

# Chapter 3

# Pixel-wise Scene Understanding

In this chapter, we introduce a series of works using pixel-wise representation for scene understanding. In Chapter 3.1, we present a self-supervise deep learning system to estimate depth for active stereo system [295]. In Chapter 3.2, we target on more challenging single view dense pixel-wise scene understanding and leverage synthetic to improve their performance [297]. In Chapter 3.3, we focus on depth completion as an example to show that single view scene understanding can practically improve the quality of the depth images from RGBD sensors [294].

## 3.1 ActiveStereoNet: End-to-End Self-Supervised Learning for Active Stereo Systems

### 3.1.1 Introduction

Estimating the 3D geometry is one of the most fundamental tasks in scene understanding [177, 96]. After years of efforts from academia and industry, depth sensors have become widely available and are revolutionizing computer vision by providing additional 3D information for many hard problems, such as non-rigid reconstruction [47, 46], action recognition [55, 60] and parametric tracking [242, 243] . Although there are many types of depth sensor technologies, they all have significant limitations. Time of flight systems suffer from motion artifacts and multi-path interference [14, 13, 187]. Structured light is vulnerable to ambient illumination and multi-device interference [58, 57]. Passive stereo struggles in texture-less regions, where expensive global optimization techniques are required - especially in traditional non-learning based methods.

Active stereo offers a potential solution: an infrared stereo camera pair is used, with a pseudorandom pattern projectively texturing the scene via a patterned IR light source. (Fig. 3.1). With a proper selection of sensing wavelength, the camera pair captures a combination of active illumination and passive light, improving quality above that of structured light while providing a robust solution in both indoor and outdoor scenarios. Although this technology was introduced decades ago [193], it has only recently become available in commercial products (e.g., Intel R200 and D400 family [116]). As a result, there is relatively little prior work targeted specifically at inferring depths from active stereo images, and large scale training data with ground truth is not available yet.

Several challenges must be addressed in an active stereo system. Some are common to all stereo problems – for example, it must avoid matching occluded pixels, which causes oversmoothing, edge fattening, and/or flying pixels near contour edges. However, other problems are specific to active stereo – for example, it must process very high-resolution images to match the high-frequency patterns produced by the projector; it must avoid the many local minima arising from alternative alignments of these high frequency patterns; and it must compensate for luminance differences between projected patterns on nearby and distant surfaces. Additionally, of course, it cannot be trained with supervision from a large active stereo dataset with ground truth depths, since none is available.

This chapter proposes the first end-to-end deep learning approach for active stereo that is trained fully self-supervised [295]. It extends recent work on self-supervised passive stereo [304] to address problems encountered in active stereo. First, we propose a new reconstruction loss based on local contrast normalization (LCN) that removes low frequency components from passive IR and re-calibrates the strength of the active pattern locally to account for fading of active stereo patterns with distance. Second, we propose a window-based loss aggregation with adaptive weights for each pixel to increase its discriminability and reduce the effect of local minima in the stereo cost function. Finally, we detect occluded pixels in the images and omit them from loss computations. These new aspects of the algorithm provide significant benefits to the convergence during training and improve depth accuracy at test time. Extensive experiments demonstrate that our network trained with these insights outperforms previous work on active stereo and alternatives in ablation studies across a wide range of experiments.

### 3.1.2 Related Work

Depth sensing is a classic problem with a long history of prior work. Among the **active sensors**, Time of Flight (TOF), such as Kinect V2, emits a modulated light source and uses multiple observations of the same scene (usually 3-9) to predict a

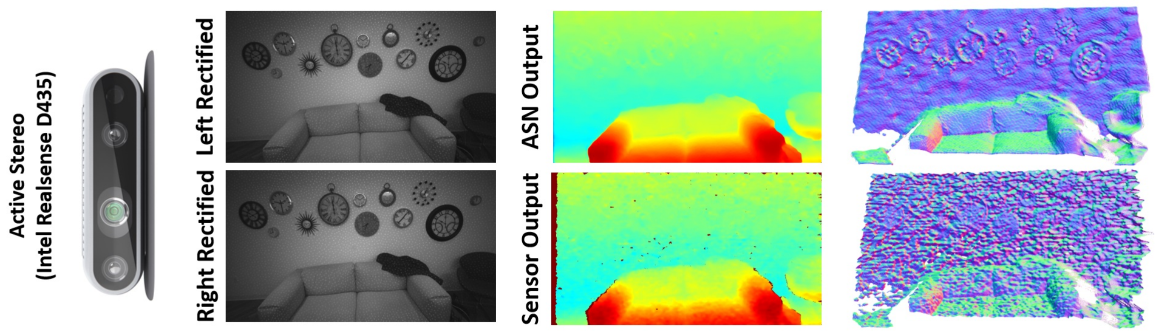


Figure 3.1: ActiveStereoNet (ASN) produces smooth, detailed, quantization free results using a pair of rectified IR images acquired with an Intel Realsense D435 camera. In particular, notice how the jacket is almost indiscernible using the sensor output, and in contrast, how it is clearly observable in our results.

single depth map. The main issues with this technology are artifacts due to motion and multipath interference [14, 13, 187]. Structure light (SL) is a viable alternative, but it requires a known projected pattern and is vulnerable to multi-device inference [58, 57]. Neither approach is robust in outdoor conditions under strong illumination.

**Passive stereo** provides an alternative approach [220, 90]. Traditional methods utilize hand-crafted schemes to find reliable local correspondences [19, 279, 111, 18, 105] and global optimization algorithms to exploit context when matching [12, 64, 134, 136]. Recent methods address these problems with deep learning. Siamese networks are trained to extract patch-wise features and/or predict matching costs [171, 288, 286, 287]. More recently, end-to-end networks learn these steps jointly, yielding better results [225, 180, 128, 114, 195, 156, 76]. However all these deep learning methods rely on a strong supervised component. As a consequence, they outperform traditional handcrafted optimization schemes only when a lot of ground-truth depth data is available, which is not the case in active stereo settings.

**Self-supervised passive stereo** is a possible solution for absence of ground-truth training data. When multiple images of the same scene are available, the images can warp between cameras using the estimated/calibrated pose and the depth, and the loss between the reconstruction and the raw image can be used to train depth estimation systems without ground truth. Taking advantage of spatial and temporal coherence, depth estimation algorithms can be trained unsupervised using monocular images [79, 73, 144], video [275, 305], and stereo [304]. However, their results are blurry and far from comparable with supervised methods due to the required strong regularization such as left-right check [79, 304]. Also, they struggle in textureless and dark regions, as do all passive methods.

**Active stereo** is an extension of the traditional passive stereo approach in which a texture is projected into the scene with an IR projector and cameras are augmented to perceive IR as well as visible spectra [137]. Intel R200 was the first attempt of commercialize an active stereo sensor, however its accuracy is poor compared to (older) structured light sensors, such as Kinect V1 [57, 58]. Very recently, Intel released the D400 family [115, 116], which provides higher resolution, $1280 \times 720$, and therefore has the potential to deliver more accurate depth maps. The build-in stereo algorithm in these cameras uses a handcrafted binary descriptor (CENSUS) in combination with a semi-global matching scheme [129]. It offers reasonable performance in a variety of settings [241], but still suffers from common stereo matching issues addressed in this paper (edge fattening, quadratic error, occlusions, holes, etc.).

**Learning-based solutions for active stereo** are limited. Past work has employed shallow architectures to learn a feature space where the matching can be performed efficiently [58, 59, 261], trained a regressor to infer disparity [57], or learned a direct mapping from pixel intensity to depth [56]. These methods fail in general scenes [56], suffer from interference and per-camera calibration [57], and/or do not work well in texture-less areas due to their shallow descriptors and local optimization schemes [58, 59]. Our paper is the first to investigate how to design an end-to-end deep network for active stereo.

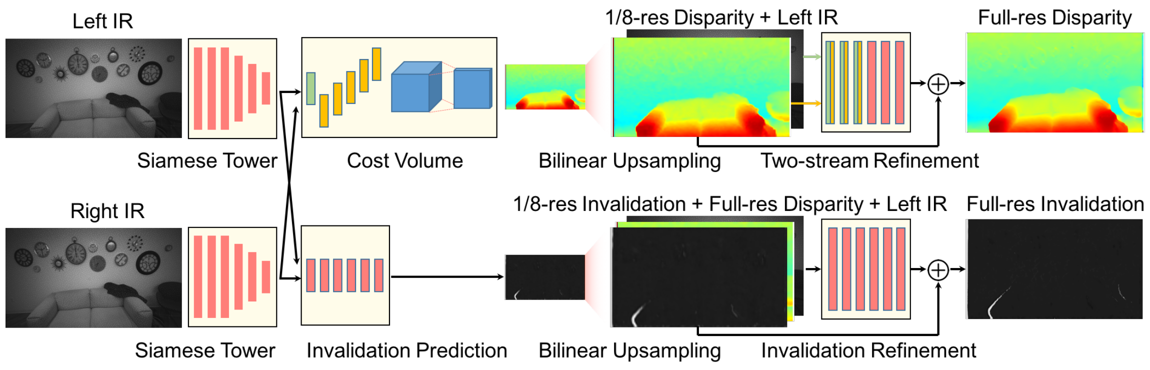


Figure 3.2: ActiveStereoNet architecture. We use a two stage network where a low resolution cost volume is built and infers the first disparity estimate. A bilinear upsampling followed by a residual network predicts the final disparity map. An "Invalidation Network" (bottom) is also trained end-to-end to predict a confidence map.

### 3.1.3 Method

In this section, we introduce the network architecture and training procedure for ActiveStereoNet. The input to our algorithm is a rectified, synchronized pair of images with active illumination (see Fig. 3.1), and the output is a pair of disparity maps at the original resolution. For our experiments, we use the recently released Intel Realsense D435 that provides synchronized, rectified $1280 \times 720$ images at 30fps. The focal length $f$ and the baseline $b$ between the two cameras are assumed to be known. Under this assumption, the depth estimation problem becomes a disparity search along the scan line. Given the output disparity $d$, the depth is obtained via $Z = \frac{bf}{d}$.

Since no ground-truth training data is available for this problem, our main challenge is to train an end-to-end network that is robust to occlusion and illumination effects without direct supervision. The following details our algorithm.

#### 3.1.3.1 Network Architecture

Nowadays, in many vision problems, the choice of the architecture plays a crucial role, and most of the efforts are spent in designing the right network. In active stereo, instead, we found that the most challenging part is the training procedure for a given deep network. In particular, since our setting is unsupervised, designing the optimal loss function has the highest impact on the overall accuracy. For this reason, we extend the network architecture proposed in [130], which has shown superior performances in many passive stereo benchmarks. Moreover, the system is computationally efficient and allows us to run on full resolution at $60Hz$ on a high-end GPU, which is desirable for real-time applications.

The overall pipeline is shown in Fig. 3.2. We start from the high-resolution images and use a siamese tower to produces feature map in 1/8 of the input resolution. We then build a low resolution cost volume of size $160 \times 90 \times 18$, allowing for a maximum disparity of 144 in the original image, which corresponds to a minimum distance of $\sim 30$ cm on the chosen sensor.

The cost volume produces a downsampled disparity map using the soft argmin operator [128]. Differently from [128] and following [130] we avoid expensive 3D deconvolution and output a $160 \times 90$ disparity. This estimation is then upsampled using bi-linear interpolation to the original resolution ($1280 \times 720$). A final residual refinement retrieves the high-frequency details such as edges. Different from [130], our refinement block starts with separate convolution layers running on the upsampled disparity and input image respectively, and merge the feature later to produce residual. This in practice works better to remove dot artifacts in the refined results.

Our network also simultaneously estimates an invalidation mask to remove uncertain areas in the result, which will be introduced in Sec. 3.1.3.4.

#### 3.1.3.2 Loss Function

The architecture described is composed of a low resolution disparity and a final refinement step to retrieve high-frequency details. A natural choice is to have a loss function for each of these two steps. Unlike [130], we are in an unsupervised setting

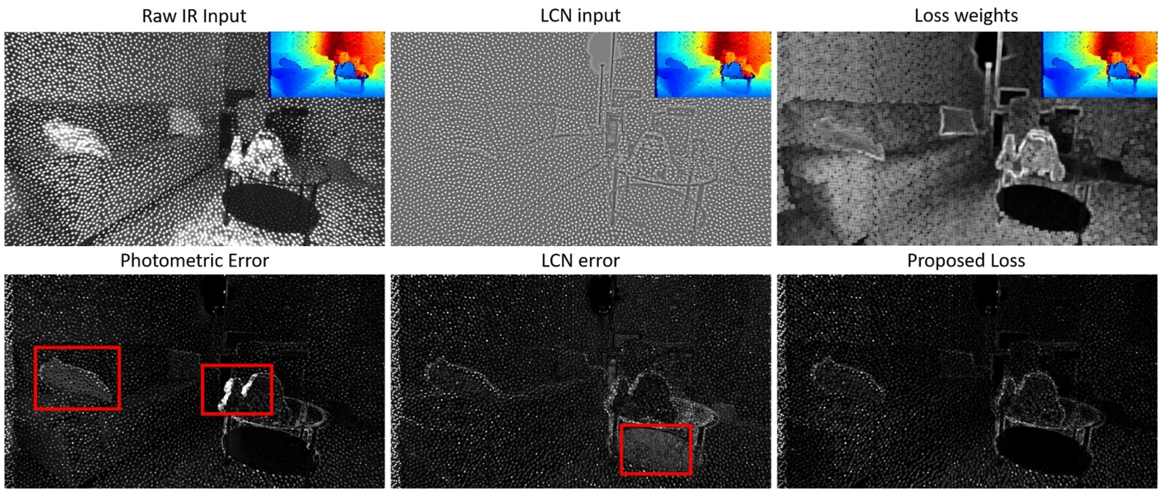


Figure 3.3: Comparisons between photometric loss (left), LCN loss (middle), and the proposed weighted LCN loss (right). Our loss is more robust to occlusions, it does not depend on the brightness of the pixels and does not suffer in low texture regions.

due to the lack of ground truth data. A viable choice for the training loss $L$ then is the photometric error between the original pixels on the left image $I^l_{ij}$ and the reconstructed left image $\hat{I}^l_{ij}$, in particular $L = \sum_{ij} \|I^l_{ij} - \hat{I}^l_{ij}\|_1$. The reconstructed image $\hat{I}^l$ is obtained by sampling pixels from the right image $I^r$ using the predicted disparity $d$, i.e. $\hat{I}^l_{ij} = I^r_{i,j-d}$. Our sampler uses the Spatial Transformer Network (STN) [117], which uses a bi-linear interpolation of 2 pixels on the same row and is fully differentiable.

However, as shown in previous work [299], the photometric loss is a poor choice for image reconstruction problems. This is even more dramatic when dealing with active setups. We recall that active sensors flood the scenes with texture and the intensity of the received signal follows the inverse square law $I \propto \frac{1}{Z^2}$, where $Z$ is the distance from the camera. In practice this creates an explicit dependency between the intensity and the distance (i.e. brighter pixels are closer). A second issue, that is also present in RGB images, is that the difference between two bright pixels is likely to have a bigger residual when compared to the difference between two dark pixels. Indeed if we consider image $I$, to have noise proportional to intensity [69], the observed intensity for a given pixel can be written as: $I_{ij} = I^{\star}_{ij} + \mathcal{N}(0, \sigma_1 I^{\star}_{ij} + \sigma_2)$, where $I^{\star}_{ij}$ is the noise free signal and the standard deviations $\sigma_1$ and $\sigma_2$ depend on the sensor [69]. It is easy to show that the difference between two correctly matched pixels $I$ and $\hat{I}$ has a residual: $\epsilon = \mathcal{N}(0, \sqrt{(\sigma_1 I^{\star}_{ij} + \sigma_2)^2 + (\sigma_3 \hat{I}^{\star}_{ij} + \sigma_4)^2})$, where its variance depends on the input intensities. This shows that for brighter pixels (i.e. close objects) the residual $\epsilon$ will be bigger compared to one of low reflectivity or farther objects.

In the case of passive stereo, this could be a negligible effect, since in RGB images there is no correlation between intensity and disparity, however in the active case the aforementioned problem will bias the network towards closeup scenes, which will have always a bigger residual. The architecture will learn mostly those easy areas and smooth out the rest. The darker pixels, mostly in distant, requiring higher matching precision for accurate depth, however, are overlooked. In Fig. 3.3 (left), we show the the reconstruction error for a given disparity map using the photometric loss. Notice how bright pixels on the pillow exhibits high reconstruction error due to the input dependent nature of the noise.

An additional issue with this loss occurs in the occluded areas: indeed when the intensity difference between background and foreground is severe, this loss will have a strong contribution in the occluded regions, forcing the network to learn to fit those areas that, however, cannot really be explained in the data.

**Weighted Local Contrast Normalization.** We propose to use a Local Contrast Normalization (LCN) scheme, that not only removes the dependency between intensity and disparity, but also gives a better residual in occluded regions. It is also invariant to brightness changes in the left and right input image. In particular, for each pixel, we compute the local mean $\mu$ and standard deviation $\sigma$ in a small $9 \times 9$ patch. These local statistics are used to normalize the current pixel intensity $I_{LCN} = \frac{I-\mu}{\sigma+\eta}$, where $\eta$ is a small constant. The result of this normalization is shown

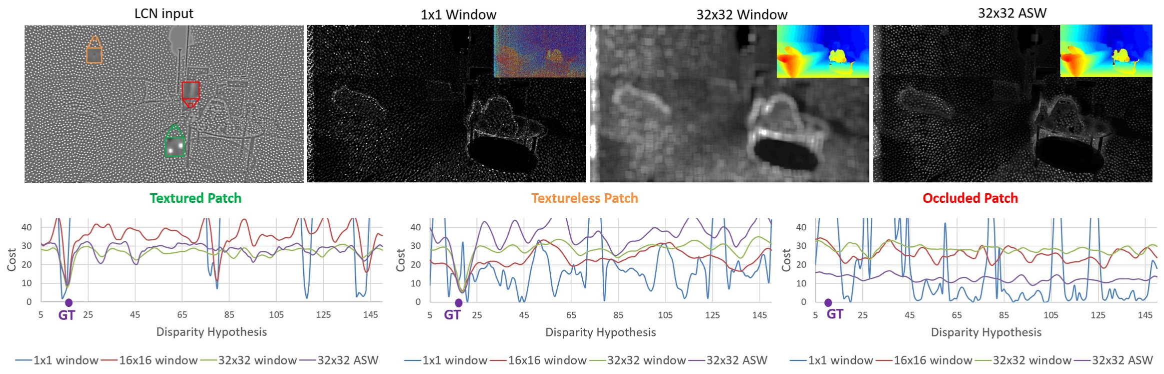


Figure 3.4: Cost volume analysis for a textured region (green), textureless patch (orange) and occluded pixel (red). Notice how the window size helps to resolve ambiguous (textureless) areas in the image, whereas in occluded pixels the lowest cost will always lead to the wrong solution. However large windows oversmooth the cost function and they do not preserve edges, where as the proposed Adaptive Support Weight loss aggregates costs preserving edges.

in Fig. 3.3, middle. Notice how the dependency between disparity and brightness is now removed, moreover the reconstruction error (Fig. 3.3, middle, second row) is not strongly biased towards high intensity areas or occluded regions.

However, LCN suffers in low texture regions when the standard deviation $\sigma$ is close to zero (see the bottom of the table in Fig. 3.3, middle). Indeed these areas have a small $\sigma$ which will would amplify any residual together with noise between two matched pixels. To remove this effect, we re-weight the residual $\epsilon$ between two matched pixel $I_{ij}$ and $\hat{I}^l_{ij}$ using the local standard deviation $\sigma_{ij}$ estimated on the reference image in a $9 \times 9$ patch around the pixel $(i, j)$. In particular our reconstruction loss becomes: $L = \sum_{ij} \|\sigma_{ij}(I^l_{LCNij} - \hat{I}^l_{LCNij})\|_1 = \sum_{ij} C_{ij}$. Example of weights computed on the reference image are shown in Fig. 3.3, top right and the final loss is shown on the bottom right. Notice how these residuals are not biased in bright areas or low textured regions.

### 3.1.3.3 Window-based Optimization

We now analyze in more details the behavior of the loss function for the whole search space. We consider a textured patch (green), a texture-less one (orange) and an occluded area (red) in an LCN image (see Fig. 3.4). We plot the loss function for every disparity candidate in the range of $[5, 144]$. For a single pixel cost (blue curve), notice how the function exhibits a highly non-convex behavior (w.r.t. the disparity) that makes extremely hard to retrieve the ground truth value (shown as purple dots). Indeed a single pixel cost has many local minima, that could lie far from the actual optimum. In traditional stereo matching pipelines, a cost aggregation robustifies the final estimate using evidence from neighboring pixels. If we consider a window around each pixel and sum all the costs, we can see that the loss becomes smoother for both textured and texture-less patch and the optimum can be reached (see Fig. 3.4, bottom graphs). However as a drawback for large windows, small objects and details can be smooth out by the aggregation of multiple costs and cannot be recovered in the final disparity.

Traditional stereo matching pipelines aggregate the costs using an adaptive support (ASW) scheme [280], which is very effective, but also slow hence not practical for real-time systems where approximated solutions are required [138]. Here we propose to integrate the ASW scheme in the training procedure, therefore it does not affect the runtime cost. In particular, we consider a pixel $(i, j)$ with intensity $I_{ij}$ and instead of compute a per-pixel loss, we aggregate the costs $C_{ij}$ around a $2k \times 2k$ window following: $\hat{C_{ij}} = \frac{\sum_{x=i-k}^{i+k-1} \sum_{y=j-k}^{j+k-1} w_{x,y} C_{ij}}{\sum_{x=i-k}^{i+k-1} \sum_{y=j-k}^{j+k-1} w_{x,y}}$, where $w_{xy} = \exp(-\frac{|I_{ij} - I_{xy}|}{\sigma_w})$, with $\sigma_w = 2$. As shown in Fig. 3.4 right, this aggregates the costs (i.e. it smooths the cost function), but it still preserves the edges. In our implementation we use a $32 \times 32$ during the whole training phase. We also tested a graduated optimization approach [189, 98], where we first optimized our network using $64 \times 64$ window and then reduce it every 15000 iterations by a factor of 2, until we reach a single pixel loss. However this solution led to very similar results compared to a single pixel loss during the whole training.

#### 3.1.3.4 Invalidation Network

So far the proposed loss does not deal with occluded regions and wrong matches (i.e. textureless areas). An occluded pixel does not have any useful information in the cost volume even when brute-force search is performed at different scales (see in Fig. 3.4, bottom right graph). To deal with occlusions, traditional stereo matching methods use a so called left-right consistency check, where a disparity is first computed from the left view point ($d_l$), then from the right camera ($d_r$) and invalidate those pixels with $|d_l - d_r| > \theta$. Related work use a left-right consistency in the loss minimization [79], however this leads to oversmooth edges which become flying pixels (outliers) in the pointcloud. Instead, we propose to use the left-check as a hard constraint by defining a mask for a pixel $(i, j)$: $m_{ij} = |d_l - d_r| < \theta$, with $\theta = 1$ disparity. Those pixels with $m_{ij} = 0$ are ignored in the loss computation. To avoid a trivial solution (i.e. all the pixels are invalidated), similarly to [305], we enforce a regularization on the number of valid pixels by minimizing the cross-entropy loss with constant label 1 in each pixel location. We use this mask in both the low-resolution disparity as well as the final refined one.

At the same time, we train an invalidation network (fully convolutional), that takes as input the features computed from the Siamese tower and produces first a low resolution invalidation mask, which is then upsampled and refined with a similar architecture used for the disparity refinement. This allows, at runtime, to avoid predicting the disparity from both the left and the right viewpoint to perform the left-right consistency, making the inference significantly faster.

### 3.1.4 Experiments

We performed a series of experiments to evaluate ActiveStereoNet (ASN). In addition to analyzing the accuracy of depth predictions in comparison to previous work, we also provide results of ablation studies to investigate how each component of the proposed loss affects the results. In the supplementary material we also evaluate the applicability of our proposed self-supervised loss in passive (RGB) stereo, showing improved generalization capabilities and compelling results on many benchmarks.

#### 3.1.4.1 Dataset and Training Schema

We train and evaluate our method on both real and synthetic data.

For the *real dataset*, we used an Intel Realsense D435 camera [116] to collect 10000 images for training in an office environment, plus 100 images in other *unseen* scenes for testing (depicting people, furnished rooms and objects).

For the *synthetic dataset*, we used Blender to render IR and depth images of indoor scenes such as living rooms, kitchens, and bedrooms, as in [58]. Specifically, we render synthetic stereo pairs with 9 cm baseline using projective textures to simulate projection of the Kinect V1 dot pattern onto the scene. We randomly move the camera in the rendered rooms and capture left IR image, right IR image as well as ground truth depth. Examples of the rendered scenes are showed in Fig. 3.8, left. The synthetic training data consists of 10000 images and the test set is composed of 1200 frames comprehending new scenes.

For both real and synthetic experiments, we trained the network using RMSprop [248]. We set the learning rate to 1e$-$4 and reduce it by half at $\frac{3}{5}$ iterations and to a quarter at $\frac{4}{5}$ iterations. We stop the training after 100000 iterations, that are usually enough to reach the convergence. Although our algorithm is self-supervised, we *did not* fine-tune the model on any of the test data since it reduces the generalization capability in real applications.

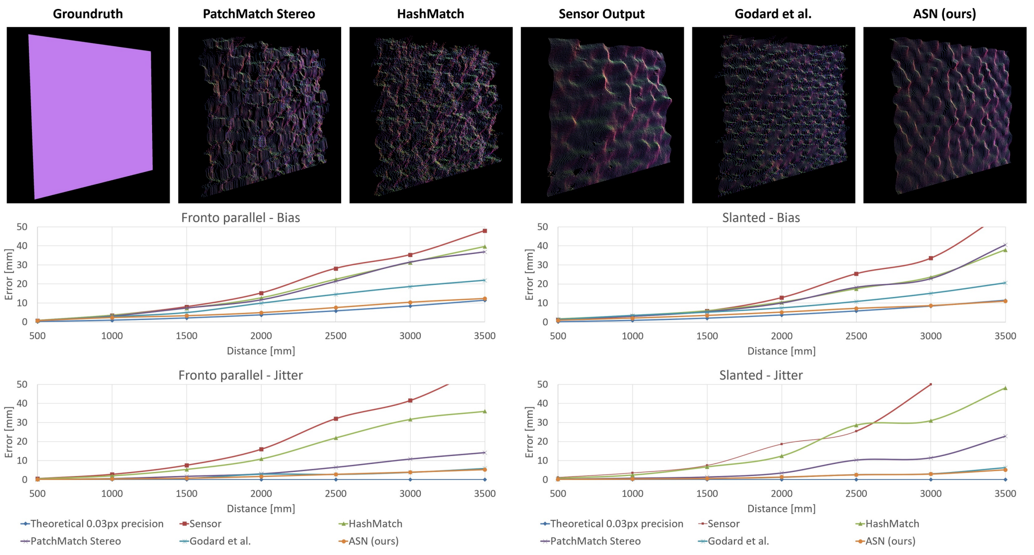


Figure 3.5: Quantitative Evaluation with state of the art. We achieve one order of magnitude less bias with a subpixel precision of 0.03 pixels with a very low jitter (see text). We also show the predicted pointclouds for various methods of a wall at 3000mm distance. Notice that despite the large distance (3m), our results is the less noisy compared to the considered approaches.

#### 3.1.4.2 Stereo Matching Evaluation

In this section, we compare our method on real data with state of the art stereo algorithms qualitatively and quantitatively using traditional stereo matching metrics, such as jitter and bias.

**Bias and Jitter.** It is known that a stereo system with baseline $b$, focal length $f$, and a subpixel disparity precision of $\delta$, has a depth error $\epsilon$ that increases quadratically with respect to the depth $Z$ according to $\epsilon = \frac{\delta Z^2}{bf}$ [240]. Due to the variable impact of disparity error on the depth, naive evaluation metrics, like mean error of disparity, does not effectively reflect the quality of the estimated depth. In contrast, we first show error of depth estimation and calculate corresponding error in disparity.

To assess the subpixel precision of ASN, we recorded 100 frames with the camera in front of a flat wall at distances ranging from 500 mm to 3500 mm, and also 100 frames with the camera facing the wall at an angle of 50 deg to assess the behavior on

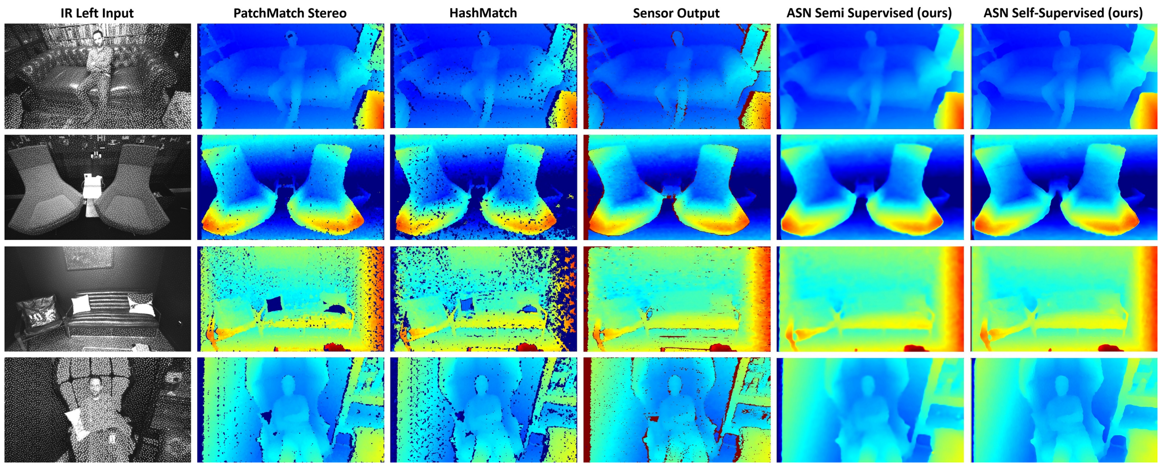


Figure 3.6: Qualitative Evaluation with state of the art. Our method produces detailed disparity maps. State of the art local methods [19, 59] suffer from textureless regions. The semi-global scheme used by the sensor [129] is noisier and it oversmooths the output.

slanted surfaces. In this case, we evaluate by comparing to "ground truth" obtained with robust plane fitting.

To characterize the precision, we compute *bias* as the average $\ell_1$ error between the predicted depth and the ground truth plane. Fig. 3.5 shows the bias with regard to the depth for our method, sensor output [129], the state of the art local stereo methods (PatchMatch [19], HashMatch [59]), and our model trained using the state of the art unsupervised loss [79], together with visualizations of point clouds colored by surface normal. Our system performs significantly better than the other methods at all distances, and its error does not increase dramatically with depth. The corresponding subpixel disparity precision of our system is $1/30th$ of a pixel, which is obtained by fitting a curve using the above mentioned equation (also shown in Fig. 3.5). This is one order of magnitude lower than the other methods where the precision is not higher than 0.2 pixel.

To characterize the noise, we compute the *jitter* as the standard deviation of the depth error. Fig. 3.5 shows that our method achieves the lowest jitter at almost every depth in comparison to other methods.

**Comparisons with State of the Art.** More qualitative evaluations of ASN in challenging scenes are shown in Fig. 3.6. As can be seen, local methods like PatchMatch stereo [19] and HashMatch [59] do not handle mixed illumination with both active and passive light, and thus produce incomplete disparity images (missing pixels shown in black). The sensor output using a semi-global scheme is more suitable for this data [129], but it is still susceptible to image noise (note the noisy results in the fourth column). In contrast, our method produces complete disparity maps and preserves sharp boundaries.

More examples on real sequences are shown in Fig. 3.8 (right), where we show point clouds colored by surface normal. Our output preserves all the details and exhibits a low level of noise. In comparison, our network trained with the self-supervised method by Godard et al. [79] over-smooths the output, hallucinating geometry and flying pixels. Our results are also free from the texture copying problem, most likely because we use a cost volume to explicitly model the matching function rather than learn directly from pixel intensity. Even though the training data is mostly captured from office environment, we find ASN generalize well to various testing scenes, e.g. living room, play room, dinning room, and objects, e.g. person, sofas, plants, table, as shown in figures.

#### 3.1.4.3 Ablation Study

In this section, we evaluate the importance of each component in the ASN system. Due to the lack of ground truth data, most of the results are qualitative – when looking at the disparity maps, please pay particular attention to noise, bias, edge fattening, flying pixels, resolution, holes, and generalization capabilities.

