USING GENERATIVE DEEP LEARNING TO CREATE HIGH-QUALITY MODELS FROM 3D SCANS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Angela Dai
August 2018

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Pat Hanrahan, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Silvio Savarese

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Thomas Funkhouser,

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

With recent developments in both commodity range sensors as well as mixed reality devices (such as the Microsoft Kinect, Microsoft Hololens, Intel RealSense, Apple iPhone X, etc.), capturing and creating 3D models of the world around us has become increasingly important. As the world around us lives in a three-dimensional space, such 3D models will not only facilitate capture and display for content creation but also provide a basis for fundamental scene understanding, from semantic understanding to virtual interactions.

Leveraging data from commodity range sensors to reconstruct 3D scans of a scene has shown significant promise towards 3D model creation of real-world environments. However, the quality of reconstructed 3D scans has yet to reach that of artist-created 3D models – in particular, 3D scans always suffer from incompleteness, due to the many occlusions in real-world scenes as well as physical limitations of range sensors. Such incomplete 3D models are both unsuitable visually, and moreover, provides only a limited basis for higher-level scene reasoning (e.g., virtual interactions in mixed reality scenarios will not be accurate in unknown or missing regions).

This work focuses on the task of scan completion, that is, from a 3D scan with partial geometry due to occlusions and scanning patterns, we aim to infer the missing geometry. We introduce a generative formulation for scan completion, leveraging deep learning techniques to create high-quality, complete models from 3D scans. We approach this problem as a conditional generative task, where we condition on an input partial scan and aim to learn 'part'-wise similarity between scans to infer the complete model. First, we begin by focusing on the more constrained problem of completing scans of isolated shapes. We then expand upon this to design a generative

approach for completion of general 3D scans of arbitrary scenes, directly addressing the challenge of varying scene sizes in 3D. This not only provides scan completion at scale, producing geometrically complete 3D models, but also provides a basis for higher-level scene reasoning such as that required for virtual interactions or physical simulations.

Acknowledgments

My time at Stanford has been uniquely defined by both the support and advice that Pat has given me as my advisor, as well as an equal amount of freedom to pursue my ideas and drive them forward. He makes research and running a group look bright and effortless; it's truly admirable, and I can only hope to strive for a fraction of it. I'm very grateful to have had the opportunity to have Pat as my advisor.

I also wish to thank Tom Funkhouser, Silvio Savarese, Leo Guibas, and Dan Yamins for serving on my orals committee. Tom has given me support and advice from my undergraduate time at Princeton, and I am both happy and lucky to have had the chance to work with him again. Leo offers great wisdom, always providing new perspectives, and advice from Silvio is highly valued. Without my friends and collaborators, I would not be here. Matthias Nießner's guidance has helped shaped and inspired my pursuit into capturing 3D models of the world, and I'm so grateful for the time and mentorship of a clueless first-year student. Manolis Savva and Angel Chang have contributed invaluably towards both research ideas and projects, as well as helped to teach me a great deal about making research happen – and planning ahead for it. I'd also like to thank the amazing set of people who have made day-to-day research much more fun along with their insightful advice: Michael Zollhöfer, Justus Thies, Christian Theobalt, Daniel Ritchie, and Feng Xie (who helped jumpstart my interest in graphics many years ago).

I would also like to thank Professor Michael J. Flynn, whose Stanford Graduate Fellowship has supported me throughout my time at Stanford, and given me invaluable freedom to explore my ideas and research. I'm very grateful for the financial support from Google Tango, and in particular Jürgen Strum and Martin Bokeloh,

who have also become wonderful collaborators.

Finally, thank you to Mom, Dad, and Adam for your love and support. You've been my strongest supporters, and I'm so grateful for your unceasing support and encouragement throughout my endeavors.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Although 2D media abounds in the billions of images and videos we capture and consume, the reality behind them lives in three-dimensional space. In taking a photo, we capture a 2D projection of the 3D environment around us. And so in order to create content – whether it is representative of the real world (e.g., recreating New York in the *Avengers* film) or largely fantastical (e.g., the floating landscapes of the film *Avatar*) – we must consider this problem of creating and modeling environments in 3D. A 3D model of an environment enables not only free-form exploration from arbitrary viewpoints but also paves the way for potential interactions in a game-like environment.

Thus a large focus in recent computer graphics is on the problem of content creation. For instance, designing 3D content for movies or games is an incredibly time-consuming task often involving hundreds or thousands of artists working for several months on end. Moreover, with recent developments in mixed reality technologies and platforms comes an increasing need for 3D content, since with augmented and virtual reality we need to know the scale and location of objects in the scene in order to both provide realistic visuals as well as enable higher-level reasoning about the scene. For instance, we must know the scene geometry in order to have virtual pokémon walk under or over tables or other obstacles rather than pass directly through them, or to enable a household robot to place a cup on top of a table. One approach towards mitigating this content creation problem is through capturing high-quality 3D models

of real-world environments, which could then be directly used for these mixed reality scenarios or higher-level scene understanding.

In fact, recent advances in commodity range sensors, which provide color and depth video data at 30Hz, and have shown incredible promise towards potentially creating such 3D models of real-world environments. However, the quality of such reconstructed 3D scans has yet to reach that of artist-created 3D models – in particular, there is markedly noticeable difference in geometry quality. The geometry of 3D scans is almost always lacking: in addition to potential tracking errors and physical and resolution limitations from sensors, there are always occlusions in complex scenes. Such occlusions make it practically impossible to capture fully complete models of arbitrary real-world scenes. This incompleteness makes the models unsuitable, as it is both visually lacking, and moreover, provides only a limited basis for higher-level scene reasoning (e.g., virtual interactions in mixed reality scenarios will not be accurate in unknown or missing regions). Thus we look to create *geometrically complete* 3D models of real-world environments, i.e. representing the physical reality behind captured measurements.

Concurrent to advances in 3D scanning and reconstruction is the growth in availability of 3D CAD models, e.g., millions of 3D models freely available in the Trimble 3D Warehouse[1]. While these models also remain far from the quality of professional artist created content – often very simplistic representations – they are nonetheless geometrically complete. We thus aim to leverage this information about the construction of complete (albeit unrealistic) models using machine learning techniques to learn to infer the complete geometry underlying the observations underlying a partial 3D scan.

Generating these complete models is the first step towards commodity 3D scanning for content creation and consumer-grade mixed reality. Completing scans nonetheless already opens up a variety of possibilities for augmented reality, with the potential for applications such as comprehensive collision detection or inpainting for interior redesign applications (e.g., completing missing regions when cutting out an object and moving it around). Towards this goal, this dissertation focuses on leveraging

[1] `https://3dwarehouse.sketchup.com/`

generative deep learning techniques to learn to create high-quality, complete models from 3D scans.

Figure 1.1: While significant progress has been made in 3D scanning and reconstruction, achieving reconstructed 3D scans with faithful geometric and visual fidelity to the original physical environments (top), the geometric quality remains noticeably behind that of artist-created content (bottom; images by Blender Foundation, © copyright Blender Foundation | www.sintel.org). In particular, note the incomplete geometry, particularly from differing viewpoints (top left vs top right)

1.1 Why 3D Scans?

Acquiring a 3D representation of an environment not only allows us to generate arbitrary image views from any camera position, but also provides precise information about the scale and location of points in the scene. Such information is requisite for myriad applications, such as robots or cars which need to navigate through complex 3D environments and thus must be able to construct an accurate spatial map. Furthermore, in addition to the vast amounts of time and manual effort spent on

current content creation pipelines in the entertainment industry, the burgeoning development of augmented and virtual reality provides a significant demand for not only 3D content but 3D models of the environments around us. Augmented reality aims to augment the physical, real world with various types of information, and so must know exactly where objects are in order to tag them with perceptual information. With virtual reality, the aim is instead to mask out the real world and provide an immersive experience of any environment, whether realistic or fantastical. Creating 3D models of our environments not only facilitates creation of shared virtual spaces, but since the current, real environment nonetheless exists and must be accounted for, we must also obtain an understanding of the real-world surrounding environment.

Recent advances in 3D scanning and reconstruction have opened an exciting avenue to address this problem of creating 3D models of real-world environments. These approaches take as input a color video stream, and often a depth input stream, of observations of an environment. They then perform camera tracking to optimize for the camera poses for each frame in the input video sequence(s) as well as reconstruction to fuse the observations from these camera locations into a unified 3D model of the observed surfaces in the scene. In particular, since the introduction of the Microsoft Kinect, commodity range sensors, providing both color and depth video stream data at real-time rates, have enabled 3D scanning and reconstruction for the consumer market. Bringing these 3D scans to high-quality 3D models would enable a new world of possibilities for applications like mixed reality. In particular, a geometrically complete 3D model in this context will enable a multitude of applications, including the basis for realistic physical interactions for virtual agents (e.g., game characters), autonomous agents (e.g., robots), or physical media (e.g., sound).

1.2 The Challenge of Incompleteness

With the promise of 3D scanning and reconstruction as an avenue for 3D content creation, we developed a real-time 3D reconstruction approach, BundleFusion [18], which performs global camera pose optimization in tandem with scene reconstruction, enabling efficient capture of 3D scans at quality comparable to state-of-the-art offline

approaches. Unfortunately, while the resulting 3D scans from state-of-the-art 3D reconstruction approaches show strong visual and geometric fidelity to the original, physical environments, scan quality remains far behind that of content created by professional artists. In Figure 1.1, we see at the top a 3D scan reconstructed by BundleFusion. From a top-down view of the captured scan, we can see and recognize all the major objects in the scene; however, with just a change in camera view we can see that the scanned model is in fact quite incomplete in various regions. In particular, the chair legs are missing as they are thin and metallic (as chair legs often are), and highly reflective (or refractive) surfaces register little-to-no depth information with depth sensors, which are light-based. Under the table is difficult to scan (although not impossible) due to very dark lighting (lack of visual features), but moreover adds a significant cost in user time and maneuvering – even then, regions behind the desktop computer would remain occluded. Additionally, the seat of the chair on the right and the walls behind the monitors are missing geometry due to occlusions from the table and monitors, respectively. In contrast, the 3D model created by a professional artist (Figure 1.1, bottom) contains clean, sharp geometry for the entirety of the scene.

Furthermore, in an effort to collect a large-scale dataset of 3D scans – for the purpose of powering 3D semantic inference with deep learning techniques (see Appendix B for more detail) – myself and several other students captured over $1,500$ scans, reconstructed with a state-of-the-art reconstruction algorithm. While about a third of us were "experts" in reconstruction in that we knew how to capture data well-suited to state-of-the-art tracking algorithms, the resulting set of 3D scans nonetheless contained significant missing regions, as seen in Figure 1.2.

This incompleteness of 3D scanned models is a significant barrier for use in content creation scenarios or for reasoning about environments. Exploration of an environment would fail to be immersive with the occasional missing geometry; virtual interactions would also be unreliable in such regions (e.g., a virtual pet walking through missing chair legs); various physical simulations like re-lighting or producing sound effects would not be accurate without the full scene geometry; and reasoning about scanned environments (e.g, inferring semantics or detecting object instances) is substantially more challenging when data is missing. Thus we address this challenge

of incompleteness in this dissertation, with the aim of inferring the physical reality underlying the measurements captured by a 3D scan.

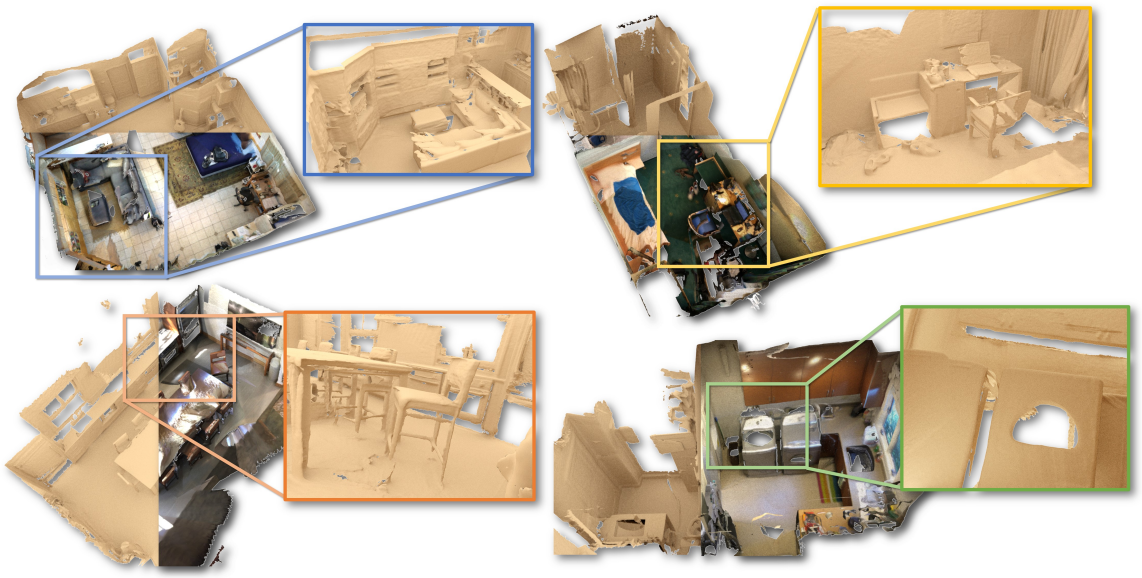

Figure 1.2: The incompleteness of 3D scans is exemplified in the data acquisition for our ScanNet [16] dataset, where scans even from expert users contain significant missing regions.

1.3 Learning to Complete 3D Scans

In this dissertation, we formulate the problem of scan completion as a generative task conditioned on an input partial scan. We leverage generative deep learning techniques to learn models which exploit part-wise similarity within and between scans of scenes to infer missing geometry. While individual scenes are typically distinct from each other as a whole, they are typically composed of similar components, e.g., furniture such as chairs, tables, cabinets, beds, etc, as well as similar structure, e.g., chairs often near tables.

We approach this problem as a conditional generative task, where we condition on an input partial scan and aim to learn 'part'-wise similarity between scans to infer the complete model. While scenes are typically quite distinct from each other as a whole,

they are typically composed of similar components, e.g., furniture such as chairs, tables, cabinets, beds, etc, as well as similar structure, e.g., chairs often near tables. In this dissertation, we develop several approaches to learn models designed to from such similarities, additionally exploiting the significant quantities complete synthetic models available. We first focus on the simpler problem of scans of isolated shapes, and then tackle the problem of large-scale scenes. Generating large scale output is a significant challenge, particularly for deep learning in 3D, due to cubic growth in resolution – previous such approaches have thus limited their scope to single objects or single image frames rather than full scenes.

In particular, this dissertation makes the following contributions:

- *An end-to-end approach for generating a complete 3D model from an input RGB-D video sequence.* From an input RGB-D video sequence, it provides a real-time 3D reconstruction algorithm for efficiently capturing globally consistent 3D scans. It then shows how to learn a generative model conditioned on these scans to create high-quality output meshes.

- *Generative model for completing scans of shapes.* It shows how to formulate the problem of completing scans of isolated objects as a generative task, conditioned on an input partial scan. The method is built to operate on a 3D scan based on common 3D reconstruction representations, and maps partial scans into an embedding space where correlation with complete models are learned, producing complete 3D meshes as output.

- *Generative model for completing scans of scenes.* It provides a new algorithm for completing large-scale (e.g., room- to building-floor-scale) scans of scenes, in particular addressing the challenge of varying scene sizes in 3D. Leveraging a combination of fully-convolutional networks and a coarse-to-fine hierarchy, this approach is the first to demonstrate scan completion at scale on arbitrary scenes.

- *Domain transfer from synthetic to real data.* It shows that the approaches proposed are capable of domain adaptation from synthetic scans to real-world

scans, achieving convincing scan completion for real-world scans where there is no available ground truth.

1.4 Dissertation Structure

This dissertation is organized following the pipeline from input RGB-D sequence to complete mesh generation.

Chapter 2 provides an overview of current real-time 3D scanning and reconstruction methods as well as various approaches to scan completion. It further discusses the common data representations for 3D scans as well as several synthetic CAD model datasets which provide complete albeit simplistic geometry.

Chapter 3 describes how to efficiently acquire globally consistent 3D scans of environments using BundleFusion for camera pose optimization and surface reconstruction.

Chapter 4 focuses on completing scans of shapes. It provides a formulation of shape completion as a generative task, leveraging a database of CAD model priors to produce high-quality complete meshes of shapes.

Chapter 5 focuses on completing scans of scenes. It addresses the challenge of varying-sized, large-scale scenes, which we see in practice with real scans.

Chapter 6 discusses the resulting pipeline from scan acquisition to complete mesh generation, and suggests several avenues for future work towards the aim of commodity 3D scanning for content creation.

Chapter 2

Background

Creating 3D models of the real world has seen an extensive amount of research over the last couple of decades through the field of 3D reconstruction. The introduction of commodity depth sensors in recent years has spurred momentum in real-time reconstruction, as these sensors are capable of providing live streams of both depth and color data. Since commodity sensors are most suited towards potential end applications like consumer-grade mixed reality devices, we focus on this scenario.

However, the incompleteness of raw 3D reconstruction results is ill-suited towards immediate use in practical applications such as content creation or mixed reality. Various approaches towards inferring such missing geometry have been developed, from more traditional optimization based hole-filling approaches to methods leveraging recent advances in machine learning – particularly generative deep learning.

This chapter provides an overview following the pipeline of scan acquisition to complete mesh generation. In particular, it first discusses approaches towards real-time 3D reconstruction for scan acquisition as well as the data representations used to represent 3D scans. It then gives a brief overview of the synthetic counterparts of these 3D scans, i.e., synthetic CAD models, for which there is complete geometric information and an opportunity to learn how to generate complete structures from partial observations. Finally, it provides a discussion of various methods for completing 3D scans.

2.1 Real-Time 3D Reconstruction

Here, we focus on the scenario of real-time reconstruction with commodity handheld RGB-D sensors – well-suited both for portable, efficient scan capture as well as applications such as augmented or virtual reality. Such reconstruction approaches take as input a RGB-D video sequence and produce as output a 3D representation of the scanned environment. Note that for our aim of scan completion we do not strictly require a reconstruction approach that runs in real time; however, in practice the reconstruction results can reach that of offline methods, and the efficiency of capture is a significant gain in a tractable pipeline from reconstruction to completion. There are two major challenges in the 3D reconstruction process: scene reconstruction and camera tracking. With scene reconstruction, we refer specifically to the creation of a 3D model from a sequence of color and depth images along with the camera poses they were taken from, i.e., fusing these observations into a unified 3D model. Camera tracking involves solving for these camera poses associated with the input depth and color images.

The process of scene reconstruction typically falls into two main types of approaches: point-based methods and implicit volumetric methods. Point-based methods [88, 116, 38, 47, 120] use an unstructured set of points to represent the surface geometry of the scene, and an new input depth map is fused into the global representation by point merging (and instantiation where there are no corresponding points found). This creates a memory-efficient, explicit representation of the scene; however, spatial structure is lost, i.e. we do not obtain a continuous surface.

Implicit volumetric methods represent the surface geometry with an implicit function defined on a volumetric grid. Most common is the volumetric fusion approach by Curless and Levoy [15], which represents the scene with an implicit truncated signed distance field (TSDF), where each voxel in a volumetric grid stores the distance to the nearest surface, the sign indicates visibility from the camera, and the surface can be extracted as the isosurface at 0 (e.g., by Marching Cubes [61]). Incoming depth maps are fused into the model through a weighted averaging scheme, which effectively regularizes noise and provides efficient updates. This volumetric fusion approach is

used by the prominent KinectFusion [72, 40], which showed real-time 3D reconstruction of small scenes with a commodity handheld sensor, along with its successors [87, 118, 127, 9, 74, 18].

The original volumetric fusion approach of Curless and Levoy [15] as well as KinectFusion [72, 40] were limited in scalability due to their representation of the 3D scene as uniform volumetric grid, whose cubic growth makes large-scale scanning quickly intractable. Thus a variety of efficient data structures for real-time volumetric fusion have been proposed, exploiting the sparsity of the TSDF representation to create more efficient spatial subdivision strategies. Moving-volume approaches [87, 118] maintain a limited active volume size but stream data out-of-core as the sensor moves. Hierarchical data structures [127, 9] trade off more efficient spatial subdivision with more computationally complex updates. A spatial-hashing-based data structure for real-time 3D reconstruction [74] has also been proposed to provide both memory efficiency as well as $\mathcal{O}(1)$ updates to the data structure. Such sparse scene representations (coupled with efficient data management of the active volume) can effectively eliminate scale restrictions to commodity sensor scanning (e.g., 4mm-level voxel resolution).

While these approaches enable volumetric fusion at scale, camera pose estimates were largely computed through a frame-to-model iterative closest point (ICP) algorithm [5, 10, 89]. Frame-to-frame or frame-to-model ICP can be very efficient, for each new frame only aligning it to the previous frame or current fused model, respectively. Thus it has been commonly used for real-time reconstruction, providing very efficient camera tracking. However, this local alignment strategy suffers from drift, as each new frame introduces small relative errors which easily accumulate over a long sequence of frames, causing distortions in the 3D model. Even small pose errors, seemingly negligible on a small local scale, can accumulate to dramatic error in the final 3D model [74]. For instance, the common problem of loop closure, where we need to localize a frame to a previously visited location (e.g., scanning around a room and coming back to the starting position), typically cannot be handled with such local tracking approaches, as there is no global registration to recognize previously visited locations and enough error has accumulated over time to produce inconsistent

tracking; an example is shown later in the top right of Figure 3.7. Global pose optimization can mitigate these tracking issues with local, frame-to-model registration, often employing bundle adjustment [110] or pose graph optimization [54] to solve for all camera poses. Pose graph optimization solves for a camera trajectory from a set of relative pose measurements, distributing the error across the graph. With bundle adjustment, camera poses are solved for by minimizing reprojection error of correspondences found between multiple image observations. Such correspondences are often found through sparse keypoint detection and matching; dense correspondences can also be employed but often come at a higher computation cost and small basin of convergence for the pose optimization.

Most research on achieving globally consistent 3D models at scale from RGB-D input requires offline processing and access to all input frames. [57, 129, 131, 130, 12] provide for globally consistent models by optimizing across the entire pose trajectory, but require minutes or even hours of processing time, meaning *real-time* revisiting or refinement of reconstructed areas is infeasible, and large-scale scan collection intractable. Real-time, drift-free pose estimation is a key focus in the simultaneous localization and mapping (SLAM) literature. Many real-time monocular RGB methods have been proposed, including sparse methods [51], semi-dense [26, 31] or direct methods [66, 25]. While impressive tracking results have been shown using only monocular RGB sensors, these approaches do not generate detailed dense 3D models, which is our aim.

As pose estimation from range data based on variants of the ICP algorithm is relatively brittle, improvements have also been developed in the form of incorporation of color data [117] or global pose correction such as pose graph optimization [103], loop closure detection [119], incremental bundle adjustement [121, 29], or recovery from tracking failures by image or keypoint-based relocalization [33, 111]. Such online correction typically runs at the cost of time (seconds to minutes to perform a global optimization step [119, 29]), rely on computing optimized camera poses prior to fusion limiting the ability to refine the model afterwards [103], or use point-based representations that limit quality and lack general applicability where continuous surfaces are

needed [121]. The recent BundleFusion [18] approach introduces a parallelizable optimization framework supporting globally-consistent model creation at real-time rates. Such real-time 3D reconstruction enables efficient creation of 3D models representative of real-world scenes, from which we can then infer the complete mesh geometry underlying the 3D scan.

2.2 Synthetic 3D Model Datasets

Our aim is to create high-quality, geometrically complete 3D models representing the physical world underlying a partial scan observation. While real-world scans remain incomplete, synthetic 3D models created by humans with 3D modeling software (e.g., SketchUp, Autodesk Maya, etc.) are complete, and offer and opportunity to learn the construction of a complete model. Ideally we would like to learn from a large collection of professional artist created models which closely mimic real-world scenes; unfortunately, such models are costly and not readily available.

Recently there have been efforts to collect open libraries and datasets of publicly available 3D models created by more novice users. Early efforts to build collections of 3D shapes offered relatively limited sets of models, with approximately $1,800$ shapes in the well-known Princeton Shape Benchmark [95]. Inspired by the large-scale data collection efforts for 2D images such as ImageNet [20], the ShapeNet [8] dataset was developed, comprising several million models of which a core set of approximately $51,300$ are cleaned and annotated with category labels and alignments. These models are largely simpler than real-world objects; despite this, they are clean, sharp, and complete – qualities lacking in 3d scans.

Similar efforts have been made towards collecting datasets of 3D models of scenes. SceneNet [36] provides a collection of 57 synthetic scenes. In the following SceneNet RGB-D [65], it further provides a scene generation pipeline which randomly samples scene layouts and 3D objects to augment to arbitrary new scenes. This enables generation of numerous different scenes, although since they are not manually designed but

random, they appear more unnatural and chaotic. The SUNCG dataset [100] contains approximately $45,000$ from the Planner5D website[1] in which users create scenes by constructing scene layouts and adding furniture from an object library. Note that the number of unique objects remains rather limited (approximately $2,600$) as the users aim to model room layouts and furniture arrangements rather than individual 3D shapes.

Using these synthetic models, which have complete geometry, we can simulate through a virtual scanning process synthetic 3D scans which then have corresponding complete ground truth models, which we can leverage to help learn a generative model for scan completion.

2.3 3D Shape and Scene Completion

Completing 3D shapes has a long history in geometry processing and is often applied as a post-process to raw, captured 3D data. Traditional methods typically focus on filling small holes by fitting local surface primitives such planes or quadrics, or by using a continuous energy minimization [101, 71, 128]. Many surface reconstruction methods that take point cloud inputs can be seen as such an approach, as they aim to fit a surface and treat the observations as data points in the optimization process; e.g., Poisson Surface Reconstruction [45, 46]. Such methods perform well for filling small, local holes, but struggle with missing global structures (e.g., a missing chair leg). To address larger holes, other shape completion methods have been developed, including approaches that leverage symmetries in meshes or point clouds [109, 67, 77, 97, 102] or part-based structural priors derived from a database [106]. Although these methods show impressive results, using pre-defined regularities fundamentally limits the shape space to that of hand-crafted design.

One can also approach scan completion by way of replacing scanned geometry with aligned synthetic models retrieved from a database [69, 93, 49, 58, 94], as the synthetic models are known to be complete. Such approaches assume identical or near-identical database matches for objects in the 3D scans, which is very difficult

[1] `https://planner5d.com/`

to achieve in practice. This assumption can be relaxed by allowing modification of the retrieved models, e.g., by non-rigid registration such that they better fit the scan [76, 85]. While this non-rigid deformation of models from a database improves shape coverage, it is still difficult to generalize to large changes in global structure.

To improve generalization to new scans, data-driven structured prediction methods show promising results. One of the first such methods is Voxlets [30], which uses a random decision forest to predict unknown voxel neighborhoods, producing a final model through a weighted average of the predicted results.

Recently, we have seen inspiration from the success of deep learning coupled with very large collections of image datasets in the image domain. Here, rather than strictly defining the features we wish to analyze, deep learning enables powerful feature learning and abstraction – demonstrating impressive success in image classification tasks. For image generation, a substantially more difficult task, significant improvements have also been demonstrated through deep learning based models. With these advances in machine learning and a growing availability of 3D model databases, 3D generative tasks have begun to be studied as well. In particular for scan completion, 3D ShapeNets [8] learns a 3D convolutional deep belief network from a shape database. While this network was developed for object recognition, in order to improve object classification performance, it also completes 3D objects. Their approach operates on a ternary volumetric grid representing occupancy and unknown space, and predicts the occupancy of the complete 3D shape. Since large resolutions are difficult to handle with convolutional neural networks, with even more dramatic increase in computational cost and memory for larger filter banks in 3D than in 2D, the model operates on a 30^3 grid.

Several other works have followed, using 3D convolutional neural networks (CNNs) for object completion. The 3D-EPN approach [19] learns to complete shapes by predicting a completed model with a generative 3D CNN conditioned on a partial scan input. The resulting coarse completion is then refined using a shape synthesis step leveraging a database of CAD model priors, generating high-quality shapes at much higher resolution (128^3) than the $\sim 30^3$ resolution commonly used for 3D deep learning. This approach is shown to generalize to real-world scans of objects. However,

the global completion and local synthesis in this approach are not end-to-end trainable, so potential global errors in the coarse prediction or limitations from the coarse resolution are difficult to fix in the later synthesis. Han et. al. [35] presents a shape completion approach which learns both global prediction and local refinement in an end-to-end fashion for synthetic shapes. To more efficiently represent and process 3D volumes, hierarchical 3D CNNs have been proposed [84, 115]. The same hierarchical strategy can be also used for generative approaches which output higher-resolution 3D models [83, 107, 37, 35]. One can also increase the spatial extent of a 3D CNN with dilated convolutions [126]. This approach has recently been used for predicting missing voxels and semantic inference for a single depth frame [100]. However, these methods operate on a fixed-sized volume whose extent is determined at training time. Hence, they focus on processing either a single object or a single depth frame, rather than a full scene, which can vary in size from a few meters to tens of meters in length.

Chapter 3

Acquiring 3D Scans

In order to acquire 3D models of real-world environments, we aim to reconstruct them from RGB-D video sequences of the environment. We can then track the frames and fuse together these observations in order to construct a 3D model of a scene. In particular, for our aim of constructing complete 3D models, we must perform this reconstruction in real time – real-time performance enables live feedback of the current reconstruction state, thus allowing users to react to and focus on scanning missing regions. With commodity RGB-D sensors such as the Microsoft Kinect, we are able to capture RGB-D data at 30Hz with a consumer device, and so we aim to perform real-time reconstruction using such RGB-D sensors in order to capture geometrically complete 3D scans of real-world scenes.

In this chapter, we describe BundleFusion, our approach for capturing real-world 3D scans in real time. BundleFusion enables efficient capture of large-scale environments, constructing globally-consistent reconstructions from which a 3D mesh model of a scene can be extracted as output. As we will see in this chapter, while BundleFusion enables capture of high-quality surface reconstructions, with real-time performance aiding more complete scanning, but it does not solve the problem of capturing complete 3D models of a scene. Thus it provides the first step in our scan completion pipeline; from the scan acquired using BundleFusion, we will generate complete 3D models in Chapters 4 and 5.

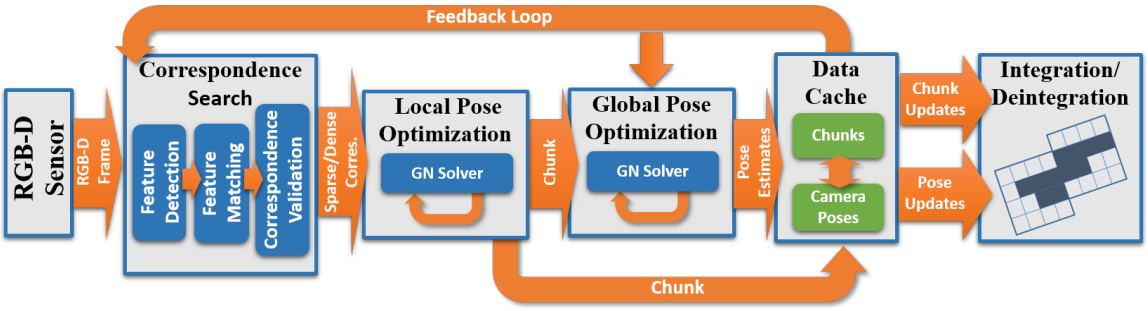

Figure 3.1: BundleFusion's global pose optimization takes as input the RGB-D stream of a commodity sensor, detects pairwise correspondences between the input frames, and performs a combination of local and global alignment steps using sparse and dense correspondences to compute per-frame pose estimates.

Here, we discuss the core components of BundleFusion for real-time 3D scan acquisition: large-scale scanning and globally-consistent camera tracking. In order to scale to capture of large scenes in real time, BundleFusion builds upon the spatial voxel hashing data structure proposed by VoxelHashing [74]. To provide robust camera tracking, we develop a fully parallelizable sparse-to-dense global pose optimization framework, maintaining global structure with implicit loop closures while achieving high local reconstruction accuracy. In addition, the scene reconstruction is dynamically updated in correspondence with these global pose updates, in order to provide a consistent scene model for interactive reconstruction. As proof of concept, we then captured and reconstructed 1513 3D scans using BundleFusion, which became the core of the ScanNet dataset [16] (see Appendix B for more detail).

3.1 Large-Scale 3D Scanning

In order to scale to large scenes in real time, we obtain a dense scene reconstruction using a sparse volumetric representation and fusion [74]. Scene geometry is reconstructed by incrementally fusing all input RGB-D data into an implicit truncated signed distance (TSDF) representation, following Curless and Levoy [15]. The TSDF is defined over a volumetric grid of voxels; to store and process this data, we employ the state-of-the-art sparse volumetric voxel hashing approach proposed by Nießner et

al. [74]. This approach scales well to the scenario of large-scale surface reconstruction, since empty space neither needs to be represented nor addressed; the TSDF is stored in a sparse volumetric grid based on spatial hashing. Following the original approach, we also use voxel blocks of $8 \times 8 \times 8$ voxels. In contrast to the work of Nießner et al. [74], we allow for RGB-D frames to both be *integrated* into the TSDF as well as *de-integrated* (i.e., adding and removing frames from the reconstruction). We ensure that these two operations are symmetric, i.e., one inverts the other, thus enabling dynamic updates to the scene according to the global pose optimization; see Section 3.3 for more detail.

3.2 Globally-Consistent Tracking

The core of BundleFusion is an efficient global pose optimization algorithm which operates in unison with a large-scale, real-time 3D reconstruction framework; see Figure 3.1. At every frame, we continuously run pose optimization and update the reconstruction according to the newly-computed pose estimates. We do not strictly rely on temporal coherence, allowing for free-form camera paths, instantaneous relocalization, and frequent revisiting of the same scene region. This makes our approach robust towards sensor occlusion, fast frame-to-frame motions and featureless regions.

We take as input the RGB-D stream captured by a commodity depth sensor. To obtain global alignment, we perform a sparse-then-dense global pose optimization: we use a set of sparse feature correspondences to obtain a coarse global alignment, as sparse features inherently provide for loop closure detection and relocalization. This alignment is then refined by optimizing for dense photometric and geometric consistency. Sparse correspondences are established through pairwise Scale-Invariant Feature Transform (SIFT) [62] feature correspondences between all input frames (see Section 3.2.1). That is, detected SIFT keypoints are matched against all previous frames, and carefully filtered to remove mismatches, thus avoiding false loop closures.

To make real-time global pose alignment tractable, we perform a hierarchical local-to-global pose optimization (see Section 3.2.2) using the filtered frame correspondences. On the first hierarchy level, every consecutive n frames compose a *chunk*,

which is locally pose optimized under the consideration of its contained frames. On the second hierarchy level, all chunks are correlated with respect to each other and globally optimized. This is akin to hierarchical submapping [63]; however, instead of analyzing global connectivity once all frames are available, our new method forms *chunks* based on the current temporal window. Note that this is our only temporal assumption; between chunks there is no temporal reliance.

This hierarchical two-stage optimization strategy reduces the number of unknowns per optimization step and ensures our method scales to large scenes. Pose alignment on both levels is formulated as energy minimization problem in which both the filtered sparse correspondences, as well as dense photometric and geometric constraints are considered (see Section 3.2.3).

3.2.1 Global Pose Alignment

We first describe the details of our real-time global pose optimization strategy, which is the foundation for *online*, globally-consistent 3D reconstruction. Input to our approach is the live RGB-D stream $\mathbf{S} = \{f_i = (\mathcal{C}_i, \mathcal{D}_i)\}_i$ captured by a commodity sensor. We assume spatially and temporally aligned color $\mathcal{C}_i$ and depth data $\mathcal{D}_i$ at each frame, captured at 30Hz and 640×480 pixel resolution. The goal is to find a set of 3D correspondences between the frames in the input sequence, and then find an optimal set of rigid camera transforms $\{\mathcal{T}_i\}$ such that all frames align as best as possible. The transformation $\mathcal{T}_i(\mathbf{p}) = \mathbf{R}_i\mathbf{p} + \mathbf{t}_i$ (rotation $\mathbf{R}_i$, translation $\mathbf{t}_i$) maps from the local camera coordinates of the i-th frame to the world space coordinate system; we assume the first frame defines the world coordinate system.

Feature Correspondence Search

In our framework, we first search for sparse correspondences between frames using efficient feature detection, feature matching, and correspondence filtering steps. These sparse correspondences are later used in tandem with dense photometric correspondences, but since accurate sparse correspondences are crucial to attaining the basin of convergence of the dense optimization, we elaborate on their search and filtering

below. For each new frame, SIFT features are detected and matched to the features of all previously seen frames. We use SIFT as it accounts for the major variation encountered during hand-held RGB-D scanning, namely: image translation, scaling, and rotation. Potential matches between each pair of frames are then filtered to remove false positives and produce a list of valid pairwise correspondences as input to global pose optimization. Our correspondence search is performed entirely on the GPU, avoiding the overhead of copying data (e.g., feature locations, descriptors, matches) to the host. We compute SIFT keypoints and descriptors at $4 - 5$ ms per frame, and match a pair of frames in ≈ 0.05ms (in parallel). We can thus find full correspondences in real-time against up to over 20K frames, matched in a hierarchical fashion, for every new input RGB-D image.

3.2.2 Hierarchical Optimization

In order to run at real-time rates on up to tens of thousands of RGB-D input frames, we apply a hierarchical optimization strategy. The input sequence is split into short *chunks* of consecutive frames. On the lowest hierarchy level, we optimize for local alignments within a *chunk*. On the second hierarchy level, chunks are globally aligned against each other, using representative *keyframes* with associated features per chunk.

Local Intra-Chunk Pose Optimization Intra-chunk alignment is based on chunks of $N_{chunk} = 11$ consecutive frames in the input RGB-D stream; adjacent chunks overlap by 1 frame. The goal of local pose optimization is to compute the best intra-chunk alignments $\{\mathcal{T}_i\}$, relative to the first frame of the chunk, which locally defines the reference frame. To this end, valid feature correspondences are searched between all pairs of frames of the chunk, and then the energy minimization approach described in Section 3.2.3 is applied, jointly considering both these feature correspondences *and* dense photometric and geometric matching. Since each chunk only contains a small number of consecutive frames, the pose variation within the chunk is small, and we can initialize each of the $\mathcal{T}_i$ to the identity matrix. To ensure that the local pose optimization result after convergence is sufficiently accurate, we compute the re-projection error between each pair of images within the chunk using the optimized

local trajectory. If the re-projection error is too large for any pair of images, the chunk is discarded and not used in the global optimization.

Per-Chunk Keyframes Once a chunk has been completely processed, we define the RGB-D data from the first frame in the chunk to be the chunk's *keyframe*. We also compute a representative aggregate *keyframe feature set*. Based on the optimized pose trajectory of the chunk, we compute a coherent set of 3D positions of the intra-chunk feature points in world space. These 3D positions may contain multiple instances of the same real-world point, found in separate pairwise frame matches. Thus, to obtain the keyframe feature set, we aggregate the feature point instances that have previously found (intra-chunk) matches. Those that coincide in 3D world space are merged to one best 3D representative in the least squares sense. This keyframe feature set is projected into the space of the keyframe using the transformations from the frames of origin, resulting in a consistent set of feature locations and depths. Note that once this global keyframe and keyframe feature set is created, the chunk data (i.e., intra-chunk features, descriptors, correspondences) can be discarded as it is not needed in the second layer pose alignment.

Global Inter-Chunk Pose Optimization Sparse correspondence search and filtering between global keyframes is analogous to that within a chunk, but on the level of all keyframes and their feature sets. If a global keyframe does not find any matches to previously seen keyframes, it is marked as invalid but kept as a candidate, allowing for re-validation when it finds a match to a keyframe observed in the future. The global pose optimization computes the best global alignments $\{\mathcal{T}_i\}$ for the set of all global keyframes, thus aligning all chunks globally. Again, the same energy minimization approach from Section 3.2.3 is applied using both sparse and dense constraints. Intra-chunk alignment runs after each new global keyframe has found correspondences. The pose for a global keyframe is initialized with the delta transform computed by the corresponding intra-chunk optimization, composed with the previous global keyframe pose.

3.2.3 Pose Alignment as Energy Optimization

Given a set of 3D correspondences between a set of frames $\mathbf{S}$ (frames in a chunk or keyframes, depending on hierarchy level), the goal of pose alignment is to find an optimal set of rigid camera transforms $\{\mathcal{T}_i\}$ per frame i (for simpler notation, we henceforth write i for f_i) such that all frames align as best as possible. We parameterize the 4×4 rigid transform $\mathcal{T}_i$ using matrix exponentials based on skew-symmetric matrix generators [68], which yields fast convergence. This leaves 3 unknown parameters for rotation, and 3 for translation. For ease of notation, we stack the degrees of freedom for all $|\mathbf{S}|$ frames in a parameter vector $\mathcal{X}$. We phrase the alignment problem as a variational non-linear least squares minimization problem in the unknown parameters $\mathcal{X}$, defining an alignment objective based on both sparse features *and* dense photometric and geometric constraints:

$$E_{\text{align}}(\mathcal{X}) = w_{\text{sparse}} E_{\text{sparse}}(\mathcal{X}) + w_{\text{dense}} E_{\text{dense}}(\mathcal{X}).$$

Here, w_{sparse} and w_{dense} are weights for the sparse and dense matching terms, respectively. w_{dense} is linearly increased; this allows the sparse term to first find a good global structure, which is then refined with the dense term (as the poses fall into the basin of convergence of the dense term, it becomes more reliable), thus achieving coarse-to-fine alignment. Note that depending on the optimization hierarchy level, the reference frame is the first frame in the chunk (for intra-chunk alignment), or the first frame in the entire input sequence (for global inter-chunk alignment). Hence, the reference transform $\mathcal{T}_0$ is not a free variable and left out from the optimization.

Sparse Matching In the sparse matching term, we minimize the sum of distances between the world space positions over all feature correspondences between all pairs of frames in $\mathbf{S}$:

$$E_{\text{sparse}}(\mathcal{X}) = \sum_{i=1}^{|\mathbf{S}|} \sum_{j=1}^{|\mathbf{S}|} \sum_{(k,l) \in \mathbf{C}(i,j)} \left\| \mathcal{T}_i \mathbf{p}_{i,k} - \mathcal{T}_j \mathbf{p}_{j,l} \right\|_2^2 .$$

Here, $\mathbf{p}_{i,k}$ is the k-th detected feature point in the i-th frame. $\mathbf{C}_{i,j}$ is the set of

all pairwise correspondences between the i-th and the j-th frame. Geometrically speaking, we seek the best rigid transformations $\mathcal{T}_i$ such that the Euclidean distance over all the detected feature matches is minimized.

Dense Matching We additionally use dense photometric and geometric constraints for fine-scale alignment. To this end, we exploit the dense pixel information of each input frame's color $\mathcal{C}_i$ and depth $\mathcal{D}_i$. Evaluating the dense alignment is computationally more expensive than the previous sparse term. We therefore evaluate it on a restricted set $\mathbf{E}$ of frame pairs, $\mathbf{E}$ contains a frame pair (i, j) if their camera angles are similar (within 60°, to avoid glancing angles of the same view) and they have non-zero overlap with each other; this can be thought of as encoding the edges (i, j) of a sparse matching graph. The optimization for both dense photometric and geometric alignment is based on the following energy:

$$E_{\text{dense}}(\mathcal{T}) = w_{\text{photo}} E_{\text{photo}}(\mathcal{T}) + w_{\text{geo}} E_{\text{geo}}(\mathcal{T}).$$

Here, w_{photo} is the weight of the photometric term and w_{geo} of the geometric term, respectively. For the dense photo-consistency term, we evaluate the error on the gradient $\mathcal{I}_i$ of the luminance of $\mathcal{C}_i$ to gain robustness against lighting changes:

$$E_{\text{photo}}(\mathcal{X}) = \sum_{(i,j)\in\mathbf{E}} \sum_{k=0}^{|\mathcal{I}_i|} \left\| \mathcal{I}_i(\pi(\mathbf{d}_{i,k})) - \mathcal{I}_j(\pi(\mathcal{T}_j^{-1}\mathcal{T}_i\mathbf{d}_{i,k})) \right\|_2^2.$$

Here, π denotes the perspective projection, and $\mathbf{d}_{i,k}$ is the 3D position associated with the k-th pixel of the i-th depth frame. Our geometric alignment term evaluates a point-to-plane metric to allow for fine-scale alignment in the tangent plane of the captured geometry:

$$E_{\text{geo}}(\mathcal{X}) = \sum_{(i,j)\in\mathbf{E}} \sum_{k=0}^{|\mathcal{D}_i|} \left[\mathbf{n}_{i,k}^T (\mathbf{d}_{i,k} - \mathcal{T}_i^{-1}\mathcal{T}_j\pi^{-1}\left(\mathcal{D}_j\left(\pi(\mathcal{T}_j^{-1}\mathcal{T}_i\mathbf{d}_{i,k})\right)\right)) \right]^2.$$

Here, $\mathbf{n}_{i,k}$ is the normal of the k-th pixel in the i-th input frame. Correspondences that project outside of the input frame are ignored, and we apply ICP-like pruning

Figure 3.2: Global pose optimization robustly detects and resolves loop closure. Note, while data is first integrated at slightly wrong locations, the volumetric representation improves over time as soon as better pose estimates are available.

based on distance and normal constraints after each optimization step. For the dense photometric and geometric constraints, we downsample $\mathcal{I}_i$ and $\mathcal{D}_i$, to 80×60 pixels. Note that for the global pose optimization, the result of optimizing densely at every keyframe is effectively reset by the sparse correspondence optimization, since the 3D positions of the correspondences are fixed. Thus we only perform the dense global keyframe optimization after the user has indicated the end of scanning.

3.3 Interactive Reconstruction

Key to live, globally consistent reconstruction is updating the 3D model based on newly-optimized camera poses. We thus monitor the continuous change in the poses of each frame to update the volumetric scene representation through *integration* and *de-integration* of frames. That is, in order to update the pose of a frame with an improved estimate, we remove the RGB-D image at the old pose with a new real-time de-integration step, and re-integrate it at the new pose. Based on this strategy, errors in the volumetric representation due to accumulated drift or dead reckoning in feature-less regions can be fixed as soon as better pose estimates are available. Thus, the volumetric model continuously improves as more RGB-D frames and refined pose estimates become available; e.g., if a loop is closed (cf. Figure 3.2).

3.3.1 Integration and De-integration

Integration of a depth frame $\mathcal{D}_i$ occurs as follows. For each voxel, $\mathbf{D}(\mathbf{v})$ denotes the signed distance of the voxel, $\mathbf{W}(\mathbf{v})$ the voxel weight, $d_i(\mathbf{v})$ the projective distance (along the z axis) between a voxel and $\mathcal{D}_i$, and $w_i(\mathbf{v})$ the integration weight for a sample of $\mathcal{D}_i$. For data integration, each voxel is then updated by

$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v})+w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v})+w_i(\mathbf{v})}, \quad \mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) + w_i(\mathbf{v}).$$

We can reverse this operation to de-integrate a frame, updating each voxel by

$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v})-w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v})-w_i(\mathbf{v})}, \quad \mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) - w_i(\mathbf{v}).$$

We can thus update a frame in the reconstruction by de-integrating it from its original pose and integrating it with a new pose. This is crucial for obtaining high-quality reconstructions in the presence of loop closures and revisiting, since the already integrated surface measurements must be adapted to the continuously changing stream of pose estimates.