**Self-supervised vs Supervised.** Here we perform more evaluations of our self-supervised model on synthetic data when supervision is available as well as on real

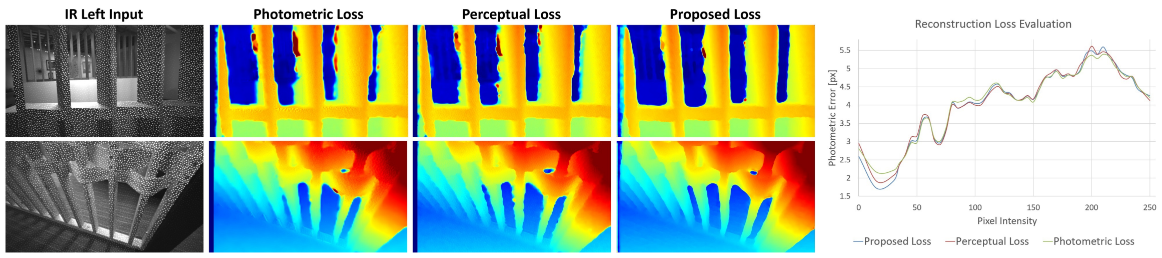


Figure 3.7: Ablation study on reconstruction loss. Same networks, trained on 3 different reconstruction losses. Notice how the proposed WLCN loss infers disparities that better follow the edges in these challenging scenes. Photometric and Perceptual losses have also a higher level of noise. On the right, we show how our loss achieves the lowest reconstruction error for low intensity pixels thanks to the proposed WLCN.

data using the depth from the sensor as supervision (together with the proposed loss). Quantitative evaluation on synthetic data (Fig. 3.8, left bottom), shows that the supervised model (blue) achieves a higher percentage of pixels with error less than 5 disparity, however for more strict requirements (error less than 2 pixels) our self-supervised loss (red) does a better job. This may indicate overfitting of the supervised model on the training set. This behavior is even more evident on real data: the model was able to fit the training set with high precision, however on test images it produces blur results compared to the self-supervised model (see Fig. 3.6, ASN Semi Supervised vs ASN Self-Supervised).

**Reconstruction Loss.** We next investigate the impact of our proposed WLCN loss (as described in Sec. 3.1.3.2) in comparison to a standard photometric error (L1) and a perceptual loss [122] computed using feature maps from a pre-trained VGG network. In this experiment, we trained three networks with the same parameters, changing only the reconstruction loss: photometric on raw IR, VGG conv-1, and the proposed WLCN, and investigate their impacts on the results.

To compute accurate metrics, we labeled the occluded regions in a subset of our test case manually (see Fig. 3.9). For those pixels that were not occluded, we computed the photometric error of the raw IR images given the predicted disparity image.

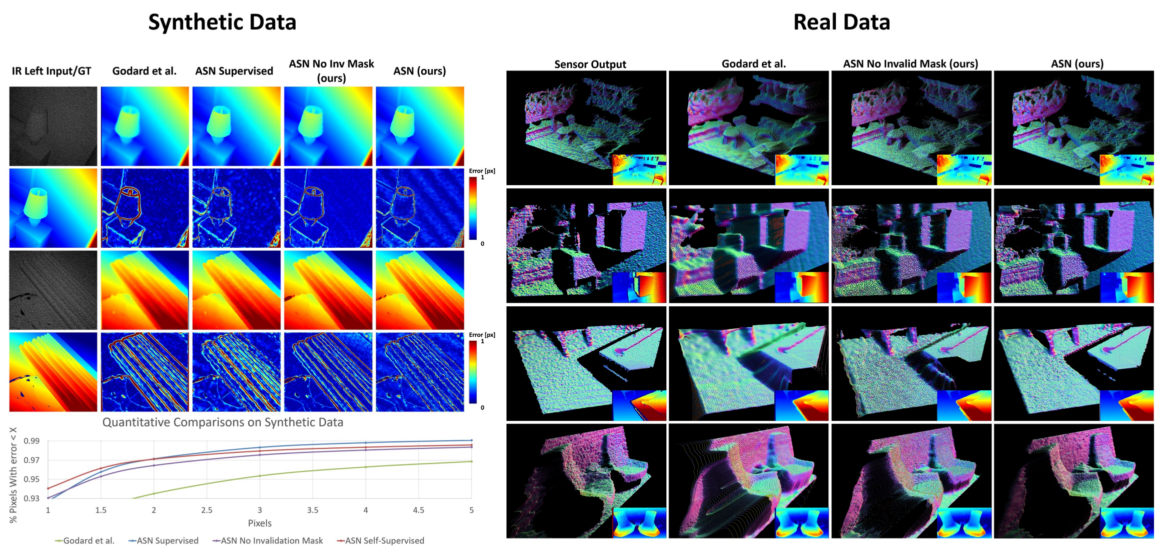


Figure 3.8: Evaluation on Synthetic and Real Data. On synthetic data (left), notice how our method has the highest percentage of pixels with error smaller than 1 disparity. We also produce sharper edges and less noisy output compared to other baselines. The state of the art self-supervised method by Godard et al. [79] is very inaccurate near discontinuities. On the right, we show real sequences from an Intel RealSense D435 where the gap between [79] and our method is even more evident: notice flying pixels and oversmooth depthmaps produced by Godard et al. [79]. Our results has higher precision than the sensor output.

In total we evaluated over 10M pixels. In Fig. 3.7 (right), we show the photometric error of the raw IR images for the three losses with respect to the pixel intensities. The proposed WLCN achieves the lowest error for small intensities, showing that the loss is not biased towards bright areas. For the rest of the range the losses get similar numbers. Please notice that our loss achieves the lowest error even we did not explicitly train to minimize the photometric reconstruction. Although the numbers may seem similar, the effect on the final disparity map is actually very evident. We show some examples of predicted disparities for each of the three different losses in Fig. 3.7 (left). Notice how the proposed WLCN loss suffers from less noise, produces crisper edges, and has a lower percentage of outliers. In contrast, the perceptual loss highlights the high frequency texture in the disparity maps (i.e. dots), leading to noisy estimates. Since VGG conv-1 is pre-trained, we observed that the responses are high on bright dots, biasing the reconstruction error again towards close up scenes. We also tried a variant of the perceptual loss by using the output from our Siamese tower as the perceptual feature, however the behavior was similar to the case of using the VGG features.

**Invalidation Network.** We next investigate whether excluding occluded region from the reconstruction loss is important to train a network – i.e., to achieve crisper edges and less noisy disparity maps. We hypothesize that the architecture would try to overfit occluded regions without this feature (where there are no matches), leading to higher errors throughout the images. We test this quantitatively on synthetic images by computing the percentage of pixels with disparity error less than $x \in [1, 5]$. The results are reported in Fig. 3.8. With the invalidation mask employed, our model outperforms the case without for all the error threshold (Red v.s Purple curve, higher is better). We further analyze the produced disparity and depth maps on both synthetic and real data. On synthetic data, the model without invalidation mask shows gross error near the occlusion boundary (Fig. 3.8, left top). Same situation happens on real data (Fig. 3.8, right), where more flying pixels exhibiting when no invalidation mask is enabled.

As a byproduct of the invalidation network, we obtain a confidence map for the depth estimates. In Fig. 3.9 we show our predicted masks compared with the ones predicted with a left-right check and the photometric error. To assess the performances, we used again the images we manually labeled with occluded regions and computed the average precision (AP). Our invalidation network and left right check achieved the highest scores with an AP of 80.7% and 80.9% respectively, whereas the photometric error only reached 51.3%. We believe that these confidence maps could be useful for many higher-level applications.

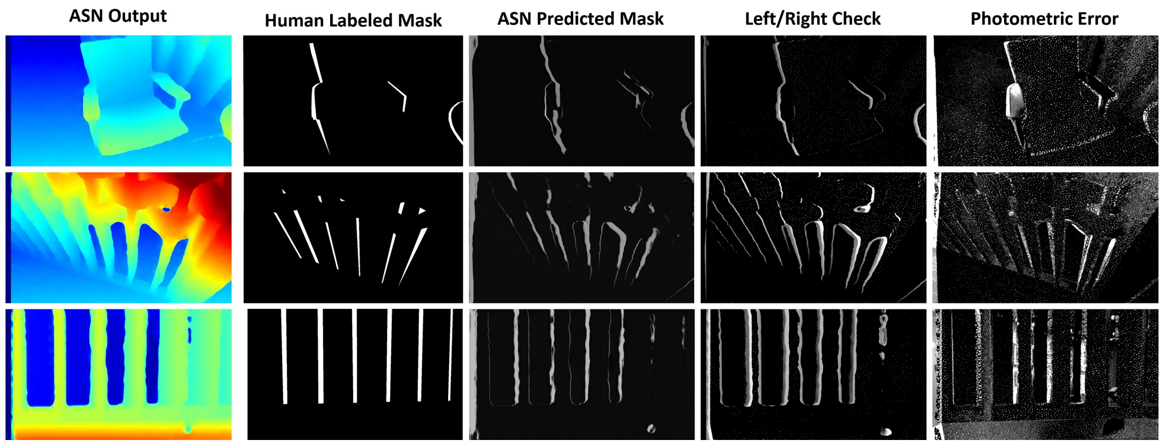


Figure 3.9: Invalidation Mask prediction. Our invalidation mask is able to detect occluded regions and it reaches an average precision of 80.7% (see text).

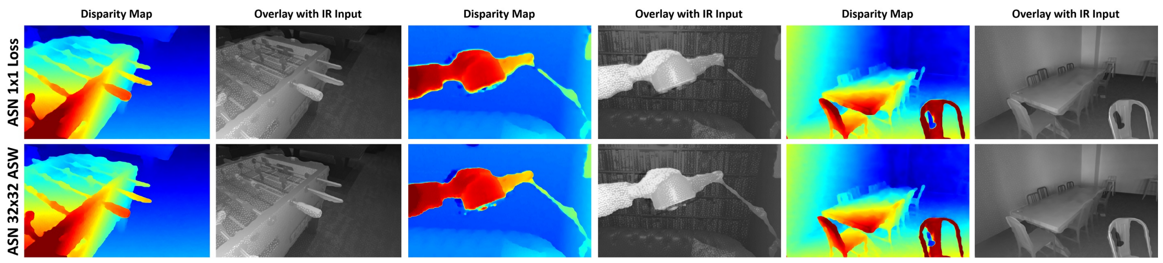


Figure 3.10: Comparison between single pixel loss and the proposed window based optimization with adaptive support scheme. Notice how the ASW is able to recover more thin structures and produce less edge fattening.

**Window based Optimization.** The proposed window based optimization with Adaptive Support Weights (ASW) is very important to get more support for thin structures that otherwise would get a lower contribution in the loss and treated as outliers. We show a comparison of this in Fig. 3.10. Notice how the loss with ASW is able to recover hard thin structures with higher precision. Moreover, our window based optimization also produces smoother results while preserving edges and details. Finally, despite we use a window-based loss, the proposed ASW strategy has a reduced amount of edge fattening.

### 3.1.5 Conclusion

We presented ActiveStereoNet (ASN) [295] the first deep learning method for active stereo systems, which produce pixel-wise dense depth from a pair of active IR images. We designed a novel loss function to cope with high-frequency patterns, illumination effects, and occluded pixels to address issues of active stereo in a self-supervised setting. We showed that our method delivers very precise reconstructions with a subpixel precision of 0.03 pixels, which is one order of magnitude better than other active stereo matching methods. Compared to other approaches, ASN does not oversmooth details, and it generates complete depthmaps, crisp edges, and no flying pixels. As a byproduct, the invalidation network is able to infer a confidence map of the disparity that can be used for high level applications requiring occlusions handling. Numerous experiments show state of the art results on different challenging scenes with a runtime cost of 15ms per frame using an NVidia Titan X.

## 3.2 Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks

### 3.2.1 Introduction

Multiple images usually enables more reliable scene understanding such as stereo matching from two images for depth estimation, however multiple calibrated cameras/images require additional hardware and algorithm support, which may not be available all the time. On the other hand, single image based scene understanding is extremely useful as it can be run under simple setting and the widely existing internet photos. Apparently, the performance of single image based scene understanding may not be on par with the multiple image case, and vision tasks sometimes are even ill-posed, e.g. depth estimation from a single color image. One of the most promising approaches to tackle this is to use a data-driven method. However, real world data is very limited for most of these tasks, such as the widely used indoor RGBD dataset for normal prediction introduced by Silberman *et al.* [227], which contains a mere 1449 images. Such datasets are not trivial to collect due to various requirements such as sensing technology for depth and normal [227, 229] and excessive human effort for semantic segmentation [158, 53]. Moreover, current datasets lack pixel level accuracy due to sensor noise or labeling error (Fig. 3.11).

This has recently led to utilizing synthetic data in the form of 2D render pairs (RGB image and per-pixel label map) from digital 3D models [2, 45, 94, 293, 233, 185]. However, there are two major problems that have not been addressed: (1) studies of how indoor scene context affect training have not been possible due to the lack of large scene datasets, so training is performed mostly on repositories with independent 3D

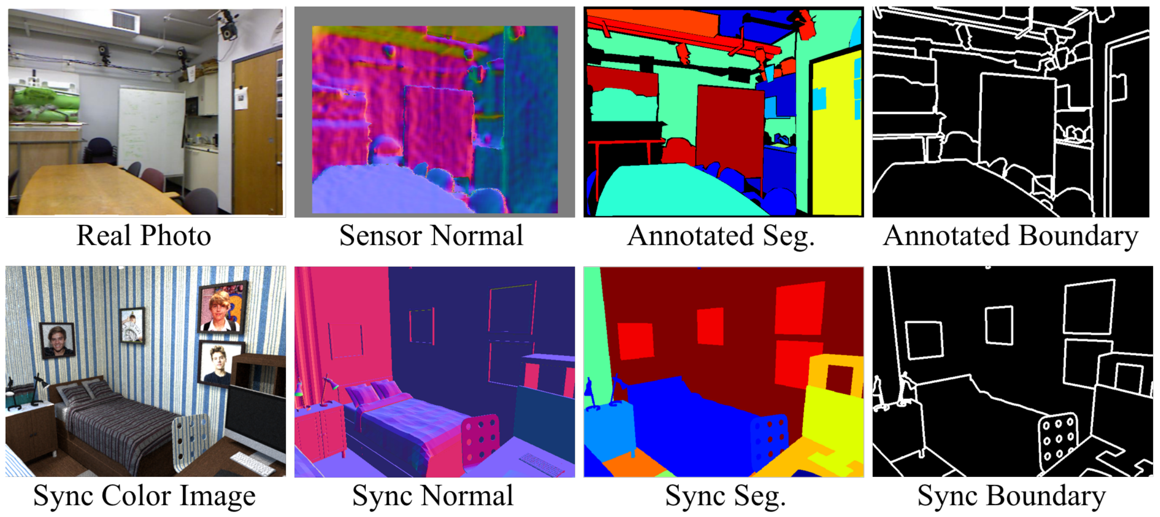


Figure 3.11: Real data (top) vs. synthetic data (bottom). For the real data, note the noise in normal map and the diminishing accuracy at object boundaries in the semantic labels.

objects [25]; and (2) systematic studies have not been done on how such data should be rendered; unrealistic rendering methods often are used in the interest of efficiency.

To address these problems, we introduce a large scale (400K images) synthetic dataset that is created from 45K 3D houses designed by humans [297]. Using such realistic indoor 3D environments enable us to create 2D images for training in realistic context settings where support constructs (e.g. such as walls, ceilings, windows) as well as light sources exist together with common household objects. Since we have access to the source 3D models, we can generate dense per-pixel training data for all tasks, virtually with no cost.

Complete control over the 3D scenes enables us to systematically manipulate both outdoor and indoor lighting, sample as many camera viewpoints as required, use the shapes in-context or out-of-context, and render with either simple shading methods, or physically based based rendering. For three indoor scene understanding tasks, namely normal prediction, semantic segmentation, and object edge detection, we study how different lighting conditions, rendering methods, and object context effects performance.

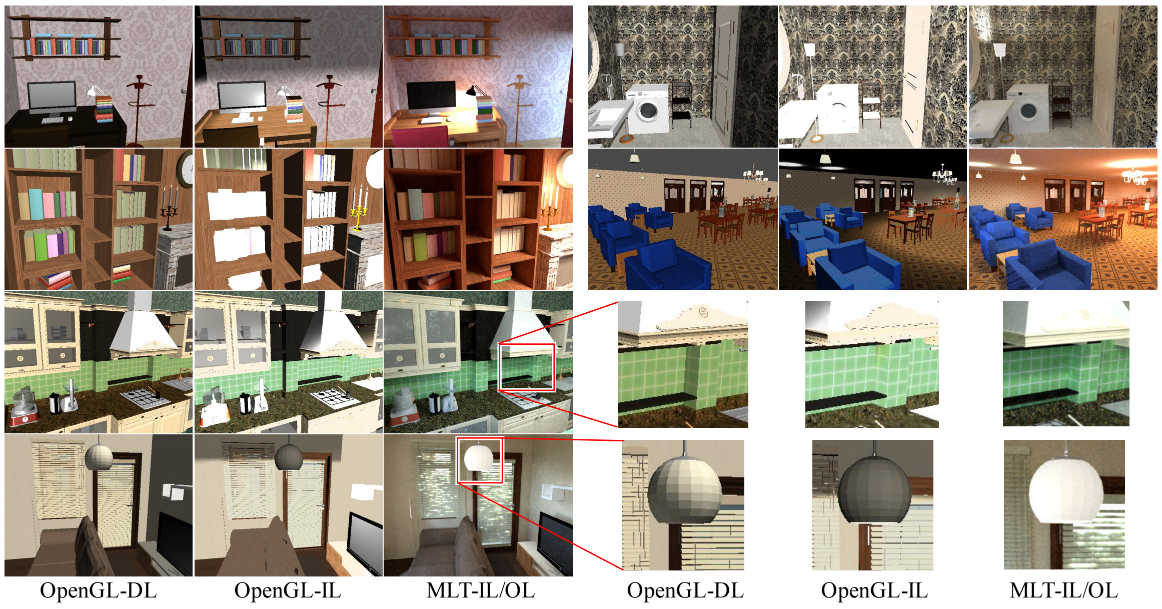


Figure 3.12: Render output examples with OpenGL-DL, OpenGL-IL, and MLT-IL/OL. The physically based rendering with proper illumination provides the best rendering quality with soft shadow and realistic material, highlighted in the zoomed in view. First two rows show four typical examples in our dataset, last two rows show two examples with zoomed in views.

We use our data to train deep convolutional neural networks for per-pixel prediction of semantic segmentation, normal prediction, and object boundary prediction, followed by finetuning on real data. Our experiments show that for all three indoor scene understanding tasks, we improve over the state of the art performance. We also demonstrate that physically based rendering with realistic lighting and soft shadows (which is not possible without context) is superior to other rendering methods.

In summary, our main contributions are as follows:

- We introduce a dataset with 400K synthetic image instances where each instance consists of three image renders with varying render quality, per-pixel accurate normal map, semantic labels and object boundaries. The dataset will be released.

- We demonstrate how different rendering methods effect normal, segmentation, and edge prediction tasks. We study the effect of object context, lighting and rendering methodology on performance.
- We provide pretrained networks that achieve the state of the art on all of the three indoor scene understanding tasks after fine-tuning.

### 3.2.2 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 [233, 185, 86], optical flow [45], semantic segmentation [93, 94, 293, 207], and investigating object features [2, 125].

Su *et al.* [233] 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.* [45] 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.* [185] 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 realistic context for the object under consideration.

Handa *et al.* [93, 94] introduced a laboriously created 3D scene dataset and demonstrated the usage on semantic segmentation training. However, their data consisted 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.* [293] proposed to replace objects in depth images with 3D models from ShapeNet [25]. 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 context relevance.

Xiang *et al.* [268] 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 with 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.* [207] demonstrated collecting synthetic data from 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 [210] 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.

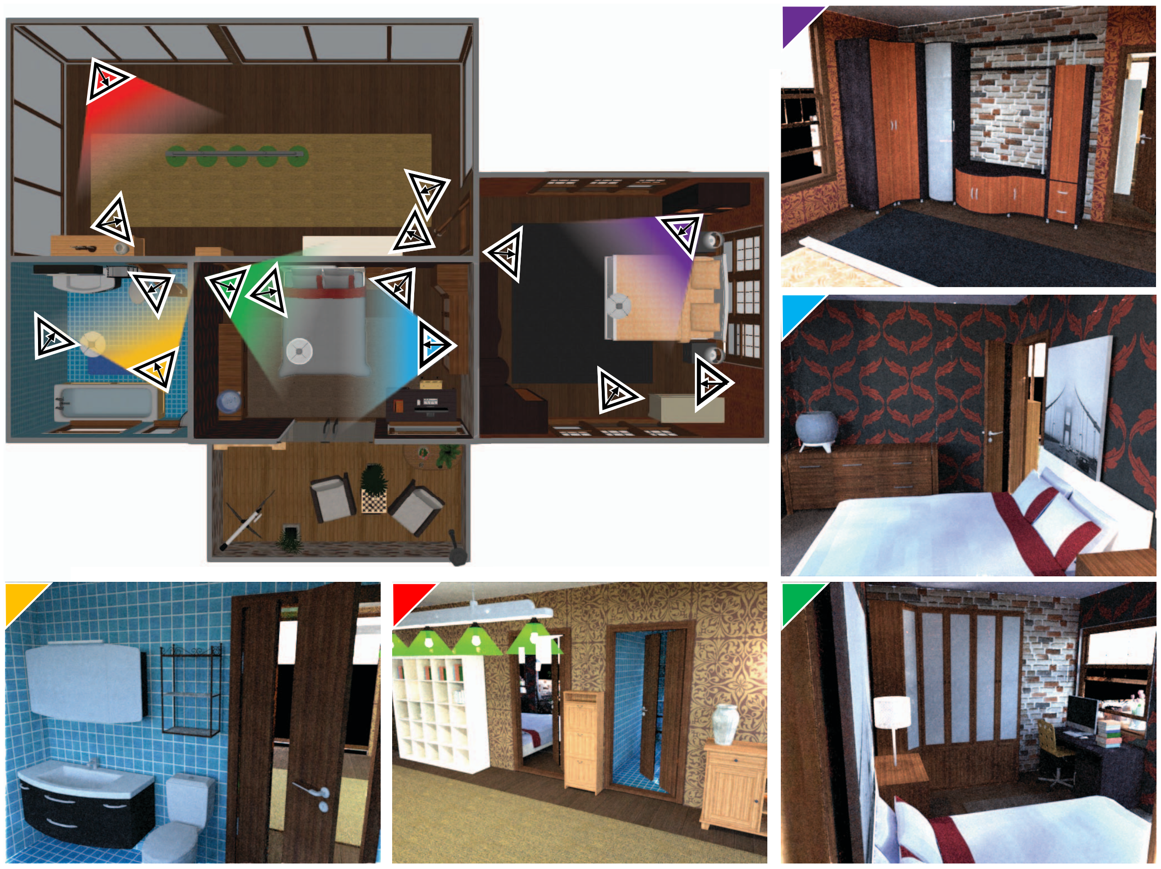


Figure 3.13: Typical camera samples in our dataset, and corresponding images rendered from these viewpoints.

### 3.2.3 Data

We use a collection of 3D scene models downloaded from the Planner5D website [201]. In total there are 45622 scenes with over 5000K instances of 2644 unique objects among 84 object categories. Objects are provided with surface materials, including reflectance, texture, and transparency, which are used to obtain photo-realistic renderings. One of the important aspects of this dataset is the fact that indoor layouts, furniture/object alignment, and surface materials are designed by people to replicate existing settings, or to plan for new upgrades for current homes. We will release our version of this data, together with minor improvements we have made (such as adding indoor light source labels, as explained later in this section), as well as our camera settings in order to facilitate repeatability.

#### 3.2.3.1 Camera Sampling.

For each scene, we select a set of cameras with a process that seeks a diverse set of views seeing many objects in context. Our process starts by selecting the "best" camera for each of six horizontal view direction sectors in every room. For each of the six views, we sample a dense set of cameras on a 2D grid with 0.25 resolution, choosing a random viewpoint within each grid cell, a random horizontal view direction within the 60 degree sector, a random height 1.5-1.6m above the floor, and a downward tilt angle of 11 degrees, while excluding viewpoints within 10cm of any obstacle to simulate typical human viewing conditions. For each of these cameras, we render an item buffer and count the number of pixels covered by "objects" in the image (everything except wall, ceiling, and floor), and select the one with the highest pixel count amongst those seeing at least three different objects covering at least 1% of the pixels. This process yields 6N candidate cameras for N rooms. Fig. 3.13 shows the cameras sampled from an example house.

#### 3.2.3.2 Image Rendering

We render images from these selected cameras using four combinations of rendering algorithms and lighting conditions, ranging from fast/unrealistic rendering with directional lights using the OpenGL pipeline to physically-based rendering with local lights using Mitsuba.

**OpenGL with Directional Lights (OpenGL-DL).** Our first method renders images with the OpenGL pipeline. The scene is illuminated with three lights: a single directional headlight pointing along the camera view direction and two directional lights pointing in nearly opposite diagonal directions with respect to the scene. No local illumination, shadows, or indirect illumination is included.

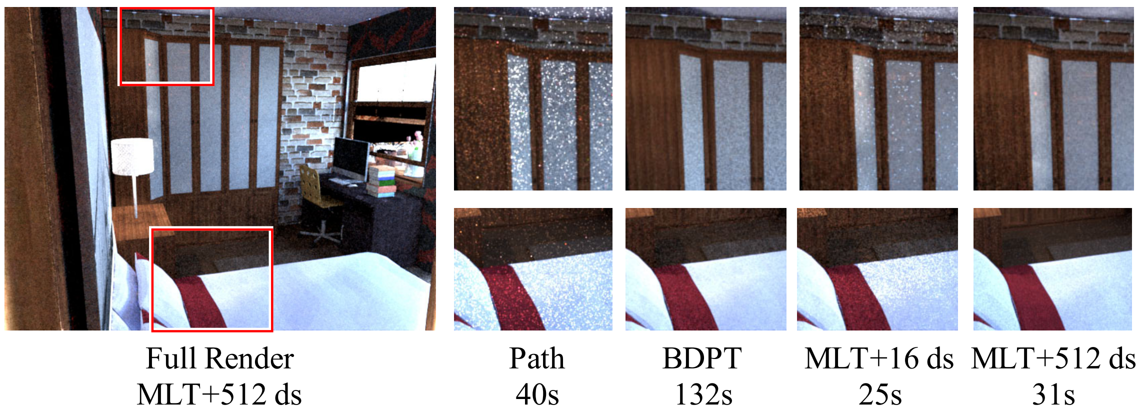


Figure 3.14: Quality and running time of different rendering techniques. Path tracer does not converge well and introduce white dots artifacts. Bidirectional path tracer works well but is very slow. Metropolis Light Transport (MLT) with low sampler rate for direct illumination still occasionally introduce white dot artifacts. We take MLT with high sampler rate for direct illumination.

**OpenGL with Indoor Lights (OpenGL-IL).** Our second method also uses the OpenGL pipeline. However, the scene is augmented with local lights approximating the emission of indoor lighting appliances. For each object emitting light, we create a set of OpenGL point lights and spot lights approximating its emission patterns. We then render the scene with these lights enabled (choosing the best 8 lights sources for each object based on illumination intensity), and no shadows or indirect illumination is included.

**Physically Based Rendering with Outdoor Lights (MLT-OL).** Our third method replicates the physics of correct lighting as much as possible to generate photo-realistic rendering. In order to do so, we setup outdoor illumination which is in the form of an environment mapping with real high-definition spherical sky panoramas. The environment map that replicates outdoor lighting is cast through windows and contributes to the indoor lighting naturally. All windows are set as fully transparent to prevent artifacts on glasses and facilitate the outdoor lights to pass through. We use Mitsuba [184] for physically based rendering. We use Path

Space Metropolis Light Transport (MLT) integrator [257] since it handles complicate structure and materials more efficiently. A comparison of rendering quality versus time with different integrators is shown in Fig. 3.14. We can see that MLT integrator with direct illumination sampler rate 512 produces almost artifact-free renderings with affordable computation time.

**Physically Based Rendering with Indoor Lights (MLT-IL/OL).** We also setup indoor illumination for light resulting from lighting appliances in the scene. However, the 3D dataset is labeled at the object level (e.g. lamp), and the specific *light generating* parts (e.g. bulb) is unknown. Therefore, we manually labeled all *light generating* parts of objects in order to generate correct indoor lighting. For light appliances that do not have a bulb, representing geometry in cases where bulb is deemed to be not seen, we manually added a spherical bulb geometry at the proper location. The bulb geometries of the lighting appliances are set as area emitter to work as indoor lights. As an efficient approximation of the translucent lamp shade, we set lamp shade geometries to be area emitter as well with very low radiance value. Similar to the outdoor lighting, we use Mitsuba and MLT integrator for physically based indoor lights. Fig. 3.12 shows several examples of images generated by different rendering techniques under the same camera. We can see, especially from the zoomed in view, that MLT-IL/OL produces soft shadow and natural looking materials.

#### 3.2.3.3 Image Selection

The final step of our image synthesis pipeline is to select a subset of images to use for training. Ideally, each of the images in our synthetic training set will be similar to ones found in a test set (e.g., NYUv2). However not all of them are good due to insufficient lighting or atypical distributions of depths (e.g., occlusion by a close-up object). We perform a selection procedure to keep only the images that are similar to those in NYUv2 dataset in terms of color and depth distribution. Specifically, we first compute a normalized color histogram for each real image in the NYUv2 dataset. For each image rendered by MLT-IL/OL, we also get the normalized color histograms and calculate the histogram similarity with those from NYUv2 as the sum of minimal value of each bin (Fig. 3.15). Then for each synthesized image, we assign it the largest similarity compared with all NYUv2 images as the score and do the same for the depth channel. Finally, we select all the images with color score and depth score both larger than 0.75. This process selects 391,522 images from the original 681,660 rendered images. Those images form our synthetic training set, and is referred as **MLT** in the latter part of this paper.

#### 3.2.3.4 Ground Truth Generation

We generate per-pixel ground truth images encoding surface normal, semantic segmentation, and object boundary for each image. Since we have the full 3D model and camera viewpoints, generating these ground images can be done via rendering with OpenGL (e.g., with an item buffer).

### 3.2.4 Indoor Scene Understanding Tasks

We investigate three most critical scene understanding tasks: (1) surface normal estimation, (2) semantic segmentation, and (3) object boundary detection. For all tasks we show how our method and synthetic data compares with state of the art works in the literature. Specifically, we compare with Eigen *et al.* [50] for normal estimation, with Long *et al.* [166] and Yu *et al.* [281] for semantic segmentation, and with Xie *et al.* [276] for object boundary detection. We perform these comparisons systematically using different rendering conditions introduced in Section 3.2.3. In addition, for normal estimation, we also add *object without context* rendering, which allows us to investigate the importance of context when using synthetic data as well.

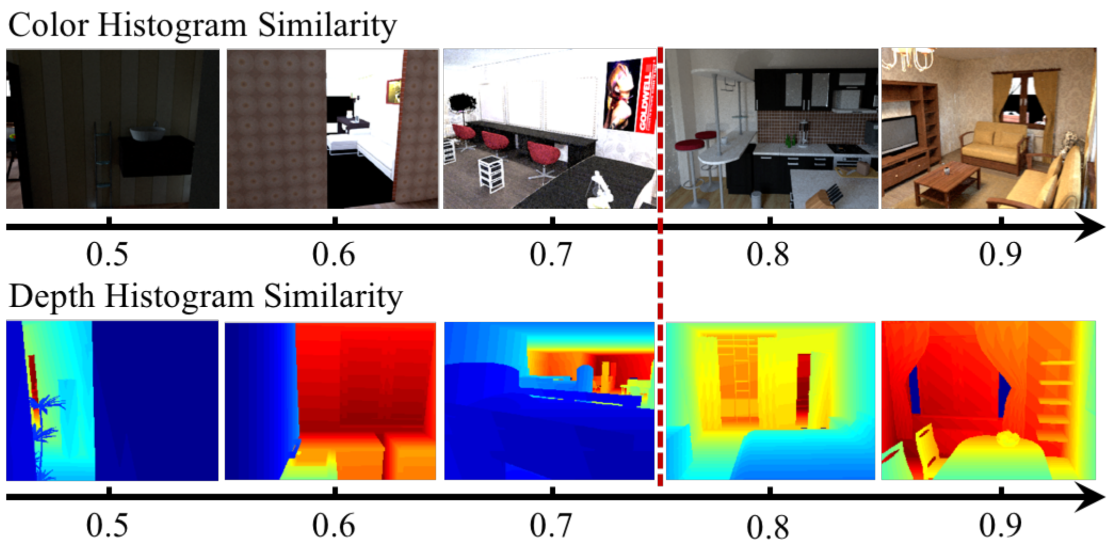


Figure 3.15: Histogram similarity between synthetic data and real data from NYUv2, based on which we do the image selection.

#### 3.2.4.1 Normal Estimation

**Method.** We utilize a fully convolutional network [166] (FCN) with skip-layers for normal estimation, by combining multi-scale feature maps in VGG-16 network [228] to perform normal estimation. Specifically, the front-end encoder remains the same as conv1-conv5 in VGG-16, and the decoder is symmetric to the encoder with convolution and unpooling layers. To generate high resolution results and alleviate the vanishing gradient problems, we use skip links between each pair of corresponding convolution layers in downstream and upstream parts of the network. To further compensate the loss of spatial information with max pooling, the network remembers pooling switches in downstream, and uses them as unpooling switches at upstream in the corresponding layer. We use the inverse of the dot product between the ground truth and the estimation as loss function similar to Eigen *et al.* [50].