3.4 Reconstruction Results

For live scanning, we use a *Structure Sensor*[1] mounted to an iPad Air. RGB-D data is streamed at 30Hz with 640×480 color and depth. Note that we are agnostic to the type of used depth sensor. We stream the captured RGB-D data via a wireless network connection to a desktop machine that runs our global pose optimization and reconstructs a 3D model in real time. Visual feedback of the reconstruction is streamed live to the iPad to aid in the scanning process and help capture more complete 3D scans. Reconstruction results of scenes captured using our live system are shown in Figure 3.3. Our real-time, global tracking enables capture of quite complete 3D models with high local quality of geometry and texture of various large-scale indoor scenes. While real-time feedback enables better coverage of a scene during scanning,

[1] `http://structure.io/`

Figure 3.3: Large-scale reconstruction results: BundleFusion's real-time global pose optimization outperforms current state-of-the-art online reconstruction systems. The globally aligned 3D reconstructions are at a quality that was previously only attainable offline. Our real-time performance also enables capture of more complete scans; however, there still remain holes, largely due to occlusions.

helping achieve relatively complete 3D reconstructions, these reconstructed models do remain geometrically incomplete, largely in occluded regions that are difficult to impossible to scan.

Qualitative Comparison First, we compare to the real-time 3D reconstruction approach of Nießner et al. [74], see Figure 3.7. In contrast to their work, which builds on frame-to-model tracking and suffers from the accumulation of camera drift, we are able to produce drift-free reconstructions at high fidelity. Our novel global pose optimization framework implicitly handles loop closure, recovers from tracking failures, and reduces geometric drift. Note that most real-time fusion methods (e.g., [40, 72, 9, 74]) share the same frame-to-model ICP tracking algorithm, and therefore suffer from notable drift. Figure 3.4 shows a comparison of our approach with the

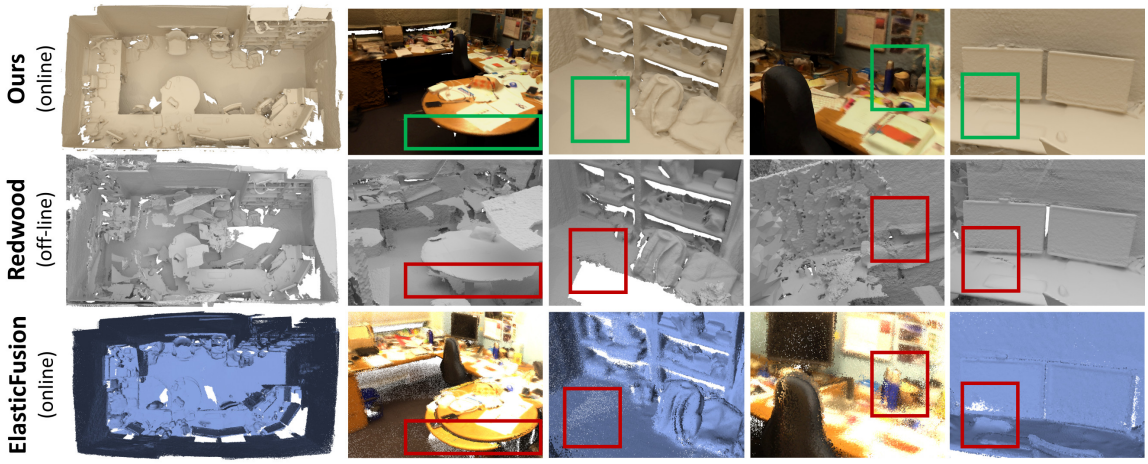

Figure 3.4: BundleFusion's real-time global pose optimization (top) delivers a reconstruction quality on par or even better than the off-line Redwood [12] (middle) and the ElasticFusion [121] (bottom) system. Note that Redwood does not use color information, and was not able to resolve all loop closures in this challenging scan.

online *ElasticFusion* approach of Whelan et al. [121] and the offline Redwood approach [12], using their rigid variant. ElasticFusion captures surfel maps using dense frame-to-model tracking and explicitly handles loop closures using non-rigid warping. In contrast, our dynamic de-integration and integration of frames mitigates issues with warping artifacts in rigid structures, and moreover produces a high quality continuous surface. Since our approach does not rely on explict loop closure detection, it scales better to scenarios with many loop closures. While the Redwood approach takes several hours (2.3h - 13.2h for each of our sequences), we achieve comparable quality and better reconstruction of small-scale detail at real-time rates. Note that Redwood does not take color information into account, thus struggling with sequences that contain fewer geometric features.

Performance We measure the performance of our pipeline on an Intel Core i7 3.4GHz CPU (32GB RAM). For compute, we use a combination of a NVIDIA GeForce GTX Titan X and a GTX Titan Black. The Titan X is used for volumetric reconstruction, and the Titan Black for correspondence search and global pose optimization. BundleFusion runs with a framerate well beyond 30Hz (see Figure 3.5) for all shown

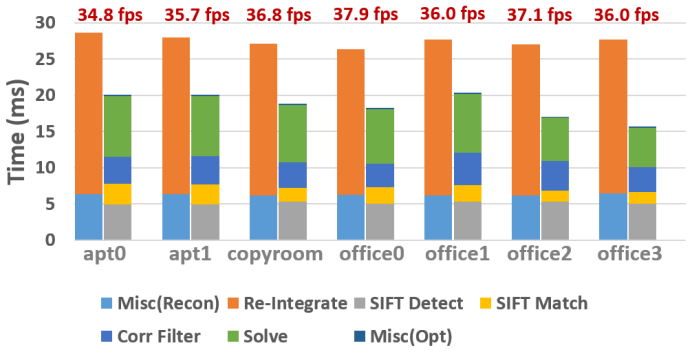

Figure 3.5: Performance Evaluation: BundleFusion runs beyond 30Hz for all used test sequences. Computation is split up over two GPUs (left bar and right bars).

test sequences. Note that the global dense optimization runs in < 500ms at the end of the sequences.

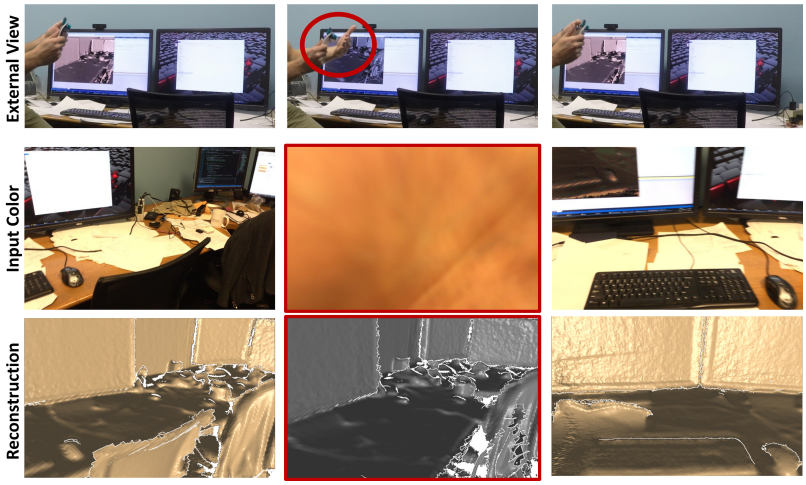

Figure 3.6: Recovery from tracking failure: BundleFusion is able to detect (gray overlay) and recover from tracking failure; i.e., if the sensor is occluded or observes a featureless region.

Recovery from Tracking Failure If a new keyframe cannot be aligned successfully, we assume tracking is lost and do not integrate surface measurements. An example scanning sequence is shown in Figure 3.6. To indicate tracking failure, the reconstruction is shown with a gray overlay. Based on this cue, the user is able to

recover the method by moving back to a previously scanned area. Note that there is no temporal nor spatial coherence required, as our method globally matches new frames against all existing data. Thus, scanning may be interrupted, and continued at a completely different location.

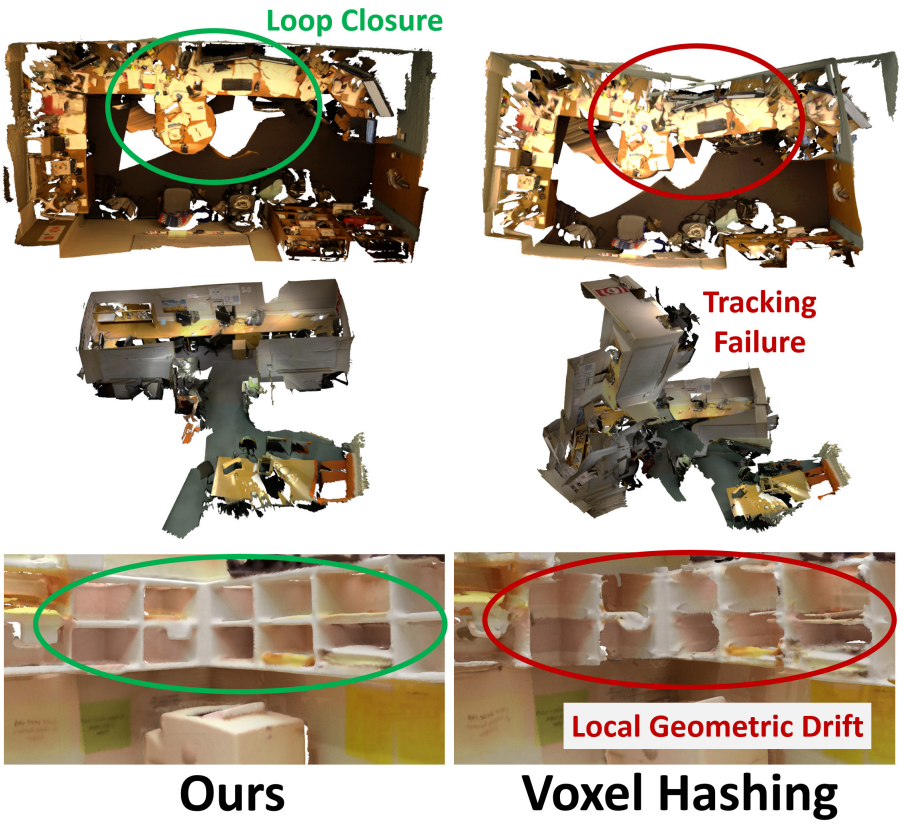

Figure 3.7: Comparison to the VoxelHashing approach of Nießner et al. [74]: in contrast to the frame-to-model tracking of VoxelHashing, our novel global pose optimization implicitly handles loop closure (top), robustly detects and recovers from tracking failures (middle), and greatly reduces local geometric drift (bottom).

Loop Closure Detection and Handling Our global pose optimization approach detects and handles loop closures transparently (see Figure 3.2), since the volumetric scene representation is continuously updated to match the stream of computed pose estimates. This allows incrementally fixing loop closures over time by means of

integration and *de-integration* of surface measurements.

3.4.1 Reconstructing a Large-Scale Dataset of 3D scans

In addition to the scenes reconstructed for the quantitative and qualitative evaluation, we further demonstrate the value of the BundleFusion approach on a large scale. Using BundleFusion, we were able to construct the ScanNet dataset [16]. All 1513 scans of ScanNet were reconstructed using BundleFusion, providing solid proof of concept of its robustness and efficiency. Additionally, approximately half of these scans were reconstructed by users who had previously had no experience in 3D scanning. We can thus pave the way for large-scale data collection of 3D scans – ScanNet provided an order of magnitude more densely reconstructed real-world scans than previous RGB-D datasets. This opens up many new avenues for powering higher-level inference in 3D using data-hungry modern machine learning methods like deep learning; see Appendix B for more detail.

3.5 Discussion

This chapter presented an approach for real-time reconstruction, whose efficient processing and live feedback enables users to scan and reconstruct 3D models of real-world scenes at high quality and completeness. The online global pose optimization enables capture of globally-consistent scans visually and geometrically representative of the original physical environments, and the ability to operate on commodity sensor data is well-suited towards augmented and virtual reality scenarios. The live feedback is crucial for achieving completeness in reconstructed models. In Figure 3.8, we see a comparison of scan quality and completeness between novice user scanning for the first time and an expert user well-versed in 3D scanning and reconstruction algorithms. While both achieve good scan quality, the expert user is able to fully leverage the real-time feedback to achieve a more complete scan. However, there do still remain holes and missing regions in the reconstructed 3D scans; such holes make

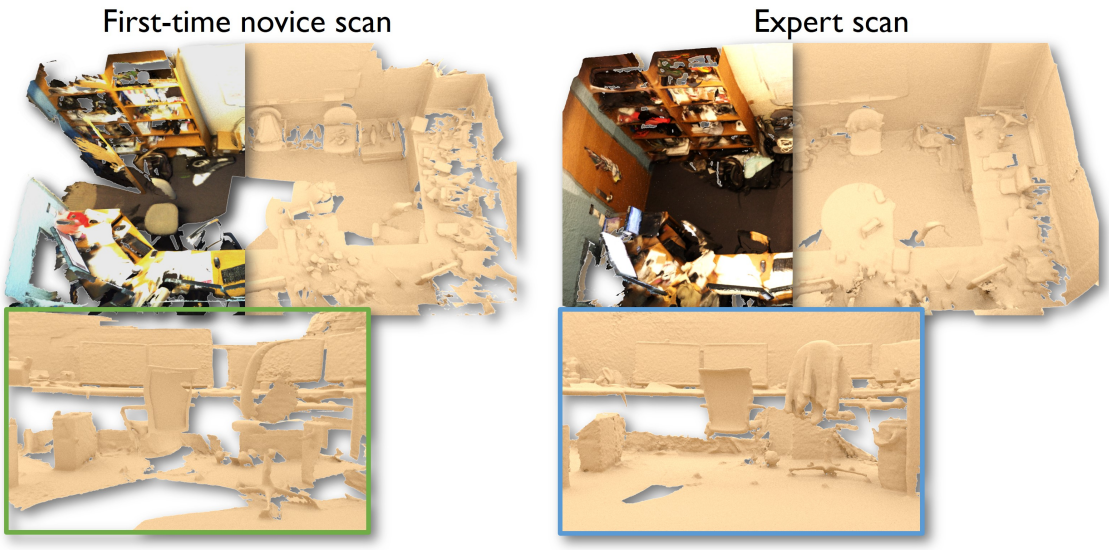

Figure 3.8: Comparison of scan quality and completeness from our approach between a novice user scanning for the first time and an expert user well-versed in 3D scanning and reconstruction algorithms. Note that both achieve good surface quality in scanned regions, and the expert user can fully leverage the real-time feedback to achieve a relatively complete scan, although there still remain holes from occlusions and a few missed regions.

the resulting 3D models ill-suited for content creation or mixed reality scenarios. Additionally, many of these are regions which are difficult to impossible to scan, e.g., geometry behind monitors is occluded and the sensor does not fit behind them.

Thus our aim in the following chapters is to infer the scene geometry in these missing, unseen regions. We acquire (partial) 3D scans using BundleFusion, and from these scans, we develop a learned approach towards generating 3D models of the complete scene underlying these 3D scan observations. To produce geometrically complete 3D models of real-world scenes, which can then be consumed by various applications (e.g., enabling accurate exploration and interactions in mixed reality), we present a generative deep learning approach conditional on these partial scans in Chapters 4 and 5.

Chapter 4

Formulating Scan Completion as a Generative Task for 3D Shapes

From the scans acquired as described by Chapter 3, we aim to produce complete 3D models. In particular, inspired by successes of generative deep learning approaches for images [34, 24, 21, 80, 56, 55, 125, 75, 81], we aim to formulate scan completion as 3D generative task. However, since real-world scans are of arbitrary size (e.g., ranging from several meters in length for a typical room to tens of meters for a building floor), and generating high resolution output remains challenging, we first focus on scan completion for 3D shapes. With isolated objects, we can temporarily bypass the challenge of generating large-scale output by scaling all objects into the same fixed volume size (impractical for 3D scenes, due to the high variance in sizes). We leave the problem of scan completion for scenes to be addressed in Chapter 5.

To complete scans of 3D shapes, we design a generative model which, conditioned on a partial scan, learns the complete 3D model underlying the partial scan observation. In this chapter, we propose a 3D-Encoder-Predictor Network (3D-EPN) based on volumetric convolutional neural networks to learn the scan completion process for shapes. By leveraging deep neural networks, we can not only learn to fill small holes but also learn global structures, and generalize global geometric structure to learn a mapping from partial scans to complete shapes, thus completing large holes with missing structural information (e.g., largely missing chair legs).

To fully exploit 3D convolutional neural networks, we represent both the partial scan and complete model with volumetric grids. In particular, we employ implicit distance fields, using a truncated signed distance field to encode the partial scan, as is used in our scan acquisition, and a truncated distance field to represent the complete shape. Here, we leverage the sign of the implicit distance field to encode known and unknown information, so we create unsigned distance fields as output. This representation not only enables encoding more resolution than a binary occupancy grid, but also allows us to extract mesh output as the isosurface of the predicted distance field, producing complete 3D meshes.

4.1 A Generative Model for Predicting Coarse Completed Global Structure

The goal of our 3D-EPN is to take a partial 3D scan of an object as input, and predict a completed 3D shape as output, i.e., learn a generative model for complete 3D models conditioned on a partial scan. We represent each model in a 3D voxel grid encoding a truncated signed distance field for the partial scan, and truncated distance field for the completed shape.

Our 3D-EPN design loosely follows the idea of autoencoders, similar to Dosovitskiy [24]. Unlike traditional autoencoder networks that reconstruct the original input and learn an efficient encoding, we aim to fill in missing data from partial input scans. In our case, the network learns a correlation of partial and complete models at training time, which at test time regresses a completed model with constraints given by known surfaces or free space information. At a high level, the goal is to map all partial scans into a shared, embedded space which we correlate with the complete models. We design the training process such that we learn this mapping, as well as the reconstruction from it, even under largely missing data. The main challenge of this process is generating new information – i.e., filling in the missing data from unseen views – by generalizing geometric structures. The network needs to encode general rules of 3D model design, and generalize across different shape instances. Here, the

main objective is the ability to reconstruct a complete mesh from the latent space while respecting the constraints of known data points.

4.1.1 Network Architecture

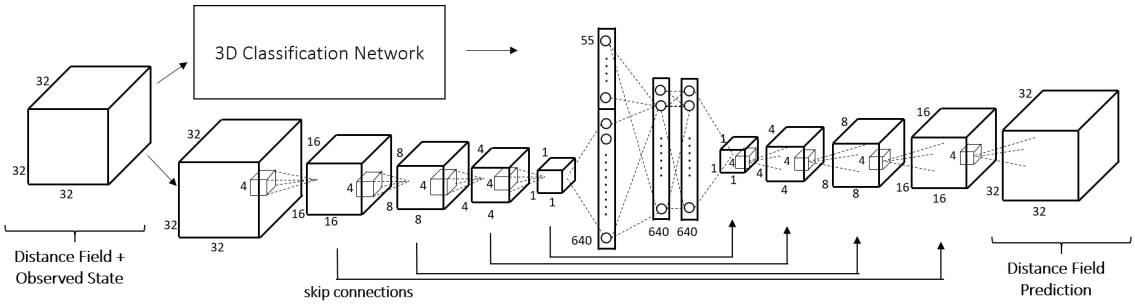

Figure 4.1: Network architecture of our 3D Encoder-Predictor Network.

We propose a 3D deep network that consumes a partial scan obtained from volumetric fusion [15], and predicts the distance field values for the missing voxels. Both our input and output are represented as volumetric grids with two channels representing the input TSDF; the first channel encodes the distance field and the second known/unknown space; see Section 4.1.2. Note that the binary known/unknown channel encodes a significant amount of knowledge as well, it will let the network know what missing areas it should focus on.

Our network is composed of two parts and is visualized in Figure 4.1. The first part is a 3D encoder, which compresses the input partial scan. The compressed stream is then concatenated with the semantic class predictions of a 3D-CNN shape classifier into a hidden space volume, in order to specifically encourage the network to leverage semantic information. The joined semantic features and input partial scan features are compressed by two fully-connected layers which embed the scan and its semantic information into the latent space. This encoder helps the network summarize global context from the input scan – both the observed distance values, known empty space, and class prediction. The second part is a predictor network that uses 3D up-convolutions to grow the hidden volume into a 32^3 full size output of estimated distance field values. Based on the global context summarized by the

encoder network, the predictor net is able to infer missing values. In addition, we add skip connections – similar to a U-net architecture [86] – between the corresponding encoder and predictor layers, visualized at the bottom of Figure 4.1. These skip connections help guide the output according to the input, since we want the output to look similar to the input in known regions already seen by the camera. The data from these connections is then concatenated with the intermediary output of the upconvolutions, thus doubling the feature map size. This way, we ensure propagation of local structure of the input data and make sure it is preserved in the generated output predictions.

We use ReLU and batch normalization for all the layers (except the last one) in the network. We use a masked L1 loss that computes the difference of ground truth distance field and predicted ones. Only the error in the unknown regions is counted; the known occupied and known empty voxels are masked out and enforced to match up the input. We use the ADAM optimizer [50] with 0.001 learning rate and momentum 0.9. The learning rate is decayed by half every 20 epochs. For $153,540$ training samples, it takes ≈ 3 days to train the model to convergence (about half as long without the skip connections).

4.1.2 Generating Supervised Training Data

In order to provide supervised training data, we aim to generate realistic ground truth scanning patterns to virtually scan 3D CAD models. We use models from the ShapeNet model database [8], which provide ground truth complete models, and virtually scan these models to create partial input scans. We simultaneously train on a subset of 8 categories of ShapeNetCore (see Section 4.2), comprising a total of 25590 object instances (the test set is composed of 5384 models).

For training, we generate complete distance fields using a 3D digital differential analyzer [2]. To generate partial reconstructions, we virtual scan the complete 3D model, roughly mimicking real-world scanning. We generate depth maps from a random trajectory around a given model with our custom virtual DirectX renderer. The obtained depth maps store range values in normalized device coordinates. We

backproject these to metric space (in m) by using Kinect intrinsics. The extrinsic camera parameters define the rigid transformation matrices which provide alignment for all generated views. All views are integrated into a shared volumetric grid using the volumetric fusion approach by Curless and Levoy [15], where the voxel grid's extent is defined by the model bounding box. Note that the ground truth poses are given by the virtual camera parameters used for rendering and the models are aligned with respect to the voxel grid. As a result, we obtain a truncated signed distance field (TSDF) for a given (virtual) scanning trajectory. This representation also encodes known free space; i.e., all voxels in front of an observed surface point are known to be empty. The sign of the distance field encodes this: a positive sign is known-empty space, zero is on the surface, and a negative sign indicates unknown values. This additional information is crucial for very partial views; see Figure 4.2. For training the 3D-EPN, we separate our the sign value from the absolute distance values, and feed them into the network in separate channels; see Section 4.1.1.

For each model, we generate a set of trajectories with different levels of partialness/completeness in order to reflect real-world scanning with a hand-held commodity RGB-D sensor. These partial scans form the training input. The ground truth counterpart is generated using a distance field transform based on a 3D scanline method [2]; here, we obtain a *perfect* (unsigned) distance field (DF). We choose to represent the ground truth as an unsigned distance field because it is non-trivial to robustly retrieve the sign bit from arbitrary 3D CAD models (some are closed, some not, etc.). In our training tasks, we use six different partial trajectories per model. This serves as data augmentation strategy, and results in a total of $153,540$ training samples of our 3D-EPN.