**Object without Context.** To facilitate a systematic comparison with object-centric synthetic data, where correct context is missing, we use shapes from

| Pre-Train | Finetune | Selection | Mean (°) ↓ | Median(°) ↓ | 11.25° (%) ↑ | 22.5° (%) ↑ | 30°(%) ↑ |
|---|---|---|---|---|---|---|---|
| Eigen *et al.* [50] | | | 22.2 | 15.3 | 38.6 | 64.0 | 73.9 |
| NYUv2 | | | 27.30 | 21.12 | 27.21 | 52.61 | 64.72 |
| MLT Object | - | - | 48.78 | 47.49 | 3.56 | 12.79 | 21.35 |
| MLT-OL | - | No | 49.33 | 42.30 | 7.47 | 23.24 | 34.09 |
| MLT-IL/OL | - | No | 29.33 | 22.62 | 24.00 | 49.78 | 61.35 |
| MLT-IL/OL | - | Yes | 28.59 | 22.61 | 25.25 | 49.80 | 61.81 |
| OpenGL-DL | - | Yes | 36.89 | 31.97 | 15.21 | 35.32 | 47.14 |
| OpenGL-IL | - | Yes | 35.93 | 30.91 | 15.48 | 36.67 | 48.67 |
| OpenGL-IL | NYUv2 | Yes | 23.65 | 15.71 | 37.91 | 62.91 | 72.59 |
| MLT-IL/OL | NYUv2 | Yes | **22.06** | **14.78** | **39.60** | **65.61** | **75.25** |

Table 3.1: **Performance of Normal Estimation on NYUv2 with different training protocols.** The first three column lists the dataset for pretraining and finetuning, and if image selection is done. The evaluation metrics are mean and median of angular error, and percentage of pixels with error smaller than 11.25°, 22.5°, and 30°.

ShapeNet[25], in addition to the rendering methodologies introduced in Sec. 3.2.3.2. We randomly pick 3500 models from furniture related categories (e.g. bed, chair, cabinet, etc.) and set up 20 cameras from randomly chosen distances and viewing directions. More specifically, we place the model at the center of a 3D sphere and uniformly sample 162 points on the sphere by subdividing it into faces of an icosahedron. For each camera a random vertex of the icosachedron is selected. This point defines a vector together with the sphere center. The camera is placed at a random distance from the center between 1.5× to 4.5× of object bounding box diagonal, and points towards the center.

**Training.** We directly pretrain on our synthetic data, followed by finetuning on NYUv2 similar to the [5]. We use RMSprop [248] to train our network. The learning rate is set as $1 \times 10{-}3$, reducing to half every $300K$ iterations for the pretraining; and $1 \times 10{-}4$ reducing to half every $10K$ iterations for finetuning. The color image is zero-centered by subtracting 128. We use the procedure provided by [227] to generate the ground truth surface normals on NYUv2 as it provides more local details resulting in more realistic shape representation compared to others [172]. The ground truth also provides a score for each pixel indicating if the normal converted from local depth is

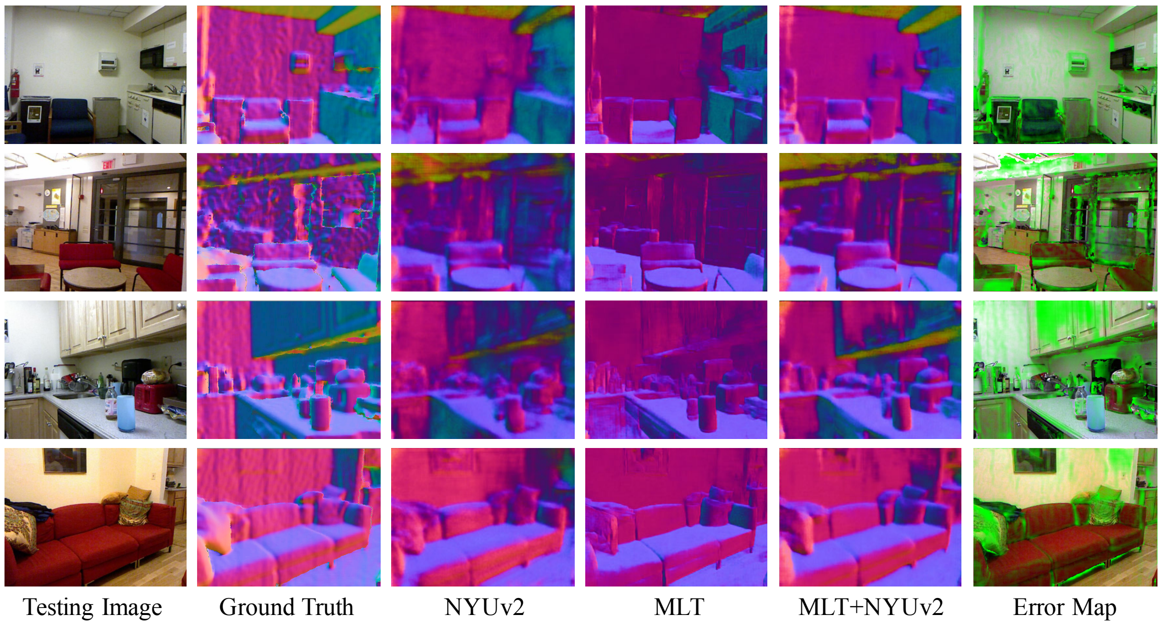


Figure 3.16: **Normal estimation results.** The pretrained model on MLT provides more local details, and model further finetuned on NYUv2 provides the best performance. The last column shows color image overlaid with angular error map. We can see a considerable amount of error happens on wall where ground truth is noisy.

reliable. We use only reliable pixels during the training. Refer to our supplementary material for details on the ground truth generation process.

**Experiments.** We conduct normal estimation experiments on NYUv2 with different training protocols. First, we directly train on NYUv2. Then we pretrain on various of MLT and OpenGL render settings respectively and finetune on NYUv2. Table 3.1 shows the performance. We can see that:

- The model pretrained on MLT and finetuned on NYUv2 (the last row) achieves the best performance, which outperforms the state of the art.
- Without finetuning, pretrained model on MLT significantly outperforms model pretrained on OpenGL based rendering and achieves similar performance with the model directly trained on NYUv2. This shows that physically based rendering with correct illumination is essential to encode useful information for normal prediction task.

- The model trained with images after image selection achieves better performance than using all rendered images, which demonstrates that good quality of training image is important for the pretraining.
- The MLT with both indoor and outdoor lighting significantly outperforms the case with only outdoor lighting, which suggests the importance of indoor lighting.

Fig. 3.16 shows visual results for normal estimation on NYUv2 test split. We can see that the result from the model pretrained on MLT rendering provides sharper edges and more local details compared to the one from the model further finetuned on NYUv2, which is presumably because of the overly smoothed and noisy ground truth. Fig. 3.16 last-column visualizes the angular error of our result compared to the ground truth, and we can see that a significant portion of the error concentrates on the walls, where our purely flat prediction is a better representation of wall normals. On the other hand, the ground truth shows significant deviation from the correct normal map. Based on this observation, we highlight the importance of high quality of ground truth. It is clear that training on synthetic data helps our model outperform and correct the NYUv2 ground truth data at certain regions such as large flat areas.

#### 3.2.4.2 Semantic Segmentation

**Method.** We use the network model proposed in [281] for semantic segmentation. The network structure is adopted from the VGG-16 network [228], however using dilated convolution layers to encode context information, which achieves better performance than [166] on NYUv2 in our experiments. We initialize the weights using the VGG-16 network [228] trained on ImageNet classification task using the procedure described in [281]. We evaluate on the same 40 semantic classes as [87].

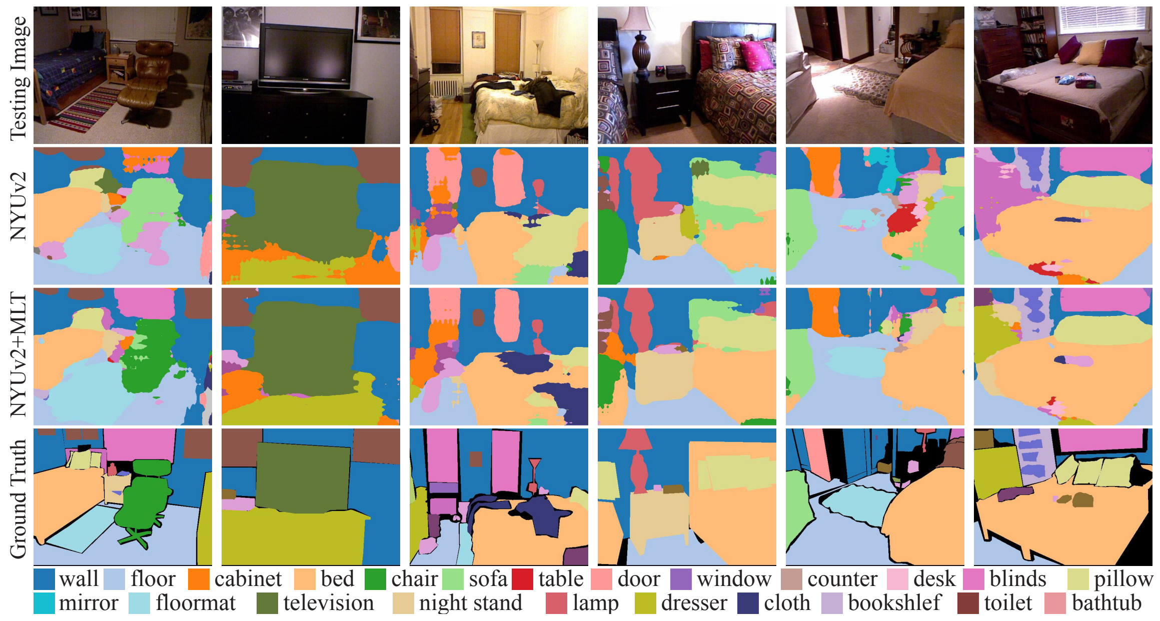


Figure 3.17: **Semantic Segmentation results.** The model pretrained on synthetic rendering data gives more accurate segmentation result. For example the model trained only with NYU data mis-labeled the chair and blind in the first column, while the model pretrained on the synthetic data is able to correctly segment them out.

**Training.** To use synthetic data for pretraining, we map our synthetic ground truth labels to the appropriate class name in these 40 classes[1]. However, note that there are several objects that are not present in our synthetic data (e.g. books, papers) as shown in Fig. 3.18. We first initialize the network with pretrained weights from ImageNet. We then follow with pretraining on our synthetic dataset, and finally finetune on NYUv2. We also replicate the corresponding state of the art training schedules by pretraining on ImageNet, followed directly by finetuning on NYUv2, for comparison. We use stochastic gradient descent with learning rate of $1 \times 10^{-5}$ for training on synthetic data and NYUv2.

**Experiments.** We use the average pixel-level intersection over union (IoU) to evaluate performance on semantic segmentation. We pretrained the model on our synthetic data with different rendering method: depth, OpenGL with and without local light,

[1]Refer to our supplementary material for the exact cross-match.

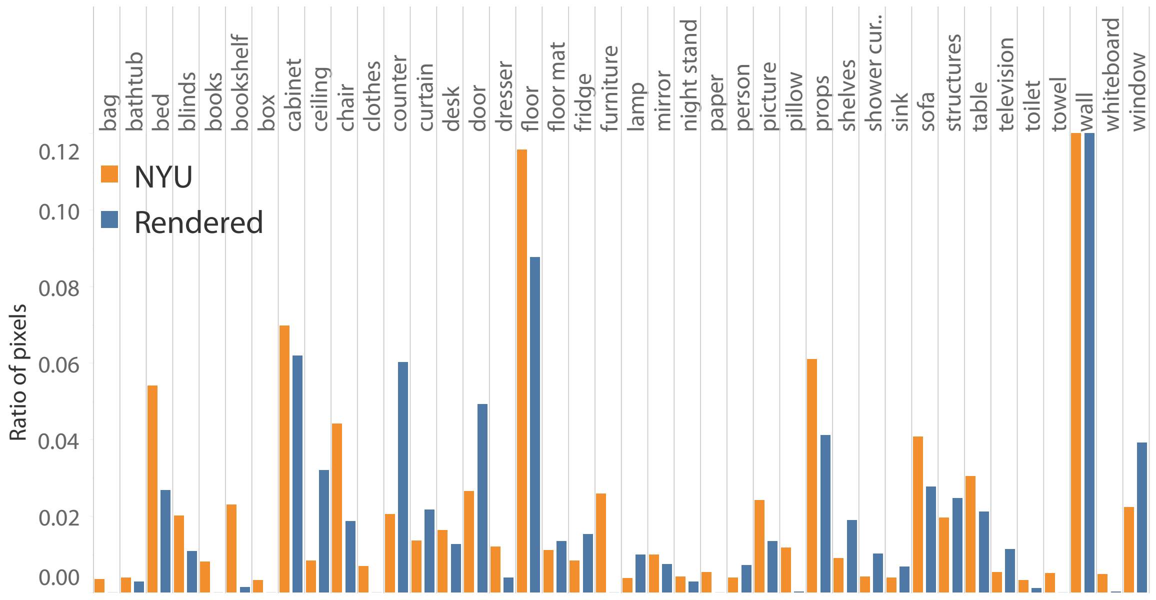


Figure 3.18: Distribution of classes in our data.

and with MLT rendering. For the depth based model we encode the depth using HHA same as [86]. Overall, pretraining on synthetic help improve the performance in semantic segmentation, compared to directly training on NYUv2 as seen in Fig. 3.17, and Table 3.2.4.2. This shows that the synthetic data helps the network learn richer high level context information than limited real data.

Handa *et al.* [94] use only rendered depth to train their 11 class semantic segmentation model due to the lack of realistic texture and material in their dataset (see HHA results in Table 3.2.4.2). However, our results demonstrate that color information is critical for more fine gained semantic segmentation task: in the 40 class task Model trained with color information achieves significantly better performance. For the color based models, pretraining on physically based rendering images helps to achieve better performance than pretraining on OpenGL rendering. This finding is consistent with normal estimation experiments. On the other hand, OpenGL rendering with local lights perform similar to OpenGL rendering without lights.

| Input | Pre-train | Mean IoU |
|---|---|---|
| HHA | ImageNet | 4.1 |
|  | ImageNet+OpenGL | 4.3 |
| RGB | Long *et al.* [166] | 31.6 |
|  | Yu *et al.* [281] | 31.7 |
|  | ImageNet + OpenGL-DL | 32.8 |
|  | ImageNet + OpenGL-IL | 32.9 |
|  | ImageNet + MLT-IL/OL | **33.2** |

Table 3.2: **Performance of Semantic Segmentation on NYUv2 with different training setting.** All models are fine-tuned on NYUv2.

#### 3.2.4.3 Object Boundary Detection

**Method.** We adopt Xie *et al.* 's [276] network architecture for object boundary detection task as they reported performance on NYUv2. The network starts with the front end of VGG-16, followed by a set of auxiliary-output layers, which produce boundary maps in multiple scales from fine to coarse. Each of these boundary maps are trained under supervision of the ground truth on corresponding scale. A weighted-fusion layer then learns the weight to combine boundary outputs in multi-scale to produces the final result (refer to our supplementary material for the detailed network structure). To evaluate the network, we follow the setting in [87], where the boundary ground truth is defined as the boundary of instance level segmentation.

**Training.** Similar to the semantic segmentation, we first initialize the network with pretrained weights on ImageNet. We then pretrain on our synthetic dataset, and fine-tune on NYUv2. We also replicate the state of the art training procedure by pretraining on ImageNet, and directly finetune on NYUv2, for comparison. To highlight the difference between multiple rendering techniques, we only train on color image without using depth. We follow the same procedure introduced in [276]. The standard stochastic gradient descend is used for optimization. The learning rate is initially set to be smaller ($2 \times 10^{-7}$), to deal with larger image resolution of NYUv2, and is reduced even more, to 1/10 after each $10K$ iterations on NYUv2. For synthetic data,

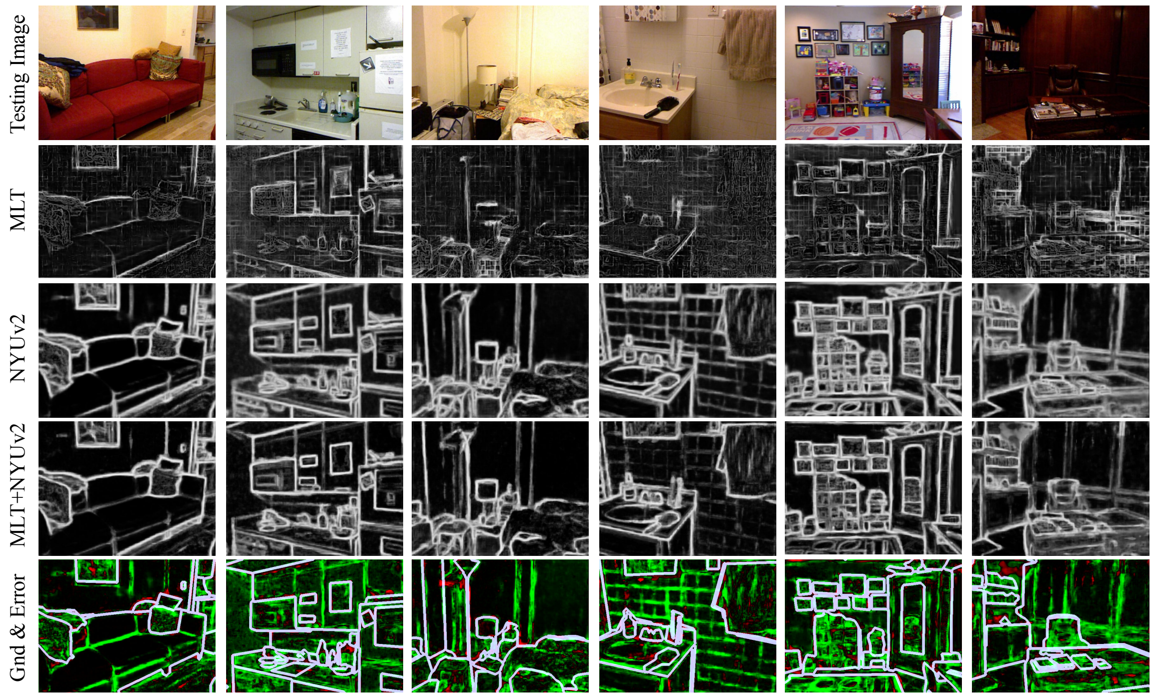


Figure 3.19: **Boundary estimation results.** The last row shows ground truth overlaid with the difference between model without (NYUv2) and with (MLT+NYUv2) synthetic data pretraining. Red and green indicates pixels enhanced and suppressed by MLT+NYUv2. The model with synthetic data pretraining successfully suppresses texture and background edges compared to the model without.

| Pre-train | Finetune | OSD↑ | OIS↑ | AP↑ | R50↑ |
|---|---|---|---|---|---|
| NYUv2 [276] | - | 0.713 | 0.725 | 0.711 | 0.267 |
| OpenGL-IL | - | 0.523 | 0.555 | 0.511 | 0.504 |
| MLT-IL/OL | - | 0.604 | 0.621 | 0.587 | 0.749 |
| OpenGL-IL | NYUv2 | 0.716 | 0.729 | 0.715 | **0.893** |
| MLT-IL/OL | NYUv2 | **0.725** | **0.736** | **0.720** | 0.887 |

Table 3.3: **Performance of boundary detection on NYUv2.**

similar to our procedure in like normal estimation task, the learning rate is reduced every $300k$ iterations.

**Experiments.** We train the model proposed in Xie *et al.* 's [276] with multiple different protocols and show our comparison and evaluation on NYUv2 in Table 3.3. Following the setting of [276], we take the average of the output from 2nd to 4th multiscale layers as the final result and perform non-maximum suppression and edge thinning. We use the ground truth in [87], and evaluation metrics in [43].

We train with the code released by [276] and achieve the performance shown in the first row of Table 3.3. We could not replicate the exact number in the paper but we were fairly close, which might be due to the randomized nature of training procedure. We first finetune the model based on the ImageNet initialization on the synthetic dataset and further finetune on NYUv2. Table 3.3 shows that the synthetic data pretraining provides consistent improvement on all evaluation metrics. Consistently, we see the model pretrained with MLT rendering achieves the best performance.

Fig. 3.19 shows a comparison between results from different models. Pretrained model on synthetic data, prior to finetuning on real data produces sharper results but is more sensitive to noise. The last column highlights the difference between model with and without pretraining on our synthetic data. We can see that edges within objects themselves as well as the ones in the background (green) are suppressed and true object boundary (red) are enhanced by the model with pretraining on synthetic.

### 3.2.5 Conclusion

We investigate the impact of the rendering quality when using synthetic data during pretraining for single image based scene understanding [297]. We introduce a large-scale synthetic dataset with 400K rendered images of contextually meaningful 3D indoor scenes with different lighting and rendering settings, as well as 45K indoor scenes they were rendered from. We show that pretraining on our physically based rendering with realistic indoor and outdoor lights boost indoor scene understanding tasks' performance. Our experiments show that our approach helps us perform better than state of the art in surface normal estimation, semantic segmentation, and edge detection tasks.

## 3.3 Deep Depth Completion of a Single RGB-D Image

### 3.3.1 Introduction

Not only as a compromised solution when multiple cameras are unavailable, single image based scene understanding can also benefit and improve other vision tasks. In this project, we study how to improve the depth map quality for commercial depth sensor using single image based scene understanding [294].

Despite recent advances in depth sensing technology, commodity-level RGB-D cameras like Microsoft Kinect, Intel RealSense, and Google Tango still produce depth images with missing data when surfaces are too glossy, bright, thin, close, or far from the camera. These problems appear when rooms are large, surfaces are shiny, and strong lighting is abundant – e.g., in museums, hospitals, classrooms, stores, etc. Even in homes, depth images often are missing more than 50% of the pixels (Fig. 3.20).

The goal of this work is to complete the depth channel of an RGB-D image captured with a commodity camera (i.e., fill all the holes). Though depth inpainting has received a lot of attention over the past two decades [254], it has generally been addressed with hand-tuned methods that fill holes by extrapolating boundary surfaces [178] or with Markovian image synthesis [44]. Newer methods have been proposed to estimate depth de novo from color using deep networks [51]. However, they have not been used for depth completion, which has its own unique challenges:

**Training data.** Large-scale training sets are not readily available for captured RGB-D images paired with "completed" depth images (e.g., where ground-truth depth is provided for holes). As a result, most methods for depth estimation are trained and evaluated only for pixels that are captured by commodity RGB-D cam-

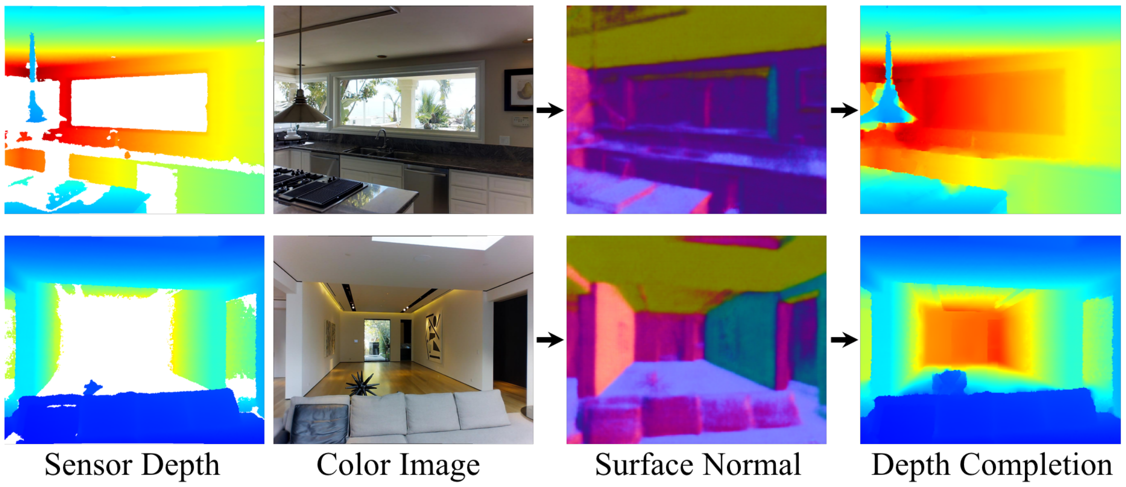


Figure 3.20: **Depth Completion.** We fill in large missing areas in the depth channel of an RGB-D image by predicting normals from color and then solving for completed depths.

eras [227]. From this data, they can at-best learn to reproduce observed depths, but not complete depths that are unobserved, which have significantly different characteristics. To address this issue, we introduce a new dataset with 105,432 RGB-D images aligned with completed depth images computed from large-scale surface reconstructions in 72 real-world environments.

**Depth representation.** The obvious approach to address our problem is to use the new dataset as supervision to train a fully convolutional network to regress depth directly from RGB-D. However, that approach does not work very well, especially for large holes like the one shown in the bottom row of Fig. 3.20. Estimating absolute depths from a monocular color image is difficult even for people [183]. Rather, we train the network to predict only local differential properties of depth (surface normals and occlusion boundaries), which are much easier to estimate [135]. We then solve for the absolute depths with a global optimization.

**Deep network design:**

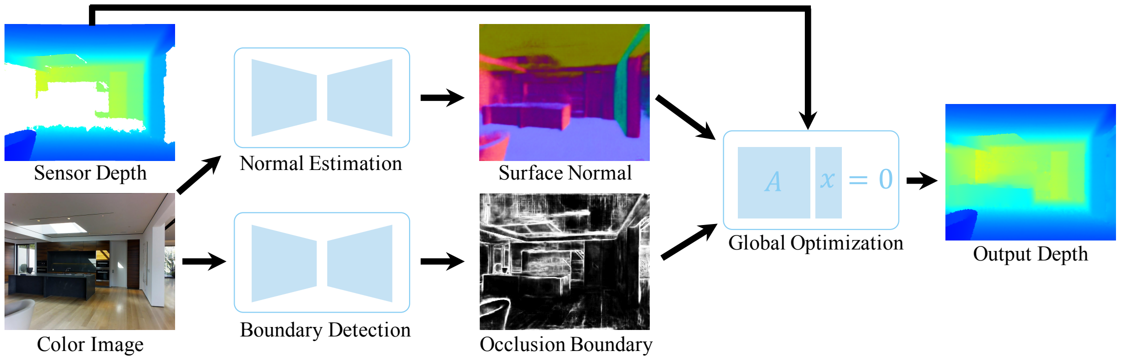


Figure 3.21: **System pipeline.** Given an input RGB-D image, we predict surface normals and occlusion boundaries from color, and then solve for the output depths with a global linear optimization regularized by the input depth.

**Deep network design.** There is no previous work on studying how best to design and train an end-to-end deep network for completing depth images from RGB-D inputs. At first glance, it seems straight-forward to extend previous networks trained for color-to-depth (e.g., by providing them an extra depth channel as input). However, we found it difficult to train the networks to fill large holes from depth inputs – they generally learn only to copy and interpolate the input depth. It is also challenging for the network to learn how to adapt for misalignments of color and depth. Our solution is to provide the network with only color images as input (Figure 3.21). We train it to predict local surface normals and occlusion boundaries with supervision. We later combine those predictions with the input depths in a global optimization to solve back to the completed depth. In this way, the network predicts only local features from color, a task where it excels. The coarse-scale structure of the scene is reconstructed through global optimization with regularization from the input depth.

Overall, our main algorithmic insight is that it is best to decompose RGB-D depth completion into two stages: 1) prediction of surface normals and occlusion boundaries only from color, and 2) optimization of global surface structure from those predictions with soft constraints provided by observed depths. During experiments we find with this proposed approach has significantly smaller relative error than alternative approaches. It has the extra benefit that the trained network is independent of the observed depths and so does not need to be retrained for new depth sensors.

### 3.3.2 Related Work

There has been a large amount of prior work on depth estimation, inpainting, and processing.

**Depth estimation.** Depth estimation from a monocular color image is a long-standing problem in computer vision. Classic methods include shape-from-shading [292] and shape-from-defocus [239]. Other early methods were based on hand-tuned models and/or assumptions about surface orientations [107, 217, 218]. Newer methods treat depth estimation as a machine learning problem, most recently using deep networks [51, 275]. For example, Eigen et al. first used a multiscale convolutional network to regress from color images to depths [51, 50]. Laina et al. used a fully convolutional network architecture based on ResNet [146]. Liu et al. proposed a deep convolutional neural field model combining deep networks with Markov random fields [160]. Roy et al. combined shallow convolutional networks with regression forests to reduce the need for large training sets [211]. All of these methods are trained only to reproduce the raw depth acquired with commodity RGB-D cameras. In contrast, we focus on depth completion, where the explicit goal is to make novel predictions for pixels where the depth sensor has no return. Since these pixels are often missing in the raw depth, methods trained only on raw depth as supervision do not predict them well.

**Depth inpainting.** Many methods have been proposed for filling holes in depth channels of RGB-D images, including ones that employ smoothness priors [104], fast marching methods [80, 162], Navier-Stokes [11], anisotropic diffusion [161], background surface extrapolation [178, 186, 246], color-depth edge alignment [28, 289, 306], low-rank matrix completion [278], tensor voting [141], Mumford-Shah functional optimization [164], joint optimization with other properties of intrinsic images [7], and patch-based image synthesis [33, 44, 74]. Recently, methods have been proposed for inpainting *color* images with auto-encoders [256] and GAN architectures [198]. However, prior work has not investigated how to use those methods for inpainting of depth images. This problem is more difficult due to the absence of strong features in depth images and the lack of large training datasets, an issue addressed in this paper.

**Depth super-resolution.** Several methods have been proposed to improve the spatial resolution of depth images using high-resolution color. They have exploited a variety of approaches, including Markov random fields [174, 42, 170, 196, 224], shape-from-shading [92, 283], segmentation [169], and dictionary methods [70, 132, 175, 250]. Although some of these techniques may be used for depth completion, the challenges of super-resolution are quite different – there the focus is on improving spatial resolution, where low-resolution measurements are assumed to be complete and regularly sampled. In contrast, our focus is on filling holes, which can be quite large and complex and thus require synthesis of large-scale content.

**Depth reconstruction from sparse samples.** Other work has investigated depth reconstruction from color images augmented with sparse sets of depth measurements. Hawe et al. investigated using a Wavelet basis for reconstruction [97]. Liu et al. combined wavelet and contourlet dictionaries [163]. Ma et al. showed that providing $\sim$100 well-spaced depth samples improves depth estimation over color-only methods by two-fold for NYUv2 [173], yet still with relatively low-quality results. These methods share some ideas with our work. However, their motivation is to reduce the cost of sensing in specialized settings (e.g., to save power on a robot), not to complete data typically missed in readily available depth cameras.

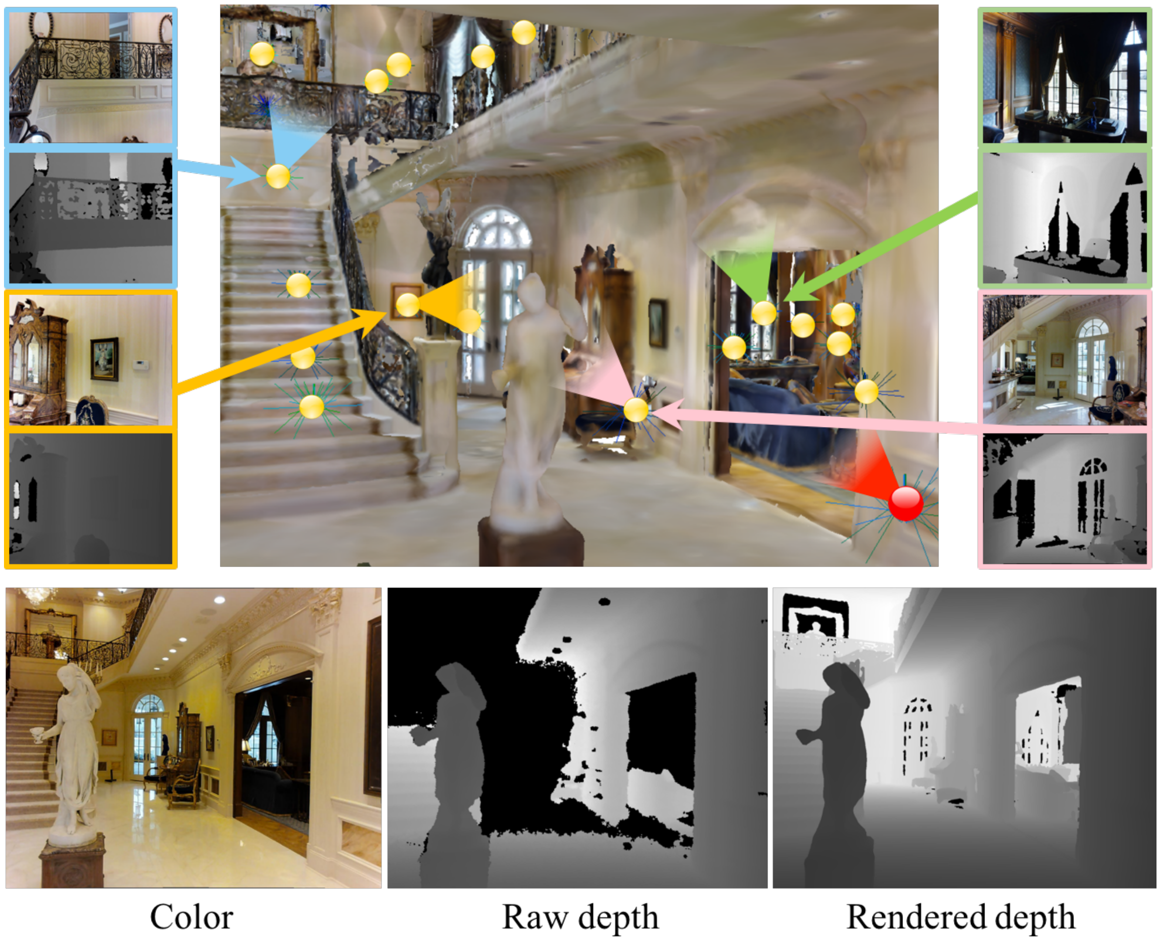


Figure 3.22: **Depth Completion Dataset.** Depth completions are computed from multi-view surface reconstructions of large indoor environments. In this example, the bottom shows the raw color and depth channels with the rendered depth for the viewpoint marked as the red dot. The rendered mesh (colored by vertex in large image) is created by combining RGB-D images from a variety of other views spread throughout the scene (yellow dots), which collaborate to fill holes when rendered to the red dot view.

### 3.3.3 Method

In this paper, we investigate how to use a deep network to complete the depth channel of a single RGB-D image. Our investigation focuses on the following questions: "how can we get training data for depth completion?," "what depth representation should we use?," and "how should cues from color and depth be combined?."

#### 3.3.3.1 Dataset

The first issue we address is to create a dataset of RGB-D images paired with completed depth images.

A straight-forward approach to this task would be to capture images with a low-cost RGB-D camera and align them to images captured simultaneously with a higher cost depth sensor. This approach is costly and time-consuming – the largest public datasets of this type cover a handful of indoor scenes (e.g., [197, 219, 278]).

Instead, to create our dataset, we utilize existing surface meshes reconstructed from multi-view RGB-D scans of large environments. There are several datasets of this type, including Matterport3D [24], ScanNet [35], SceneNN [113], and SUN3D [89, 273], to name a few. We use Matterport3D. For each scene, we extract a triangle mesh $M$ with $\sim$1-6 million triangles per room from a global surface reconstruction using screened Poisson surface reconstruction [126]. Then, for a sampling of RGB-D images in the scene, we render the reconstructed mesh $M$ from the camera pose of the image viewpoint to acquire a completed depth image D*. This process provides us with a set of RGB-D $\rightarrow$ D* image pairs without having to collect new data.

Fig. 3.22 shows some examples of depth image completions from our dataset. Though the completions are not always perfect, they have several favorable properties for training a deep network for our problem [181]. First, the completed depth images generally have fewer holes. That's because it is not limited by the observation of one camera viewpoint (e.g., the red dot in Fig. 3.22), but instead by the union of all observations of all cameras viewpoints contributing to the surface reconstruction (yellow dots in Fig. 3.22). As a result, surfaces distant to one view, but within range of another, will be included in the completed depth image. Similarly, glossy surfaces that provide no depth data when viewed at a grazing angle usually can be filled in with data from other cameras viewing the surface more directly (note the completion of the shiny floor in rendered depth). On average, 64.6% of the pixels missing from the raw depth images are filled in by our reconstruction process.

Second, the completed depth images generally replicate the resolution of the originals for close-up surfaces, but provide far better resolution for distant surfaces. Since the surface reconstructions are constructed at a 3D grid size comparable to the resolution of a depth camera, there is usually no loss of resolution in completed depth images. However, that same 3D resolution provides an effectively higher pixel resolution for surfaces further from the camera when projected onto the view plane. As a result, completed depth images can leverage subpixel antialiasing when rendering high resolution meshes to get finer resolution than the originals (note the detail in the furniture in Fig. 3.22).

Finally, the completed depth images generally have far less noise than the originals. Since the surface reconstruction algorithm combines noisy depth samples from many camera views by filtering and averaging, it essentially de-noises the surfaces. This is especially important for distant observations (e.g., >4 meters), where raw depth measurements are quantized and noisy.

In all, our dataset contains 117,516 RGB-D images with rendered completions, which we split into a training set with 105,432 images and a test set with 12,084 images.

#### 3.3.3.2 Depth Representation

A second interesting question is “what geometric representation is best for deep depth completion?”

A straight-forward approach is to design a network that regresses completed depth from raw depth and color. However, absolute depth can be difficult to predict from monocular images, as it may require knowledge of object sizes, scene categories, etc.

Instead, we train the network to predict local properties of the visible surface at each pixel and then solve back for the depth from those predictions.

Previous work has considered a number of indirect representations of depth. For example, Chen et al. investigated relative depths [27]. Charkrabarti et al. proposed depth derivatives [23]. Li et al. used depth derivatives in conjunction with depths [152]. We have experimented with methods based on predicted derivatives. However, we find that they do not perform the best in our experiments (see Section 3.3.4).

Instead, we focus on predicting surface normals and occlusion boundaries. Since normals are differential surface properties, they depend only on local neighborhoods of pixels. Moreover, they relate strongly to local lighting variations directly observable in a color image. For these reasons, previous works on dense prediction of surface normals from color images produce excellent results [5, 50, 150, 262, 297]. Similarly, occlusion boundaries produce local patterns in pixels (e.g., edges), and so they usually can be robustly detected with a deep network [49, 297].

A critical question, though, is how we can use predicted surface normals and occlusion boundaries to complete depth images. Several researchers have used predicted normals to refine details on observed 3D surfaces [95, 188, 277], and Galliani *et al.* [72] used surface normals to recover missing geometry in multi-view reconstruction for table-top objects. However, nobody has ever used surface normals before for depth estimation or completion from monocular RGB-D images in complex environments.

Unfortunately, it is theoretically not possible to solve for depths from only surface normals and occlusion boundaries. There can be pathological situations where the depth relationships between different parts of the image cannot be inferred only from normals. For example, in Fig. 3.23(a), it is impossible to infer the depth of the wall seen through the window based on only the given surface normals. In this case, the visible region of the wall is enclosed completely by occlusion boundaries (contours)

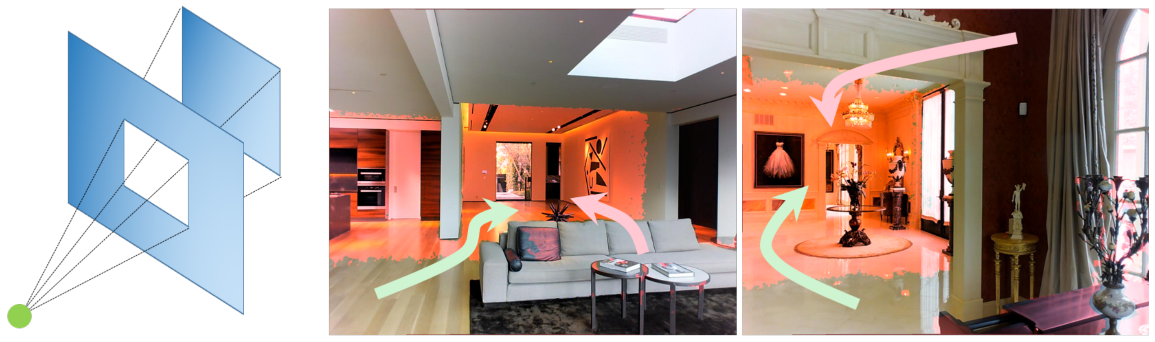


Figure 3.23: **Using surface normals to solve for depth completion.** (a) An example of where depth cannot be solved from surface normal. (b) The area missing depth is marked in red. The red arrow shows paths on which depth cannot be integrated from surface normals. However in real-world images, there are usually many paths through connected neighboring pixels (along floors, ceilings, etc.) over which depths can be integrated (green arrows).

from the perspective of the camera, leaving its depth indeterminate with respect to the rest of the image.

In practice, however, for real-world scenes it is very unlikely that a region of an image will both be surrounded by occlusion boundaries AND contain no raw depth observations at all (Fig. 3.23(b)). Therefore, we find it practical to complete even large holes in depth images using predicted surface normals with coherence weighted by predicted occlusion boundaries and regularization constrained by observed raw depths. During experiments, we find that solving depth from predicted surface normals and occlusion boundaries results in better depth completions than predicting absolute depths directory, or even solving from depth derivatives (see Section 3.3.4).

#### 3.3.3.3 Network Architecture and Training

A third interesting question is “what is the best way to train a deep network to predict surface normals and occlusion boundaries for depth completion?”

For our study, we pick the deep network architecture proposed in Zhang et.al because it has shown competitive performance on both normal estimation and boundary detection [297]. The model is a fully convolutional neural network built on the back-bone of VGG-16 with symmetry encoder and decoder. It is also equipped with short-cut connections and shared pooling masks for corresponding max pooling and unpooling layers, which are critical for learning local image features. We train the network with "ground truth" surface normals and silhouette boundaries computed from the reconstructed mesh.

After choosing this network, there are still several interesting questions regarding how to training it for depth completion. The following paragraphs consider these questions with a focus on normal estimation, but the issues and conclusions apply similarly for occlusion boundary detection.

**What loss should be used to train the network?** Unlike past work on surface normal estimation, our primary goal is to train a network to predict normals *only for pixels inside holes* of raw observed depth images. Since the color appearance characteristics of those pixels are likely different than the others (shiny, far from the camera, etc.), one might think that the network should be supervised to regress normals only for these pixels. Yet, there are fewer pixels in holes than not, and so training data of that type is limited. It was not obvious whether it is best to train only on holes vs. all pixels. So, we tested both and compared.

We define the observed pixels as the ones with depth data from both the raw sensor and the rendered mesh, and the unobserved pixels as the ones with depth from the rendered mesh but not the raw sensor. For any given set of pixels (observed, unobserved, or both), we train models with a loss for only those pixels by masking out the gradients on other pixels during the back-propagation.

Qualitative and quantitative results comparing the results for different trained models are shown in supplemental material. The results suggest that the models trained with all pixels perform better than the ones using only observed or only unobserved pixels, and ones trained with rendered normals perform better than with raw normals.

**What image channels should be input to the network?** One might think that the best way to train the network to predict surface normals from a raw RGB-D image is to provide all four channels (RGBD) and train it to regress the three normal channels. However, surprisingly, we find that our networks performed poorly at predicting normals for pixels without observed depth when trained that way. They are excellent at predicting normals for pixels with observed depth, but not for the ones in holes – i.e., the ones required for depth completion. This result holds regardless of what pixels are included in the loss.

We conjecture that the network trained with raw depth mainly learns to compute normals from depth directly – it fails to learn how to predict normals from color when depth is not present, which is the key skill for depth completion. In general, we find that the network learns to predict normals better from color than depth, even if the network is given an extra channel containing a binary mask indicating which pixels have observed depth [291]. For example, in Fig. 3.24, we see that the normals predicted in large holes from color alone are better than from depth, and just as good as from both color and depth. Quantitative experiments support this finding in Table 3.4.

This result is very interesting because it suggests that we can train a network to predict surface normals from color alone and use the observed depth *only as regularization* when solving back for depth from normals (next section). This strategy of separating "prediction without depth" from "optimization with depth" is compelling for two reasons. First, the prediction network does not have to be retrained for different depth sensors. Second, the optimization can be generalized to take a variety of

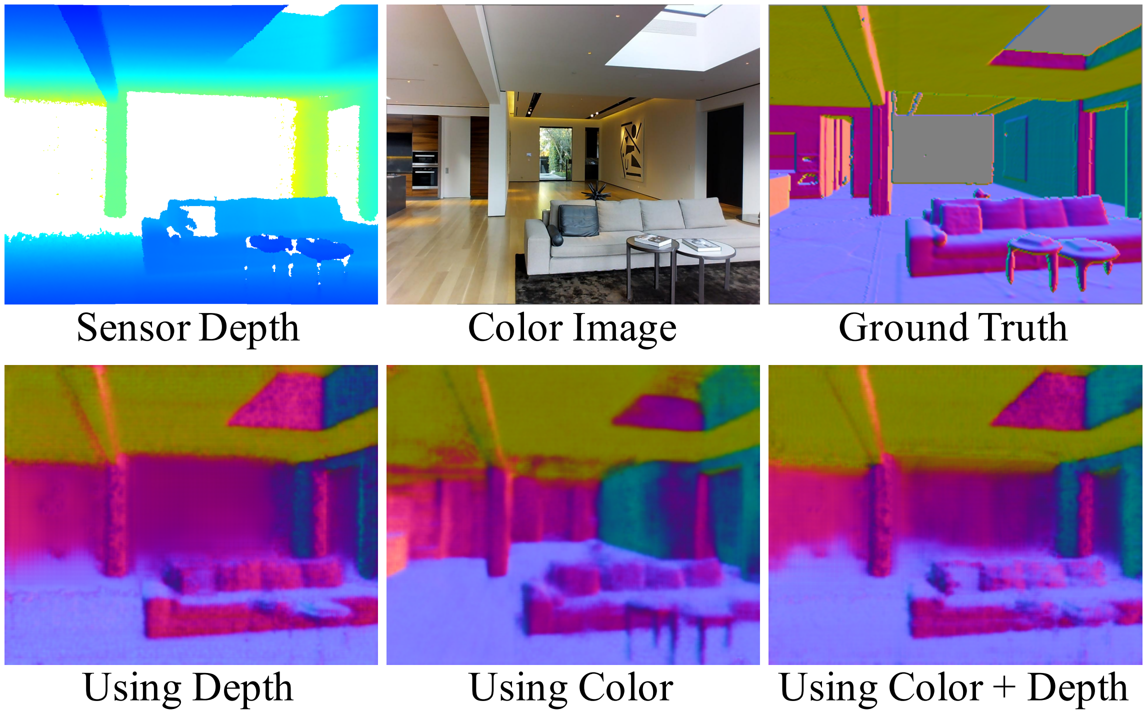


Figure 3.24: **Surface normal estimation for different inputs.** The top row shows an input color image, raw depth, and the rendered normal. The bottom row shows surface normal predictions when the inputs are depth only, color only, and both. The middle one performs the best for the missing area, while comparable elsewhere with the other two models even without depth as input.

| | Depth Completion | | | | | | | Surface Normal Estimation | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Input | Rel↓ | RMSE↓ | 1.05↑ | 1.10↑ | 1.25↑ | $1.25^2$↑ | $1.25^3$↑ | Mean↓ | Median↓ | 11.25↑ | 22.5↑ | 30↑ |
| Depth | 0.107 | 0.165 | 38.89 | 48.54 | 61.12 | 73.57 | 80.98 | 35.08 | 23.07 | 27.6 | 49.1 | 58.6 |
| Both | 0.090 | 0.124 | 40.13 | 51.26 | 64.84 | 76.46 | 83.05 | 35.30 | 23.59 | 26.7 | 48.5 | 58.1 |
| Color | 0.089 | 0.116 | 40.63 | 51.21 | 65.35 | 76.64 | 82.98 | 31.13 | 17.28 | 37.7 | 58.3 | 67.1 |

Table 3.4: **Effect of different inputs to our deep network.** We train models taking depth, color, and both respectively for surface normal estimation and depth completion. Using only color as input achieves similar performance as the case with both.

depth observations as regularization, including perhaps sparse depth samples [173]. This is investigated experimentally in Section 3.3.4.

#### 3.3.3.4 Optimization

After predicting the surface normal image $N$ and occlusion boundary image $B$, we solve a system of equations to complete the depth image $D$. The objective function is defined as the weighted sum of squared errors with four terms:

$$\begin{aligned} E &= \lambda_D E_D + \lambda_S E_S + \lambda_N E_N B \\ E_D &= \sum_{p \in T_{obs}} ||D(p) - D_0(p)||^2 \\ E_N &= \sum_{p,q \in N} || < v(p,q), N(p) > ||^2 \\ E_S &= \sum_{p,q \in N} ||D(p) - D(q))|^2 \end{aligned} \tag{3.1}$$

where $E_D$ measures the distance between the estimated depth $D(p)$ and the observed raw depth $D_0(p)$ at pixel $p$, $E_N$ measures the consistency between the estimated depth and the predicted surface normal $N(p)$, $E_S$ encourages adjacent pixels to have the same depths. $B \in [0, 1]$ down-weights the normal terms based on the predicted probability a pixel is on an occlusion boundary (B(p)).

In its simplest form, this objective function is non-linear, due to the normalization of the tangent vector $v(p, q)$ required for the dot product with the surface normal in $E_N$. However, we can approximate this error term with a linear formation by foregoing the vector normalization, as suggested in [188]. In other settings, this approximation would add sensitivity to scaling errors, since smaller depths result in shorter tangents and potentially smaller $E_N$ terms. However, in a depth completion setting, the data term $E_D$ forces the global solution to maintain the correct scale by enforcing consistency with the observed raw depth, and thus this is not a significant problem.

Since the matrix form of the system of equations is sparse and symmetric positive definite, we can solve it efficiently with a sparse Cholesky factorization (as implemented in cs_cholsol in CSparse [37]). The final solution is a global minimum to the approximated objective function.

This linearization approach is critical to the success of the proposed method. Surface normals and occlusion boundaries (and optionally depth derivatives) capture only local properties of the surface geometry, which makes them relatively easy to estimate. Only through global optimization can we combine them to complete the depths for all pixels in a consistent solution.

### 3.3.4 Experimental Results

We ran a series of experiments to test the proposed methods. Unless otherwise specified, networks were pretrained on the SUNCG dataset [232, 297] and fine-tuned on the training split of the our new dataset using only color as input and a loss computed for all rendered pixels. Optimizations were performed with $\lambda_D = 10^3$, $\lambda_N = 1$, and $\lambda_S = 10^{-3}$. Evaluations were performed on the test split of our new dataset.

We find that predicting surface normals and occlusion boundaries from color at 320x256 takes $\sim$0.3 seconds on a NVIDIA TITAN X GPU. Solving the linear equations for depths takes $\sim$1.5 seconds on a Intel Xeon 2.4GHz CPU.

#### 3.3.4.1 Ablation Studies

The first set of experiments investigates how different test inputs, training data, loss functions, depth representations, and optimization methods affect the depth prediction results (further results can be found in the supplemental material).

Since the focus of our work is predicting depth where it is unobserved by a depth sensor, our evaluations measure errors in depth predictions only for pixels of test images *unobserved* in the test depth image (but present in the rendered image). This is the opposite of most previous work on depth estimation, where error is measured only for pixels that are observed by a depth camera.

When evaluating depth predictions, we report the median error relative to the rendered depth (Rel), the root mean squared error in meters (RMSE), and percentages of pixels with predicted depths falling within an interval ($[\delta = |predicted - true|/true]$), where $\delta$ is 1.05, 1.10, 1.25, $1.25^2$, or $1.25^3$. These metrics are standard among previous work on depth prediction, except that we add thresholds of 1.05 and 1.10 to enable finer-grained evaluation.

When evaluating surface normal predictions, we report the mean and median errors (in degrees), plus the percentages of pixels with predicted normals less than thresholds of 11.25, 22.5, and 30 degrees.

**What data should be input to the network?** Table 3.4 shows results of an experiment to test what type of inputs are best for our normal prediction network: color only, raw depth only, or both. Intuitively, it would seem that inputting both would be best. However, we find that the network learns to predict surface normals better when given only color (median error = 17.28° for color vs. 23.07° for both), which results in depth estimates that are also slightly better (Rel = 0.089 vs. 0.090). This difference persists whether we train with depths for all pixels, only observed pixels, or only unobserved pixels (results in supplemental material). We expect the reason is that the network quickly learns to interpolate from observed depth if it is available, which hinders it from learning to synthesize new depth in large holes.

The impact of this result is quite significant, as it motivates our two-stage system design that separates normal/boundary prediction only from color and optimization with raw depth.

**What depth representation is best?** Table 3.5 shows results of an experiment to test which depth representations are best for our network to predict. We train networks separately to predict absolute depths (D), surface normals (N), and depth derivatives in 8 directions (DD), and then use different combinations to complete the depth by optimizing Equation 3.1. The results indicate that solving for depths from predicted normals (N) provides the best results (Rel = 0.089 for normals (N) as compared to 0.167 for depth (D), 0.100 for derivatives (DD), 0.092 for normals and derivatives (N+DD). We expect that this is because normals represent only the orientation of surfaces, which is relatively easy to predict [135]. Moreover, normals do not scale with depth, unlike depths or depth derivatives, and thus are more consistent across a range of views.

| B | Rep | Rel↓ | RMSE↓ | 1.05↑ | 1.10↑ | 1.25↑ | $1.25^2$↑ | $1.25^3$↑ |
|---|---|---|---|---|---|---|---|---|
| - | D | 0.167 | 0.241 | 16.43 | 31.13 | 57.62 | 75.63 | 84.01 |
| No | DD | 0.123 | 0.176 | 35.39 | 45.88 | 60.41 | 73.26 | 80.73 |
|  | N+DD | 0.112 | 0.163 | 37.85 | 47.22 | 61.27 | 73.70 | 80.83 |
|  | N | 0.110 | 0.161 | 38.12 | 47.96 | 61.42 | 73.77 | 80.85 |
| Yes | DD | 0.100 | 0.131 | 37.95 | 49.14 | 64.26 | 76.14 | 82.63 |
|  | N+DD | 0.092 | 0.122 | 39.93 | 50.73 | 65.33 | **77.04** | **83.25** |
|  | N | **0.089** | **0.116** | **40.63** | **51.21** | **65.35** | 76.74 | 82.98 |

Table 3.5: **Effect of predicted representation on depth accuracy.** "DD" represents depth derivative, and "N" represents surface normal. We also evaluate the effect of using boundary weight. The first row shows the performance of directly estimating depth. Overall, solving back depth with surface normal and occlusion boundary gives the best performance.

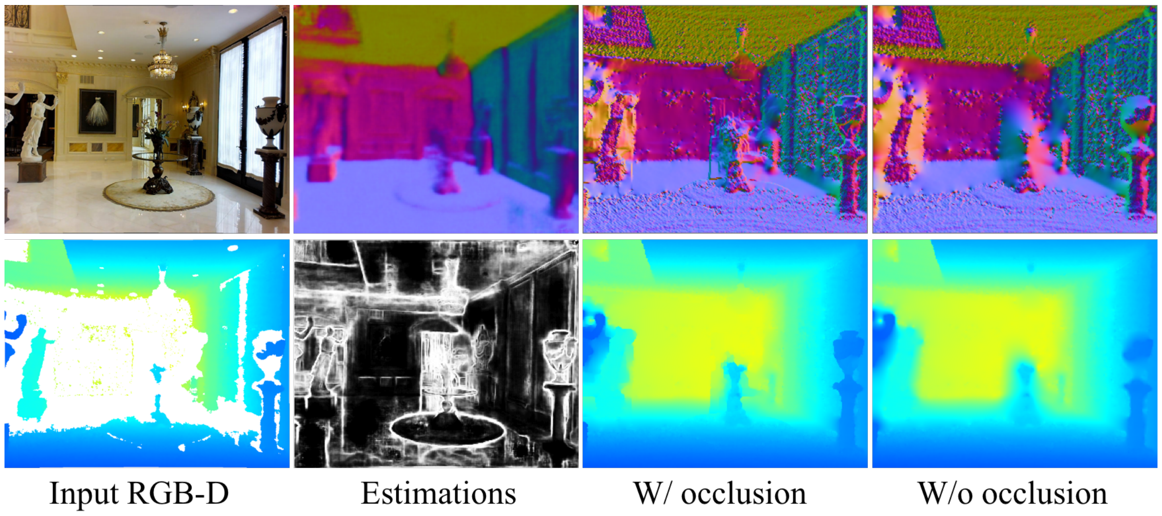


Figure 3.25: **Effect of occlusion boundary prediction on normals.** The 2nd column shows the estimated surface normal and occlusion boundary. The 3rd and 4th column shows the output of the optimization with/without occlusion boundary weight. To help understand the 3D geometry and local detail, we also visualize the surface normal computed from the output depth. The occlusion boundary provides information for depth discontinuity, which help to maintain boundary sharpness.

**Does prediction of occlusion boundaries help?** The last six rows of Table 3.5 show results of an experiment to test whether down-weighting the effect of surface normals near predicted occlusion boundaries helps the optimizer solve for better depths. Rows 2-4 are without boundary prediction ("No" in the first column), and Rows 5-7 are with ("Yes"). The results indicate that boundary predictions improve the results by $\sim$19% (Rel = 0.089 vs. 0.110). This suggests that the network is on average correctly predicting pixels where surface normals are noisy or incorrect, as shown qualitatively in Fig. 3.25.

**How much observed depth is necessary?** Fig. 3.26 shows results of an experiment to test how much our depth completion method depends on the quantity of input depth. To investigate this question, we degraded the input depth images by randomly masking different numbers of pixels before giving them to the optimizer to solve for completed depths from predicted normals and boundaries. The two plots shows curves indicating depth accuracy solved for pixels that are observed (left) and unobserved (right) in the original raw depth images. From these results, we see that the optimizer is able to solve for depth almost as accurately when given only a small fraction of the pixels in the raw depth image. As expected, the performance is much worse on pixels unobserved by the raw depth (they are harder). However, the depth estimations are still quite good when only a small fraction of the raw pixels are provided (the rightmost point on the curve at 2000 pixels represents only 2.5% of all pixels). This results suggests that our method could be useful for other depth sensor designs with sparse measurements. In this setting, our deep network would not have to be retrained for each new dense sensor (since it depends only on color), a benefit of our two-stage approach.

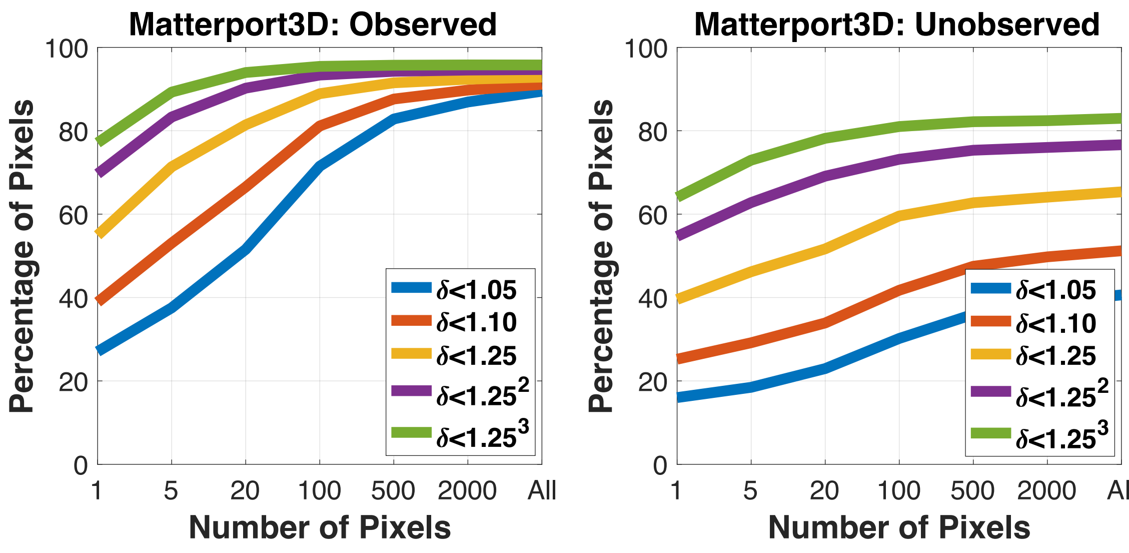


Figure 3.26: **Effect of sparse raw depth inputs on depth accuracy.** The depth completion performance of our method w.r.t number of input pixels with depth. The plot shows that depth estimation on unobserved pixels is harder than the observed. It also shows that our method works well with only a small number of sparse pixels, which is desirable to many applications.

#### 3.3.4.2 Comparison to Baseline Methods

The second set of experiments investigates how the proposed approach compares to baseline depth inpainting and depth estimation methods.

**Comparison to Inpainting Methods** Table 3.6 shows results of a study comparing our proposed method to typical non-data-driven alternatives for depth inpainting. The focus of this study is to establish how well-known methods perform to provide a baseline on how hard the problem is for this new dataset. As such, the methods we consider include: a) joint bilinear filtering [227] (Bilateral), b) fast bilateral solver [8] (Fast), and c) global edge-aware energy optimization [65] (TGV). The results in Table 3.6 show that our method significantly outperforms these methods (Rel=0.089 vs. 0.103-0.151 for the others). By training to predict surface normals with a deep network, our method learns to complete depth with data-driven priors, which are stronger than simple geometric heuristics. The difference to the best of the tested hand-tuned approaches (Bilateral) can be seen in Fig. 3.27.

| Method | Rel↓ | RMSE↓ | 1.05↑ | 1.10↑ | 1.25↑ | $1.25^2$↑ | $1.25^3$↑ |
|---|---|---|---|---|---|---|---|
| Smooth | 0.151 | 0.187 | 32.80 | 42.71 | 57.61 | 72.29 | 80.15 |
| Bilateral [227] | 0.118 | 0.152 | 34.39 | 46.50 | 61.92 | 75.26 | 81.84 |
| Fast [8] | 0.127 | 0.154 | 33.65 | 45.08 | 60.36 | 74.52 | 81.79 |
| TGV [65] | 0.103 | 0.146 | 37.40 | 48.75 | 62.97 | 75.00 | 81.71 |
| Ours | **0.089** | **0.116** | **40.63** | **51.21** | **65.35** | **76.74** | **82.98** |

Table 3.6: **Comparison to baseline inpainting methods.** Our method significantly outperforms baseline inpainting methods.

**Comparison to Depth Estimation Methods.** Table 3.7 shows results for a study comparing our proposed method to previous methods that estimate depth only from color. We consider comparisons to Chakrabarti et al. [23], whose approach is most similar to ours (it uses predicted derivatives), and to Laina et al. [146], who recently reported state-of-the-art results in experiments with NYUv2 [227]. We finetune [23] on our dataset, but use pretrained model on NYUv2 for [146] as their training code is not provided.

Of course, these depth estimation methods solve a different problem than ours (no input depth), and alternative methods have different sensitivities to the scale of depth values, and so we make our best attempt to adapt both their and our methods to the same setting for fair comparison. To do that, we run all methods with only color images as input and then uniformly scale their depth image outputs to align perfectly with the true depth at one random pixel (selected the same for all methods). In our case, since Equation 3.1 is under-constrained without any depth data, we arbitrarily set the middle pixel to a depth of 3 meters during our optimization and then later apply the same scaling as the other methods. This method focuses the comparison on predicting the “shape” of the computed depth image rather than its global scale.

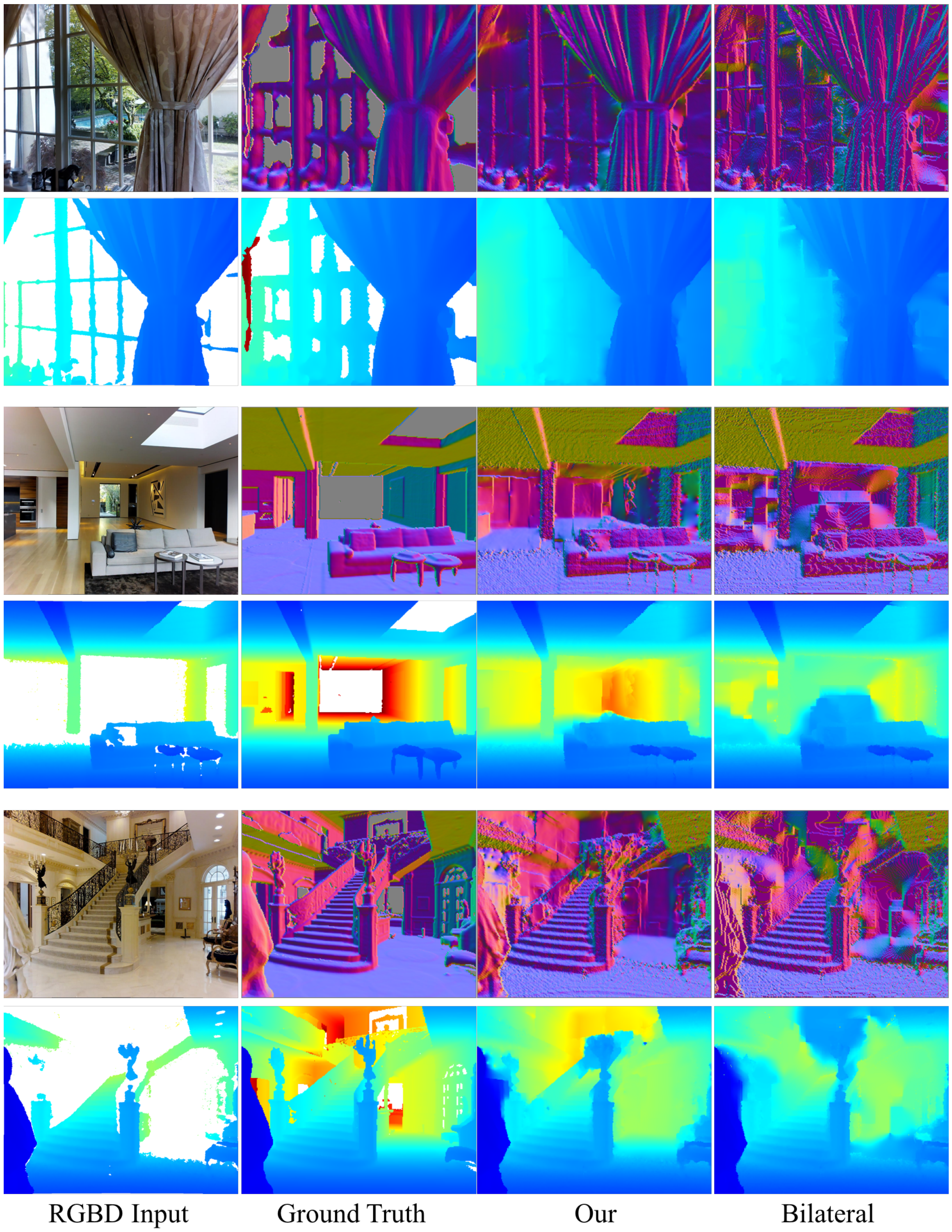


Figure 3.27: **Comparison to inpainting with a joint bilateral filter.** Our method learns better guidance from color and produce comparatively sharper and more accurate results.