In order to train our 3D-EPN, we generate training pairs of TSDF and DF at resolutions of 32^3.

4.2 Evaluation

In this section we evaluate our generative shape completion approach on both synthetic shapes from the ShapeNet dataset [8] as well as real-world range scans from

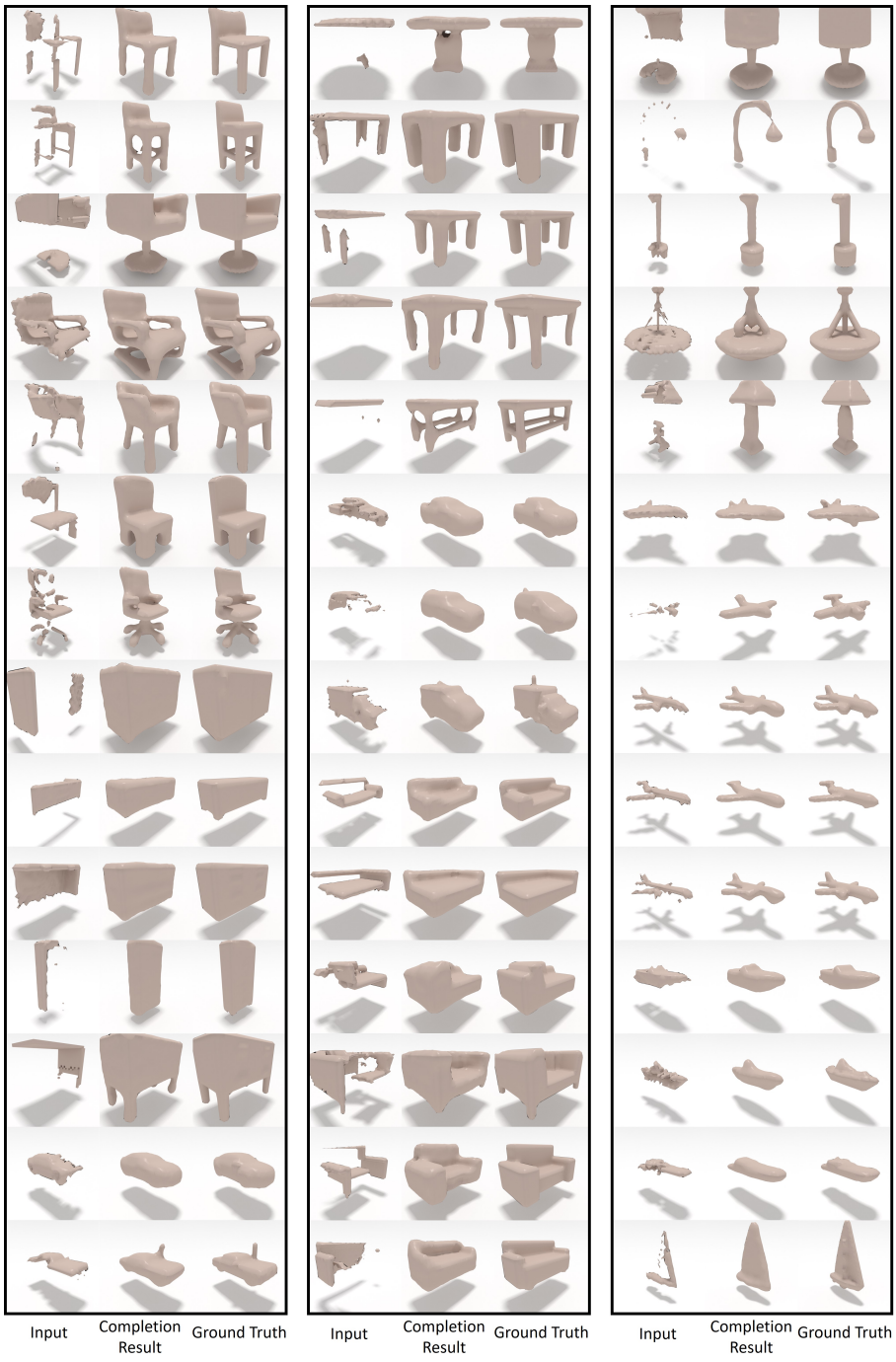

Figure 4.2: Examples of shape completion with our method (note that our approaches operates on all shape types using the same trained models).

the dataset proposed by Qi et. al. [79]. Across all experiments, we train the 3D-CNN classifier network, the 3D-EPN, and the 3D retrieval network on the same train/test split for ShapeNet [8], with the 3D-EPN trained on a subset of eight classes: namely, airplanes, tables, cars, chairs, sofas, dressers, lamps, and boats. Quantitative evaluations are obtained for a test set of 1200 models. When a distance field representation is available, we extract the isosurface using Matlab's *isosurface* function. However, some baselines directly predict meshes; in these cases, we use those for rendering and evaluation. Our main results on synthetic data and real data are visualized in Figures 4.2 and 4.6, respectively.

4.2.1 Evaluation Metrics

To quantitatively evaluate the quality of our completions, we compute the ℓ_1 error of the unknown regions against the ground truth distance field (in voxel space, up to truncation distance of 2.5 voxels).

4.2.2 Comparison to Previous Work

Method	ℓ_1-Err (32^3)
Poisson [45, 46]	1.90
ShapeRecon [85]	0.97
3D ShapeNets [123]	0.91
Ours (3D-EPN)	0.51
Ours (3D-EPN-class)	0.48
Ours (3D-EPN-unet)	0.38
Ours (final) (3D-EPN-unet-class)	**0.37**

Table 4.1: Quantitative shape completion results on synthetic ground truth data, using single depth map scans as input. We measure the ℓ_1 error of the unknown regions against the ground truth distance field (in voxel space, up to truncation distance of 2.5 voxels).

In Figure 4.3, we compare against state-of-the-art shape completion methods. Since some methods are designed to complete shapes from single depth images, we

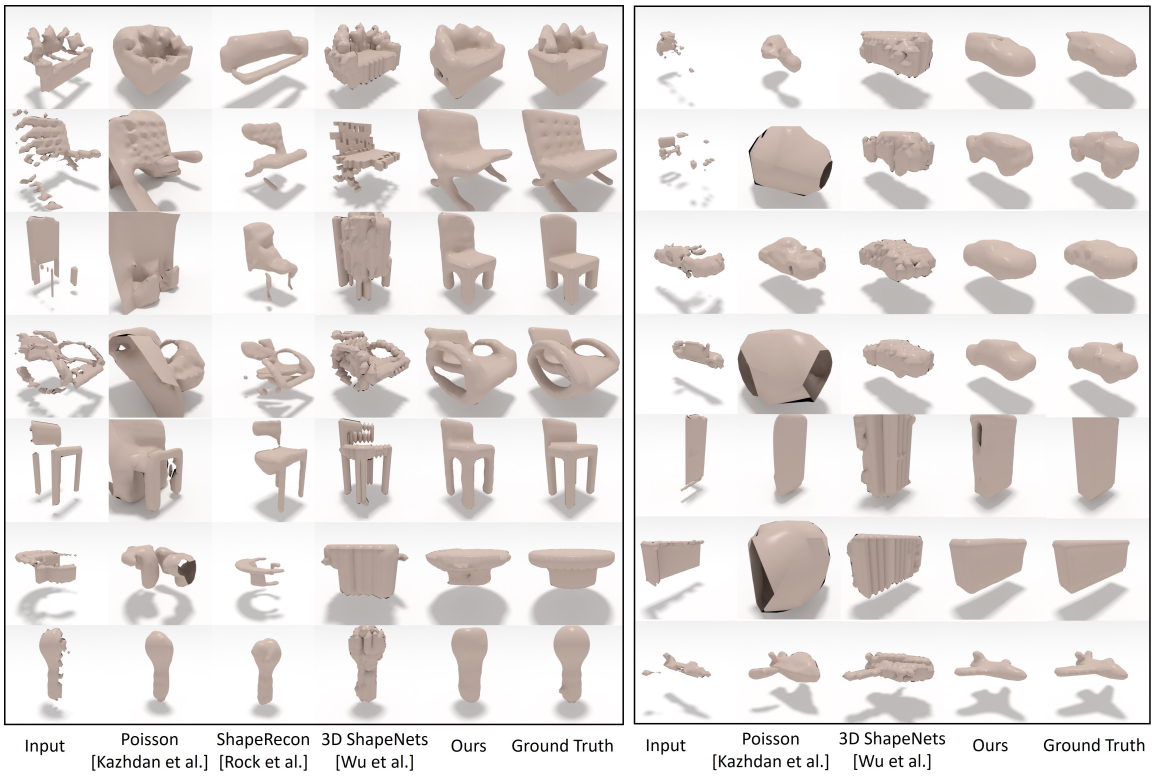

Figure 4.3: Qualitative evaluation on ShapeNet [8]. We show results on a variety of different scenes and compare against [45, 85, 123]. Note that ShapeRecon is only trained on a subset of categories (left). Our 3D-EPN successfully captures missing global structures even in relatively large unknown regions.

use only single image scans for input. Poisson surface reconstruction [45, 46] is mostly used to obtain complete surfaces on dense point clouds, but it cannot infer missing structures. ShapeRecon [85] performs slightly better, but overall, it is heavily dependent on finding good nearest neighbors; the available implementation was also trained only on a subset of classes. 3D ShapeNets [123] is most similar to our method, but it is a fully generative model, which in practice hurts performance since it addresses a more general task. A quantitative evaluation on the same dataset is shown in Table 4.1. Overall, our 3D-EPN performs best, and it efficiently leverages the 3D-CNN class vector input.

4.2.3 Benefit of Completion for Classification

In Table 4.2, we address the question of whether it is possible to use the 3D-EPN to improve accuracy on classification and retrieval tasks. For a given partial scan, there are two options for performing classification. In the first variant, we train the 3D-CNN of Qi et al. [79] on partial input to reflect the occlusion patterns of the test data. In the second variant, we first run our 3D-EPN and obtain a completed 32^3 output; we use this result as input to the 3D-CNN which is now trained on complete shapes. In both cases, the exact same partial test inputs are used; however, with the intermediate completion step, performance for both classification and shape retrieval increases significantly.

	3D-CNN /w Partial Train	3D-EPN + 3D-CNN /w Complete Train
Classification	90.9%	**92.6%**
Shape Retrieval	90.3%	**95.4%**

Table 4.2: Effect of 3D-EPN predictions on classification and shape retrieval tasks. We train a 3D-CNN classification network [79] on partial (left) and complete (right) ShapeNet models. The retrieval accuracy is computed from the classes of the top 3 retrieved neighbors. Performance improves significantly when we use the 3D-EPN predictions as an intermediary result. Note that the test task is the same for both cases since they use the same test input.

4.2.4 Ablation Study

We study various design choices for our shape completion pipeline. We evaluate several variants of the 3D-EPN network architecture, and analyze the benefits of using skip connections (strong performance gain) as well as explicitly using learned semantic class information (lesser performance gain). We further show that our model generalizes well enough to leverage multi-class training to outperform individual models trained specifically for each class. In addition, we show that our 3D-EPN maintains low completion error over varying degrees of partialness in input scans.

Network Variants

Table 4.3 shows a quantitative evaluation of our network on a test set of input partial scans with varying trajectory sizes ($\geq$ 1 camera views). Note this is in contrast to Table 4.1, where we test completion from single depth frames only; since real-world scans can contain a varying number of views, we evaluate our approach on scans from a variety of trajectory sizes. Our 3D-EPN with skip connections and class vector performs best, informing the best shape synthesis results.

Method	ℓ_1-Err (32^3)
Ours (3D-EPN)	0.382
Ours (3D-EPN-class)	0.376
Ours (3D-EPN-unet)	0.310
Ours (final) (3D-EPN-unet-class)	**0.309**

Table 4.3: Quantitative shape completion results on synthetic ground truth data for input partial scans with varying trajectory sizes. We measure the ℓ_1 error of the unknown regions against the ground truth distance field (in voxel space, up to truncation distance of 2.5 voxels).

Single Class vs. Multi-Class Training

Table 4.4 evaluates different training options for performance over multiple object categories. We aim to answer the question whether we benefit from training a separate network for each class separately (first column). Table 4.4 compares the results of training separate networks for each class with a single network trained over all classes (with and without class information). Our networks trained over all classes combined performs better than training over each individual class, as there is significantly more training data, and the network leveraging class predictions performs the best.

Varying Degrees of Completeness

Figure 4.4 shows an evaluation and comparisons against 3D ShapeNets [8] on different test datasets with varying degrees of partialness. Even for highly partial input, our

Category (# train models)	Separate EPN-unets (known class) ℓ_1-Error	EPN-unet w/o Class ℓ_1-Error	EPN-unet /w Class **Ours Final** ℓ_1-Error
Chairs (5K)	0.477	**0.409**	0.418
Tables (5K)	0.423	**0.368**	0.377
Sofas (2.6K)	0.478	0.421	**0.392**
Lamps (1.8K)	0.450	0.398	**0.388**
Planes (3.3K)	0.440	**0.418**	0.421
Cars (5K)	0.271	0.266	**0.259**
Dressers (1.3K)	0.453	0.387	**0.381**
Boats (1.6K)	0.380	0.364	**0.356**
Total (25.7K)	0.422	0.379	**0.374**

Table 4.4: Quantitative evaluations of 32^3 3D-EPNs; from left to right: separate networks have been trained for each class independently (at test time, the ground truth class is used to select the class network); a single network is used for all classes, but no class vector is used; our final result uses a single network trained across all classes and we input a probability class vector into the latent space of the 3D-EPN.

method achieves relatively low completion errors. Compared to previous work, the error rate of our method is relatively stable with respect to the degree of missing data.

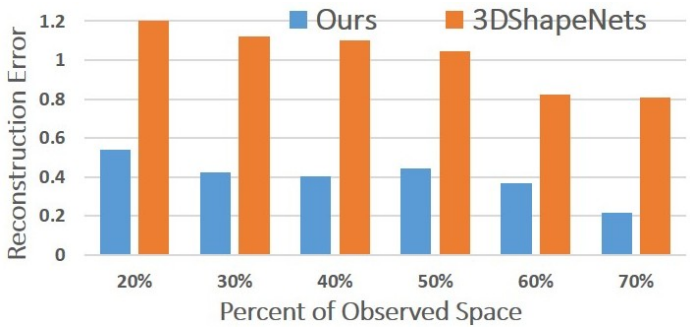

Figure 4.4: Quantitative evaluation of shape completion using our 3D-EPN and 3D ShapeNets [123] on different degrees of partial input. For this task, we generate several test sets with partial observed surfaces ranging from 20% to 70%. Even for very partial input, we obtain relatively low reconstruction errors, whereas 3D ShapeNets becomes more unstable.

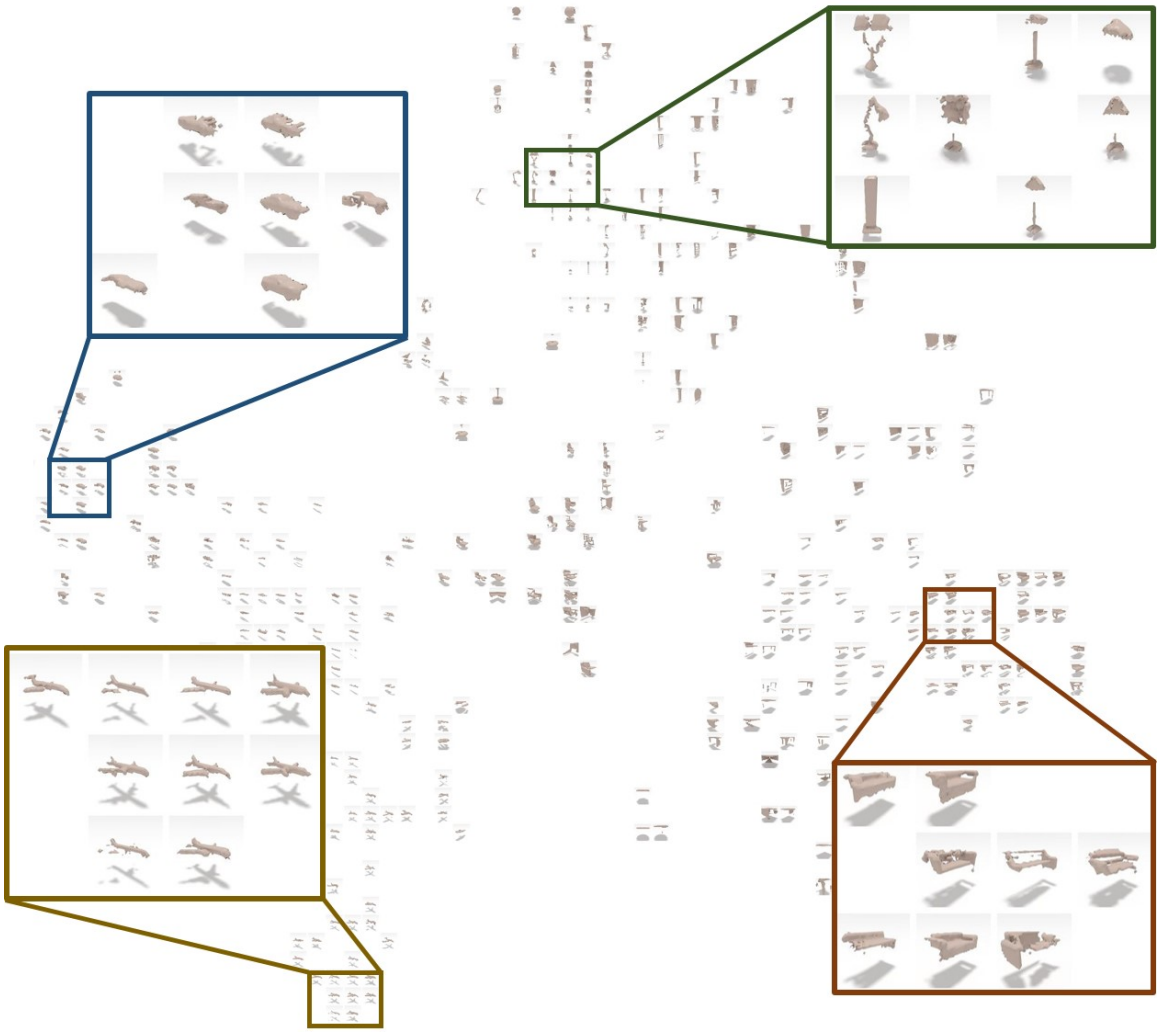

Figure 4.5: t-SNE visualization of the latent vectors in our 3D-EPN trained for shape completion without class information (3D-EPN-unet). The rendered images show input partial scans. Four zoom-ins are shown for regions of cars (top left), lamps (top right), airplanes (bottom left) and sofas (bottom right).

Shape Embeddings

Fig. 4.5 shows a t-SNE visualization of the latent vectors in our 3D-EPN trained for shape completion without using class information (3D-EPN-unet). For a set of test input partial scans, we extract their latent vectors (the 512-dimensional vector after the first fully-connected layer and before up-convolution) and then use t-SNE

to reduce their dimension to 2 as (x, y) coordinates. Images of the partial scans are displayed according to these coordinates. It is interesting to note that the input partial scans are naturally clustered according to their semantic categories, over varying degrees of occlusion, even without explicitly introducing a classification network into the completion pipeline Shapes with similar geometry tend to lie near each other, although they have varying degrees of occlusion. The visualization indicates that our 3D-EPN is truly learning the semantics of the partial scans, even with a supervision task that is only for shape completion.

4.2.5 Effect of Data Representation

An important question for generative learning in 3D is the data representation that is used. We focus on volumetric representations, and evaluate the effect of different volumetric surface representations for shape completion in Table 4.5. The simplest representation is a binary grid with 1 indicating occupied surface and 0 indicating empty or unknown regions. We can additionally add camera visibility information to create a ternary grid, indicating known occupied, known empty, and unknown space – this additional information gives improved completion; intuitively, it gives the 3D-EPN explicit information as to where the unknown regions are that might need to be completed. Furthermore, we can encode the surface as a distance field rather than a binary grid, where each voxel stores the distance to the nearest surface. This not only conforms with common 3D scanning representations, but also encodes more information in voxels representing empty space, and is capable of encoding some super-resolution. Using a distance field we again see improved performance, and adding the camera visibility information in as a signed distance field provides the best performance.

That is, there are two major characteristics of the representation which affect the 3D-EPN performance. First, a smooth function, as realized by signed and unsigned distance fields, provides better performance (and super-resolution encoding) than a discrete representation. Second, explicitly storing known-free vs unknown space

Surface Rep.	ℓ_1-Error (32^3)	ℓ_2-Error (32^3)
Binary Grid	0.653	1.160
Ternary Grid	0.567	0.871
Distance Field	0.417	0.483
Signed Distance Field	**0.379**	**0.380**

Table 4.5: Quantitative evaluation of the surface representation used by our 3D-EPN. In our final results, we use a signed distance field input; it encodes the ternary state of known-free space, surface voxels, and unknown space, and is a smooth function. It provides the lowest error compared to alternative volumetric representations.

encodes information in addition to the voxels on the surface, improving output completion quality. The signed distance field representation combines both advantages.