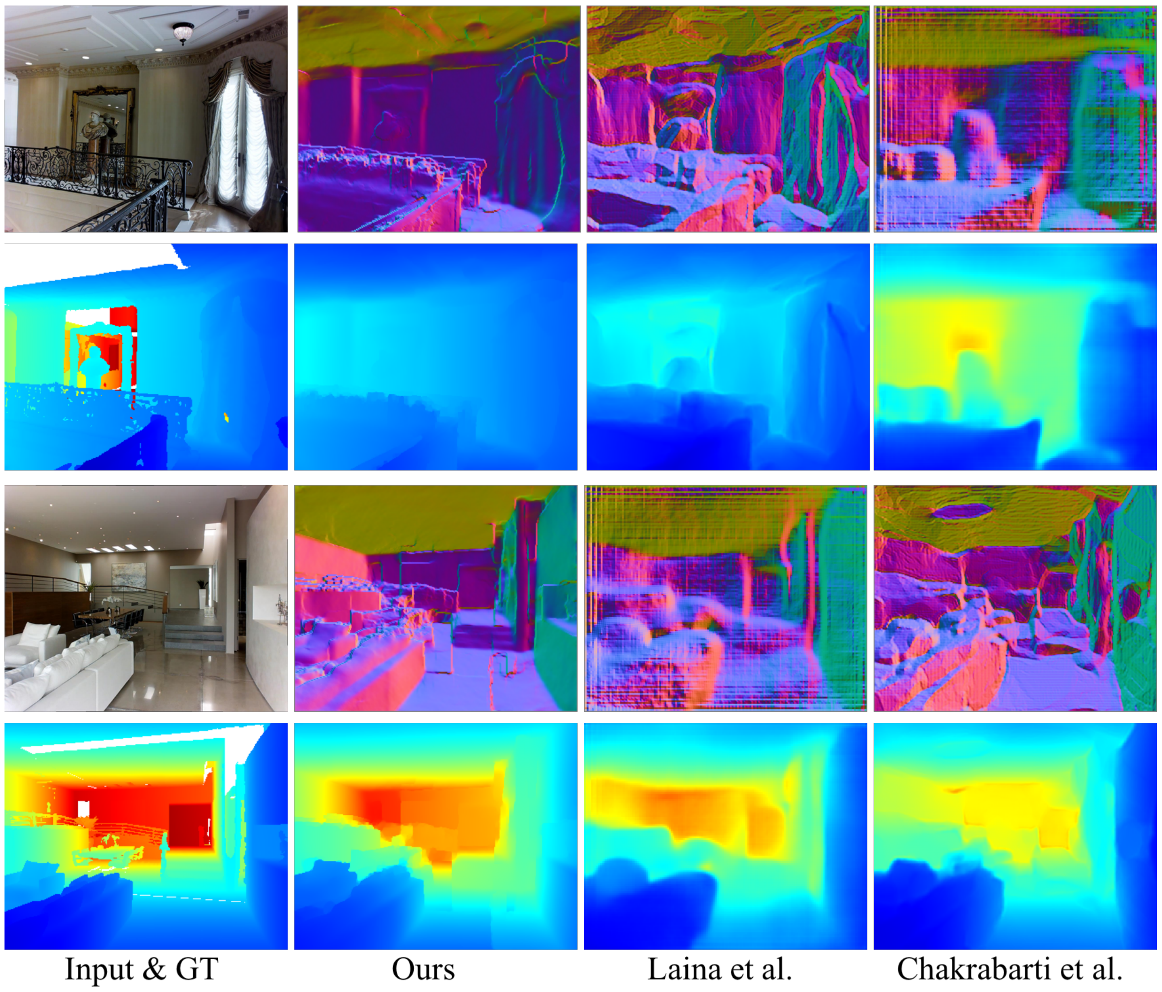


Figure 3.28: **Comparison to deep depth estimation methods.** We compare with the state of the art methods under the depth estimation setting. Our method produces not only accurate depth value but also large scale geometry as reflected in the surface normal.

Results of the comparison are shown in Fig. 3.28 and Table 3.7. From the qualitative results in Fig. 3.28, we see that our method reproduces both the structure of the scene and the fine details best – even when given only one pixel of raw depth. According to the quantitative results shown in Table 3.7, our method is 23-40% better than the others, regardless of whether evaluation pixels have observed depth (Y) or not (N). These results suggest that predicting surface normals is a promising approach to depth estimation as well.

| Obs | Meth | Rel$\downarrow$ | RMSE$\downarrow$ | 1.05$\uparrow$ | 1.10$\uparrow$ | 1.25$\uparrow$ | $1.25^2\uparrow$ | $1.25^3\uparrow$ |
|---|---|---|---|---|---|---|---|---|
| Y | [146] | 0.190 | 0.374 | 17.90 | 31.03 | 54.80 | 75.97 | 85.69 |
|  | [23] | 0.161 | 0.320 | 21.52 | 35.5 | 58.75 | 77.48 | 85.65 |
|  | Ours | **0.130** | **0.274** | **30.60** | **43.65** | **61.14** | **75.69** | **82.65** |
| N | [146] | 0.384 | 0.572 | 8.86 | 16.67 | 34.64 | 55.60 | 69.21 |
|  | [23] | 0.352 | 0.610 | 11.16 | 20.50 | 37.73 | 57.77 | 70.10 |
|  | Ours | **0.283** | **0.537** | **17.27** | **27.42** | **44.19** | **61.80** | **70.90** |

Table 3.7: **Comparison to deep depth estimation methods.** We compare with Laina et al. [146] and Chakrabarti et al.[23]. All the methods perform worse on unobserved pixels than the observed pixels, which indicates unobserved pixels are harder. Our method significantly outperform other methods.

### 3.3.5 Conclusion

This chapter describes an example of leveraging single image based scene understanding to enhance vision tasks. A deep learning framework is proposed to complete the depth channel of an RGB-D image acquired with a commodity RGB-D camera. It provides two main research contributions. First, it proposes to complete depth with a two stage process where surface normals and occlusion boundaries are predicted from color, and then completed depths are solved from those predictions. Second, it learns to complete depth images by supervised training on data rendered from large-scale surface reconstructions. During tests with a new benchmark, we find the proposed approach outperforms previous baseline approaches for depth inpainting and estimation.

# Chapter 4

# Holistic Scene Understand

In this chapter, we study the scene understanding using holistic representation. In Chapter 4.1, we show how to produce bounding box scene understanding from a single color panorama and demonstrate the importance of scene context for 3D object detection and room layout estimation [296]. In Chapter 4.2, we propose a deep learning system that produce bounding box representation directly with context encoded explicitly [293].

## 4.1 PanoContext: A Whole-room 3D Context Model for Panoramic Scene Understanding

### 4.1.1 Introduction

Recognizing 3D objects from an image has been a central research topic since the computer vision field was established [208]. While the past decade witnesses rapid progress on bottom-up object detection methods [63, 53, 255, 263, 78], the improvement brought by the top-down context cue is rather limited, as demonstrated in standard benchmarks (e.g. PASCAL VOC[53]). In contrast, there are strong psychophysical evidences that context plays a crucial role in scene understanding for humans [16, 249].

We believe that one of the main reasons for this gap is because the field of view (FOV) for a typical camera is only about 15% of that of the human vision system. The approximate FOV of a single human eye is about 95°. Two eyes give us almost 180° FOV. Considering the movement of eyeballs (head rotation excluded, peripheral vision included), the horizontal FOV of the human vision system is as high as 270°. However, the FOV of a typical camera is much smaller. For example, on standard full-frame cameras, the horizontal FOV is only 39.6° (or 54.4°) with a standard 50mm lens (or with a 35mm wide-angle lens). This problem is exemplified in Fig. 4.1. The narrow FOV hinders the context information in several ways. Firstly, a limited FOV sees only a small fraction of all scene objects, and therefore, observes little interplay among them. For example, on average, there is only 1.5 object classes and 2.7 object instances per image in PASCAL VOC. Secondly, the occurrence of an object becomes unpredictable with a small FOV. For example, a typically bedroom should have at least one bed, which can serve as a strong context cue. But in a bedroom picture of small FOV (Fig. 4.1), there might or might not be a bed, depending on the direction the camera looks at. Given a much limited FOV, it is unfair to ask computer

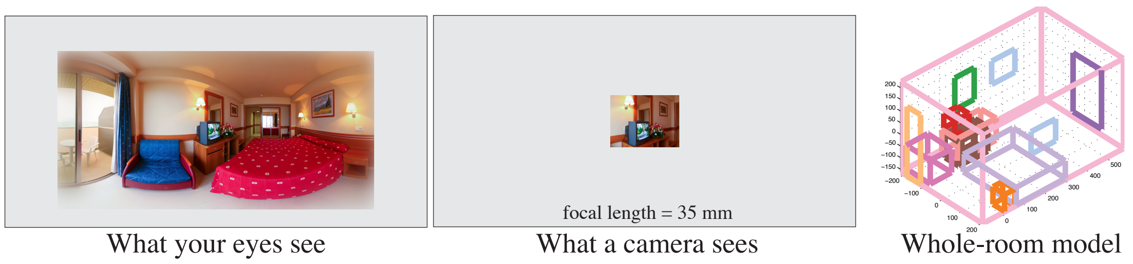


Figure 4.1: **Comparison of field-of-view.** A camera with narrow FOV might not see a bed in a bedroom which complicates the context model.

vision algorithms to match the performance of human vision. Therefore, we advocate the use of panoramic images in scene understanding, which nowadays can be easily obtained by camera arrays (e.g. Google Streetview), special lenses (e.g. `0-360.com`), smartphones (e.g. `cycloramic.com`) or automatic image stitching algorithms (e.g. [168, 20, 21]).

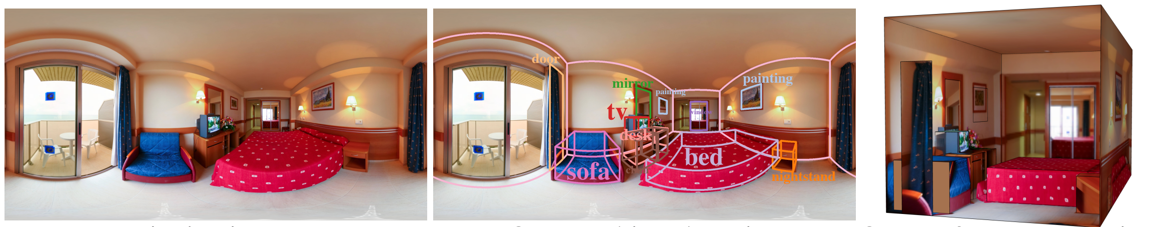
We present a whole-room 3D context model to address the indoor scene understanding problem from a single panorama (Fig. 4.2) [296]. In a panorama, characteristic scene objects such as beds and sofas are usually visible despite occlusion, so that we can jointly optimize the room layout and object detection to exploit the contextual information in its full strength. Our output is a 3D room layout with recognized scene objects represented by their 3D bounding boxes. We use context to decide number of instances, to assign object semantics, to sample objects, to valid whole-room hypotheses, to extract room model feature, to reconstruct 3D and to adjust final result. An example of input and output are provided in Fig. 4.2.

Our method consists of two steps: bottom-up hypotheses generation and holistic hypotheses ranking. It starts by generating hypotheses for the room layout and object bounding boxes in a bottom-up fashion using a variety of image evidences, e.g. edge, segmentation, orientation map and geometric context. 3D scene hypotheses are formed from these hypothesized room layouts and object bounding boxes. A trained Support Vector Machine (SVM) [54] ranks these 3D scene hypotheses and chooses the best one. Finally, we locally refine good hypotheses to further maximize their

SVM scores. The SVM is trained utilizing both image information and room structure constrains from our training data, which consists of high-resolution panorama images with detail object annotations and 3D ground truth reconstructed using the 2D annotations.

The whole-room contextual information is critical in many key steps of our system. During *hypothesis generation*, the object categories are predicted based on its relative location in the room. We sample the number of object instances according to the typical distribution of each object category, guided by the pairwise position relationship among objects. During *hypothesis ranking*, we firstly align each hypothesis with the 3D rooms from the training set to tell if it is valid. This non-parametric room alignment captures high order relationship among all objects, which cannot be represented well by pairwise constraints. Secondly, we also build a room model for each hypothesis. This room model includes color and texture statistics for the foreground and background. Since we know all the objects and room layout in 3D, we can calculate these statistics easily from image regions unoccluded by objects.We use this model to judge how well a hypothesis explains the image evidences. Thirdly, because we map each hypothesis to 3D space by reconstructing the objects and room layouts, a wrong room layout hypothesis is typically ranked low by the SVM, since it often produces unreasonable 3D bounding boxes of objects. This implicit 3D interaction between objects and room layout enables us to identify many bad hypotheses. During *final adjustment*, we also use the object number distribution and pairwise context model to guide the search.

As demonstrated in our experiments, we can recognize objects using only 3D contextual information (without a classifier to discriminate object categories based on image feature), and still achieve a comparable performance with the state-of-the-art object detector [63], which learns a mapping from image region feature to object category. This shows that context is as powerful as object appearance and much more



Input: a single-view panorama Output: object detection Output: 3D reconstruction

Figure 4.2: **Input and output.** Taken a full-view panorama as input, our algorithm detects all the objects inside the panorama and represents them as bounding boxes in 3D, which also enables 3D reconstruction from a single-view.

useful than we previously thought. The root of context model being under-utilized is partly because the regular FOVs are too small.

In the following section, we will describe our algorithm in greater details. We will also talk about the construction of a 3D panorama data set and present experiments to evaluate the algorithm in Sec. 4.1.4. In Sec. 4.1.2, we will discuss the relation of our proposed method with existing ones.

### 4.1.2 Related Work

There are many exceptional works that inspired the design of our algorithm. The surface orientation and depth estimation from a single image is studied in [39, 34, 106, 218, 107, 110, 109, 108]. The state-of-the-art of single view room layout estimation can be found in [100, 149, 85, 91, 300, 259, 285, 101, 148, 200, 284, 199, 102, 222, 272, 83, 214, 215, 32, 38, 301, 221, 223, 26, 71]. Our work extends them to full-view panorama to fully exploit the contextual information. There are also many great works that model context and object relations [204, 252, 31, 30, 29, 41, 145, 238, 237, 234, 236, 235] and parse a scene [151, 103, 91] in a unified way. Although they have some success on reasoning about 3D, their main focus is still on 2D, while our context model is fully in 3D. For scene parsing grammar, several approaches such as And-Or graph, stochastic grammar, or probabilistic languages have been proposed [265, 9, 245, 176, 91, 253, 153, 154]. Our data-driven sampling and discriminative training provides a simple but powerful way to combine the bottom-up image evidence and top-down context information. Same with our assumptions, 3D cuboids are also a popular representation for scene understanding [100, 274, 101, 301, 221, 157, 66, 270]. For object recognition datasets, there are several main-stream datasets that contain object annotation in regular pictures [53, 269, 213, 271, 10, 40, 226, 212]. Our panorama dataset is the first annotated panorama dataset for scene understanding, and we also provide ground truth in 3D. For using panoramas in computer vision tasks, there are several projects focus on scene viewpoint recognition, localization, image extrapolation and warping [269, 192, 298, 99]. Recently, the rapid increase of popularity of RGB-D sensors enables many seminar works on scene understanding in 3D [230, 266, 84, 87, 227, 120, 133, 290, 119, 303, 157]. We expect our approach can also be naturally extended into RGB-D panoramas or RGB-D scanned 3D rooms [273].

### 4.1.3 Method

As shown in Fig. 4.2, our input is a panorama covering 360° horizontal and 180° vertical FOV represented in equirectangular projection. Our output is a 3D box representation of the sceneWe adopt the Manhattan world assumption, assuming that the scene consists of 3D cuboids aligned with three principle directions[1].

Our method first generates whole-room hypotheses and then ranks them holistically. The challenge for hypotheses generation is to maintain high recall using a managable number of hypotheses, while the challenge for holistic ranking is to have high precision. To generate hypotheses, we first estimate vanishing points by Hough Transform based on the detected line segments (Sec. 4.1.3.1). We then generate 3D room layout hypotheses from line detections and verify them with the computed

[1]We focus on indoor scenes only, although our algorithm may be generalized to outdoor scenes as well. We assume an object can either stand on the ground, sit on another object, or hang on a wall (i.e. no object floats in space). We also assume that the height of camera center is 1.6 meters away from the floor to obtain a metric 3D reconstruction.

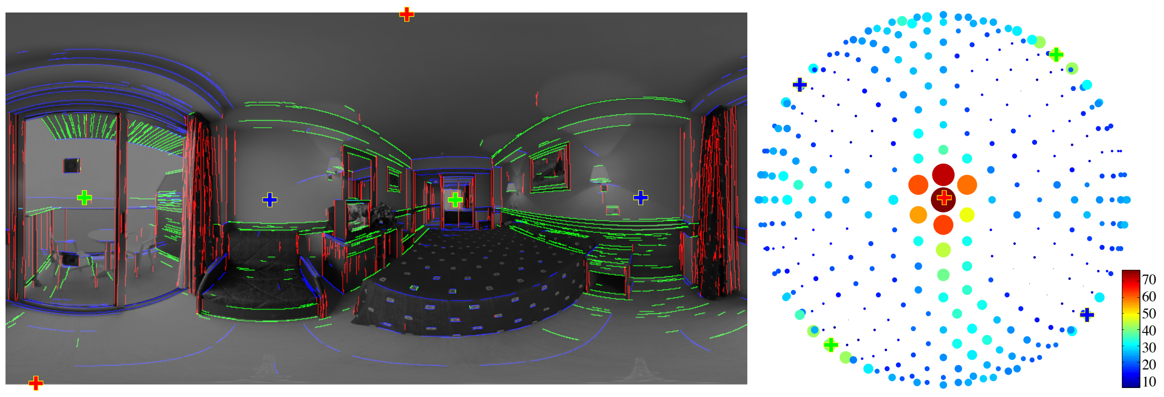


Figure 4.3: **Hough transform for vanishing point detection.** The **left** image shows the detected edges and vanishing points. The colors indicate the edge directions. The **right** image shows the votes on each bin of the half sphere. The sizes and colors both indicate the number of votes.

geometric context and orientation map on the panorama (Sec. 4.1.3.2). For objects, we generate 3D cuboid hypotheses using rectangle detection and image segmentation (Sec. 4.1.3.3). Next, we use sampling to generate full scene hypotheses, each of which has a 3D room and multiple 3D objects inside (Sec. 4.1.3.4). To choose the best hypothesis that is supported by the image evidence and structurally meaningful (i.e. satisfying all context constraints), we extract various features and train a SVM to rank these hypotheses holistically (Sec. 4.1.3.5). Finally, we locally adjust the top hypothesis and search for a solution that maximizes the SVM score by adding, deleting and swapping an object.

#### 4.1.3.1 Vanishing point estimation for panoramas

We detect line segments on the panorama and use them to vote for the vanishing directions (Fig. 4.3). To take full advantage of previous line segment detection works on standard photo,we convert a panorama image to a set of perspective images, and run the state-of-the-art Line Segment Detection (LSD) algorithm [258] in each perspective image, and warp all detected line segments back to the panorama.

A line segment in 3D space corresponds to a section of a great circle on a sphere, and displays as a curve in panorama. For each line $\mathbf{l}$, we use $\mathbf{n}$ to denote the normal of the plane where its great circle lies in. The vanishing direction $\mathbf{v}$ associated with the line $\mathbf{l}$ should be perpendicular to $\mathbf{n}$. We use a hough transform [112] to find all vanishing directions. We uniformly divide the unit sphere into bins by recursively dividing triangles of a icosahedron. A line segments $\mathbf{l}$ will vote for a bin whose center $\mathbf{n}_b$ satisfies $\mathbf{n}_b \cdot \mathbf{n} = 0$. We then find three mutually orthogonal bins with maximal sum of votes as three vanishing directions. After that, we snap all line segments to align with their vanishing directions.

#### 4.1.3.2 Room layout hypothesis generation

Because the room layout is essential to generate good object cuboid hypotheses in 3D, we first obtain some good room layouts to reduce the burden of 3D object detection in the next step. We randomly generate many room layout hypotheses and keep those consistent with a pixel-wise surface normal direction map estimated on panorama.

A 3D room layout can be generated by sampling line segments as room corners [100]. Geometrically, five lines determine a cuboid in 3D space except some degenerative cases. We classify each line segment with two labels from top/bottom, front/back, and right/left according to its association to the vanishing directions, and randomly sample five non-degenerative lines to form a room layout hypothesis. To reduce the number of hypotheses while keeping the good ones, we use the surface normal consistency with a pixel-wise surface direction estimation from panorama to rank these hypotheses and choose the top 50 (since the recall starts to saturate around 50 in Fig. 4.11).

Orientation Map (OM) [149] and Geometric Context (GC) [100] provide pixel-wise surface normal estimation for ordinary perspective images. We convert a panorama into several overlapping perspective images, and apply OM and GC on these images

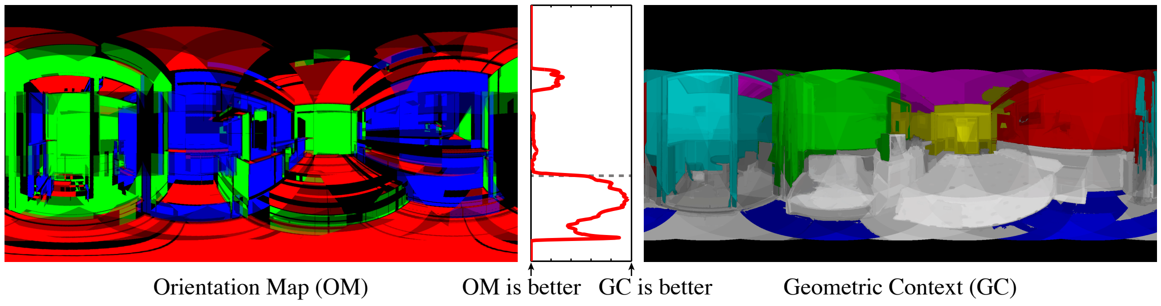


Figure 4.4: **OM vs. GC.** Here shows the OM and GC for the panorama image in Fig. 4.1. The curve at the center shows the accuracy comparison between OM and GC as the vertical view angle changes (data is from all training images). We can clearly see that OM is better at the upper part while GC is better at the lower part, and there is a clear threshold to combine them.

respectively and project results back to the panorama. From our training data with manually marked ground truth wall orientations, we observe that GC provides better normal estimation at the bottom (probably because the model was trained using images looking slightly downwards), and OM works better at the top half of an image (probably less cluttered.), as shown in Fig. 4.4. Therefore, we combine the top part of OM and the bottom part of GC to evaluate the room layout. As can be seen from Fig. 4.11(left), the recall rate is significantly improved by combining OM and GC. Fig. 4.7 shows some good room layout hypotheses, which are generally very close to the ground truth.

#### 4.1.3.3 3D object hypotheses generation

After generating a set of good 3D room layout hypotheses, the next step is to generate 3D cuboid hypotheses for major objects in the room. To obtain high recall for hypotheses generation, we use two complementary approaches: a detection-based method to apply a 2D rectangle detector to the projections of a panorama along the three principle directions, and a segmentation-based method to segment the image and fit a 3D cuboid to each segment by sampling its boundaries.

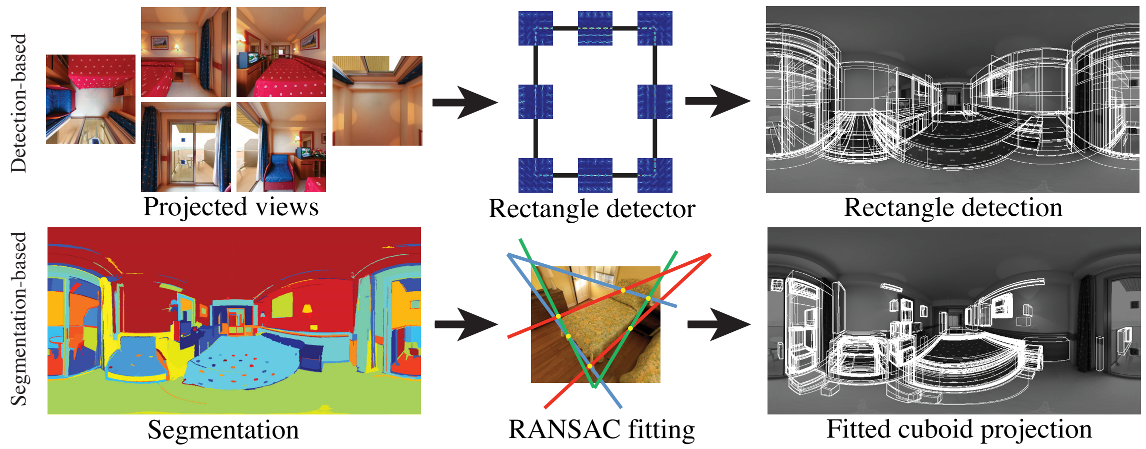


Figure 4.5: **Two ways to generate object hypotheses:** detection and segmentation.

**Detection-based cuboid generation:** We project the input panorama orthographically to six axis-aligned views, and run a rectangle detector in each projection respectively (Fig. 4.5(top)). Our rectangle detector is similar as Deformable Part Model [63] but without spring-like constraints. We define a part at each corner and the middle of each edge of the rectangle. We use the SUN primitive dataset [274] containing 382 annotated cuboid images, and transform each cuboid surface to an axis aligned rectangle to train each part detector independently. During testing, we first compute the response maps of all part detectors, and sum up them according to the models. We set a low threshold to ensure high recall.We then generate cuboid hypotheses from the 3D rectangles.

**Segmentation-based cuboid generation:** Some scene objects, such beds and sofas, do not have strong edges, and cannot be reliably detected by the rectangle detection. Therefore, we generate additional cuboid hypotheses from image segmentation (Fig. 4.5(bottom)) by selective search [255]. Specifically, for each segment, we evaluate how well its shape can be explained by the projection of a cuboid. We create many cuboids by randomly sampling 6 rays at the segment boundary passing through the three vanishing points. Among these cuboids, we choose the best one whose projection has the largest intersection over union score with the segment.

#### 4.1.3.4 Whole-room scene hypotheses generation

After obtain a hypothesis pool for room layout and objects, we generate a pool of whole-room hypotheses, each consisting of a room layout with several cuboid objects inside. To achieve high recall with a manageble number of hypotheses, we classify the semantic meaning of each cuboid and use pairwise context constraints to guide our sampling.

**Semantic label:** Intuitively, the semantic object type is strongly correlated with the cuboid shape and its 3D locations in the room. We train a random forest classifier to estimate the semantic type of a cuboid according to its size, aspect ratio and relative position in the room. And we achieve the multiple-label classification accuracy at around 70% This shows that the context between room and objects is very strong.

**Pairwise constraint:** There are strong pairwise context constraints between scene objects. For instance, nightstands are usually nearby a bed, and a TV set often faces a bed. For two object types, we collect all instances of the pair $\{\langle \mathbf{p}_1, \mathbf{p}_2 \rangle\}$ from our training database. We register all these instances by a rotation in the ground plane to ensure the closest wall to $\mathbf{p}_1$ is on its left. We then take the displacement vector $\mathbf{p}_1 - \mathbf{p}_2$ as a sample, and capture the pairwise location constraint by all collected samples. Such a set of samples are plotted in Fig. 4.6(a). When testing the validity of a pair of objects, we compute their displacement vector and search for the $K$ nearest neighbors in the sample set. The mean distance to the $K$ nearest neighbors will be transferred to a probability by a sigmoid function.

**Whole-room sampling:** We generate a complete scene hypothesis as following,

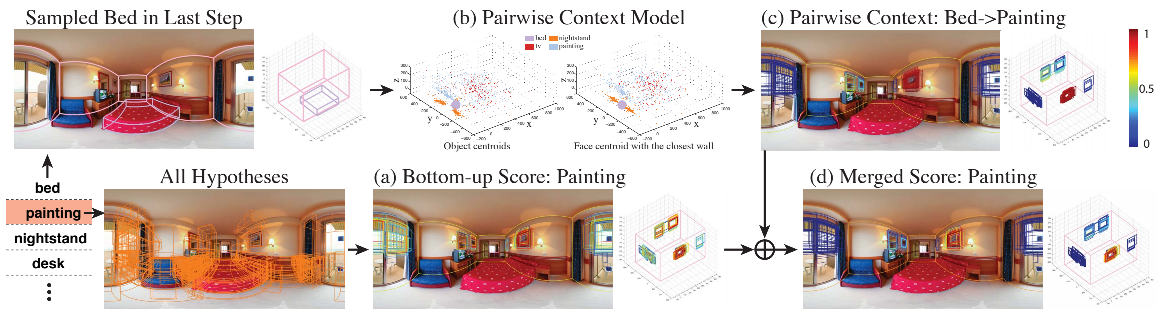

Sampled Bed in Last Step
(b) Pairwise Context Model
(c) Pairwise Context: Bed->Painting
Object centroids
Face centroid with the closest wall
bed
painting
nightstand
desk
All Hypotheses
(a) Bottom-up Score: Painting
(d) Merged Score: Painting
1
0.5
0

Figure 4.6: **Example sampling pipeline.** Here we show an example of sampling a painting guided by context, given that a bed is already sampled in previous steps. (a) the bottom-up scores of painting hypotheses, (b) pairwise context statistics to show how objects of different categories locates around a bed, (c) the pairwise context constraint from the sampled bed, (d) the scores for merging bottom-up scores and pairwise context.

1. Randomly select a room layout according to their scores evaluated by GC and OM (higher score with a higher probability).

2. Decide the number of instances for each object type and the sampling order for different object types according to statistic prior. Fig. 4.6 shows an example of order list on left side.

3. Start from the first object. Search for cuboids of the selected object type, and randomly choose a cuboid according to bottom up score, e.g. rectangle detection score, semantic classifier score. Hypothesis with higher score would be sampled with higher probability.

4. Go to the next object, we combine the bottom-up scores and the pairwise context constraint with all the previously selected objects. A new object will be randomly selected according to merged scores. For example, the unary bottom-up score is effective in pruning invalid hypotheses (Fig. 4.6(a)), and pairwise score can further enhance it (Fig. 4.6(c)). As shown in Fig. 4.6(d), the rectangles on head of the bed are further highlighted, and those on windows are

depressed. We can see the hypotheses around the true painting are all with high score and thus we have a higher chance to get a correct object.

5. Given all the sampled object so far, repeat the previous step until all the instances have been sampled.

Comparing with completely random sampling, our method can avoid obviously unreasonable scene hypotheses, and thus ensure high recall with a managable number of samples. Fig. 4.7 shows some sampling results.

#### 4.1.3.5 Data-driven holistic ranking

After generating a large list of whole-room hypotheses, we train a SVM model to rank them and choose the best hypothesis, holistically for the whole-room.

**Linear SVM:** Our goal is to learn a mapping from a panorama $\mathbf{x}$ to a scene parsing result $\mathbf{y}$. Because $\mathbf{y}$ is a structural output, we formulate the problem as a 0-1 loss structural SVM [121], i.e. a binary linear SVM[2]. We define a feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ for a panorama $\mathbf{x}$ and its hypothesis $\mathbf{y}$. The binary label $l$ indicates whether $\mathbf{y}$ is close enough to the manually annotated ground truth $\mathbf{y}^*$, i.e. $l = [\Delta(\mathbf{y}, \mathbf{y}^*) < \epsilon]$. During training, for each panorama $\mathbf{x}_n$, we sample $M$ hypotheses $\{\mathbf{y}_n^m\}^{m=1,\cdots,M}$. We use all $N$ panoramic images from our training set to train the binary SVM by $MN$ pairs of $\{\langle \mathbf{f}(\mathbf{x}_n, \mathbf{y}_n^m), l_n^m \rangle\}_{n=1,\cdots,N}^{m=1,\cdots,M}$. Since we typically have hundreds of panoramas and thousands of hypotheses, there are about a million training data for the SVM[3]. During testing, the SVM score is used to choose the best hypothesis with maximal SVM score.