4.2.6 Results on Real Shapes

In Figure 4.6, we show examples of shape completion on real-world range scans. The test scans are part of the RGB-D test set of the work of Qi et al. [79], and have been captured with a PrimeSense sensor. The dataset includes reconstructions and frame alignment obtained through VoxelHashing [74] as well as mesh objects which have been manually segmented from the surrounding environment. For the purpose of testing our mesh completion method, we only use the first depth frame as input (left column of Figure 4.6). We use our 3D-EPN trained as described on purely synthetic data from ShapeNet [8]. As we can see, our method is able to produce faithful completion results even for highly partial input data. Although the results are compelling for both the intermediate 3D-EPN predictions, as well our final output, the completion quality looks visually slightly worse than the test results on synthetic data. We attribute this to the fact that the real-world sensor characteristics of the PrimeSense are different from the synthetically-generated training data used to train our model. We believe a better noise model, reflecting the PrimeSense range data, could alleviate this problem (at the moment we don't simulate sensor noise). Another option would be to generate training data from real-world input, captured with careful scanning and complete scanning patterns; e.g., using the dataset captured by Choi et

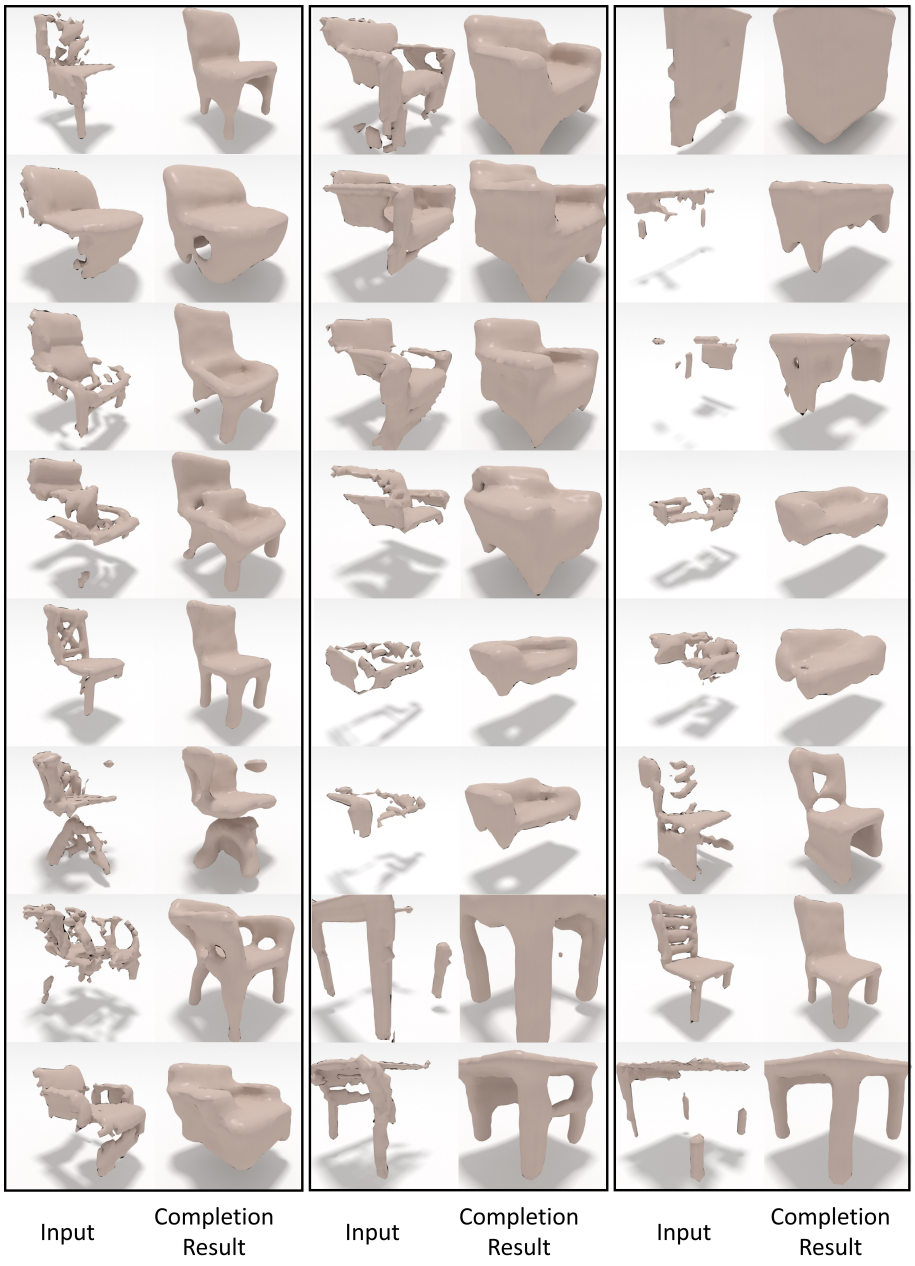

Figure 4.6: Shape completion from our method on real-world range scans from commodity sensors (here, a PrimeSense is used). We visualize partial input, 3D-EPN predictions, and our final results. In addition, we show the retrieved shapes as intermediate results on the right. Note that although the retrieved models look clean, they are inherently different from the input with respect to global structure.

al. [14]. However, we did not further explore this direction in the context of the paper, as our goal was to learn the completions from actual ground truth input. In addition to 3D-EPN predictions and our final results, we show the intermediate shape retrieval results. These models are similar; however, they differ significantly from the partial input with respect to global geometric structure. Our final results thus combine the advantages of both the global structure inferred by our 3D-EPN, as well as the local detail obtained through the shape synthesis optimization process.

4.3 Discussion

This chapter presented a generative model for 3D shape completion, producing high-quality complete shapes by leveraging generative deep learning. We additionally show the effectiveness of the signed distance field representation for 3D scans over other volumetric representations. For a task such as shape completion, the use of skip connections is very effective, as there is significant information in the input that is useful to transfer around a bottleneck layer. While using semantic class information from an explicit classification network provides improvement in completion quality, the gain is somewhat marginal. Additionally, using our model trained only on synthetic data, we are still able to show faithful completion results on real-world scans, indicating that our virtual scanning effectively mimics real-world data characteristics, and that synthetic and real shapes can still look similar enough at a part level to exploit for machine learning.

The most fundamental limitation here is the fixed, limited resolution at which the completion model operates. In practice, we do not have scans of only isolated objects. It is also not practical to take the approach of resizing all scans to fit into the same volumetric grid for general scans of arbitrary scenes, as we would not only lose the inherent scale information, but moreover larger scans would lose a considerable amount of information if resized to fit into typical tractable output resolutions for volumetric 3D CNNs ($32^3, 64^3$). Thus in Chapter 5, we will adapt our conditional generative scan completion model to operate on general indoor scans of room-scale scenes.

Chapter 5

A Generative Model for Scan Completion for Large-Scale Scenes

Now that we have constructed a generative model for completion 3D scans of shapes in Chapter 4, the largest challenge in generalizing this model to complete scans of arbitrary scenes is handling large and varying sizes. Here we are also faced with the curse of dimensionality for volumetric 3D data. In this chapter, we describe ScanComplete, which extends our generative model for scan completion to operate on large 3D environments without restrictions on spatial extent. As with our shape completion formulation, we represent partial scans and complete models with implicit (signed) distance fields, designing our model to predict the complete distance field representing the physical environment, conditioned on a partial scan of the scene.

In order to process scans at scale, we leverage fully-convolutional neural networks that can be trained on smaller subvolumes but applied to arbitrarily-sized scene environments at test time. This enables efficient processing of 3D scans of very large indoor scenes: we show examples with bounds of up to $1480 \times 1230 \times 64$ voxels ($\approx 70 \times 60 \times 3$m). In addition, to obtain high-quality completed output, the model must use a sufficiently high resolution to predict fine-scale detail. However, it must also consider a sufficiently large context to recognize large structures and maintain global consistency. To reconcile these competing concerns, we propose a coarse-to-fine strategy in which the model predicts a multi-resolution hierarchy of outputs. The

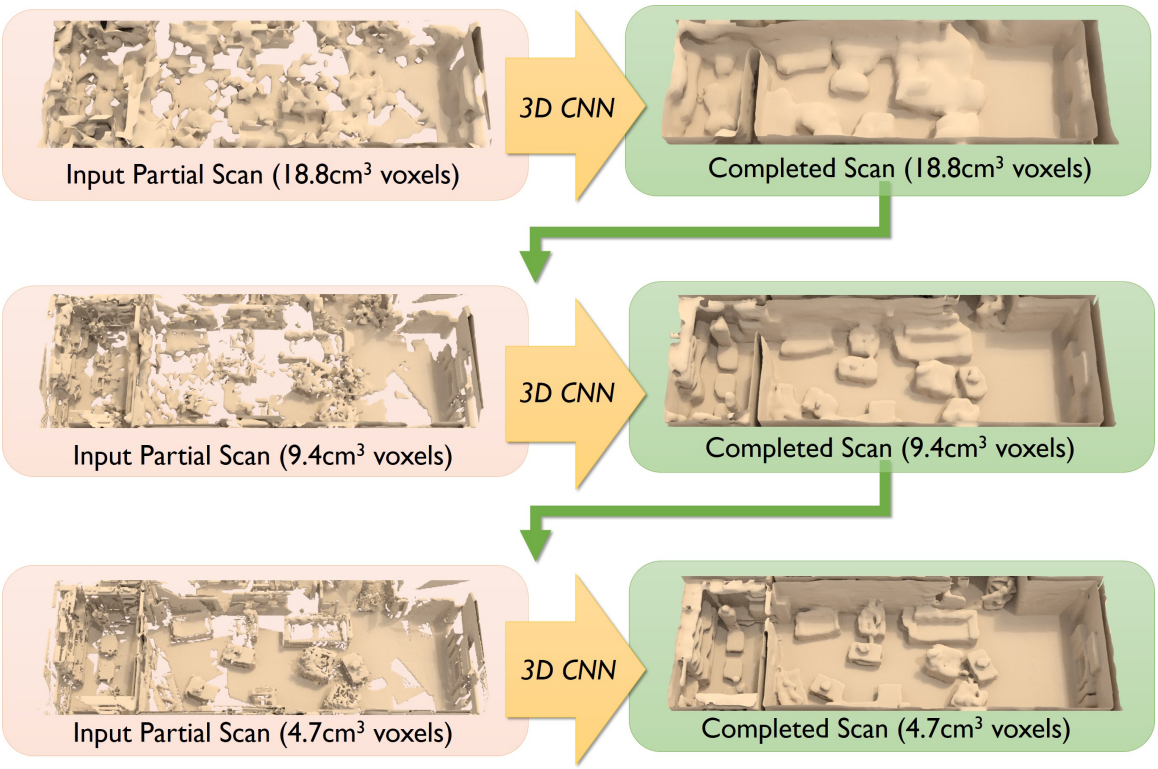

Figure 5.1: Overview of our method: we propose a hierarchical coarse-to-fine approach, where each level takes a partial 3D scan as input, and predicts a completed scan at the respective level's voxel resolution using our autoregressive 3D CNN architecture (see Figure 5.3). The next hierarchy level takes as input the output of the previous level, and is then able to refine the results. This process allows leveraging a large spatial context while operating on a high local voxel resolution. In the final result, we see both global completion as well as local surface detail.

first hierarchy level predicts scene geometry at low resolution but large spatial context. Following levels use a smaller spatial context but higher resolution, and take the output of the previous hierarchy level as input in order to leverage global context.

Using ScanComplete, we show scene completion at unprecedented spatial extents. In addition, we show that our architecture design is flexible towards other tasks aside from scene completion – we can also outperform state of the art in the task of 3D semantic segmentation. Furthermore, we demonstrate that it is possible to train our model on synthetic data and transfer it to completion of real RGB-D scans taken from commodity scanning devices.

5.1 Network Design

The ScanComplete method takes as input a partial 3D scan, represented by a truncated signed distance field (TSDF) stored in a volumetric grid. The TSDF is generated from depth frames following the volumetric fusion approach of Curless and Levoy [15], which has been widely adopted by modern RGB-D scanning methods [72, 40, 74, 42, 18]. We feed this partial TSDF into our new volumetric neural network, which outputs a truncated, unsigned distance field (TDF). At train time, we provide the network with a target TDF, which is generated from a complete ground-truth mesh. The network is trained to output a TDF which is as similar as possible to this target complete TDF.

Our network uses a fully-convolutional architecture with three-dimensional filter banks. Its key property is its invariance to input spatial extent, which is particularly critical for completing large 3D scenes whose sizes can vary significantly. That is, we can train the network using random spatial crops sampled from training scenes, and then test on different spatial extents at test time.

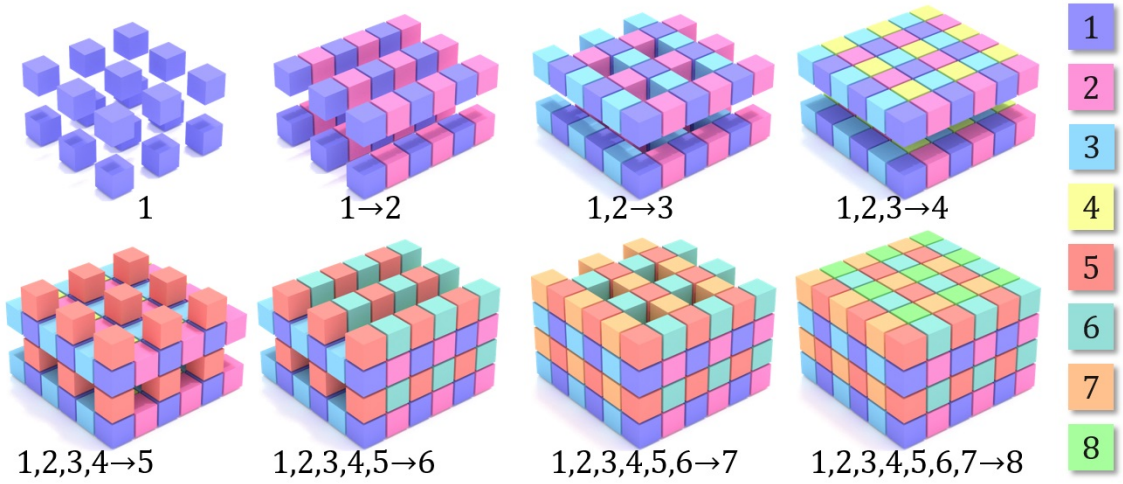

Figure 5.2: Our model divides volumetric space into eight interleaved voxel groups, such that voxels from the same group do not neighbor each other. It then predicts the contents of these voxel groups autoregressively, predicting voxel group i conditioned on the predictions for groups $1 \dots i-1$. This approach is based on prior work in autoregressive image modeling [82].

The memory requirements of a volumetric grid grow cubically with spatial extent, which limits manageable resolutions. Small voxel sizes capture local detail but lack spatial context; large voxel sizes provide large spatial context but lack local detail. To get the best of both worlds while maintaining high resolution, we use a coarse-to-fine hierarchical strategy. Our network first predicts the output at a low resolution in order to leverage more global information from the input. Subsequent hierarchy levels operate at a higher resolution and smaller context size. They condition on the previous level's output in addition to the current-level incomplete TSDF. We use three hierarchy levels, with a large context of several meters ($\sim 6\text{m}^3$) at the coarsest level, up to a fine-scale voxel resolution of $\sim 5\text{cm}^3$; see Figure 5.1.

Our network uses an autoregressive architecture based on that of Reed et al. [82]. We divide the volumetric space of a given hierarchy level into a set of eight voxel groups, such that voxels from the same group do not neighbor each other; see Figure 5.2. The network predicts all voxels in group one, followed by all voxels in group two, and so on. The prediction for each group is conditioned on the predictions for the groups that precede it. Thus, we use eight separate networks, one for each voxel group; see Figure 5.2.

We also explore multiple options for the training loss function which penalizes differences between the network output and the ground truth target TDF. As one option, we use a deterministic ℓ_1-distance, which forces the network to focus on a single mode. This setup is ideal when partial scans contain enough context to allow for a single explanation of the missing geometry. As another option, we use a probabilistic model formulated as a classification problem, i.e., TDF values are discretized into bins and their probabilities are weighted based on the magnitude of the TDF value. This setup may be better suited for very sparse inputs, as the predictions can be multi-modal.

Our ScanComplete network architecture for a single hierarchy level is shown in Figure 5.3. It is a fully-convolutional architecture operating directly in 3D, which makes it invariant to different training and testing input data sizes.

At each hierarchy level, the network takes the input partial scan as input (encoded

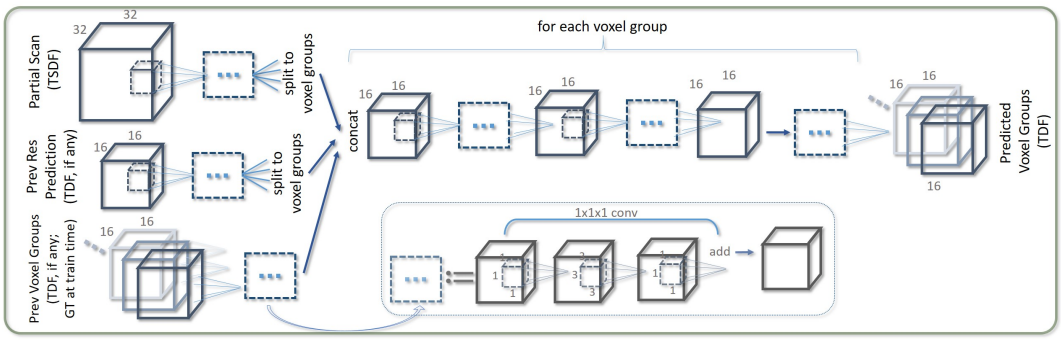

Figure 5.3: Our ScanComplete network architecture for a single hierarchy level. We take as input a TSDF partial scan, and autoregressively predict both the completed geometry and semantic segmentation. Our network trains for all eight voxel groups in parallel, as we use ground truth for previous voxel groups at train time. In addition to input from the current hierarchy level, the network takes the predictions from the previous level (i.e., next coarser resolution as input), if available; cf. Figure 5.1.

as an TSDF in a volumetric grid) as well as the previous low-resolution TDF prediction (if not the base level) and any previous voxel group TDF predictions. Each of the input volumes is processed with a series of 3D convolutions with $1 \times 1 \times 1$ convolution shortcuts. They are then all concatenated feature-wise and further processed with 3D convolutions with shortcuts. At the end, the network splits into two paths, one outputting the geometric completion, and the other outputting semantic segmentation, which are measured with an ℓ_1 loss and voxel-wise softmax cross entropy, respectively. An overview of the architectures between hierarchy levels is shown in Figure 5.1.

5.1.1 Increasing the Receptive Field

The coarse-to-fine hierarchy is designed to increase the effective receptive field of the network without incurring the cubic cost of increasing the size of the filter kernels. The coarsest resolution of the hierarchy is capable of capturing a large amount of global information from the input. This is then input into the next level of the hierarchy in an autoregressive fashion, thus also transferring some of this global information. Our hierarchy of $18.8\text{cm}^3 \rightarrow 9.4\text{cm}^3 \rightarrow 4.7\text{cm}^3$ enables use to complete large global structures as well as finer level detail.

To train the hierarchy, we feed ground truth volumes as the previous voxel group inputs to the network at train time. For the previous hierarchy level input, however, we feed in volumes predicted by the previous hierarchy level network. Initially, we trained on ground-truth volumes here, but found that this tended to produce highly over-smoothed final output volumes. We hypothesize that the network learned to rely heavily on sharp details in the ground truth volumes that are sometimes not present in the predicted volumes, as the network predictions cannot perfectly recover such details and tend to introduce some smoothing. By using previous hierarchy level predicted volumes as input instead, the network must learn to use the current-level partial input scan to resolve details, relying on the previous level input only for more global, lower-frequency information (such as how to fill in large holes in walls and floors). The one downside to this approach is that the networks for each hierarchy level can no longer be trained in parallel. They must be trained sequentially, as the networks for each hierarchy level depend on output predictions from the trained networks at the previous level. Ideally, we would train all hierarchy levels in a single, end-to-end procedure. However, current GPU memory limitations make this intractable.

5.2 Decoupling Train and Test Sizes

The fully-convolutional network design of ScanComplete enables decoupling of train and test sizes. We can thus train on cropped subvolumes of a full scene, while testing on the entire scene in $\mathcal{O}(1)$ forward passes. In Figure 5.4, we visualize the subvolumes used for training our fully-convolutional network on the three hierarchy levels of our network. By randomly selecting a large variety of these subvolumes as ground truth pairs for training, we are able train our network such that it generalizes to varying spatial extents at test time.

5.3 Generating Supervised Training Data

To train our ScanComplete CNN architecture, we prepare training pairs of partial TSDF scans and their complete TDF counterparts. We generate training examples

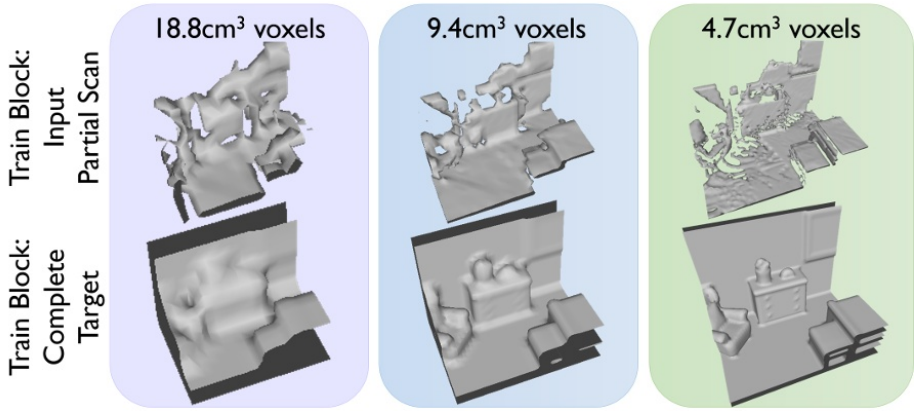

Figure 5.4: Subvolume train-test pairs of our three hierarchy levels.

from SUNCG [100], using 5359 train scenes and 155 test scenes from the train-test split from prior work [100]. As our network requires only depth input, we virtually scan depth data by generating scanning trajectories mimicking real-world scanning paths. To do this, we extract trajectory statistics from the ScanNet dataset [16] and compute the mean and variance of camera heights above the ground as well as the camera angle between the look and world-up vectors. For each room in a SUNCG scene, we then sample from this distribution to select a camera height and angle.

Within each 1.5m^3 region in a room, we select one camera to add to the training scanning trajectory. We choose the camera c whose resulting depth image $D(c)$ is most similar to depth images from ScanNet. To quantify this similarity, we first compute the histogram of depth of values $H(D(c))$ for all cameras in ScanNet, and then compute the average histogram, $\bar{H}$. We then compute the Earth Mover's Distance between histograms for all cameras in ScanNet and $\bar{H}$, i.e., $\text{EMD}(H(D(c)), \bar{H})$ for all cameras c in ScanNet. We take the mean μ_{EMD} and variance σ^2_{EMD} of these distance values. This gives us a Gaussian distribution over distances to the average depth histogram that we expect to see in real scanning trajectories. For each candidate camera c, we compute its probability under this distribution, i.e., $\mathcal{N}(\text{EMD}(H(D(c)), \bar{H}), \mu_{\text{EMD}}, \sigma_{\text{EMD}})$. We take a linear combination of this term with the percentage of pixels in $D(c)$ which cover scene objects (i.e., not floor, ceiling, or wall), reflecting the assumption that people tend to focus scans on interesting objects

rather than pointing a depth sensor directly at the ground or a wall. The highest-scoring camera c^* under this combined objective is added to the training scanning trajectory. This way, we encourage a realistic scanning trajectory, which we use for rendering virtual views from the SUNCG scenes.

For rendered views, we store per-pixel depth in meters. We then volumetrically fuse [15] the data into a dense regular grid, where each voxel stores a truncated signed distance value. We set the truncation to $3\times$ the voxel size, and we store TSDF values in voxel-distance metrics. We repeat this process independently for three hierarchy levels, with voxel sizes of 4.7cm^3, 9.4cm^3, and 18.8cm^3.

We generate target TDFs for training using complete meshes from SUNCG. To do this, we employ the level set generation toolkit by Batty [4]. For each voxel, we store a truncated distance value (no sign; truncation of $3\times$ voxel size), as well as a semantic label of the closest object to the voxel center. As with TSDFs, TDF values are stored in voxel-distance metrics, and we repeat this ground truth data generation for each of the three hierarchy levels.

For training, we uniformly sample subvolumes at 3m intervals out of each of the train scenes. We keep all subvolumes containing any non-structural object voxels (e.g., tables, chairs), and randomly discard subvolumes that contain only structural voxels (i.e., wall/ceiling/floor) with 90% probability. This results in a total of $225,414$ training subvolumes. We use voxel grid resolutions of $[32 \times 16 \times 32]$, $[32 \times 32 \times 32]$, and $[32 \times 64 \times 32]$ for each level, resulting in spatial extents of $[6\text{m} \times 3\text{m} \times 6\text{m}]$, $[3\text{m}^3]$, $[1.5\text{m} \times 3\text{m} \times 1.5\text{m}]$, respectively. A sample test volume is visualized in Figure 5.4. For testing, we test on entire scenes. Both the input partial TSDF and complete target TDF are stored as uniform grids spanning the full extent of the scene, which varies across the test set. Our fully-convolutional architecture allows training and testing on different sizes and supports varying training spatial extents.

Note that the sign of the input TSDF encodes known and unknown space according to camera visibility, i.e., voxels with a negative value lie behind an observed surface and are thus unknown. In contrast, we use an unsigned distance field (TDF) for the ground truth target volume, since all voxels are known in the ground truth. One could argue that the target distance field should use a sign to represent space

inside objects. However, this is infeasible in practice, since the synthetic 3D models from which the ground truth distance fields are generated are rarely watertight. The use of implicit functions (TSDF and TDF) rather than a discrete occupancy grid allows for better gradients in the training process; this is demonstrated by a variety of experiments on different types of grid representations in prior work [19].

5.4 Evaluation

We evaluate our ScanComplete approach on both scans of synthetic scenes from the SUNCG dataset [100] as well as real-world scans from the ScanNet dataset [16]. Across all experiments, we train our model on the same train/test scenes for SUNCG as defined by SSCNet. [100] (5359 train scenes, 115 test scenes). Note that we do not use the exact same frames from these scenes, since SSCNet and thus their data generation as well was designed for single frames, and we instead simulate scanning trajectories. Our main results on synthetic data and real data are visualized in Figures 5.5 and 5.7, respectively. In addition, we show that our ScanComplete architecture can be applied to other tasks in addition to completion: we can also predict per-voxel semantic class labels as well as jointly prediction completion and semantics, see Section 5.5.

5.4.1 Evaluation Metric

To quantitatively evaluate our completion quality, we evaluate using ℓ_1 distances with respect to the entire target scene volume (*entire*), predicted surface (*pred. surf.*), target surface (*target surf.*), and unknown space (*unk. space*). The ℓ_1 distances are measured in voxel units, using a truncation of 3 voxels. Note that these metrics shouls be evaluated as a whole, as they can all be biased in certain directions, e.g., the entire scene volume is dominated by empty space, it is easier to achieve lower predicted surface error with less predicted surface, etc. All evaluations are performed on the highest voxel resolution of $\sim 5\text{cm}^3$.

5.4.2 Ablation Study

Hierarchy Levels	Probabilistic/ Deterministic	Autoregressive	Input Size	ℓ_1-Err (entire)	ℓ_1-Err (pred. surf.)	ℓ_1-Err (target surf.)	ℓ_1-Err (unk. space)
1	prob. (#quant=256)	non-autoreg.	32	0.248	0.311	0.969	0.324
1	prob. (#quant=256)	autoreg.	16	0.226	**0.243**	0.921	0.290
1	prob. (#quant=256)	autoreg.	32	0.218	0.269	0.860	0.283
1	prob. (#quant=32)	autoreg.	32	0.208	0.252	0.839	0.271
1	prob. (#quant=16)	autoreg.	32	0.212	0.325	0.818	0.272
1	prob. (#quant=8)	autoreg.	32	0.226	0.408	0.832	0.284
1	det.	non-autoreg.	32	0.248	0.532	0.717	0.330
1	det.	autoreg.	16	0.217	0.349	0.808	0.282
1	det.	autoreg.	32	0.204	0.284	0.780	0.266
3 (gt train)	prob. (#quant=32)	autoreg.	32	0.336	0.840	0.902	0.359
3 (pred. train)	prob. (#quant=32)	autoreg.	32	0.202	0.405	0.673	0.251
3 (gt train)	det.	autoreg.	32	0.303	0.730	0.791	0.318
3 (pred. train)	det.	autoreg.	32	**0.182**	0.419	**0.534**	**0.225**

Table 5.1: Quantitative scene completion results for different variants of our completion-only model evaluated on synthetic SUNCG ground truth data. We measure the ℓ_1 error against the ground truth distance field (in voxel space, up to truncation distance of 3 voxels). Using an autoregressive model with a three-level hierarchy and large input context size gives the best performance.

We first evaluate different architecture variants for geometric scene completion in Table 5.1. We test on 155 SUNCG test scenes, varying the following architectural design choices:

- **Hierarchy Levels:** our three-level hierarchy (*3*) vs. a single 4.7cm-only level (*1*). For the three-level hierarchy, we compare training on ground truth volumes (*gt train*) vs. predicted volumes (*pred. train*) from the previous hierarchy level.
- **Probabilistic/Deterministic:** a probabilistic model (*prob.*) that outputs per-voxel a discrete distribution over some number of quantized distance value bins (*#quant*) vs. a deterministic model that outputs a single distance value per voxel (*det.*).
- **Autoregressive:** our autoregressive model that predicts eight interleaved voxel groups in sequence (*autoreg.*) vs. a non-autoregressive variant that predicts all voxels independently (*non-autoreg.*).
- **Input Size:** the width and depth of the input context at train time, using either *16* or *32* voxels

How much does the hierarchy help? While a single hierarchy level of 5cm only produces passable completion results, we see a noticeable gain in completion quality from a three-level hierarchy, in particular in the *target surf.* and *unk. space* errors. In particular, the hierarchy helps predict considerably more missing surface regions, as the problem of completing large holes is made easier at low resolutions.

How much does input context size help? We see that increasing the input context size from 16 to 32 improves completion results with both probabilistic and deterministic variants. Similar to the benefits of using a hierarchy, the more context that is seen helps to infer missing global structure.

Does an autoregressive model help? Under both probabilistic and deterministic variants, the autoregressive model outperforms the non-autoregressive one. This underscores the benefits of modeling 3D generation as a product of conditional distributions, similar to what we see in the image domain [113, 114, 91, 82].

Is it better to have a deterministic or probabilistic model? For a probabilistic model, reducing the number of quantization bins from 256 to 32 improves completion (further reduction reduces the discrete distribution's ability to approximate a continuous distance field). Overall, for our scene completion task, a deterministic model performs better than a probabilistic one. Intuitively we aim to capture a single output mode—the physical reality behind the captured 3D scan.

All together, an autoregressive, deterministic, full hierarchy with the largest spatial context provides the highest accuracy.

5.4.3 Results on Synthetic Scenes

We compare our method to alternative scene completion methods in Table 5.2. As a baseline, we compare to Poisson Surface Reconstruction [45, 46]. We also compare to 3D-EPN, which was designed for completing single objects, as opposed to scenes [19]. Additionally, we compare to SSCNet, which completes the subvolume of a scene viewed by a single depth frame [100]. For this last comparison, in order to complete the entire scene, we fuse the predictions from all cameras of a test scene into one volume, then evaluate ℓ_1 errors over this entire volume. Our method achieves lower

reconstruction error than all the other methods. Note that while jointly predicting semantics along with completion does not improve on completion, Table 5.3 shows that it significantly improves semantic segmentation performance.

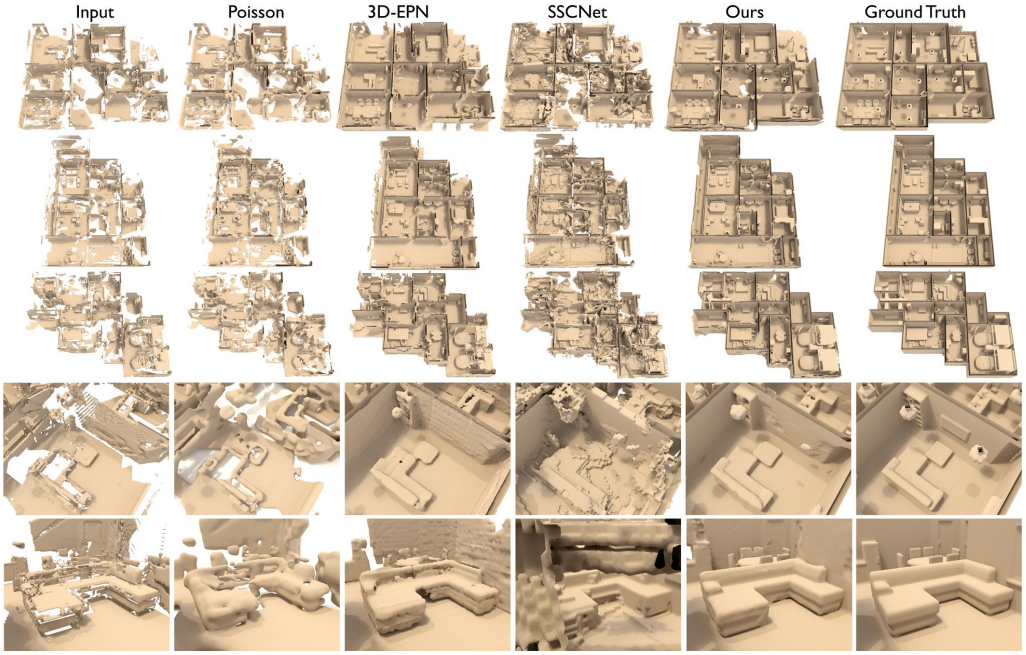

Figure 5.5: Completion results on synthetic SUNCG scenes; left to right: input, Poisson Surface Reconstruction [46], 3D-EPN [19], SSCNet [100], Ours, ground truth.

We show a qualitative comparison of our completion against state-of-the-art methods in Figure 5.5. For these results, we use the best performing architecture according to Table 5.1. We can run our method on arbitrarily large scenes as test input, thus predicting missing geometry in large areas even when input scans are highly partial, and producing more complete results as well as more accurate local detail. Note that our method is $\mathcal{O}(1)$ at test time in terms of forward passes; we run more efficiently than previous methods which operate on fixed-size subvolumes and must iteratively make predictions on subvolumes of a scene, typically $\mathcal{O}(wd)$ for a $w \times h \times d$ scene.

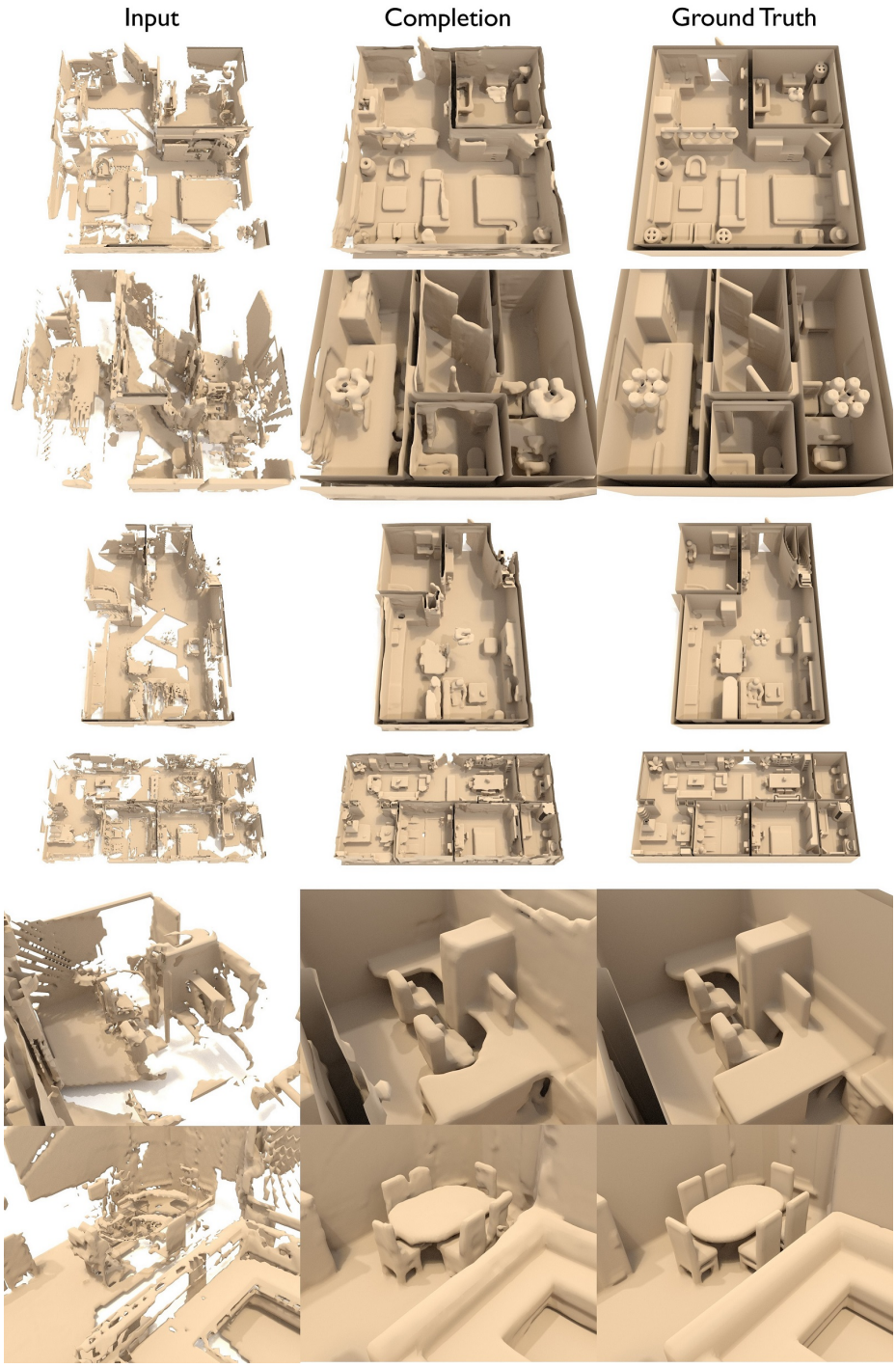

Figure 5.6: Additional completion results on synthetic SUNCG scenes

Method	ℓ_1-Err (entire)	ℓ_1-Err (pred. surf.)	ℓ_1-Err (target surf.)	ℓ_1-Err (unk. space)
Poisson Surface Reconstruction [45, 46]	0.531	1.178	1.695	0.512
SSCNet [100]	0.536	1.106	0.931	0.527
3D-EPN (unet) [19]	0.245	0.467	0.650	0.302
Ours	**0.182**	**0.419**	**0.534**	**0.225**

Table 5.2: Quantitative scene completion results for different methods on synthetic SUNCG data. We measure the ℓ_1 error against the ground truth distance field in voxel space, up to truncation distance of 3 voxels (i.e., 1 voxel corresponds to 4.7cm^3). Our method outperforms others in reconstruction error.

5.4.4 Results on Real Scenes

We also show qualitative completion results on real-world scans in Figure 5.7. We run our model on scans from the publicly-available RGB-D ScanNet dataset [16], which has data captured with an Occiptal Structure Sensor, similar to a Microsoft Kinect or Intel PrimeSense sensor. Again, we use the best performing network according to Table 5.1. We see that our model, trained only on synthetic data, learns to generalize and transfer to real data.

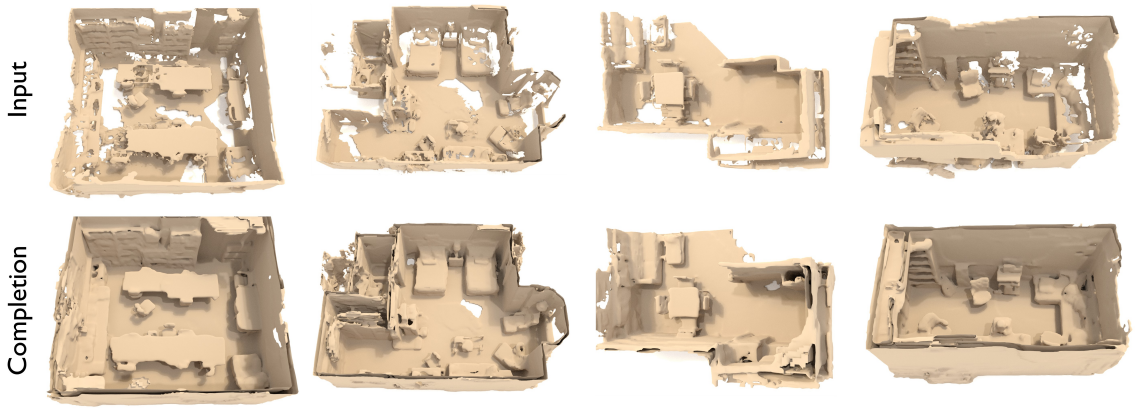

Figure 5.7: Completion results on real-world scans from ScanNet [16]. Despite being trained only on synthetic data, our model is also able to complete many missing regions of real-world data.

5.5 Other Applications of ScanComplete: Semantic Segmentation

We can also use our ScanComplete approach to predict other per-voxel characteristics, in place of or in addition to per-voxel distance values for scan completion. In particular, we leverage our scan completion approach to further inform semantic reasoning about the scenes, inferring 3D semantic segmentation over the scans. We can predict per-voxel semantic class labels from an input partial scan, in place of the predicted distance field values, but moreover, we can jointly predict both a completed scan and its semantic segmentation. In the latter case, we have two volumetric grids output from our network, and the model is autoregressive on both completion and semantics.

Figure 5.8: Semantic voxel labeling results on SUNCG; from left to right: input, SSCNet [100], ScanNet [16], Ours, and ground truth.

In Table 5.3, we evaluate and compare our semantic segmentation on the SUNCG

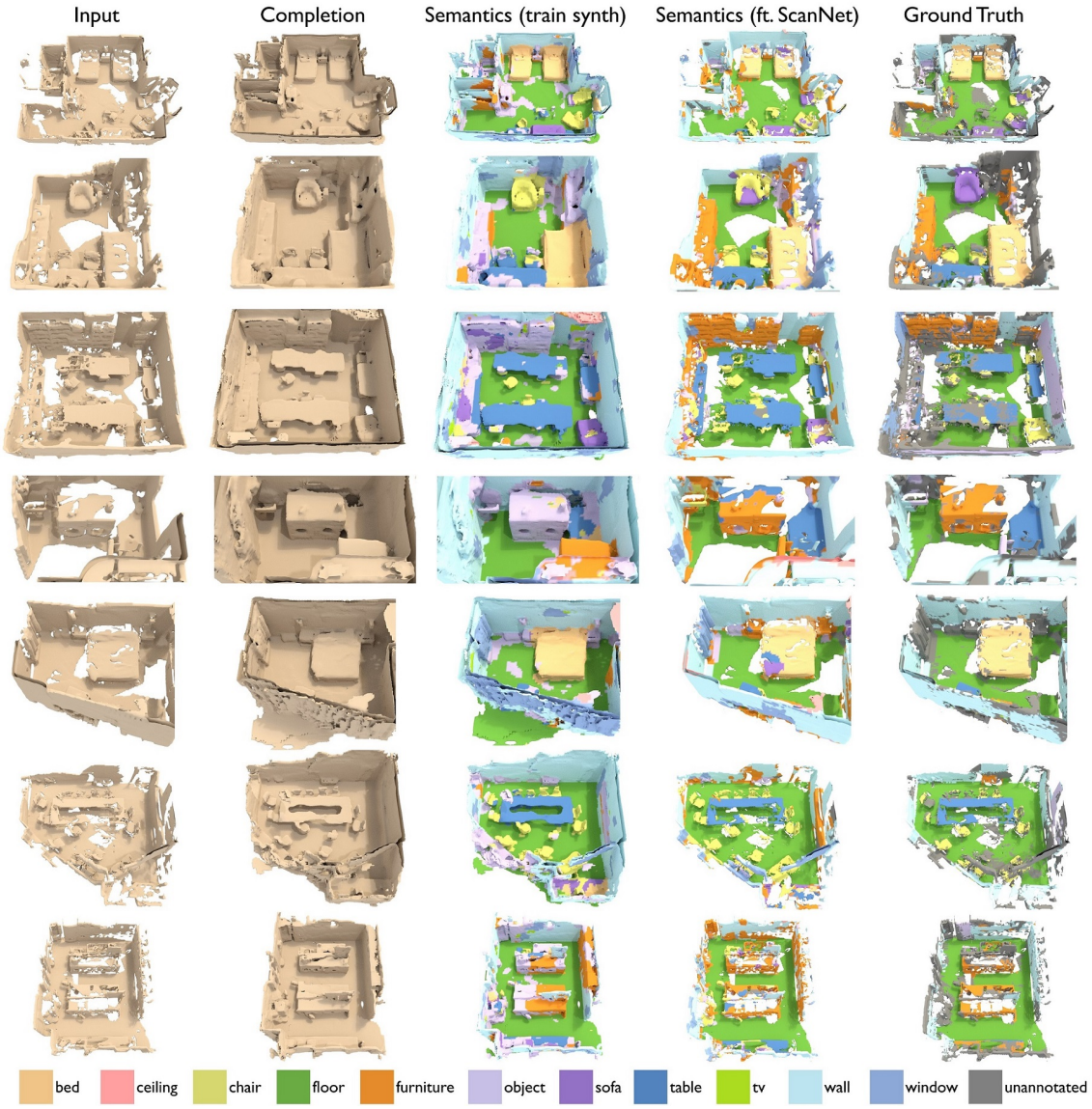

Figure 5.9: Results on ScanNet for our completion and semantic voxel labeling predictions.

dataset. We use the same train/test split for SUNCG as before, and use the SUNCG 11-label set. While our semantic-only model already produces good semantic predictions, we see a significant gain in performance when jointly predicting both completion and semantics. Intuitively, explicitly learning the full scene geometry gives better semantic performance as semantic segmentation is easier with complete models than

with partial data. This demonstrates how our approach towards scan completion also enables further reasoning and understanding of these scenes, as we see in the improvements in semantic segmentation.

Note that jointly predicting per-voxel semantic segmentation does not improve completion results but rather produces similar completion results as when training for completion only. In this case, an explicit semantic loss can artificially group together objects that are very different geometrically as well as separate objects which have similar parts; for instance, beanbag chairs, office chairs, and swings are all grouped into 'chair', as well as bathtub and sink into 'object', and objects like chairs and tables which often share common parts are separated. Moreover, intuitively, our completion-only network must already learning relevant semantic information, in order to achieve compelling completion results; we can see similar behavior from shape completion exemplified in Fig. 4.5, where completion-only training already groups together semantically similar objects without any explicit semantic input or loss.

Figure 5.8 shows qualitative semantic segmentation results on SUNCG scenes. Our ability to process the entire scene at test time, in contrast to previous methods which operate on fixed subvolumes, along with the autoregressive, joint completion task, produces more globally consistent and accurate voxel labels.

	bed	ceil.	chair	floor	furn.	obj.	sofa	table	tv	wall	wind.	avg
(vis) ScanNet [16]	44.8	90.1	32.5	75.2	41.3	25.4	51.3	42.4	9.1	60.5	4.5	43.4
(vis) SSCNet [100]	67.4	95.8	41.6	90.2	42.5	40.7	50.8	58.4	20.2	59.3	**49.7**	56.1
(vis) Ours [sem-only, no hier]	63.6	92.9	41.2	58.0	27.2	19.6	55.5	49.0	9.0	58.3	5.1	43.6
(vis) Ours [sem-only]	**82.9**	96.1	48.2	67.5	**64.5**	40.8	**80.6**	61.7	14.8	69.1	13.7	58.2
(vis) Ours [no hier]	70.3	97.6	58.9	63.0	46.6	34.1	74.5	**66.5**	**40.9**	**86.5**	43.1	62.0
(vis) **Ours**	80.1	**97.8**	**63.4**	**94.3**	59.8	**51.2**	77.6	65.4	32.4	84.1	48.3	**68.6**
(int) SSCNet [100]	65.6	81.2	48.2	76.4	49.5	49.8	61.1	57.4	14.4	74.0	36.6	55.8
(int) Ours [no hier]	68.6	96.9	55.4	71.6	43.5	36.3	75.4	**68.2**	**33.0**	**88.4**	33.1	60.9
(int) **Ours**	**82.3**	**97.1**	**60.0**	**93.2**	**58.0**	**51.6**	**80.6**	66.1	26.8	86.9	**37.3**	**67.3**

Table 5.3: Semantic labeling accuracy on SUNCG scenes. We measure per-voxel class accuracies for both the voxels originally visible in the input partial scan (*vis*) as well as the voxels in the intersection of our predictions, SSCNet, and ground truth (*int*). Note that we show significant improvement over a semantic-only model that does not perform completion (*sem-only*) as well as the current state-of-the-art.