[2]We use a 0-1 loss structural SVM because the ranking among the bad hypotheses is unimportant, and we only want to find the best one. Our experiment also shows that it is very slow to train a general structural SVM using the standard cutting plane algorithm, and the general structural SVM is very sensitive to the loss function. A 0-1 loss structure SVM is basically a binary linear SVM that can be trained efficiently and is robust to the loss-function.

[3]To learn the SVM efficiently with less restriction on computation memory, we use all the positive data and randomly choose a subset of negative data to train an initial version of the SVM. We then repeatedly test the SVM on the remaining negative samples, and add the false positives into the SVM training, until there are no hard negatives (similar to hard negative mining in object detection [63]).

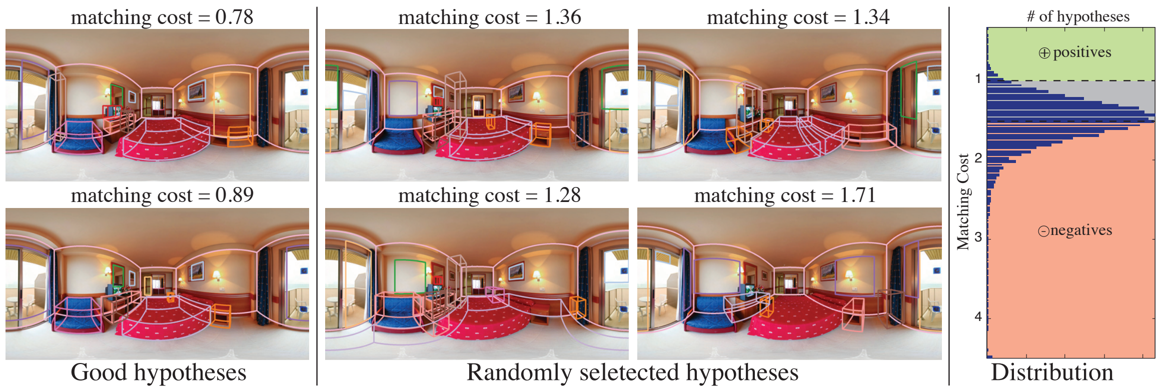


Figure 4.7: **Whole-room hypotheses.** On the left we show some good hypotheses selected based on matching cost. At the center we show some random hypotheses to visualize the hypothesis space.

**Matching cost:** $\Delta(\mathbf{y}, \mathbf{y}^*)$ measures the difference between a whole-room hypothesis $\mathbf{y}$ and its ground truth $\mathbf{y}^*$. We first register the two scenes by matching their vanishing directions and room centers. We compute the the average 3D distance between corresponding vertices between pairs of cuboids of the same semantic type, one from each scene. We search for bipartite matching minimal distance for each semantic label. $\Delta(\mathbf{y}, \mathbf{y}^*)$ is the sum of all bipartite matching distances plus the constant penalty for unmatched cuboids in both scenes. We use this score to decide the labels for the data to train the SVM. Because it is hard to find a good threshold, we choose two conservative thresholds to make sure all positives are good and all negatives are bad. We drop all other data in between as we cannot tell their quality reliably.

**Holistic features:** The feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a concatenation of object level feature $\mathbf{f}^{\text{object}}$ and room level feature $\mathbf{f}^{\text{room}}$. Thus, it encodes both bottom-up image evidence and top-down contextual constraints. The relative importance between all the information is learned by the SVM in a data-driven way using the training data. $\mathbf{f}^{\text{object}}$ measures the reliability of each single object. On each object hypothesis, rectangle detector score, segmentation consistency score, sum/mean/std on each channel of OM and GC, entropy of color name, and 2D projected size (area on panorama sphere) will be extracted and concatenated into a column vector. For each object category, we take the sum/mean/max/min features of all instances in the category to form the feature vector. For categories with no object, we set the features zeros. We concatenate the features for all object categories to a single vector as $\mathbf{f}^{\text{object}}$ to form the holistic feature. Since the number of categories is fixed, the total dimension is also fixed.

**Non-parametric room alignment:** The room level feature $\mathbf{f}^{\text{room}}$ checks whether the hypothesized structure, i.e. the room layout and the arrangement of all objects, can be found in reality. We propose a non-parametric data-driven brute-force context model by aligning a hypothesis with the 3D rooms from the training set. Specifically, for a hypothesis $\mathbf{y}$, we would match it with all manual annotations $\{\mathbf{y}_1^*, \mathbf{y}_2^*, \cdots, \mathbf{y}_N^*\}$ in the training set. After registering two scenes, we can efficiently compute the distances between all pairs of cuboids in the two scenes. The distance is defined as a combination of center distance, volume intersection over union, and semantic type consistency. Since our training data has limited size, we apply various transformations $\mathbf{T}$ on the ground truth rooms to increase the database diversity. Specifically, we increase/decrease the size of the room, while keep the relative positions of all objects in room unchanged, or keep their absolute distance to a wall fixed. We further allow a universal scaling on the whole scene. The room level feature $\mathbf{f}^{\text{room}}$ is defined as the 10 smallest matching costs $\Delta(\mathbf{y}, \mathbf{T}(\mathbf{y}_n^*))$ between a hypothsis $\mathbf{y}$ and these transformed rooms from the ground truth $\{\mathbf{T}(\mathbf{y}_n^*)\}$, and the accumulated sums and products of these 10 numbers.

**Room-only color model:** To consider all the objects together, we divide image region to foreground (pixels covered by objects) and background (other pixels). In each regions, we extract the same feature defined in $\mathbf{f}^{\text{object}}$. This provides context information integrating both bottom-up and top-down information, and it is part of $\mathbf{f}^{\text{room}}$.

**Local adjustment:** The top hypothesis returned by the SVM could be limited by the size of our hypothesis pool. So we apply a local refinement to the SVM returned hypothesis for a result with higher SVM score. Specifically, we can delete, add, or swap an object using the pairwise context constraints, or completely re-sample an object. If this generates a result with higher SVM score, we will accept the new result and perform local refinement again nearby the new result.

### 4.1.4 Experiments

#### 4.1.4.1 Annotated 3D panorama dataset

We collected 700 full-view panoramas for home environments from SUN360 database [269], including 418 bedrooms and 282 living rooms. We split our dataset into two halves for training and testing respectively.The data is manually annotated in house by five persons. After that, an author went through each image to correct mistakes and ensure consistency among the annotation.

To annotate panorama, we designed a WebGL annotation tool in browser, which renders a 360° panorama as a texture wrapped inside a sphere with the camera located at the center (Fig. 4.8). To annotate an object, the user first chooses one of the nine predefined viewpoints of a cuboid (shown in the black box in Fig. 4.8), because a different viewpoint requires a different set of vertices to be specified. When the user is marking these vertices, the interface will highlight the corresponding vertex on a 3D cuboid on the right. We ask the annotator to mark the 3D bounding box as tight as possible and align it with major orientations of the object. A key novelty of our tool is to first let the user to choose one of the nine predefined viewpoints for each cuboid object, and click each visible vertices guided by the instruction. We found

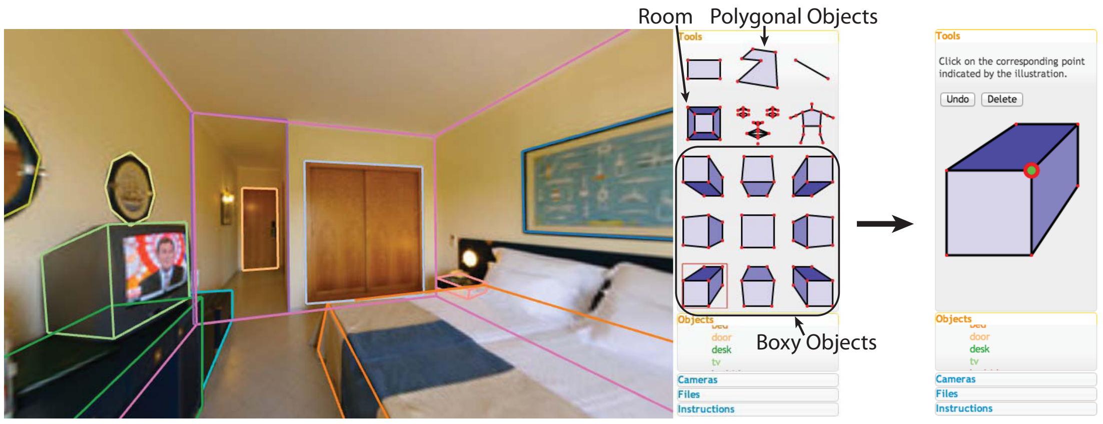


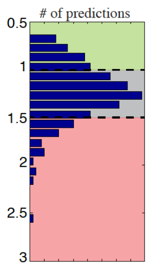
Figure 4.8: **Panoramic annotation tool.** To annotate a 3D cuboid, the user picks a viewpoint from [Tools], and clicks on the key points of the 3D bounding boxes on the panorama. The screen displays an indication about what is the next corner to click on, as shown on the right.

that this interface is much easier to use than [274], where the viewpoint is implicit. For rectangular objects (e.g. painting), we annotate its four corners using a polygon tool. To label the room layout, we design a specialized primitive to ask the user to click the eight corners of the room.

We further convert 2D annotations to 3D scene models. Assuming each object is a perfect cuboid and can only rotate horizontally around a vertical axis. The task of generating 3D scene models amounts to find the parameters for each scene object to minimize the reprojection error between the 3D cuboid and annotations. Furthermore, we enforce each object can only be at one of the following positions, standing on the ground, sitting on another object, or attaching to a wall. We formulate all these constraints to a single non-linear optimization to minimize reprojection errors from all objects.

### 4.1.4.2 Evaluation

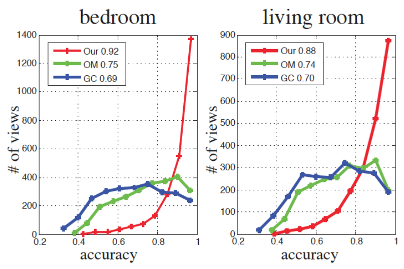
Some results for both bedroom and living rooms are shown in Fig. 4.10, where we can see that the algorithm performs reasonably.



(a) Cost

| category | accuracy | category | accuracy |
|---|---|---|---|
| background | 86.90 | wardrobe | 27.44 |
| bed | 78.58 | tv | 34.81 |
| painting | 38.70 | door | 19.40 |
| nightstand | 39.66 | chair | 9.61 |
| window | 35.58 | sofa | 11.10 |
| mirror | 38.15 | cabinet | 5.46 |
| desk | 29.55 | average | 35.00 |

(b) Semantic segmentation accuracy (bedroom)



(c) Compare with regular FOV.

Figure 4.9: **Evaluation.** (a) shows the matching cost distribution for the top hypotheses of our results (for bedroom). (b) shows the accuracy for semantic segmentation. (c) shows the distribution of views across different surface orientation prediction accuracy (see the supp. material for more).

**Matching cost to the ground truth:** The most straightforward way for evaluation is to compare the prediction with the ground truth, using the matching cost that we defined to choose label for the SVM training in Sec. 4.1.3.5. The average matching cost is 1.23, which is much better than the pure bottom-up hypotheses generation (average cost is 1.55). We show the histogram for the distributions of the matching cost in Fig. 4.9(a).

**Semantic image segmentation:** We also covert the 3D understanding results into a semantic segmentation mask on the panorama images and compare the results with the ground truth as in PASCAL VOC segmentation challenge [53]. During conversion, we use the 3D bounding boxes of the objects to create a mask with occlusion testing. To avoid the artifact of panorama projection, instead of comparing panorama segmentation results, we uniformly sample rays of different orientation on the sphere and compare the label of the prediction and ground truth. Fig. 4.9(b) shows the accuracy.

## 4.1.5 How important is larger FOV and context?

To justify the importance of FOV for context model, we conduct five types of comparison. First, we show that a larger FOV provides stronger context to recover room

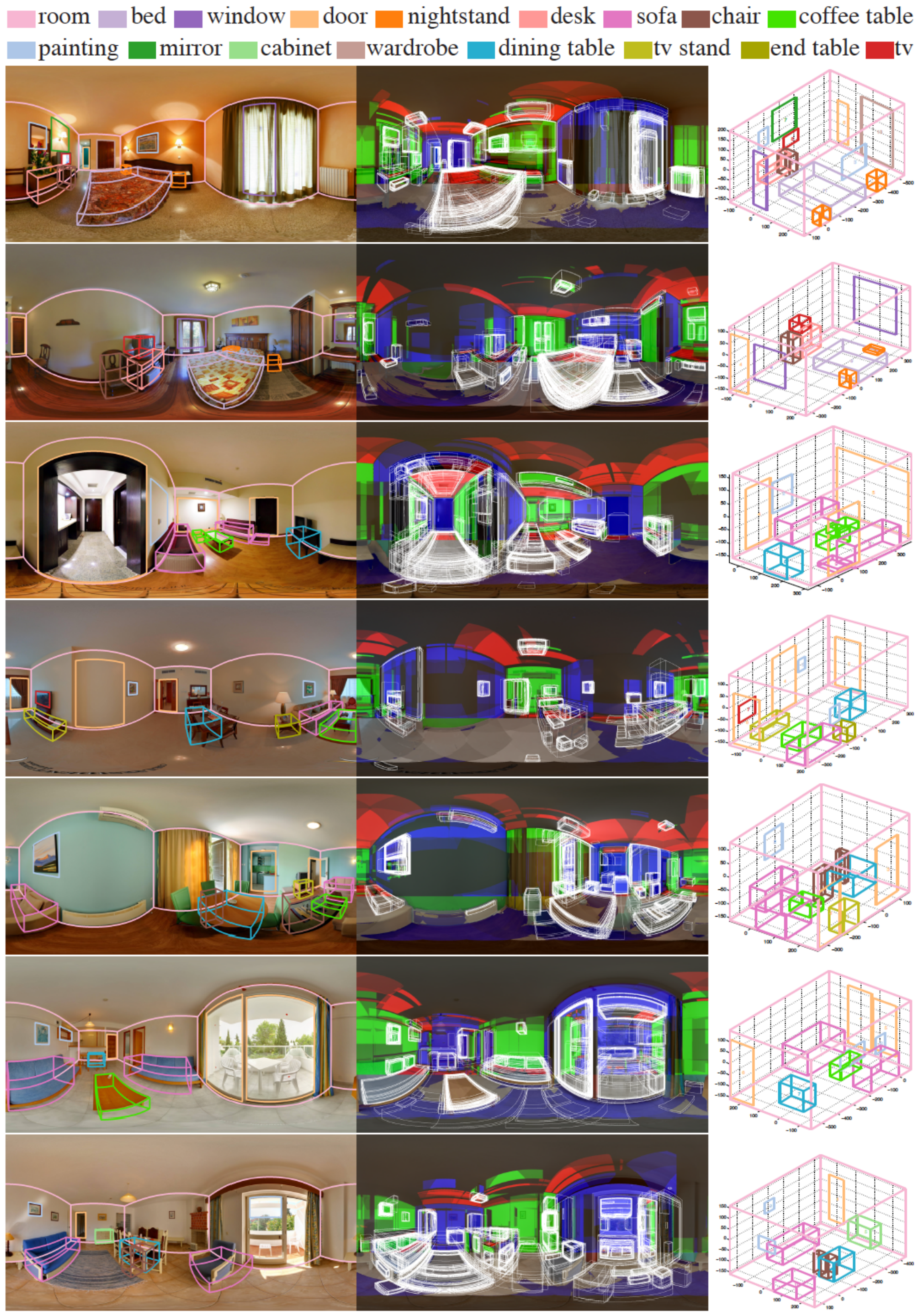


Figure 4.10: **Example results.** The first column is the input panorama and the output object detection results. The second column contains intermediate steps for generating cuboid hypotheses from bottom-up sampling as well as the combination of OM and GC. The third column is the results visualized in 3D. (Best view in color. More results are available in the supplementary material.)

layout.Second, we show that full-room context provides stronger cue for recognizing objects than an object detector, which classifies an image patch (i.e. small FOV). Third, we decompose our system and disable some key usages of global context information, to see the importance of context during sampling and ranking. Fourth, we vary the effective range of FOV in the context model to demonstrate the larger FOV enables stronger context. Finally, we combine our context model with standard object detectors, to demonstrate the complementary natural of context and local object appearance.

**Is larger FOV helpful for room layout estimation?** To demonstrate the importance of panorama, we compare our algorithm with [149] and [100] on regular field-of-view images. We warp the panorama images into perspective images using $54.4^\circ$ FOV, run [149] and [100] on these images to obtain surface orientation estimation for regular FOV images. We warp our result and the ground truth to these perspective views for comparison. From the comparison shown in Fig. 4.9(c), we can see that by using panorama, our algorithm significantly outperforms these results on regular FOV images.

**Is context as powerful as local image appearance for object detection?** Using only our top 1 prediction for each panorama images, we can compute the precision and recall for each object category. Therefore, we compare with the state-of-the-art object detector. We train DPM [63] using the SUN database [271]. To test it on panorama images, we warp a panorama into many regular perspective images and run the trained DPM on it. Fig. 4.12(a) shows the result of the comparison. We can see that our model performs better than DPM at many object categories. This demonstrates that by using only 3D contextual information without any image feature for categorization, we can still achieve a comparable performance with object detectors using image features.

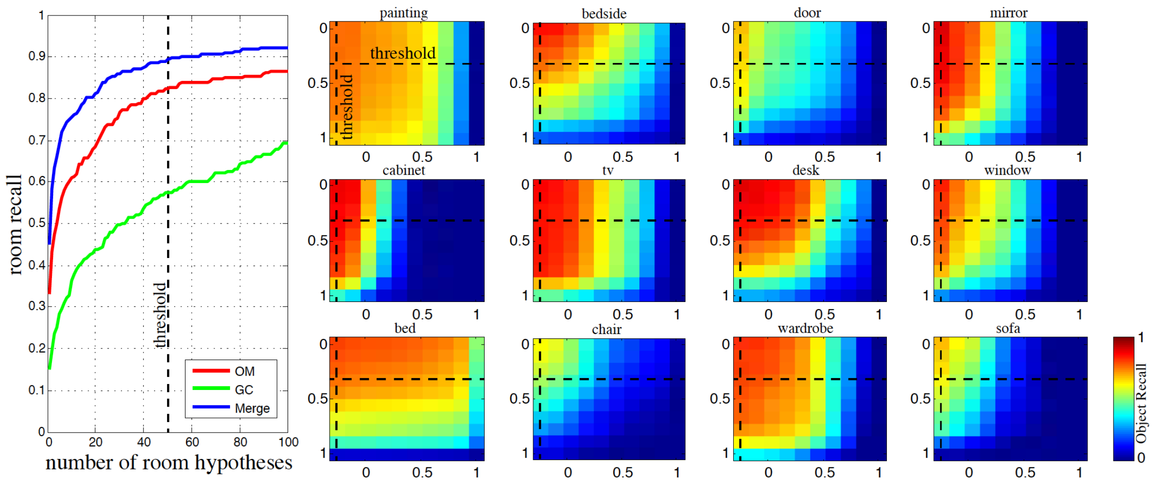


Figure 4.11: **Recall.** Left: The room recall verse the number of top room hypotheses ranked by the OM, GC and our combination of OM and GC, which shows that merging OM and GC significantly improves the result and justifies that 50 is a good threshold. Right: The object recall w.r.t. the rectangle detection score (horizontal axes) and random forest score (vertical axes).

**Is context important in sampling and ranking?** We disable the pairwise context model for sampling and the room alignment matching cost for ranking respectively and together to show the power of each one. In Fig. 4.12(a), the detection performances for nightstand and tv keep on decreasing when disabling more context model. These objects are usually in a common size and shape, and thus cannot be discriminated easily with bottom up image evidence. Context information can be especially useful under this situation. However, for painting and door, the performance does not change much. It could be that these objects usually have strong relations with the walls, and we didn't turn off the wall-object context, so the pairwise or high level context doesn't matter much. Such strong context between wall and objects further shows the advantage of using panorama, in which all the wall, floor, and ceiling are visible.

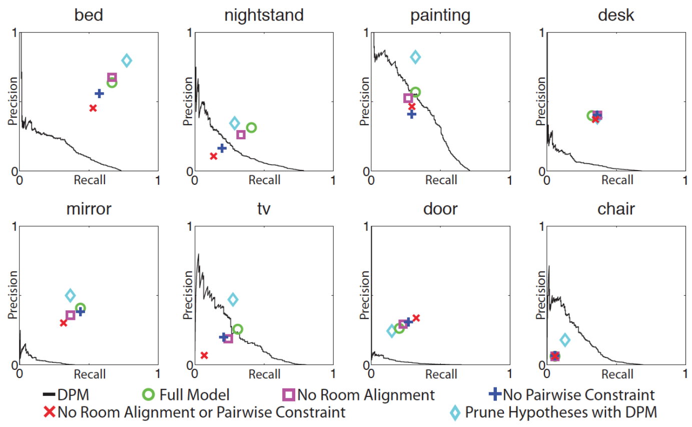
**Is larger FOV better for context?** We narrow down the FOV that is allowed to for pairwise context model. Pair of cuboids far in term of FOV cannot affect each other by pairwise context model during the whole-room hypothesis sampling. Fig.

4.12(b) shows the F-score ($\sqrt{\text{precision} \times \text{recall}}$) of object detection w.r.t. different FOVs. We can see that the F-score curves are all in a decreasing tendency when the FOV is getting smaller. It shows that the big FOV is essential in providing more context information.

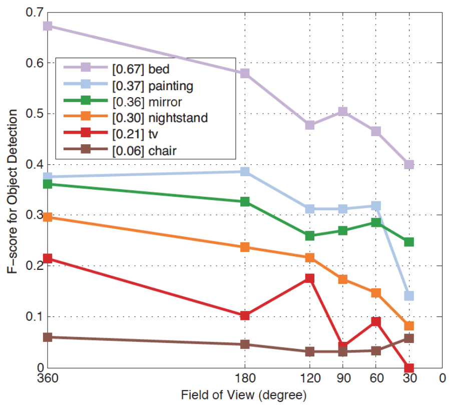
**Is context complementary with local object appearance?** We combine to our model with object detector to answer this question. We run DPM on each object hypothesis, and prune those with low score during the scene hypothesis sampling. The result is shown in Fig. 4.12(a). We can see that for categories on which both DPM and context perform well, merging them will achieve higher performance, like bed, tv, painting. It proves that the context information is complementary to image evidence. For categories that DPM does not work well, the improvement benefit from merging is very limited as expected, like mirror, desk. For the objects without much context, e.g. chair, though the performance is improved, it is still not comparable with DPM. Maybe this shows that context can hurt the detection performance for objects with flexible locations. Note that this is just a simple test to show the effect of merging DPM with our system, there are actually many parts in our model which can be improved by a strong image feature based detector. The confidence score from detector can be used as a power bottom up feature of object hypotheses during the sampling and holistic ranking.

### 4.1.6 Conclusion

We propose a whole-room 3D context model that takes a 360° panorama as input and outputs a 3D bounding box of the room and detects all major objects inside [296]. Experiments show that our model can recognize objects using only 3D contextual information without any image feature for categorization, and still achieves a comparable performance with the state-of-the-art object detector using image features for



(a) Precision-recall comparison with DPM



(b) F-score with different FOV

Figure 4.12: **Object detection.** (a) the performance of our system by partially disable some key usage of context, and the comparison with DPM. (b) F-score of our system with decreasing field of view. The performance becomes worse when the FOV is getting smaller.

categorization. We showcase that the root of context model being under-utilized is partly because regular FOVs are too small, and that context is more powerful than we thought.

## 4.2 DeepContext: Context-Encoding Neural Pathways for 3D Holistic Scene Understanding

### 4.2.1 Introduction

Chapter 4.1 introduced a holistic scene context models, which integrates both the bottom up local evidence and the top down scene context and achieves superior performance. However, it suffers from a severe drawback that the bottom up and top down stages are run separately. The bottom up stage using only the local evidence needs to generate a large quantity of noisy hypotheses to ensure a high recall, and the top down inference usually requires combinatorial algorithms, such as belief propagation or MCMC, which are computationally expensive in a noisy solution space. Therefore, the whole combined system can hardly achieve a reasonably optimal solution efficiently and robustly.

Inspired by the success of deep learning, we propose a 3D deep convolutional neural network architecture that jointly leverages local appearance and global scene context efficiently for 3D scene understanding [293]. To prevent the difficulty of inferring depth from color images, in this work we take a single depth image as the input and focus on the network design to incorporate scene context.

Designing a deep learning architecture to encode context for scene understanding is challenging. Unlike an object whose location and size can be represented with a fixed number of parameters, a scene could involve unknown number of objects and thus requires variable dimensionality to represent, which is hard to incorporate with convolutional neural network with a fixed architecture. Also, although holistic scene models allow flexible context, they require common knowledge to manually predefine relationship between objects, e.g. the relative distance between bed and nightstands. As a result, the model may unnecessarily encode weak context, ignore important context, or measure context in an over simplified way.

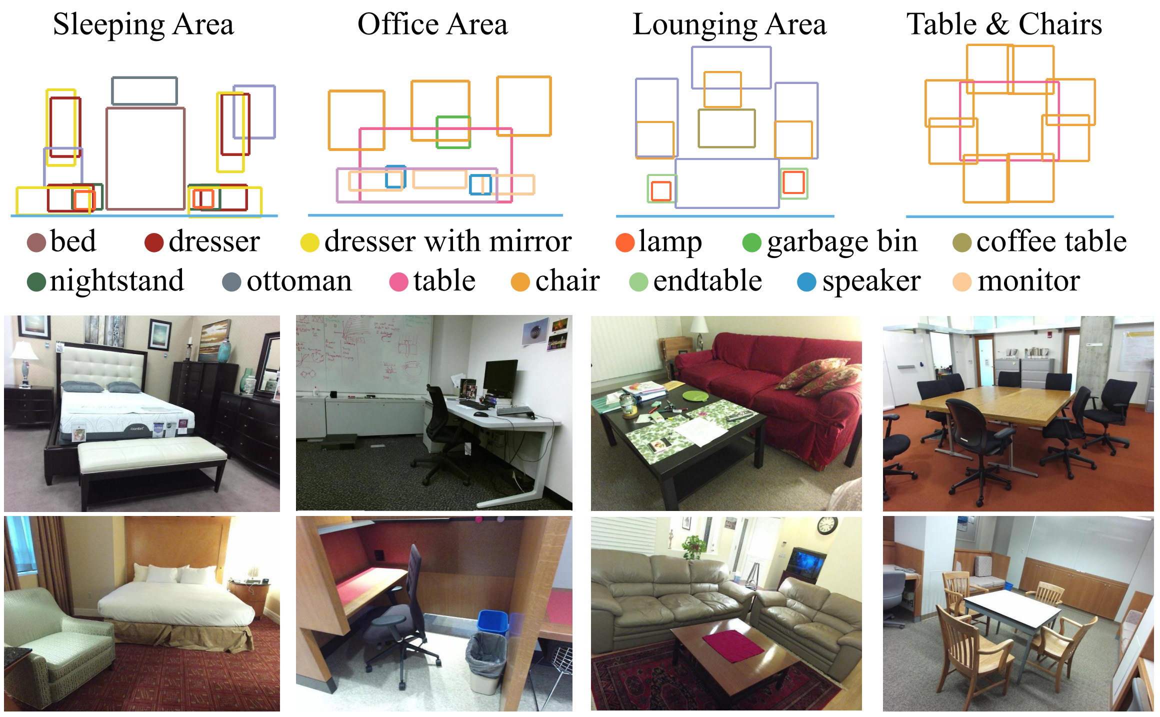


Figure 4.13: **Example of canonical scene templates (top view) and the natural images they represent.** We learn four scene templates from SUN-RGBD[229]. Each scene template encodes the canonical layout of a functional area.

To solve these issues, we propose and learn a scene representation encoded in scene templates. A scene template contains a super set of objects with strong contextual correlation that could possibly appear in a scene with relatively constrained furniture arrangements. It allows a prediction of "not present" for the involved objects so that a variety of scenes can be represented with a fixed dimensionality. A scene can be considered as a scene template with a subset of objects activated. Scene template also learns to only consider objects with strong context, and we argue that contextless objects, such as a chair can be arbitrarily placed, should be detected by a local appearance based object detector.

Each template represents a functional sub-region of an indoor scene, predefined with canonical furniture arrangements and estimated 3D anchor positions of possible objects with respect to the reference frame of the template. We incorporate these template anchors as priors in the neural architecture by designing a transformation network that aligns the input 3D scene (corresponding to the observed depth image) with the template (i.e. the canonical furniture arrangement in 3D space). The aligned 3D scene is then fed into a 3D context neural network that determines the existence and location of each object in the scene template. This 3D context neural network contains a holistic scene pathway and an object pathway using 3D Region Of Interest (ROI) pooling in order to classify object existence and regress object location respectively. Our model learns to leverage both global and local information from two pathways, and can recognize multiple objects in a single forward pass of a 3D neural network. It is noted that we do not manually define the contextual relationships between objects, but allow the network to automatically learn context in arbitrary format across all objects.

Data is yet another challenging problem for training our network. Holistic scene understanding requires the 3D ConvNet to have sufficient model capacity, which needs to be trained with a massive amount of data. However, existing RGB-D datasets for scene understanding are all small. To overcome this limitation, we synthesize training data from existing RGB-D datasets by replacing objects in a scene with those from a repository of CAD models from the same object category, and render them in place to generate partially synthesized depth images. Our synthetic data exhibits a variety of different local object appearances, while still keeping the indoor furniture arrangements and clutter as shown in the real scenes. In experiments, we use these synthetic data to pretrain and then finetune our network on a small amount of real data, whereas the same network directly trained on real data can not converge.

The contributions of this work are mainly three aspects. 1) We propose a scene template representation that enables the use of a deep learning approach for scene understanding and learning context. The scene template only encodes objects with strong context, and provides a fixed dimension of representation for a family of scenes. 2) We propose a 3D context neural network that learns scene context automatically.

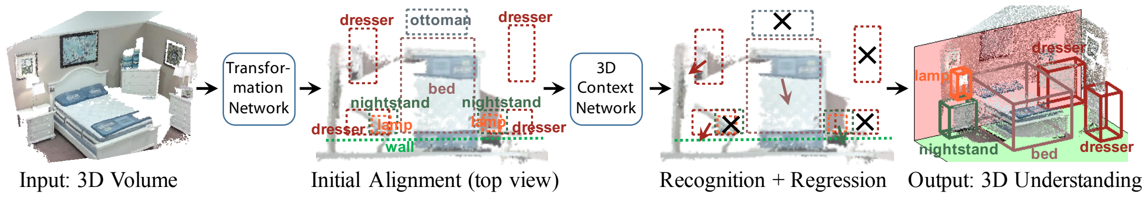


Figure 4.14: **Our deep 3D scene understanding pipeline.** Given a 3D volumetric input derived from a depth image, we first aligns the scene template with the input data. Given the initial alignment, our 3D context network estimates the existence of an object and adjusts the object location based on local object features and holistic scene feature, to produce the final 3D scene understanding result.

It leverages both global context and local appearance, and detects all objects in context efficiently in a single forward pass of the network. 3) We propose a hybrid data augmentation method, which generates depth images keeping indoor furniture arrangements from real scenes but containing synthetic objects with different appearance.

### 4.2.2 Related Work

There are some efforts incorporating holistic context model for scene understanding, which is closely related to our work. Scene context is usually manually defined as a unary term on a single object, pairwise term between a pair of objects to satisfy certain functionality [157, 284], or a more complicated hierarchy architecture [32, 165, 302]. The learned context models are usually applied on a large set of object hypotheses generated using local evidence, e.g. line segments [302] or cuboid [157], by energy minimization. Therefore high order context might be ignored or infeasible to optimize. Context can be also represented in a non-parametric way [296], which potentially enables high order context but is more computationally expensive to infer during the testing time. In contrast, our 3D context network does not require any heuristic intervene on the context and learns context automatically. We also require no object hypothesis generation, which is essential in making our method more computationally efficient.

Deep learning has been applied to 3D data, but most of these works focus on modeling objects [266] and object detection [179, 231]. Recently, some successes have been made on applying deep learning for inverse graphics [142, 143]. Our approach goes one step further to embrace the full complexity of real-world scenes to perform holistic scene understanding. Related to our transformation network, Spatial Transformation Networks [117] can learn the transformation of an input data to a canonical alignment in an unsupervised fashion. However, unlike MNIST digits (which were considered in [117]) or an individual object where an alignment to a canonical viewpoint is quite natural, it is not clear what transforms are needed to reach a canonical configuration for a 3D scene. We define the desired alignment in template coordinates and use supervised training by employing the ground truth alignments available from our training data.