We additionally apply our network, trained only on the synthetic SUNCG set, and

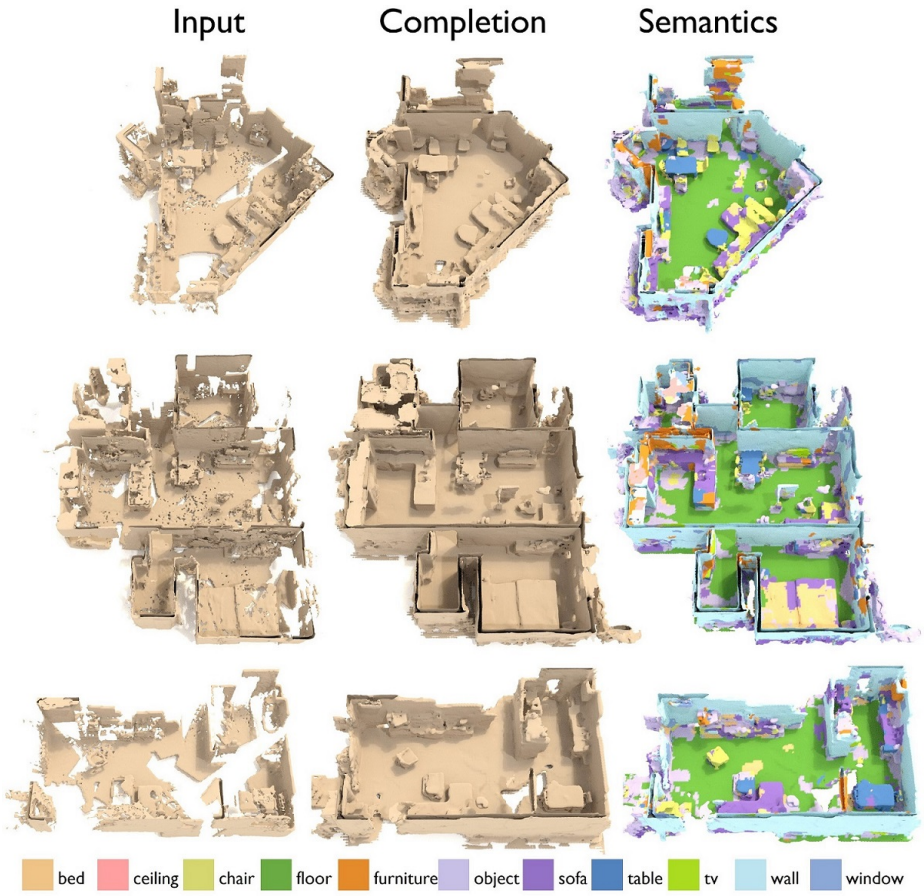

Figure 5.10: Additional results on Google Tango scans for our completion and semantic voxel labeling predictions.

use it infer missing geometry and semantic segmentation in real-world RGB-D scans. In Figure 5.9, we show results on several scenes on the publicly-available ScanNet [16] dataset; the figure visualizes real input, completion (synthetically-trained), semantics (synthetically-trained), and semantics (synthetically pre-trained and fine-tuned on the ScanNet annotations). Figure 5.10 visualizes results on another sensor modality, with scans taken from Google Tango sensors.

5.6 Discussion

This chapter demonstrated scan completion at scale. With ScanComplete, we exploit a fully-convolutional neural network that decouples train and test resolutions, thus allowing for variably-sized test scenes with unbounded spatial extents. In addition, we use a coarse-to-fine prediction strategy combined with a volumetric autoregressive network that leverages large spatial contexts while simultaneously predicting local detail. As a result, we achieve unprecedented scene completion results.

We analyzed the necessary elements for generating coherent global structure along with local detail. In particular, we see that the more input context that can be leveraged, the better. Because of this, we are able to attain improved completion in large missing regions with a coarse-to-fine hierarchy. In addition, we gain additional benefit from using an autoregressive model, which helps tractably model a joint distribution over the voxels, as well as deterministic predictions. The deterministic advantage is likely due to our problem scenario, where input partial scans often provide coverage over most of the scene – albeit limited, partial coverage – and we aim to recover the single physical reality behind these observations. For instance, a probabilistic model may be better-suited for a problem in which only a small portion of the scene has been observed, and full rooms may need to be generated. Finally, we demonstrated that our scan completion approach helps to enable higher-level scene understanding, significantly improving 3D semantic segmentation performance by jointly estimating completion and semantics.

Our work is a starting point for obtaining high-quality 3D scans from partial inputs, which is a typical problem for RGB-D reconstructions. One important aspect for future work is to further improve output resolution. Currently, our final output resolution of $\sim 5\text{cm}^3$ voxels is still not enough—ideally, we would use even higher resolutions in order to resolve fine-scale objects, e.g., cups. In addition, end-to-end training across all hierarchy levels would likely further improve performance with the right joint optimization strategy. An interesting future direction could be to combine purely generative models with conditioned input, such as GANs [34]. These networks are somewhat challenging to train, particularly for higher resolutions in 3D space,

although promising results have been shown for 2D image generation recently [44]. Another possible avenue is the incorporation of RGB information; for instance, one could enforce shading constraints to obtain fine-scale detail by borrowing ideas from recent shape-from-shading methods [122, 132].

Chapter 6

Conclusions

With the prevalence of 3D content in movies and games, and the newfound momentum of augmented and virtual reality, there will be an increasing demand for 3D content creation which cannot be tractably met through handcrafted, manual design – especially in the case of mixed reality where a 3D model of the environment is required to enable accurate exploration and interactions. 3D scanning and reconstruction provide a promising avenue towards addressing this problem, and in this dissertation we studied a generative approach towards creating high-quality 3D models from commodity RGB-D sensors, from scan acquisition with real-time 3D reconstruction to a conditional generative deep learning approach for creating geometrically complete 3D models faithful to the original physical environment. We first acquire scans with a real-time reconstruction approach that produced globally consistent 3D models. Such real-world 3D scans practically always suffer from varying degrees of incompleteness, making them unsatisfactory for use in content creation or mixed reality. We thus developed a generative formulation for scan completion leveraging deep learning techniques in 3D to create high-quality, geometrically complete meshes from partial scans. This provides a stepping stone towards fundamental elements of mixed reality, e.g., realistic interactions for a virtual agent exploring the environment or better informed physical simulations for sound or lighting, as well as potentially aiding other applications such as robot navigation or, as demonstrated in Chapter 5, improving semantic inference on 3D scans.

Concretely, in Chapter 3 we described our scan acquisition approach, building upon real-time 3D reconstruction to produce globally consistent 3D models of real-world scenes. This enabled us to easily and efficiently capture a large number of scans of real-world environments, which serve as the basis of our scan completion approach towards generating high-quality 3D models.

In Chapter 4, we presented a generative model for producing complete models from scans of 3D shapes. This model learns shared global structures of objects, and can infer missing geometry even in large unknown regions – although it remains constrained to operating on a fixed size volume.

Finally, in Chapter 5, we expanded this generative model to completion of 3D scenes, removing the spatial constraints. The model produces compelling completion results for a variety of 3D scans of rooms to building floors, with approximately 2.5cm ℓ_1 error from ground truth geometry. Furthermore, this model is capable of domain adaptation, producing convincing complete meshes for real-world 3D scans – our approach is now also a post-process option for in-house Google Tango scans.

Such a generative model for scan completion provides a first step for 3D scanning for content creation and mixed reality, already enabling improved semantic scene understanding as we demonstrated with our scan completion approach in Chapter 5. However, there are still many challenges towards making 3D models which could be used as a final graphics asset in such applications rather than enabling higher-level reasoning about 3D scenes. In terms of model geometry, we are nonetheless limited by the dense volumetric representation for producing higher resolution models, as scene data grows quickly intractable for storing a large number of scenes represented densely at sub-centimeter voxel resolution. Recently several octree-based hierarchical approaches have been proposed [84, 83, 107, 37] which show promise for using sparser volumetric representations for deep learning. Currently these approaches largely focus on producing high resolution for objects or small scenes (e.g., 256^3 volumes), and there is significant room for study in applying them for large-scale scenes. Another possibility is to take inspiration from the widely used spatial voxel hashing representation [74] for 3D reconstruction, which could provide an alternative sparse volumetric representation.

Additionally, in this dissertation we have focused on modeling scan geometry, but color and texture information are not only beneficial for understanding scenes but also are essential for creating models to be visually consumed. Here there are two main questions which can be intercorrelated: how to use color information as input to best understand a scene, and how to generate high-quality colored output. Again, we come to questions of data representation. While our $\sim$ 5cm voxel distance field can capture principal scene geometry, per-voxel colors at such resolution become extremely over-smoothed, losing visual distinctness and producing unsatisfactory visuals. As input we can consider using the original color images captured, and leveraging a combination of both 2D color features and 3D geometric features to provide complementary reasoning about a scene [17]. In this case we could also potentially leverage well-developed 2D convolutional neural networks and the vast image datasets currently available. Another alternative to a voxel-based 3D representation is textured meshes, which are consumed by typical graphics applications, but the non-euclidean nature of this data remains challenging. Several early approaches have been developed for deep learning on mesh data structures [7], although the field is still nascent.

The question of data representation in 3D for generative deep learning is a fundamental one, and still remains open. Here, there are many possibilities for innovation, and such problems are those that must be addressed to make further progress.

Appendix A

BundleFusion Experiment Details

Convergence After adding a new global keyframe, our approach requires only a few iterations to reach convergence. Figure A.1 shows convergence plots for three of the used test sequences (cf. Figure 3.3); the behavior generalizes to all other sequences. We achieve this real-time performance with the combination of our tailored data-parallel Gauss-Newton solver (efficiently handling millions of residuals and solving for over a hundred thousand unknowns), a sparse-to-dense strategy enabling convergence in only a few iterations, and a local-to-global strategy which efficiently decomposes the problem. Note that recent work provides detailed intuition as to why hand-crafted optimizers outperform existing, general solver libraries [22].

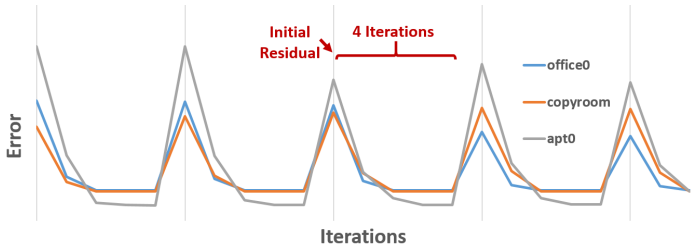

Figure A.1: Convergence analysis of the global keyframe optimization (log scale): peaks correspond to new global keyframes. Only a few iterations are required for convergence.

Additionally, we evaluate the performance of our tailored GPU-based solver against the widely-used, CPU-based Ceres solver [1]. Figure A.2 shows the performance of

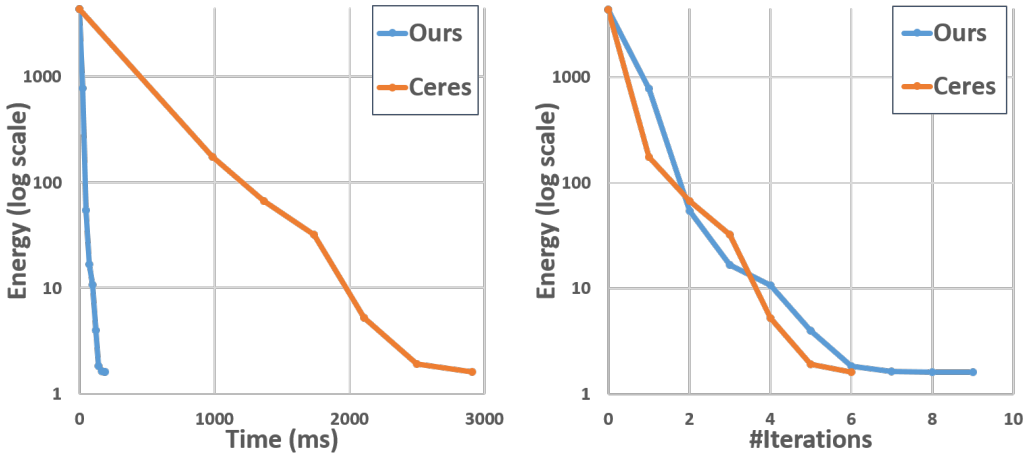

Figure A.2: Performance comparison of our tailored GPU-based solver to Ceres [1]. Both solvers are evaluated over the sparse energy term for 101 keyframes, involving 600 variables and 16339 residuals, with poses initialized to the identity.

both solvers for the sparse energy over 101 keyframes, comprising 600 variables and 16339 residuals, with poses initialized to the identity. Note that this behavior is representative of other sparse energy solves. For Ceres, we use the default Levenberg-Marquardt with a sparse normal Cholesky linear solver (the fastest of the linear solver options for this problem). While our solver takes a couple more iterations to converge without the Levenberg-Marquardt damping strategy, it still runs $\approx$ 20 times faster than Ceres while converging to the same energy minimum.

Memory Consumption We evaluate the memory consumption of our globally consistent reconstruction approach on our eight captured sequences, see Table A.1. The most significant required memory resides in RAM (CPU), i.e., 20GB for the Apt 0 sequence. It stores all RGB-D frames and depends linearly on the length of the sequence. The required device memory (GPU) is much smaller, e.g., 5.3GB (4mm voxels) and 1.9GB (1cm voxels) for the same sequence. This is well within the limits of modern graphics cards (e.g., 12 GB for GTX Titan X). We also give the amount of memory required to store and manage the TSDF (Rec) and to run the camera pose optimization, both for the sparse term (Opt-s) and the dense term (Opt-d). The footprint for storing the SIFT keypoints and correspondences (included

Table A.1: Memory consumption (GB) for the captured sequences.

	GPU						CPU
			1cm		4mm		
	Opt-d	Opt-s	Rec	$\sum$	Rec	$\sum$	
Apt 0	1.4	0.031	0.5	1.9	3.9	5.3	20.0
Apt 1	1.4	0.031	0.4	1.8	3.2	4.6	20.1
Apt 2	0.6	0.012	0.7	1.4	6.0	6.7	9.3
Copyroom	0.7	0.016	0.3	1.1	1.8	2.6	10.5
Office 0	1.0	0.021	0.4	1.4	2.5	3.5	14.4
Office 1	0.9	0.024	0.4	1.4	2.9	3.9	13.4
Office 2	0.6	0.011	0.4	1.0	3.0	3.6	8.2
Office 3	0.6	0.011	0.4	1.0	2.7	3.3	8.9

in Opt(s)) is negligibly small; i.e, 31MB for Apt 0. The longest reconstructed sequence (home_at_scan1_2013_jan_1) is part of the SUN3D dataset [124], consisting of 14785 frames ($\approx$ 8.2 minutes scan time @30Hz). This sequence has a CPU memory footprint of 34.7GB and requires 7.3GB of GPU memory (4mm voxels) for tracking and reconstruction.

Precision and Recall of Loop Closures Tab. A.2 gives the precision (i.e., the percentage of correct chunk pair correspondence detections from the set of established correspondences) and recall (i.e., the percentage of detected chunk pair correspondences from the set of ground truth correspondences), on the loop closure set of the augmented ICL-NUIM dataset. A chunk pair correspondence is determined to be in the ground truth set if their geometry overlaps by $\geq 30\%$ according to the ground truth trajectory, and a proposed chunk pair correspondence is determined to be correct if it lies in the ground truth set with reprojection error less than 0.2m, following [12]. We show our registration performance after running the SIFT matcher (Sift Raw), our correspondence filters – Key Point Correspondence Filter (SIFT + KF) and Surface Area and Dense Verification (SIFT + Verify) –, and the final result after the optimization residual pruning (Opt). As can be seen, all steps of the globally consistent camera tracking increase precision while maintaining sufficient recall.

Table A.2: Loop closure precision and recall on the synthetic augmented ICL-NUIM Dataset [12].

		Sift Raw	Sift + KF	Sift + Verify	Opt
Living 1	Precision (%)	27.0	98.0	98.2	100
	Recall (%)	47.5	40.3	39.5	39.3
Living 2	Precision (%)	25.3	92.1	92.4	100
	Recall (%)	49.3	47.4	45.9	45.7
Office 1	Precision (%)	14.1	97.7	99.6	100
	Recall (%)	49.1	48.7	48.0	47.7
Office 2	Precision (%)	10.9	90.2	96.2	100
	Recall (%)	46.0	42.4	42.1	42.0

Sparse vs. Dense Tracking In Figure A.3, we evaluate the influence of the dense tracking component of our energy function. While globally drift-free reconstructions can be obtained by sparse tracking only, the dense alignment term leads to more refined local results.

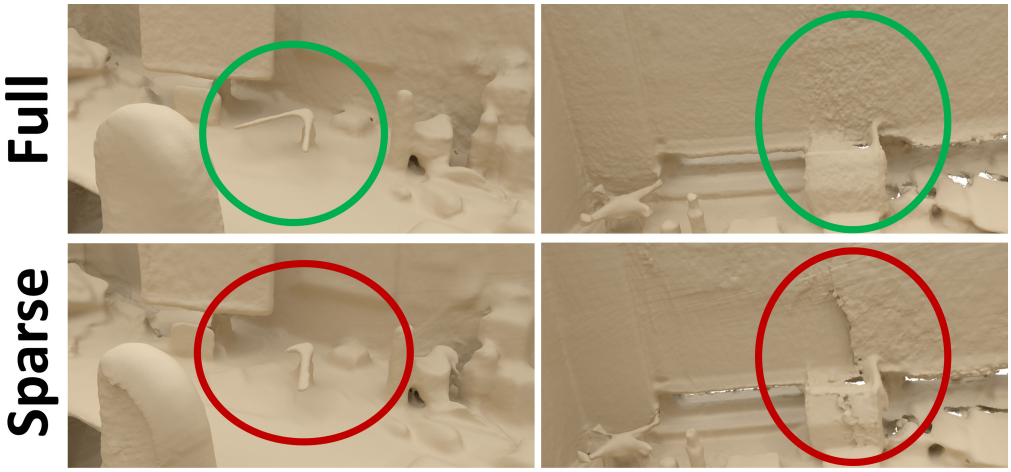

Figure A.3: Comparison of Sparse vs. Dense Alignment: the proposed dense intra- and inter- chunk alignment (top) leads to higher quality reconstructions than only the sparse alignment step (bottom).

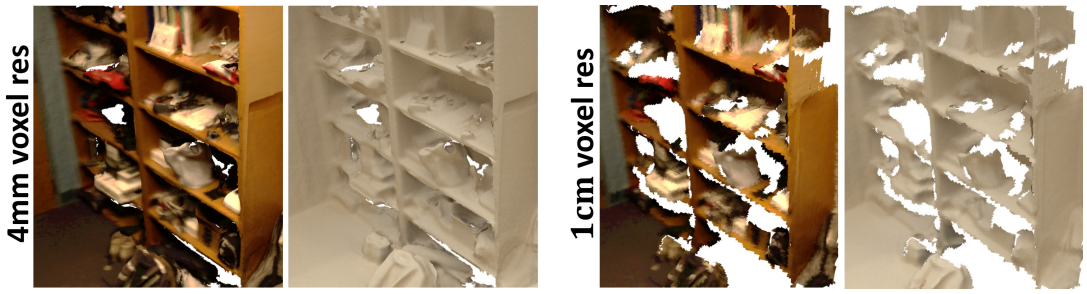

Figure A.4: Comparison of different voxel resolutions: 4mm voxel resolution (left) leads to higher-fidelity reconstructions than the coarser 1cm resolution (right). Note the generally sharper texture and the more refined geometry in case of 4mm voxels.

Voxel Resolution The impact of voxel resolution on reconstruction quality is shown in Figure A.4. As a default, we use a voxel resolution of 4mm for all reconstructions. While 1cm voxels reduce memory consumption, the quality of the reconstruction is slightly impaired.

Appendix B

Acquiring a Large-Scale Dataset of 3D Scans

3D scene understanding has recently begun to gain momentum. Research along such semantic understanding is heavily facilitated by the rapid progress of modern machine learning methods, such as neural models. One key to successfully applying theses approaches is the availability of large, labeled datasets. While much effort has been made on 2D datasets [27, 53, 59], where images can be downloaded from the web and directly annotated, the situation for 3D data is more challenging. Thus, many of the current RGB-D datasets [96, 124, 99, 39] are orders of magnitude smaller than their 2D counterparts.

One of the reasons that current 3D datasets are small is because their capture requires much more effort, and efficiently providing (dense) annotations in 3D is non-trivial. Thus, existing work on 3D datasets often fall back to polygon or bounding box annotations on 2.5D RGB-D images [96, 124, 99], rather than directly annotating in 3D. In the latter case, labels are added manually by expert users (typically by the paper authors) [39, 92] which limits their overall size and scalability.

Thus we exploit the real-time reconstruction capabilities of our BundleFusion approach (described in Chapter 3) to create globally-consistent 3D reconstructions to power the construction of a large-scale dataset of richly-annotated RGB-D scans:

Figure B.1: Example spaces in ScanNet, reconstructed with BundleFusion (c.f. Chapter 3) and annotated with instance-level object category labels through our crowdsourced annotation framework.

ScanNet. ScanNet comprises 2.5M RGB-D images in 1513 scans of real-world environments acquired in 707 distinct spaces. The sheer magnitude of this dataset is larger than any other [70, 104, 124, 98, 3, 92, 39]. However, what makes it particularly valuable for research in scene understanding is its annotation with estimated calibration parameters, camera poses, 3D surface reconstructions, textured meshes, and dense object-level semantic segmentations (see Fig. B.2). The semantic segmentations are more than an order of magnitude larger than any previous RGB-D dataset.

Dataset	Size	Labels	Annotation Tool	Reconstruction
NYU v2 [70]	464 scans	1449 frames	2D LabelMe-style [90]	none
TUM [104]	47 scans	none	-	aligned poses (Vicon)
SUN 3D [124]	415 scans	8 scans	2D polygons	aligned poses [124]
SUN RGB-D [98]	10k frames	10k frames	2D polygons + bounding boxes	aligned poses [124]
BuildingParser [3]	265 rooms	265 rooms	CloudCompare [32]	point cloud
PiGraphs [92]	26 scans	26 scans	dense 3D, by the authors [92]	dense 3D [74]
SceneNN [39]	100 scans	100 scans	dense 3D, by the authors [73]	dense 3D [13]
ScanNet (ours)	**1513 scans 2.5M frames**	**1513 scans**	**dense 3D, crowd-sourced MTurk labels also proj. to 2D frames**	**dense 3D [18]**

Table B.1: Overview of RGB-D datasets for 3D reconstruction and semantic scene understanding.

B.1 Data Acquisition

How can we design a framework that allows many people to collect and reconstruct large amounts of RGB-D data? In order to collect a large-scale dataset, we aim to allow untrained users to capture 3D scans of indoor scenes with commodity hardware. This goal drives our framework design: the RGB-D scanning system must be easy to use, and the data processing robust, automatic, with data flow handled by a tracking server. Thus, in our data acquisition pipeline, a user uses an app on an iPad mounted with a depth camera to acquire RGB-D video; the RGB-D data is then processed offline to produce a 3D reconstruction of the scene, ready to be semantically labeled. We discuss the various design choices below.

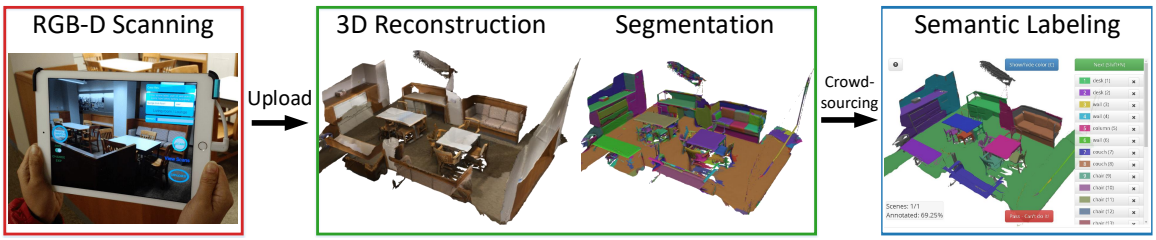

Figure B.2: RGB-D reconstruction and semantic annotation framework overview. **Left:** a novice user uses a handheld RGB-D device with our scanning interface to scan an environment. **Mid:** RGB-D sequences are uploaded to a processing server which produces 3D surface mesh reconstructions and their surface segmentations. **Right:** Semantic annotation tasks are issued for crowdsourcing to obtain instance-level object category annotations.