While many works have considered rendering synthetic data for training (a.k.a, graphics for vision, or synthesis for analysis), these efforts mostly focus on object rendering, either in color [155, 233] or depth [230]. There is also work rendering synthetic data from CAD model of complicated scenes for scene understanding [94, 293]. However, the generated depth is overly clean, and the scene layouts generated by either by algorithm or human artists are not guaranteed to be correct. In contrast, we utilize both the CAD models and real depth maps to generate more natural data with appropriate context and real-world clutter.

### 4.2.3 Method

#### 4.2.3.1 Algorithm Overview

Our approach works by first automatically constructing a set of scene templates from the training data (see Section 4.2.3.2). Rather than a holistic model for everything in the scene, each scene template only represents objects with context in a sub-area of a scene performing particular functionality. Each template defines a distribution of possible layouts of one or more instances of different object categories in a fixed dimensionality.

Given a depth map of a scene as input[4], we convert it into a 3D volumetric representation of the scene and feed it into the neural network. The neural network first infers the scene template that is suitable to represent the scene, or leaves it to a local appearance based object detector if none of the predefined scene templates is satisfied. If a scene template is chosen, the transformation network estimates the rotation and translation that aligns the scene to the inferred scene template. With this initial alignment, the 3D context network extracts both the global scene feature encoding scene context and the local object features pooled for each anchor object defined in the template, as shown in Fig.4.14. These features are concatenated together to predict the existence of each anchor object in the template and an offset to adjust its bounding box for a better object fit. The final result is an understanding of the scene with a 3D location and category for each object in the scene, as well as room layout elements including wall, floor, and ceiling, which are represented as objects in the network.

#### 4.2.3.2 Learning Scene Template

Objects with context in a functional area are usually at relatively fixed locations. For example, a sleeping area is usually composed of a bed with one or two nightstands on the side, with optional lamps on the top. Object detection is likely to succeed by searching around these canonical locations. We learn the categories of object instances and their canonical sizes and locations in the template, from the training data. Examples of each template can be seen in Fig. 4.13.

[4]Note that while all the figures in the paper contain color, our system relies only on depth as input without using any color information.

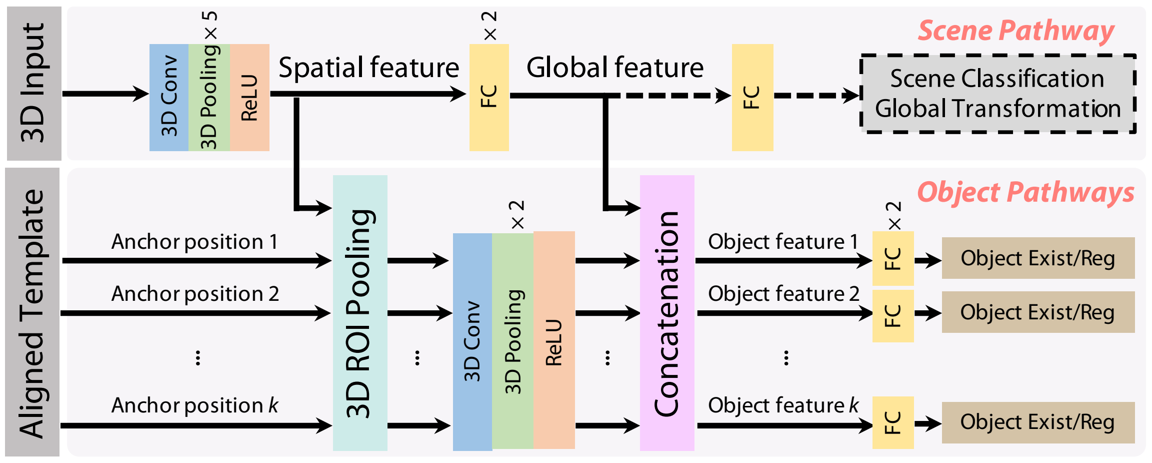


Figure 4.15: **3D context network.** The network consists of two pathways. The scene pathway takes the whole scene as input and extracts spatial feature and global feature. The object pathway pools local feature from the spatial feature. The network learns to use both the local and global features to perform object detection, including wall, ceiling, and floor.

**Data-driven Template Definition** We learn to create scene templates using the SUN-RGBD dataset consisting of 10,335 RGB-D images with 3D object bounding box annotations. These RGB-D images are mostly captured from household environments with strong context. As a first experiment of combining 3D deep learning with context, we choose four scene templates: sleeping area, office area, lounging area, and table & chair set, because they represent commonly seen indoor environments with relatively larger numbers of images provided in SUN-RGBD. Our approach can be extended to other functional areas given sufficient training data. For SUN-RGBD, 75% of the images from household scene categories can be described, fully or partially, using these four scene templates.

Our goal is to learn layouts of the scene, such that each template summarizes the bounding box location and category of all objects appearing in the training set. To enable the learning of the template, we select the images that contain a single functional area, and label them with the scene type they belong to. Other images containing arbitrary objects or multiple scene templates are not used in learning scene templates. The ground truth scene categories are used not only for learning the aforementioned templates, but also for learning the scene template classification, the transformation networks, and the 3D context networks in the following sections.

To obtain the anchor positions (i.e. common locations) for each object type in a template, we take all 3D scenes belonging to this scene template and align them with respect to the center and orientation of a major object[5]. After that, we run $k$-means clustering for each object type and use the top $k$ cluster centroids as the anchor positions and size, where $k$ is user-defined. We also include room layout elements, including wall, floor, ceiling, which are all represented as regular objects with predefined thickness. Each scene template has tens of object anchors in total for various object categories (Fig. 4.13).

**Generating Template-Based Ground Truth** To train a 3D context network using scene templates, we need to convert the original ground truth data from SUN RGB-D dataset to a template representation. Specifically, we need to associate each annotated object in the original ground truth with one of the objects defined in the scene template. Similar to above, we first align the training images with their corresponding scene templates using the center and rotation of the major object. For the rest of the objects, we run a bipartite matching between the dataset annotation and the template anchors, using the difference of center location and size as the distance, while ensuring that the objects of the same category are matched.

#### 4.2.3.3 3D Scene Parsing Network

Given a depth image as input, we first convert it into a 3D volumetric representation, using the Truncated Signed Distance Function (TSDF) [231, 191]. We use a $128 \times 128 \times 64$ grid for the TSDF to include a whole scene, with a voxel unit size of 0.05

[5]We manually choose bed for sleeping area, desk for office area, sofa for lounging area, and table for table&chair set as the major objects.

meters and a truncation value of 0.15 meters. This TSDF representation is fed into the 3D neural network such that the model runs naturally in 3D space and directly produces output in 3D.

**Scene Template Classification Network** We first train a neural network to estimate the scene template category for the input scene (Fig. 4.15, Scene pathway). The TSDF representation of the input scene is firstly fed into 3 layers of 3D convolution + 3D pooling + ReLU, and converted to a spatial feature map. After passing through two fully connected layers, the 3D spatial feature is converted to a global feature vector that encodes the information from the whole scene. The global feature is used for scene template classification with a classic softmax layer. During testing, we choose the scene template with the highest score for the input scene if the confidence is high enough ($> 0.95$). Otherwise, we do not run our method because none of the scene templates fits the input scene. Such scenes are passed to a local appearance based object detector for object detection. In practice, the four scene templates can match with more than half of the images in the SUN-RGBD dataset captured from various of indoor environments.

**Transformation Network** Given the scene template category, our method estimates a global transformation consisting of a 3D rotation and translation that aligns the point cloud of the input scene to the target predefined scene-template (Fig. 4.16). This is essentially a transformation that aligns the major object in the input scene with that from the scene template. This makes the result of this stage invariant to rotations in the input, and the wall and bounding box of objects are globally aligned to three main directions. The next part of our architecture, the 3D context network, relies on this alignment to obtain the object orientation and the location to pool feature based on 3D object anchor locations from the scene template.

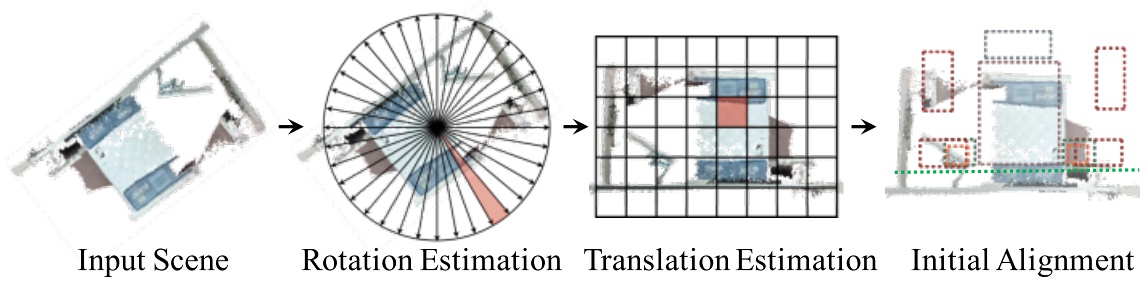


Figure 4.16: **Transformation estimation.** Our transformation network first produces global rotation and then translation to align the input scene with its scene template in 3D space. Both the rotation and translation are estimated as classification problems.

We first estimate the rotation. We assume that the gravity direction is given, e.g. from an accelerometer. In our case, this gravity direction is provided by the SUN RGB-D dataset used in our experiments. Therefore, we only need to estimate the yaw, which rotates the input point cloud in horizontal plane to the scene template viewpoint shown in Fig.4.13. We divide the 360-degree range of rotation into 36 bins and cast this problem into a classification task (Fig. 4.16). We train a 3D ConvNet using the same architecture as the scene template classification network introduced in Sec. 4.2.3.3 except generating a 36 channel output for softmax. During training, we align each training input scene to the center of the point cloud and add noise for rotations (+/- 10 degrees) and translations (1/6 of the range of the point cloud).

For translation, we apply the same network architecture to identify the translation after applying the predicted rotation. The goal is to predict the 3D offset between the centers of the major objects of the input point cloud and its corresponding scene template. To achieve this goal, we discretize the 3D translation space into a grid of $0.5\text{m}^3$ resolution with dimensions of $[-2.5, 2.5] \times [-2.5, 2.5] \times [-1.5, 1]$, and formulate this task again as a 726-way classification problem (Fig. 4.16). We tried direct regression with various loss functions, but it did not work as well as classification. We also tried an ICP-based approach, however it could not produce good results.

**3D Context Network** We now describe the context neural network for indoor scene parsing using scene templates. For each scene template defined in the previous section, a separate prediction network is trained. As shown in Fig. 4.15, the network has two pathways. The global scene pathway, given a 3D volumetric input in a coordinate system that is aligned with the template, produces both a spatial feature that preserves the spatial structure in the input data and a global feature for the whole scene. For the object pathway, we take the spatial feature map from the scene pathway as input, and pool the local 3D Region Of Interest (ROI) based on the 3D scene template for the specific object. The 3D ROI pooling is a max pooling at $6 \times 6 \times 6$ resolution, inspired by the 2D ROI pooling from [77]. The 3D pooled features are then passed through 2 layers of 3D convolution + 3D pooling + ReLU, and then concatenated with the global feature vector from the scene pathway. After two more fully connected layers, the network predicts the existence of the object (a binary classification task) as well as the offset of the 3D object bounding box (3D location and size) related to the anchor locations learned in Sec. 4.2.3.2 (a regression task using L1-smooth loss [231]). Including the global scene feature vector in the object feature vector provides holistic context information to help identify if the object exists and its location.

**Training Schema** Our 3D scene parsing network contains a series of components with a large number of parameters. We perform careful training strategy to avoid bad local optima. We first train the scene pathway alone to perform a 4-way scene classification task. After this training converges, we finetune the classification network to estimate the transformation for each individual scene template. An alternative approach is to jointly train a network for classification and transformation, however this does not perform well in practice. The object pathway is then enabled, and the two pathways are jointly finetuned to perform object detection. We found that this

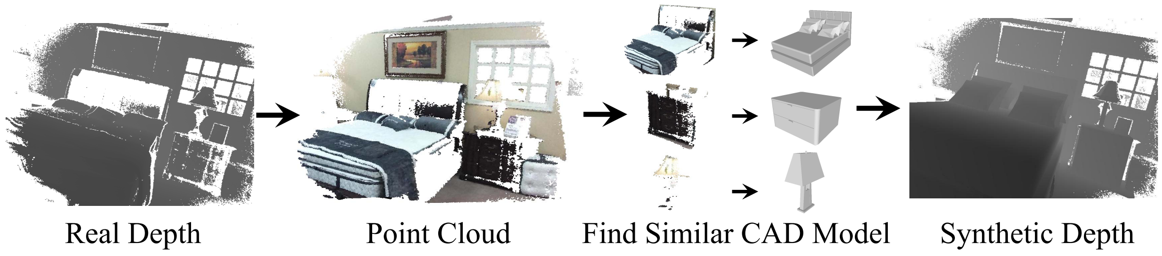


Figure 4.17: **Hybrid data synthesis.** We first search for similar CAD model for each object. Then, we randomly choose models from good matches, and replace the points in annotated bounding box with the rendered CAD model.

form of pretraining, from easy to hard task, is crucial in our experiments. Otherwise, training the four networks independently from scratch cannot produce meaningful models.

#### 4.2.3.4 Synthesizing Hybrid Data for Pre-training

In contrast to existing deep architectures for 3D [231, 266], our model takes the whole scene with multiple objects as input. As such, during training, it needs to model the different variations in the scene layout. We found the RGB-D images from the existing SUN RGB-D [229] dataset are far from sufficient. Furthermore, capturing and annotating RGB-D images on the scale of ImageNet [40] was impractical. To overcome the data deficiency problem, we increase the size of the training data by replacing the annotated objects from SUN RGB-D with CAD models of same category from ShapeNetCore dataset [25] (Fig. 4.17). This allows us to generate context-valid scenes, as the context still comes from a real environment, while changing the shapes of the objects. By replacing the annotated objects while keeping the full complexity of the areas outside the annotated bounding boxes, we could generate more realistic hybrid data partially maintaining sensor noise. This is in contrast to images generated from purely synthetic models which do not contain clutter caused by the presence of small objects.

To search for similar CAD models for annotated objects in RGB-D images, we need to define the distance between a CAD model $\mathcal{M}$, and the 3D point cloud $\mathcal{P}$ representing the object. In order to get a symmetric definition, we first put the model in the annotated 3D box, scale it to fit, render $\mathcal{M}$ with the camera parameter of the depth image, and convert the rendered depth image to a point cloud $\mathcal{V}$. This is to mimic the partial view due to self occlusion. Then, we define the distance between $\mathcal{P}$ and $\mathcal{S}$ as:

$$D(\mathcal{P}, \mathcal{S}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \Big( \min_{q \in \mathcal{V}} d(p, q) \Big) + \frac{1}{|\mathcal{V}|} \sum_{p \in \mathcal{V}} \Big( \min_{q \in \mathcal{P}} d(p, q) \Big), \tag{4.1}$$

where $d(p, q)$ is the distance between two 3D points $p$ and $q$. After acquiring a short list of similar CAD models for each object, we randomly choose one and render the depth image with the original annotation as training data.

We generate a hybrid training set that is 1,000 times bigger than the original RGB-D training set. For both of the pathways in the 3D context network, we have to train the models on this large hybrid dataset first, followed by finetuning on the real depth maps. Otherwise, the training cannot converge.

## 4.2.4 Experiments

We use the SUN RGB-D dataset [229] because they provide high quality 3D bounding box annotations of objects. As described in Section 4.2.3.2, we manually select images that can be perfectly represented by one of the scene templates, and choose 1,863 RGB-D images from SUN RGB-D. We use 1,502 depth images to learn scene templates and train the 3D scene parsing network, and the remaining 361 images for testing. We also evaluate our model for object detection on a testing set containing images that cannot be perfectly represented, e.g. containing arbitrary objects or multiple

| | bed | night-stand | dresser | coffee table | mirror dresser | end table | lamp | monitor | ottoman | sofa | chair | table |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COG [205] | 79.8 | 48.1 | 1.70 | - | - | - | - | - | - | 55.8 | 72.9 | 58.4 |
| DSS [231] | 90.3 | 52.3 | 7.60 | 52.7 | 4.40 | 13.3 | 40.2 | 15.0 | 23.7 | 71.3 | 79.1 | 75.2 |
| Ours | 89.4 | 63.3 | 19.7 | 40.5 | 16.8 | 27.9 | 41.6 | 18.2 | 13.3 | 50.3 | 44.5 | 65.9 |
| Ours + DSS | **91.8** | **66.7** | **23.4** | 50.1 | **10.0** | **35.3** | **53.6** | **23.2** | **31.5** | 62.8 | **80.2** | **77.4** |
| GT Align | 92.4 | 64.4 | 19.7 | 49.3 | 23.4 | 25.0 | 31.4 | 16.0 | 15.8 | 63.6 | 46.1 | 70.4 |
| GT Align+Scene | 94.1 | 66.3 | 19.4 | 48.9 | 23.4 | 21.7 | 31.4 | 16.1 | 15.8 | 74.6 | 50.2 | 74.0 |
| DSS, Full | 75.7 | 30.0 | 7.14 | 19.5 | 0.64 | 11.7 | 20.9 | 1.80 | 8.49 | 51.7 | 52.9 | 41.1 |
| Ours, Full | **75.8** | **44.1** | **15.7** | **25.8** | **4.99** | **12.4** | **22.4** | **3.47** | **10.7** | 49.0 | **53.2** | 30.5 |

Table 4.1: **Average precision for 3D object detection.** We (row 3) achieve comparable performance with DSS [231] (row 2). Combining two methods (row 4) achieves significantly better performance, which shows our model learns context complementary to local appearance. Our model can further achieve better performance with better alignment and scene classification. The last row shows our superior performance on extended testing set where images might not be perfectly represented by any single scene template.

scene templates, to demonstrate that our scene templates have a good generalization capability and a high impact on real scenes in the wild.

Our model uses the half data type, which represents a floating point number by 2 bytes, to reduce the memory usage. We train the model with a mini-batch of 24 depth images requiring 10GB, which nearly fills a typical 12GB GPU. However, this mini-batch size was too small to obtain reliable gradients for optimization. Therefore, we accumulate the gradients over four iterations of forward and backward without weight update, and only update the weights once afterwards. Using this approach, the effective mini-batch size is $24 \times 4$ or 96.

#### 4.2.4.1 3D object detection.

Our model recognizes major objects in a scene, which can be evaluated by 3D object detection. Qualitative parsing results are shown in Fig. 4.20. Our model finds most of the objects correctly and produces decent scene parsing results for challenging cases, e.g. heavy occlusion and missing depth. 3D context enables long range regression when initial alignment is far from correct, as shown in the 5th row. The last row shows a failure case, where our model recognizes it as a sleeping area misled by the futon with blankets. Therefore, our model overlooks the coffee table, but still predicts the wall and floor correctly and find a proper place to sleep.

Table 4.1 shows quantitative comparison to the local appearance based 3D object detector Deep Sliding Shape (DSS) [231] and also the cloud of gradient feature based context model from Ren *et al.* (COG) [205]. Our average precision (3rd row) is comparable to state-of-the-art, but only takes about 0.5 seconds to process an image for all object categories, which is about 40 times faster than DSS which takes 20 seconds per image.

**Context complements local evidence.** Fig. 4.18 shows some qualitative comparisons between our context model and the local object detector DSS [231]. We can see that our context model works significantly better in detecting objects with missing depth (the monitor in 1st and 3rd examples) and heavy occlusion (the nightstand in 2nd example). 3D context also helps to remove objects in incorrect arrangements, such as the table on top of another table, and the nightstand at the tail of the bed or in office, as shown in the result of DSS. Comparatively, DSS works better for objects that are not constrained, e.g. chairs on the right of 3rd example.

We integrate the result from DSS and our context model. The combined result achieves significantly better performance than each of the models individually, increasing the mean average precision from the 43.76% for DSS stand-alone to 50.50%. This significant improvement demonstrates that our context model provides complementary information with a local appearance based object detector.

Fig. 4.19 shows the Precision-Recall (PR) curves for some of the object categories. We can clearly see that our (green) recalls are not as high as DSS (blue) that runs in a sliding window fashion to exhaustively cover the search space. This is because our model only detects objects within the context. However, our algorithm maintains a very high precision, which applies to a broader range of working situations, with

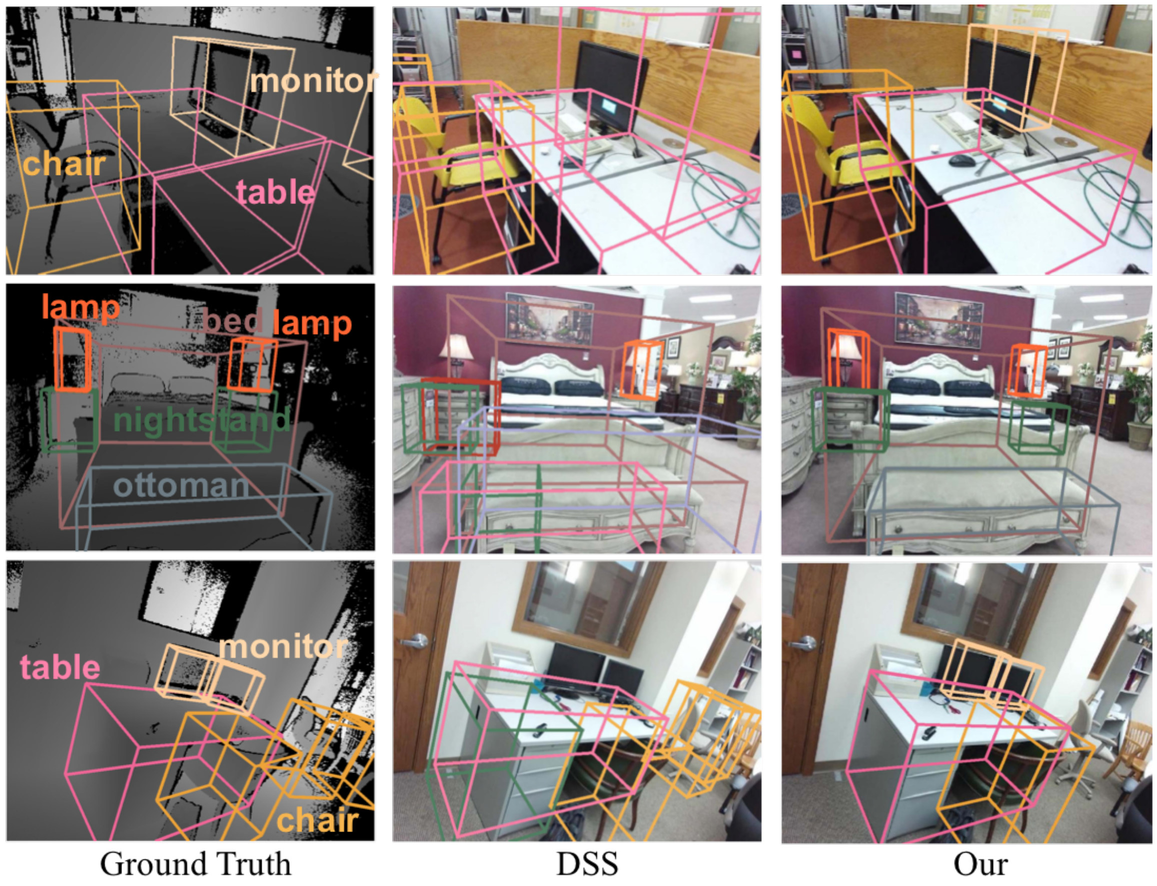


Figure 4.18: **Comparison between our context model and the local object detector DSS [231].** Our context model works well for objects with missing depth (monitors in 1st, 3rd row), heavy occlusion (nightstand in 2nd row), and prevents detections with wrong arrangement (wrong table and nightstand in DSS result).

slightly lower recall. Nevertheless, combining the result of our method and DSS (red) obtains the best performance in terms of both precision and recall.

**Generalization to imperfect scene template images.** Our method can work not only on perfect scene template images, but also images in the wild. Thanks to the template classification and alignment component, our method can find the right place in the input scene to apply the context model. To evaluate, we randomly pick 2,000 images that are not used for training from the SUN-RGBD dataset. This uniformly sampled testing set reflects the scene distribution from the dataset, and contains many images that cannot be perfectly represented by any of the scene templates. We test

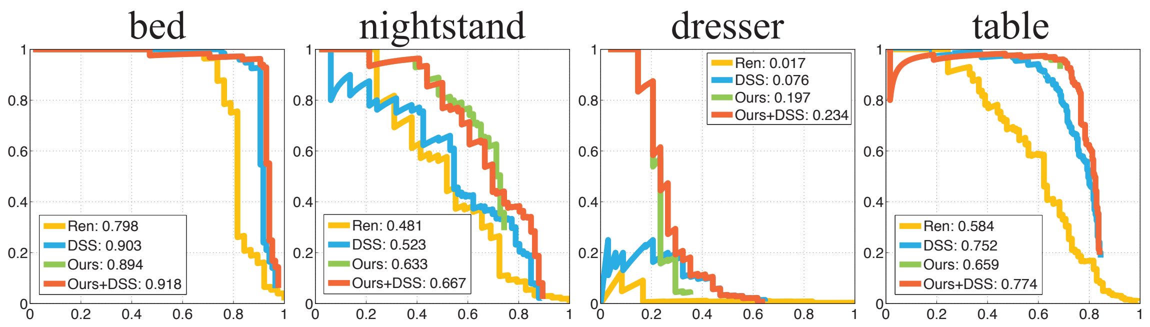


Figure 4.19: **Precision recall curves for some object categories.** We compare our algorithm with the 3D object detector DSS [231] and the cloud of gradient feature based context model Ren *et al.* [205].

DSS on this test set and achieve 26.80% mAP (Table 4.1, the 2nd last row), which is similar to the performance reported in [231]. We further run our method on testing images with the template classification confidence higher than 0.95, which ends up choosing 1,260 images. We combine our result with DSS, and the performance is shown in the last row of Table 4.1. As can be seen, our model successfully wins in 10 out of 12 categories, and improves the mAP to 29.00%. This improvement shows that our model can be applied to a variety of indoor scenes. It is also extremely effective in improving the scene understanding result in the aligned sub-area.

#### 4.2.4.2 Room Layout and Total Scene Understanding

**Layout estimation.** As part of our model, we can estimate the existence and location of the ceiling, floor, and the wall directly behind the camera view. Table 4.2 shows quantitative evaluation. We can see that the 3D context network can successfully reduce the error and predict a more accurate room layout. Note that for some scene categories, the ceiling and wall are usually not visible from the images. These cases are marked as "-".

**Scene understanding.** We use the metrics proposed in [229] to evaluate total 3D Scene Understanding accuracy. These metrics favor algorithms producing correct

| Layout Estimation (Mean/Median) | Sleeping Area | Office Area | Lounging Area | Table &Chair |
|---|---|---|---|---|
| Ceiling Initial | 0.57/0.56 | - | - | 0.84/0.71 |
| Ceiling Estimate | 0.45/0.40 | - | - | 0.72/0.44 |
| Floor Initial | 0.30/0.25 | 0.28/0.24 | 0.25/0.23 | 0.22/0.20 |
| Floor Estimate | 0.10/0.09 | 0.09/0.06 | 0.22/0.16 | 0.08/0.05 |
| Wall Initial | 0.40/0.30 | 0.70/0.60 | - | - |
| Wall Estimate | 0.22/0.08 | 0.60/0.21 | - | - |

Table 4.2: **Error (in meter) for room layout estimation.** Our network reduces the layout error upon initialization from the transformation network. Note that for some scene categories, the ceiling and wall may not be visible from the images and therefore there are no annotations (marked with "-").

| Method | Sym. | Sleeping Area | Office Area | Lounging Area | Table &Chair |
|---|---|---|---|---|---|
| ICP | No | 75.6% | 69.2% | 58.5% | 38.1% |
| ICP | Yes | 96.3% | 89.0% | 92.5% | 75.3% |
| Network | No | 92.7% | 87.9% | 71.7% | 44.3% |
| Network | Yes | 100.0% | 93.4% | 94.3% | 73.2% |

(a) Rotation Estimation Accuracy$\uparrow$

| Method | Rot. | Sleeping Area | Office Area | Lounging Area | Table &Chair |
|---|---|---|---|---|---|
| ICP | - | 0.473 | 0.627 | 1.019 | 0.558 |
| Network | GT | 0.278 | 0.246 | 0.336 | 0.346 |
| Network | Est | 0.306 | 0.278 | 0.606 | 0.332 |

(b) Translation Error (in meters) $\downarrow$

Table 4.3: **Evaluation of the transformation networks.** Our transformation network outperforms direct point cloud matching in the accuracies of both rotation and translation.

detections for all categories and accurate estimation of the free space. We compare our model with Ren *et al.* (COG) [205]. For geometry precision ($P_g$), geometry recall ($R_g$), and semantic recall ($R_r$), we achieve 71.02%, 54.43%, and 52.96%, which all clearly outperform 66.93%, 50.59%, and 47.99% from COG. Note that our algorithm uses only the depth map as input, while COG uses both color and depth.

#### 4.2.4.3 System Component Analysis

Our 3D context network relies on the initial alignment produced by scene template classification and transformation estimation model. We also investigate how these factors affect our performance.

**Transformation Prediction.** Table 4.3 reports the evaluation of template alignment. For rotation, we show the percentage of data within a 10 degree range to the ground truth. For translation, we show the distance between the estimated translation and the ground truth.

For rotation, since some scenes (especially for lounging area and table&chair set) are symmetric with respect to the horizontal plane, a correct estimation of the main direction would be enough for our purposes. Therefore, we report the accuracies both with and without symmetry [Sym.].

To compare with our neural network-based approach, we design an ICP approach based on point cloud alignment as a baseline. Given a point cloud from a testing depth map, we align it with the point cloud of each image in the training set, by exhaustively searching for the best rotation and translation, using the measurement in Section 4.2.3.4. We choose the alignment with the best aligned training depth map as our transformation. We can see that our neural network based approach significantly outperforms this baseline.

To see how sensitive our model is to the initial alignment, we evaluate our model with the ground truth alignment, and the result is shown in Table 4.1 [GT Align]. We can see that the 9 out of 12 categories are improved in terms of AP, compared to that with estimated transformation, and the overall mAP improves 2.19%.

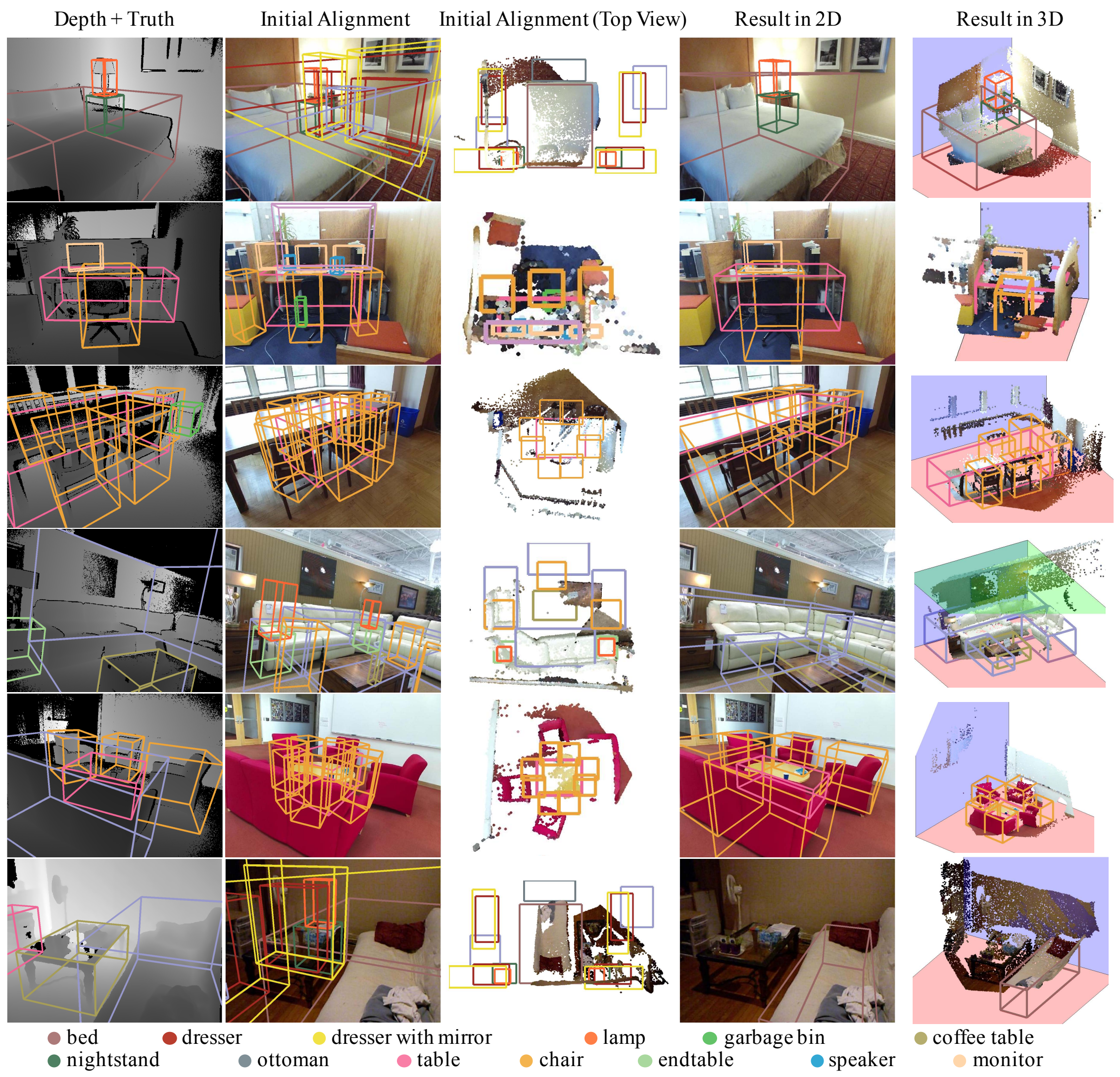


Figure 4.20: **Visualization of the qualitative results on the testset.**

**Template Classification.** The accuracy of the scene template classification is 89.5%. In addition to the ground truth transformation, we test our model with truth template category. This further improves the mAP by 1.52%.

### 4.2.5 Conclusion

We propose a 3D ConvNet architecture that explicitly encodes context and local evidence leveraging scene template and directly outputs holistic understanding [293]. The template is learned from training data to represent the functional area with relatively strong context evidence. We show that context model provides complementary information with a local object detector, which can be easily integrate. Our system has a fairly high coverage on real datasets, and achieves the state of the art performance for 3D object detection on the SUN-RGBD dataset.

# Chapter 5

# Discussion

This chapter consists of the summary of the findings from the introduced projects, shortcomings and limitations of the current scene understanding research from a few aspects, and the potential future directions to strengthen and extend the vision-based scene understanding.

## 5.1 Summary

In this thesis, we introduced a series of works for 3D indoor scene understanding. More specifically, we presented models producing dense pixel-wise representations and complete holistic representations for 3D geometry estimation and semantic prediction.

We demonstrated that pixel-wise representation enables simple and straightforward end-to-end model design (Sec. 3.1.3). It supports per-pixel variation in the output such that it is suitable for vision tasks requiring fine-grained details, such as instance boundary prediction (Sec. 3.2.4), and depth discontinuity estimation (Sec. 3.3.3.2). It also provides a data-driven framework to tackle some previously ill-posed problems by leveraging state-of-the-art learning techniques, e.g., neural network (Sec. 3.2.4). With the help of large-scale photo-realistic synthetic data (Sec. 3.2.3) and real datasets, data-driven approaches produced appealing results, like surface normal estimation, and effectively help to deal with challenging and failure cases in other vision tasks and to improve their overall performance (Sec. 3.3.1).

On the other hand, 3D holistic representation for scene understanding requires more effort on model design but allows us to leverage top-down prior knowledge, like scene context (Sec. 4.1.3). We demonstrate that scene context provides complementary information compared to a model based on local evidence and is more powerful than we expected when being exploited properly for indoor scene parsing (Sec. 4.1.5). We further integrated scene context with deep learning using an explicitly context-aware network architecture, which allows effective learning of scene context and efficient inference with both top-down and bottom-up information (Sec. 4.2.3.3).

## 5.2 Shortcomings and Future Work

### 5.2.1 Representation

It is not feasible to draw a conclusion for which representation is the best, as a trade-off has to be made between the simplicity and capacity of the representation. The best balancing between the two may vary depending on the applications.

The pixel-wise representation wins for its simplicity but suffers from the inherent redundancy which limits its capacity when scaled up. Taking a neural network model as an example, producing high-resolution detailed results needs convolutions to run on high-resolution input image and feature maps, which is contradicting with the fact that downsizing the resolution, e.g., through pooling, is necessary for both computation/memory efficiency and a large receptive field. Some network architectures have been proposed to encourage fine-grained details, such as shortcut connections (Sec. 3.2.4) and image-guided up-sampling (Sec. 3.1.3). However, the results may still be blurry as the main components of the neural network, convolution and pooling, are essentially low pass filters. Alternatively, standalone post-processing can be applied to the network output to encourage sharp results (Sec. 3.3.3.4). This however, may not be optimal overall. An end-to-end network architecture that is friendly to computational resources and produces good details still remains unavailable.

On the other hand, the holistic 3D model we propose is comparatively more compact and powerful, but requires specific non-trivial model designs for different vision tasks. Beyond the category, location, and size of an object, there are many other aspects of the scene which can be investigated and potentially integrated with the bounding box-based representation. First of all, having a more detailed shape, like a mesh model or point cloud, for each object in the 3D bounding box would be desirable. This can be achieved by running a single object based shape generation model [260] in each of the detected bounding boxes or an integrated model to output the shape details together with the upper levels. Second, other aspects of the objects can be also learned. Textures, materials, and illumination can all greatly change the appearance of the scene and thus affect the other scene understanding tasks relying on visual input. Understanding them may help to remove these variances and benefit other tasks. Recently indoor scene dataset provides high dynamic range images from multiple views [24], which provides great opportunities to study these aspects. Last but not least, the scene understanding can extend on the temporal dimension on both the input and the output side. On more and more personal electronics (e.g., cellphones and AR goggles) and operating devices (e.g., robots, cars, and surveillance cameras), we have easy access to videos which provide abundant information for scene understanding. Compared to a single image, a temporal sequence allows us to leverage coherence across frames, observe motion, and look from multiple viewpoints, which together would result in more reliable 3D scene understanding. The output can also be dynamic and change in time to understand the past, present, and future state of the scene. Especially for dynamic outputs, there are some initial trials [190, 216] but not many works exist in the literature.

### 5.2.2 Data for Scene Understanding

Due to the effectiveness of the deep learning, the pursuit of large-scale high-quality training data is always ongoing. Synthetic data allows error-free ground truth, and we have demonstrated that well-rendered synthetic data can effectively improve the pre-training stage and further improve the performance after fine-tuning on real data (Sec. 3.2.3). However, high-quality synthetic data can be expensive, and a compromise has to be made between the rendering quality and the quantity. Therefore, the domain gap between rendering and real data is still inevitably big. On the other hand, real data collection is extremely hard especially for 3D geometry. Recent studies have made efforts to speed up the physically based rendering convergence using a recurrent auto-encoder [22] or directly generate natural looking images using generative adversarial networks [81]. The first line of work follows the physics of image rendering but is not an end-to-end system and is therefore not optimized for a specific task. In contrast, the second line of work is end-to-end and specifically optimized for natural looking images, i.e., reducing the domain gap. It is however less physically driven and can result in structurally incongruent images. It might be promising if the physics insight and advanced learning techniques could be integrated to generate better synthetic data. Moreover, a concept of having virtual goggles that bring both real and synthetic images to the same domain has been proposed, and vision tasks can be done in this unified virtual domain [267]. However, the concept is only demonstrated to be possible for image generation, and there is no clear evidence showing if essential information for vision tasks is still maintained in such a virtual domain.

### 5.2.3 Relation between Semantics and Geometry

Intuitively, vision tasks should share information, and joint learning should benefit each individual task, which also stands for semantic and geometry. On one hand, geometry carries a lot of semantic information. Knowing the shape of an object may directly tell the semantic type especially for categories having strong 3D structures, such as chair, plane, and car [202, 203]; and knowing the size and relative location of an object in a room can roughly tell us the semantic category (Sec. 4.1.3). One may also conjecture that geometry provides quite a lot of complementary semantic information compared to color images, and object recognition on RGB-D images would be easier and perform better. However, adding depth images as input together with the color image does not improve the performance much [230, 231], possibly because color-only baselines are tuned well, or depth images are not an ideal representation for semantics. More research is still required to fully understand how to extract semantic information effectively from the geometry.

On the other hand, knowing semantics could also improve geometry estimation. Boundaries of homogeneous semantic regions often indicate geometric discontinuity, which helps to predict sharper boundaries in the estimated geometry [88]. The semantic category also provides a strong prior, such that shape estimation for a certain semantic category (e.g., face, hand, plane, and chair) can leverage prior shape analysis (e.g., PCA) to simplify the problem and produce better results [123, 167, 131, 124]. Nevertheless, how semantics helps to improve geometry estimation in general is not clear especially for fine-grained details [36]. One possible argument is that the semantic understanding we have so far is still at a very rough level, which brings in only a vague shape prior. For example, knowing the object is a chair still leaves large inter-class shape variation. Ideally, the impact of semantics on geometry should be studied with fine-grained or instance level semantics.

# Bibliography

[1] Amazon mechanical turk. `https://www.mturk.com/`. Accessed: 2018-09-28.

[2] 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.

[3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 2001.

[4] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

[5] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016.

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

[7] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013.

[8] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

[9] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 2013.

[10] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: a richly annotated catalog of surface appearance. *TOG*, 2013.

[11] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[12] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision*, 110(1):2–13, 2014.

[13] A. Bhandari, M. Feigin, S. Izadi, C. Rhemann, M. Schmidt, and R. Raskar. Resolving multipath interference in kinect: An inverse problem approach. In *IEEE Sensors*, 2014.

[14] A. Bhandari, A. Kadambi, R. Whyte, C. Barsi, M. Feigin, A.A. Dorrington, and R. Raskar. Resolving multi-path interference in time-of-flight imaging via modulation frequency diversity and sparse regularization. *CoRR*, 2014.

[15] Irving Biederman. Perceiving real-world scenes. *Science*, 177(4043):77–80, 1972.

[16] Irving Biederman. On the semantics of a glance at a scene. In *Perceptual organization*, pages 213–253. Routledge, 2017.

[17] Irving Biederman, Robert J Mezzanotte, and Jan C Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive psychology*, 14(2):143–177, 1982.

[18] Michael Bleyer and Margrit Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *VISAPP (2)*, pages 415–422, 2008.

[19] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011.

[20] Matthew Brown and David G Lowe. Recognising panoramas. In *ICCV*, 2003.

[21] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 2007.

[22] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics*, 36(4):98, 2017.

[23] Ayan Chakrabarti, Jingyu Shao, and Greg Shakhnarovich. Depth from a single image by harmonizing overcomplete local network predictions. In *Advances in Neural Information Processing Systems*, pages 2658–2666, 2016.

[24] Angel 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. *International Conference on 3D Vision (3DV)*, 2017.

[25] 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.

[26] Yu-Wei Chao, Wongun Choi, Caroline Pantofaru, and Silvio Savarese. Layout estimation of highly cluttered indoor scenes using geometric and semantic cues. In *ICIAP*, 2013.

[27] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Advances in Neural Information Processing Systems*, pages 730–738, 2016.

[28] Weihai Chen, Haosong Yue, Jianhua Wang, and Xingming Wu. An improved edge detection algorithm for depth map inpainting. *Optics and Lasers in Engineering*, 55:69–77, 2014.

[29] Myung Jin Choi, Joseph J Lim, Antonio Torralba, and Alan S Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[30] Myung Jin Choi, Antonio Torralba, and Alan S Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 2012.

[31] Myung Jin Choi, Antonio Torralba, and Alan S Willsky. A tree-based context model for object recognition. *PAMI*, 2012.

[32] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Understanding indoor scenes using 3D geometric phrases. In *CVPR*, 2013.

[33] Michael Ciotta and Dimitrios Androutsos. Depth guided image completion for structure and texture synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 1199–1203. IEEE, 2016.

[34] James M. Coughlan and A.L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *ICCV*, 1999.

[35] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[36] 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.

[37] Tim Davis. Csparse. *Society for Industrial and Applied Mathematics, Philadephia, PA*, 2006.

[38] Luca Del Pero, Joshua Bowdish, Bonnie Kermgard, Emily Hartley, and Kobus Barnard. Understanding bayesian rooms using composite 3D object models. In *CVPR*, 2013.

[39] Erick Delage, Honglak Lee, and Andrew Y. Ng. Automatic single-image 3D reconstructions of indoor manhattan world scenes. In *ISRR*, 2005.

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[41] Chaitanya Desai, Deva Ramanan, and Charless C Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 2011.

[42] James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, pages 291–298, 2006.

[43] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2015.

[44] David Doria and Richard J Radke. Filling large holes in lidar data by inpainting depth gradients. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 65–72. IEEE, 2012.

[45] 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.

[46] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *SIGGRAPH Asia*, 2017.

[47] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: Real-time performance capture of challenging scenes. *SIGGRAPH*, 2016.

[48] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[49] Krista A Ehinger, Wendy J Adams, Erich W Graf, James H Elder, Karthikeyan Vaiapury, Balamuralidhar Purushothaman, Arpan Pal, Swapna Agarwal, Brojeshwar Bhowmick, Ivet Rafegas, et al. Local depth edge detection in humans and deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2681–2689, 2017.

[50] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[51] 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.

[52] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[53] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[54] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[55] S. R. Fanello, I. Gori, G. Metta, and F. Odone. Keep it simple and sparse: Real-time action recognition. *JMLR*, 2013.

[56] S. R. Fanello, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, S.B. Kang, and T. Paek. Learning to be a depth camera for close-range human capture and interaction. *ACM SIGGRAPH and Transaction On Graphics*, 2014.

[57] S. R. Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. Orts Escolano, D. Kim, and S. Izadi. Hyperdepth: Learning depth from structured light without matching. In *CVPR*, 2016.

[58] Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. Ultrastereo: Efficient learning-based matching for active stereo systems. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6535–6544. IEEE, 2017.

[59] Sean Ryan Fanello, Julien PC Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip L Davidson, and Shahram Izadi. Low compute and fully parallel computer vision with hashmatch. In *ICCV*, pages 3894–3903, 2017.

[60] S.R. Fanello, I. Gori, G. Metta, and F. Odone. One-shot learning for real-time action recognition. In *IbPRIA*, 2013.

[61] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[62] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[63] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[64] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

[65] David Ferstl, Christian Reinbacher, Rene Ranftl, Matthias Rüther, and Horst Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 993–1000. IEEE, 2013.

[66] Sanja Fidler, Sven J Dickinson, and Raquel Urtasun. 3D object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, 2012.

[67] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[68] Stefan Florczyk. *Robot vision: Video-based indoor exploration with autonomous and mobile robots.* John Wiley & Sons, 2006.

[69] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 2008.

[70] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002.

[71] Axel Furlan, David Miller, Domenico G. Sorrenti, Li Fei-Fei, and Silvio Savarese. Free your camera: 3D indoor scene understanding from arbitrary camera motion. In *BMVC*, 2013.

[72] Silvano Galliani and Konrad Schindler. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5479–5487, 2016.

[73] Ravi Garg, Vijay Kumar BG, 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.

[74] Josselin Gautier, Olivier Le Meur, and Christine Guillemot. Depth-based image completion for view synthesis. In *3DTV Conference: The True Vision-capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1–4. IEEE, 2011.

[75] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011.

[76] Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5248–5257, 2017.

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

[78] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[79] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2.6, page 7, 2017.

[80] Xiaojin Gong, Junyi Liu, Wenhui Zhou, and Jilin Liu. Guided depth enhancement via a fast marching method. *Image and Vision Computing*, 31(10):695–703, 2013.

[81] 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.

[82] Michelle R Greene and Aude Oliva. The briefest of glances: The time course of natural scene understanding. *Psychological Science*, 20(4):464–472, 2009.

[83] Ruiqi Guo and Derek Hoiem. Beyond the line of sight: labeling the underlying surfaces. In *ECCV*, 2012.

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

[85] Abhinav Gupta, Scott Satkin, Alexei A. Efros, and Martial Hebert. From scene geometry to human workspace. In *CVPR*, 2011.

[86] 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.

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

[88] Christian Haene, Christopher Zach, Andrea Cohen, and Marc Pollefeys. Dense semantic 3d reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1730–1743, 2017.

[89] Maciej Halber and Thomas Funkhouser. Fine-to-coarse global registration of rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[90] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016, 2016.

[91] Feng Han and Song-Chun Zhu. Bottom-up/top-down image parsing with attribute grammar. *PAMI*, 2009.

[92] Yudeog Han, Joon-Young Lee, and In So Kweon. High quality shape from a single rgb-d image under uncalibrated natural illumination. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1617–1624, 2013.

[93] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

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

[95] Christian Hane, Lubor Ladicky, and Marc Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 381–389, 2015.

[96] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[97] Simon Hawe, Martin Kleinsteuber, and Klaus Diepold. Dense disparity maps from sparse disparity measurements. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2126–2133. IEEE, 2011.

[98] Elad Hazan, Kfir Y. Levy, and Shai Shalev-Shwartz. On graduated optimization for stochastic non-convex problems. In *ICML*, 2016.

[99] Kaiming He, Huiwen Chang, and Jian Sun. Rectangling panoramic images via warping. *TOG*, 2013.

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

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

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

[103] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*, 2008.

[104] Daniel Herrera, Juho Kannala, Janne Heikkilä, et al. Depth map inpainting under a second-order smoothness prior. In *Scandinavian Conference on Image Analysis*, pages 555–566. Springer, 2013.

[105] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008.

[106] Derek Hoiem. *Seeing the world behind the image: spatial layout for 3D scene understanding*. PhD thesis, Carnegie Mellon University, 2007.

[107] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. *ACM transactions on graphics (TOG)*, 24(3):577–584, 2005.

[108] Derek Hoiem, Alexei A Efros, and Martial Hebert. Geometric context from a single image. In *ICCV*, 2005.

[109] Derek Hoiem, Alexei A Efros, and Martial Hebert. Closing the loop in scene interpretation. In *CVPR*, 2008.

[110] Derek Hoiem, Alexei A Efros, and Martial Hebert. Putting objects in perspective. *IJCV*, 2008.

[111] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2013.

[112] Paul VC Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, volume 73, 1959.

[113] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 92–101. IEEE, 2016.

[114] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[115] Intel realsense d415. `https://click.intel.com/intelr-realsensetm-depth-camera-d415.html`. Accessed: 2018-02-28.

[116] Intel realsense d435. `https://click.intel.com/intelr-realsensetm-depth-camera-d435.html`. Accessed: 2018-02-28.

[117] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[118] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

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

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

[121] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[122] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, 2016.

[123] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[124] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*, 2018.

[125] 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.

[126] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[127] Wajahat Kazmi, Sergi Foix, Guillem Alenyà, and Hans Jørgen Andersen. Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: Analysis and comparison. *ISPRS journal of photogrammetry and remote sensing*, 88:128–146, 2014.

[128] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *CoRR, vol. abs/1703.04309*, 2017.

[129] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel RealSense Stereoscopic Depth Cameras. *CVPR Workshops*, 2017.

[130] Sameh Khamis, Sean Fanello, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, and Shahram Izadi. Stereonet: Guided hierarchical refinement for edge-aware depth prediction. In *ECCV*, 2018.

[131] Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2540–2548, 2015.

[132] Martin Kiechle, Simon Hawe, and Martin Kleinsteuber. A joint intensity and depth co-sparse analysis model for depth map super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1545–1552, 2013.

[133] Byungsoo Kim, Pushmeet Kohli, and Silvio Savarese. 3D scene understanding by Voxel-CRF. In *ICCV*, 2013.

[134] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 15–18. IEEE, 2006.

[135] Jan J Koenderink, Andrea J Van Doorn, and Astrid ML Kappers. Surface perception in pictures. *Attention, Perception, & Psychophysics*, 52(5):487–496, 1992.

[136] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001.

[137] Kurt Konolige. Projected texture stereo. In *ICRA*, 2010.

[138] J. Kowalczuk, E. T. Psota, and L. C. Perez. Real-time stereo matching on cuda using an iterative refinement method for adaptive support-weight correspondences. *IEEE Transactions on Circuits and Systems for Video Technology*, 2013.

[139] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[140] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[141] Mandar Kulkarni and Ambasamudram N Rajagopalan. Depth inpainting by tensor voting. *JOSA A*, 30(6):1155–1165, 2013.

[142] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *CVPR*, 2015.

[143] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[144] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017.

[145] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip HS Torr. Graph cut based inference with co-occurrence statistics. In *European Conference on Computer Vision*, pages 239–253. Springer, 2010.

[146] 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.

[147] Crystian Wendel M Leão, Joao Paulo Lima, Veronica Teichrieb, Eduardo S Albuquerque, and Judith Kelner. Altered reality: Augmenting and diminishing reality in real time. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 219–220. IEEE, 2011.

[148] David C. Lee, Abhinav Gupta, Martial Hebert, and Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010.

[149] David C. Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009.

[150] Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

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

[152] Jun Li, Reinhard Klein, and Angela Yao. A two-streamed network for estimating fine-scaled depth maps from single rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3372–3380, 2017.

[153] Li-Jia Li, Richard Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009.

[154] Li-Jia Li, Hao Su, Eric P Xing, and Fei-Fei Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[155] Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J Guibas. Joint embeddings of shapes and images via cnn image purification. *ACM Transactions on Graphics (TOG)*, 2015.

[156] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Linbo Qiao, Wei Chen, Li Zhou, and Jianfeng Zhang. Learning deep correspondence through prior and posterior feature constancy. *arXiv preprint arXiv:1712.01039*, 2017.

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

[158] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[159] Pamela Lipson, Eric Grimson, and Pawan Sinha. Configuration based scene classification and image indexing. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1007–1013. IEEE, 1997.

[160] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016.

[161] Junyi Liu and Xiaojin Gong. Guided depth enhancement via anisotropic diffusion. In *Pacific-Rim Conference on Multimedia*, pages 408–417. Springer, 2013.

[162] Junyi Liu, Xiaojin Gong, and Jilin Liu. Guided inpainting and filtering for kinect depth maps. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2055–2058. IEEE, 2012.

[163] Lee-Kang Liu, Stanley H Chan, and Truong Q Nguyen. Depth reconstruction from sparse samples: Representation, algorithm, and sampling. *IEEE Transactions on Image Processing*, 24(6):1983–1996, 2015.

[164] Miaomiao Liu, Xuming He, and Mathieu Salzmann. Building scene models by completing and hallucinating depth and semantics. In *European Conference on Computer Vision*, pages 258–274. Springer, 2016.

[165] Tianqiang Liu, Siddhartha Chaudhuri, Vladimir G Kim, Qixing Huang, Niloy J Mitra, and Thomas Funkhouser. Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics (TOG)*, 33(6):211, 2014.

[166] 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.

[167] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

[168] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[169] Jiajun Lu and David Forsyth. Sparse depth super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2245–2253, 2015.

[170] Jiangbo Lu, Dongbo Min, Ramanpreet Singh Pahwa, and Minh N Do. A revisit to mrf-based depth map super-resolution and enhancement. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 985–988. IEEE, 2011.

[171] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[172] Bernhard Lubor Ladickỳ, Zeisl, Marc Pollefeys, et al. Discriminatively trained dense surface normal estimation. In *European conference on computer vision*, pages 468–484. Springer, 2014.

[173] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *arXiv preprint arXiv:1709.07492*, 2017.

[174] Oisin Mac Aodha, Neill DF Campbell, Arun Nair, and Gabriel J Brostow. Patch based synthesis for single depth image super-resolution. In *European Conference on Computer Vision*, pages 71–84. Springer, 2012.

[175] Mona Mahmoudi and Guillermo Sapiro. Sparse representations for range data restoration. *IEEE Transactions on Image Processing*, 21(5):2909–2915, 2012.

[176] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Joshua B Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, 2013.

[177] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* MIT press, 1982.

[178] Kiyoshi Matsuo and Yoshimitsu Aoki. Depth image enhancement using local tangent plane approximations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3574–3583, 2015.

[179] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[180] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[181] 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*, volume 2, 2012.

[182] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–293. International Society for Optics and Photonics, 1995.

[183] Ennio Mingolla and James T Todd. Perception of solid shape from shading. *Biological cybernetics*, 53(3):137–151, 1986.

[184] Mitsuba physically based renderer. `http://www.mitsuba-renderer.org/`.

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

[186] Suryanarayana M Muddala, Marten Sjostrom, and Roger Olsson. Depth-based inpainting for disocclusion filling. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014*, pages 1–4. IEEE, 2014.

[187] N. Naik, A. Kadambi, C. Rhemann, S. Izadi, R. Raskar, and S.B. Kang. A light transport model for mitigating multipath interference in TOF sensors. *CVPR*, 2015.

[188] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM transactions on graphics (TOG)*, 24(3):536–543, 2005.

[189] Thacker Neil and Cootes Tim. Multi-resolution methods and graduated non-convexity. In *Vision Through Optimization*, 1997.

[190] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[191] 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.

[192] Kai Ni, Anitha Kannan, Antonio Criminisi, and John Winn. Epitomic location recognition. In *CVPR*, 2008.

[193] H K Nishihara. Prism: A practical mealtime imaging stereo matcher. In *Intelligent Robots: 3rd Intl Conf on Robot Vision and Sensory Controls*, volume 449, pages 134–143. International Society for Optics and Photonics, 1984.

[194] Structure io. `https://structure.io/`. Accessed: 2018-09-28.

[195] Jiahao Pang, Wenxiu Sun, JS Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *International Conf. on Computer Vision-Workshop on Geometry Meets Deep Learning (ICCVW 2017)*, volume 3.9, 2017.

[196] Jaesik Park, Hyeongwoo Kim, Yu-Wing Tai, Michael S Brown, and Inso Kweon. High quality depth map upsampling for 3d-tof cameras. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1623–1630. IEEE, 2011.

[197] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 143–152, 2017.

[198] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[199] Luca Del Pero, Joshua C. Bowdish, Daniel Fried, Bonnie D. Kermgard, Emily L. Hartley, and Kobus Barnard. Bayesian geometric modelling of indoor scenes. In *CVPR*, 2012.

[200] Luca Del Pero, Jinyan Guan, Ernesto Brau, Joseph Schlecht, and Kobus Barnard. Sampling bedrooms. In *CVPR*, 2011.

[201] Planner5D home design and interior decor in 2d and 3d. `http://www.planner5d.com/`.

[202] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[203] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[204] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV*, 2007.

[205] Zhile Ren and Erik B. Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. In *CVPR*, 2016.

[206] Zhile Ren and Erik B Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. CVPR, 2016.

[207] 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.

[208] Lawrence G. Roberts. *Machine perception of 3-D solids.* PhD thesis, Massachusetts Institute of Technology, 1963.

[209] Lawrence G Roberts. *Machine perception of three-dimensional solids.* PhD thesis, Massachusetts Institute of Technology, 1963.

[210] 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.

[211] Anirban Roy and Sinisa Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016.

[212] Bryan C Russell and Antonio Torralba. Building a database of 3D scenes from user annotations. In *CVPR*, 2009.

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

[214] Scott Satkin and Martial Hebert. 3DNN: Viewpoint invariant 3D geometry matching for scene understanding. In *ICCV*, 2013.

[215] Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3D models. In *BMVC*, 2012.

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

[217] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[218] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[219] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014.

[220] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[221] Alexander G Schwing, Sanja Fidler, Marc Pollefeys, and Raquel Urtasun. Box in the box: Joint 3D layout and object reasoning from single images. In *ICCV*, 2013.

[222] Alexander G. Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Efficient structured prediction for 3D indoor scene understanding. In *CVPR*, 2012.

[223] Alexander G Schwing and Raquel Urtasun. Efficient exact inference for 3D indoor scene understanding. In *ECCV*, 2012.

[224] Elham Shabaninia, Ahmad Reza Naghsh-Nilchi, and Shohreh Kasaei. High-order markov random field for single depth image super-resolution. *IET Computer Vision*, 2017.

[225] Amit Shaked and Lior Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. *CoRR, vol. abs/1701.00165*, 2017.

[226] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009.

[227] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012.

[228] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

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

[231] Shuran Song and Jianxiong Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *CVPR*, 2016.

[232] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[233] 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.

[234] Erik Sudderth, Antonio Torralba, William Freeman, and Alan Willsky. Describing visual scenes using transformed dirichlet processes. In *NIPS*, 2005.

[235] Erik B Sudderth and Michael I Jordan. Shared segmentation of natural scenes using dependent pitman-yor processes. In *NIPS*, 2008.

[236] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, 2005.

[237] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Depth from familiar objects: A hierarchical model for 3D scenes. In *CVPR*, 2006.

[238] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Describing visual scenes using transformed objects and parts. *IJCV*, 2008.

[239] Supasorn Suwajanakorn, Carlos Hernandez, and Steven M Seitz. Depth from focus with your mobile phone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3497–3506, 2015.

[240] Richard Szeliski. *Computer Vision: Algorithms and Applications.* Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[241] Vladimir Tankovich, Michael Schoenberg, Sean Ryan Fanello, Adarsh Kowdle, Christoph Rhemann, Max Dzitsiuk, Mirko Schmidt, Julien Valentin, and Shahram Izadi. Sos: Stereo matching in o(1) with slanted support windows. *IROS*, 2018.

[242] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, and Jamie Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *SIGGRAPH*, 2016.

[243] Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. Articulated distance fields for ultra-fast tracking of hands interacting. *Siggraph Asia*, 2017.

[244] Jay M Tenenbaum and AP Witkin. On the role of structure in vision. *Human and machine vision*, pages 481–543, 1983.

[245] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 2011.

[246] Ali K Thabet, Jean Lahoud, Daniel Asmar, and Bernard Ghanem. 3d aware correction and completion of depth maps in piecewise planar scenes. In *Asian Conference on Computer Vision*, pages 226–241. Springer, 2014.

[247] Sebastian Thrun. Simultaneous localization and mapping. In *Robotics and cognitive approaches to spatial mapping*, pages 13–41. Springer, 2007.

[248] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[249] Antonio Torralba. Contextual influences on saliency. In *Neurobiology of attention*, pages 586–592. Elsevier, 2005.

[250] Ivana Tosic and Sarah Drewes. Learning joint intensity-depth sparse representations. *IEEE Transactions on Image Processing*, 23(5):2122–2132, 2014.

[251] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[252] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[253] Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 2005.

[254] Jonas Uhrig, N Schneider, L Schneidre, U Franke, Thomas Brox, and A Geiger. Sparsity invariant cnns. In *IEEE International Conference on 3D Vision (3DV)*, 2017.

[255] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[256] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[257] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997.

[258] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2012.

[259] Huayan Wang, Stephen Gould, and Daphne Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *ECCV*, 2010.

[260] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. *ECCV*, 2018.

[261] Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. The global patch collider. *CVPR*, 2016.

[262] 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.

[263] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *ICCV*, 2013.

[264] Robert J Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *Image Understanding Systems and Industrial Applications I*, volume 155, pages 136–144. International Society for Optics and Photonics, 1979.

[265] Tianfu Wu and Song-Chun Zhu. A numerical study of the bottom-up and top-down inference processes in and-or graphs. *IJCV*, 2011.

[266] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[267] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[268] 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.

[269] Jianxiong Xiao, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012.

[270] Jianxiong Xiao and Yasutaka Furukawa. Reconstructing the world's museums. *IJCV*, 2014.

[271] 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.

[272] 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*, 2013.

[273] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[274] Jianxiong Xiao, Bryan C Russell, and Antonio Torralba. Localizing 3D cuboids in single-view images. In *NIPS*, 2012.

[275] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

[276] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[277] Wuyuan Xie, Miaohui Wang, Xianbiao Qi, and Lei Zhang. 3d surface detail enhancement from a single normal map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2325–2333, 2017.

[278] Hongyang Xue, Shengming Zhang, and Deng Cai. Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26(9):4311–4320, 2017.

[279] Kuk-Jin Yoon and In-So Kweon. Locally adaptive support-weight approach for visual correspondence search. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 924–931. IEEE, 2005.

[280] Kuk-Jin Yoon and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):650–656, 2006.

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

[282] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. In *arXiv*, 2015.

[283] Lap-Fai Yu, Sai-Kit Yeung, Yu-Wing Tai, and Stephen Lin. Shading-based shape refinement of rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1415–1422, 2013.

[284] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. In *ACM Transactions on Graphics (TOG)*, volume 30.4, page 86. ACM, 2011.

[285] Stella Yu, Hao Zhang, and Jitendra Malik. Inferring spatial layout from a single image via depth-ordered grouping. In *IEEE Workshop on Perceptual Organization in Computer Vision*, 2008.

[286] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4353–4361. IEEE, 2015.

[287] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1592–1599, 2015.

[288] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016.

[289] Hai-Tao Zhang, Jun Yu, and Zeng-Fu Wang. Probability contour guided depth map inpainting and superresolution using non-local total generalized variation. *Multimedia Tools and Applications*, pages 1–18, 2017.

[290] 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.

[291] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

[292] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.

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

[294] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[295] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. Activestereonet: End-to-end self-supervised learning for active stereo systems. *ECCV*, 2018.

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

[297] 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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[298] Yinda Zhang, Jianxiong Xiao, James Hays, and Ping Tan. Framebreak: Dramatic image extrapolation by guided shift-maps. In *CVPR*, 2013.

[299] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 2017.

[300] Yibiao Zhao and Song chun Zhu. Image parsing with stochastic scene grammar. In *NIPS*, 2011.

[301] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013.

[302] Yibiao Zhao and Song-Chun Zhu. Integrating function , geometry , appearance for scene parsing. In *manuscript*, 2014.

[303] 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*, 2013.

[304] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*, 2017.

[305] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2.6, page 7, 2017.

[306] Yifan Zuo, Qiang Wu, Jian Zhang, and Ping An. Explicit edge inconsistency evaluation model for color-guided depth map enhancement. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.