B.1.1 RGB-D Scanning

Hardware. There is a spectrum of choices for RGB-D sensor hardware. Our requirement for deployment to large groups of inexperienced users necessitates a portable and low-cost RGB-D sensor setup. We use the Structure sensor[1], a commodity range sensor with design similar to the Microsoft Kinect v1. We attach this sensor to a handheld device, in our case, an iPad Air2 (see Fig. B.2 left). The iPad RGB camera data is temporally synchronized with the depth sensor, providing synchronized depth and color capture at 30Hz. Depth frames are captured at a resolution of 640×480 and color at 1296×968 pixels. Note that while feature detection and matching for 3D reconstruction is typically easier with constant exposure and white-balancing, we enable auto-white balance and auto-exposure by default, which we found to be crucial for avoiding over- or under-saturation (e.g., when there is an open window or strong light in the scene).

Calibration. Our use of commodity RGB-D sensors necessitates unwarping of depth data and alignment of depth and color data. Since we aim for novice users, calibration in controlled lab conditions becomes impractical; instead, we focus on a portable, reproducible setup. Thus the user only needs to print out a checkerboard pattern, place it on a large, flat surface, and capture an RGB-D sequence viewing the surface from close to far away (e.g., 0.5m to 6m). This sequence, as well as a set of infrared and color frame pairs viewing the checkerboard, are uploaded by the user as input to the calibration. Our system then runs a calibration procedure based on [108, 23] to obtain intrinsic parameters for both depth and color sensors, and an extrinsic transformation of depth to color. We find that this calibration procedure is easy for users and results in improved data and consequently enhanced reconstruction quality.

User Interface. To make the capture process simple for untrained users, we designed an iOS app with a simple live RGB-D video capture UI (see Fig. B.2 left). The user provides a name and scene type for the current scan and proceeds to record

[1] `http://structure.io/`

a sequence. During scanning, a log-scale RGB feature detector point metric is shown as a "featurefulness" bar to provide a rough measure of tracking robustness and reconstruction quality in different regions being scanned. This feature was critical for providing intuition to users who are not familiar with the constraints and limitations of 3D reconstruction algorithms.

We additionally provide functionality to upload and delete individual captured sequences, as well as uploading all captured sequences. Upon confirmation of a successful upload (all files confirmed received by the data processing server), the corresponding files are automatically deleted from the scanning device.

Storage. We store scans as compressed RGB-D data on the device flash memory so that a stable internet connection is not required during scanning. The user can upload scans to the processing server when convenient by pressing an "upload" button. Our sensor units used 128GB iPad Air2 devices, allowing for several hours of recorded RGB-D video. In practice, the bottleneck was sensor battery life rather than storage space. Depth is recorded as 16-bit unsigned short values and stored using standard zLib compression. RGB data is encoded with the H.264 codec with a high bitrate of 15Mbps to prevent encoding artifacts. In addition to the RGB-D frames, we also record Inertial Measurement Unit (IMU) data, including acceleration, and angular velocities, from the Apple SDK. Timestamps are recorded for IMU, color, and depth images.

B.1.2 Surface Reconstruction

Once data has been uploaded from the iPad to our server, the first processing step is to estimate a densely-reconstructed 3D surface mesh and 6-DoF camera poses for all RGB-D frames. To conform with the goal for an automated and scalable framework, we choose methods that favor robustness and processing speed such that uploaded recordings can be processed at near real-time rates with little supervision.

Dense Reconstruction. We use volumetric fusion [15] to perform the dense reconstruction, since this approach is widely used in the context of commodity RGB-D

data. There is a large variety of algorithms targeting this scenario [72, 118, 9, 74, 42, 120, 52, 13, 120, 41, 18]. Our BundleFusion reconstruction approach (c.f. Chapter 3) was designed for this setup: it produces 3D reconstructions for portable commodity sensors like the Structure sensor, and provides real-time speed while being reasonably robust given handheld RGB-D video data. The real-time performance is critical, as this allows efficient reconstruction of thousands (or more than thousands) of scans.

For each input scan, we first run BundleFusion [18] at a voxel resolution of 1cm^3. This produces accurate pose alignments which we then use to perform volumetric integration through VoxelHashing [74] and extract a high resolution surface mesh using Marching Cubes [61] on the implicit TSDF (4mm^3 voxels). The mesh is then automatically cleaned up with a set of filtering steps to merge close vertices, delete duplicate and isolated mesh parts, and finally to downsample the mesh to high, medium, and low resolution versions (each level reducing the number of faces by half). These filtering steps are automatically applied through MeshLab[2] scripts.

Orientation. After the surface mesh is extracted, we automatically align it and all camera poses to a common coordinate frame with the z-axis as the up vector. To perform this alignment, we first extract all planar regions of sufficient size, merge regions defined by the same plane, and sort them by normal (we use a normal threshold of 25° and a planar offset threshold of 5cm). We then determine a prior for the up vector by projecting the IMU gravity vectors of all frames into the coordinates of the first frame. This allows us to select the floor plane based on the scan bounding box and the normal most similar to the IMU up vector direction.

Validation. This reconstruction process is automatically triggered when a scan is uploaded to the processing server and runs unsupervised. We automatically discard scan sequences that are short, have high residual reconstruction error, or have low percentage of aligned frames. We then manually check for and discard reconstructions with noticeable misalignments. Note that compared to other RGB-D datasets – e.g., NYU [70] or SUN RGB-D [98] –, we do not pre-process the raw depth data. Although

[2] `http://www.meshlab.net/`

methods like cross-bilateral filtering can lead to visually more appealing results and reduce holes, they often hallucinate false measures. Instead, we use volumetric fusion as part of the 3D reconstruction to regularize out noise from independent depth frames.

B.2 Data Annotation

How can we design an annotation framework to scale to the thousands of scans and millions of frames that we have collected? Similar to large-scale data collection and annotation efforts for images [90, 20, 59], we look to crowdsourcing to efficiently provide annotations for our RGB-D data. In particular, in order to efficiently annotate our RGB-D scan data for both 2D and 3D tasks, we construct our semantic annotation task in 3D, on the reconstructed meshes. Thus only the 1513 scans need to be labeled, and the labels can be projected back into the original RGB-D frames to provide pixel-level semantic segmentation annotations in 2D. This is in contrast to much prior work that uses 2D polygon annotations on RGB or RGB-D images, or 3D bounding box annotations.

After a reconstruction is produced by the processing server, we issue an annotation HIT (Human Intelligence Tasks) on the Amazon Mechanical Turk crowdsourcing market for the task of instance-level object category labeling of all surfaces in the reconstruction. The annotations are crowdsourced using web-based interfaces to maintain the overall scalability of the framework.

We developed a WebGL interface that takes as input the low-resolution surface mesh of a given reconstruction and a conservative over-segmentation of the mesh using a normal-based graph cut method [28, 43]. The crowd worker then selects segments to annotate with instance-level object category labels (see Fig. B.3). Each worker is required to annotate at least 25% of the surfaces in a reconstruction, and encouraged to annotate more than 50% before submission. Each scan is annotated by multiple workers (scans in ScanNet are annotated by 2.3 workers on average).

A key challenge in designing this interface is to enable efficient annotation by workers who have no prior experience with the task, or 3D interfaces in general. Our

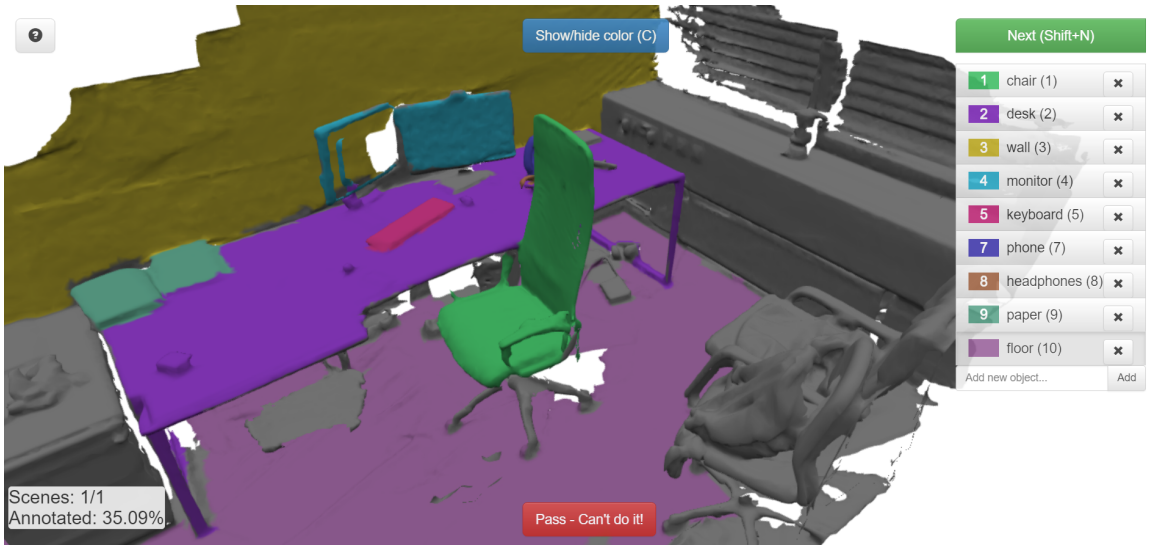

Figure B.3: Our web-based crowdsourcing interface for annotating a scene with instance-level object category labels. The right panel lists object instances already annotated in the scene with matching painted colors. This annotation is in progress at $\approx 35\%$, with gray regions indicating unannotated surfaces.

interface uses a simple painting metaphor where clicking and dragging over surfaces paints segments with a given label and corresponding color. This functions similarly to 2D painting and allows for erasing and modifying existing regions.

Another design requirement is to allow for freeform text labels, to reduce the inherent bias and scalability issues of pre-selected label lists. At the same time, it is desirable to guide users for consistency and coverage of basic object types. To achieve this, the interface provides autocomplete functionality over all labels previously provided by other workers that pass a frequency threshold (> 5 annotations). Workers are always allowed to add arbitrary text labels to ensure coverage and allow expansion of the label set.

Several additional design details are important to ensure usability by novice workers. First, a simple distance check for connectedness is used to disallow labeling of disconnected surfaces with the same label. Earlier experiments without this constraint resulted in two undesirable behaviors: cheating by painting many surfaces with a few labels, and labeling of multiple object instances with the same label. Second, the

3D nature of the data is challenging for novice users. Therefore, we first show a full turntable rotation of each reconstruction and instruct workers to change the view using a rotating turntable metaphor. Without the turntable rotation animation, many workers only annotated from the initial view and never used camera controls despite the provided instructions.

B.3 Impact of Real-world Data for 3D Semantic Understanding

Now that we have collected and annotated a large set of RGB-D scans for our ScanNet dataset, we explore the impact of using such real-world training data for various 3D semantic understanding tasks. In particular, we discuss our benchmark tasks of object classification and 3D semantic scene segmentation, and demonstrate the importance of real-world training data.

B.3.1 3D Object Classification

With the availability of large-scale synthetic 3D shape datasets such as [123, 8] and recent advances in 3D deep learning, research has developed approaches to classify objects using only geometric data with volumetric deep nets [123, 105, 64, 19, 79]. All of these methods train on purely synthetic data and focus on isolated objects. Although they show limited evaluation on real-world data, a larger evaluation on realistic scanning data is largely missing. When training data is synthetic and test is performed on real data, there is also a significant discrepancy of test performance, as data characteristics, such as noise and occlusions patterns, are inherently different.

With ScanNet, we close this gap as we have captured a sufficiently large amount of 3D data to use real-world RGB-D input for *both* training and test sets. For this task, we use the oriented bounding boxes of annotated objects in ScanNet, and isolate the contained geometry. As a result, we obtain local volumes around each object instance for which we know the annotated category. The goal of the task is to classify the object represented by a set of scanned points within a given bounding box. For this

benchmark, we use 17 categories, with $9,677$ train instances and $2,606$ test instances.

Network and training. For object classification, we follow the network architecture of the 3D Network-in-Network of [79], without the multi-orientation pooling step. In order to classify partial data, we add a second channel to the 30^3 occupancy grid input, indicating known and unknown regions (with 1 and 0, respectively) according to the camera scanning trajectory. As in Qi et al. [79], we use an SGD solver with learning rate 0.01 and momentum 0.9, decaying the learning rate by half every 20 epochs, and training the model for 200 epochs. We augment training samples with 12 instances of different rotations (including both elevation and tilt), resulting in a total training set of $111,660$ samples.

Benchmark performance. As a baseline evaluation, we run the 3D CNN approach of Qi et al. [79]. Table B.2 shows the performance of 3D shape classification with different train and test sets. The first two columns show results on synthetic test data from ShapeNet [8] including both complete and partial data. Naturally, training with the corresponding synthetic counterparts of ShapeNet provides the best performance, as data characteristics are shared. However, the more interesting case is real-world test data (rightmost two columns); here, we show results on test sets of SceneNN [39] and ScanNet. First, we see that training on synthetic data allows only for limited knowledge transfer (first two rows). Second, although the relatively small SceneNN dataset is able to learn within its own dataset to a reasonable degree, it does not generalize to the larger variety of environments found in ScanNet. On the other hand, training on ScanNet translates well to testing on SceneNN; as a result, the test results on SceneNN are significantly improved by using the training data from ScanNet. Interestingly, these results can be slightly improved when mixing training data of ScanNet with partial scans of ShapeNet (last row), indicating that there is room for more data to improve performance.

	Synthetic Test Sets		Real Test Sets	
Training Set	ShapeNet	ShapeNet Partial	SceneNN	ScanNet
ShapeNet	**92.5**	37.6	68.2	39.5
ShapeNet Partial	88.5	**92.1**	72.7	45.7
SceneNN	19.9	27.7	69.8	48.2
NYU	26.2	26.6	72.7	53.2
ScanNet	21.4	31.0	78.8	74.9
ScanNet+ShapeNet Par.	79.7	89.8	**81.2**	**76.6**

Table B.2: 3D object classification benchmark performance. Percentages give the classification accuracy over all models in each test set (average instance accuracy).

B.3.2 3D Semantic Scene Segmentation

A common task on RGB data is semantic segmentation (i.e. labeling pixels with semantic classes) [60]. With our data, we can extend this task to 3D, where the goal is to predict the semantic object label on a per-voxel basis. This task of predicting a semantic class for each visible 3D voxel has been addressed by some prior work, but using hand-crafted features to predict a small number of classes [48, 112], or focusing on outdoor environments [11, 6].

Data Generation. We first voxelize a scene and obtain a dense voxel grid with 2cm^3 voxels, where every voxel stores its TSDF value and object class annotation (empty space and unlabeled surface points have their own respective classes). We now extract subvolumes of the scene volume, of dimension $2 \times 31 \times 31 \times 62$ and spatial extent $1.5\text{m} \times 1.5\text{m} \times 3\text{m}$; i.e., a voxel size of $\approx 4.8\text{cm}^3$; the two channels represent the occupancy and known/unknown space according to the camera trajectory. These sample volumes are aligned with the xy-ground plane.For ground truth data generation, voxel labels are propagated from the scene voxelization to these sample volumes. The samples are chosen that $\geq 2\%$ of the voxels are occupied (i.e., on the surface), and $\geq 70\%$ of these surface voxels have valid annotations; samples not meeting these criteria are discarded. Across ScanNet, we generate $93,721$ subvolume examples for training, augmented by 8 rotations each (i.e., $749,768$ training samples), from 1201 training scenes. In addition, we extract $18,750$ sample volumes for testing,

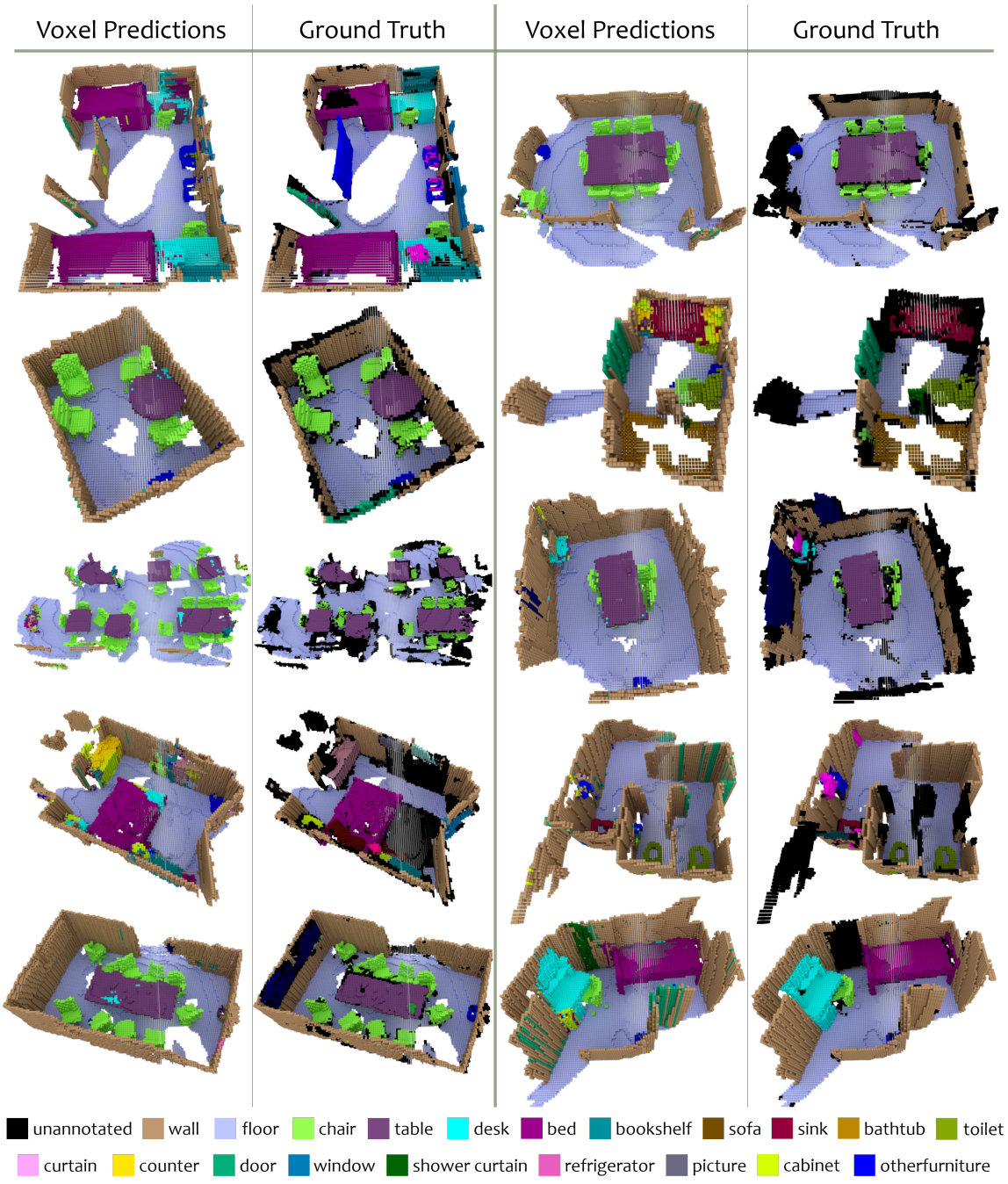

Figure B.4: 3D semantic scene segmentation of 3D scans in ScanNet using our 3D CNN architecture. Voxel colors indicate predicted or ground truth category.

Class	% of Test Scenes	Accuracy
Floor	35.7%	90.3%
Wall	38.8%	70.1%
Chair	3.8%	69.3%
Sofa	2.5%	75.7%
Table	3.3%	68.4%
Door	2.2%	48.9%
Cabinet	2.4%	49.8%
Bed	2.0%	62.4%
Desk	1.7%	36.8%
Toilet	0.2%	69.9%
Sink	0.2%	39.4%
Window	0.4%	20.1%
Picture	0.2%	3.4%
Bookshelf	1.6%	64.6%
Curtain	0.7%	7.0%
Shower Curtain	0.04%	46.8%
Counter	0.6%	32.1%
Refrigerator	0.3%	66.4%
Bathtub	0.2%	74.3%
OtherFurniture	2.9%	19.5%
Total	-	73.0%

Table B.3: 3D semantic scene segmentation accuracy on ScanNet test scenes.

which are also augmented by 8 rotations each (i.e., $150,000$ test samples) from 312 test scenes. We have 20 object class labels plus 1 class for free space.

Network and training. For the 3D semantic scene segmentation task, we propose a network which predicts class labels for a column of voxels in a scene according to the occupancy characteristics of the voxels' neighborhood. In order to infer labels for an entire scene, we use the network to predict a label for every voxel column at test time (i.e., every xy position that has voxels on the surface). The network takes as input a $2 \times 31 \times 31 \times 62$ volume and uses a series of fully convolutional layers to simultaneously predict class scores for the center column of 62 voxels. We use ReLU and batch normalization for all layers (except the last) in the network. To account for the unbalanced training data over the class labels, we weight the cross entropy loss with the inverse log of the histogram of the train data.

We use an SGD solver with learning rate 0.01 and momentum 0.9, decaying the learning rate by half every 20 epochs, and train the model for 100 epochs.

Quantitative Results. The goal of this task is to predict semantic labels for all visible surface voxels in a given 3D scene; i.e., every voxel on a visible surface receives one of the 20 object class labels. We use NYU2 labels, and list voxel classification results on ScanNet in Table B.3. We achieve a voxel classification accuracy of 73.0%, which is based purely on the geometric input (no color is used).

Note that we do not explicitly enforce prediction consistency between neighboring voxel columns when the *test volume* is slid across the xy plane. This could be achieved with a volumetric CRF [78], as used in [112]; however, our goal in this task to focus exclusively on the per-voxel classification accuracy.

B.4 Creating a 2D/3D Benchmark

In order to create a full benchmark challenge for ScanNet, providing consistent benchmarks and evaluation for various scene understanding tasks, we added several new improvements and additions.

2D/3D benchmark tasks. To leverage the full potential of the RGB-D scan data of ScanNet, which ties together both 2D and 3D data, our benchmark challenge covers several 2D and 3D tasks: 2D semantic segmentation, 2D instance segmentation, 3D semantic segmentation, 3D instance segmentation, and scene type classification. Evaluation for the semantic label and instance segmentation tasks runs in the respective domain, but a variety of input data types can be used, e.g., multiple images leveraging the video nature of the scans, the 3D geometry, etc.

Hidden test set. We collected 100 new scans, each of different scenes, spanning 20 different scene types. Only the raw RGB-D data and 3D reconstructions are publicly released, enabling a more consistent evaluation for all methods over this test set. Note that the scan collection was greatly facilitated by our automated data processing pipeline, and collection of new scans is largely bottlenecked by finding and traveling to new scenes.

Updated annotations. We updated the scene type annotations to a consistent 20-class set of scene types, as knowledge of the types of scenes that were scanned enabled more precise recategorization from the original 10 types. Additionally, we leveraged new 'merge' functionality and fixup annotation tasks to improve the consistency and coverage of the dense 3D semantic annotations, bringing surface annotation coverage from $\approx 60\%$ to $\approx 90\%$.

Automated evaluation server. To provide consistent evaluation over all benchmark tasks, we developed an automated evaluation server where users can submit the results on the hidden test set to be evaluated and ranked. 2D evaluation runs per-pixel in the image domain, and for 3D we designed the evaluation to run over the reconstructed mesh vertices. Since data formats are not as standardized for 3D as for 2D, this choice was made since it allows users to submit text files containing predictions for each vertex in the order provided in the mesh reconstruction. Moreover, the original annotations were provided on the mesh vertices, and since the meshes were originally extracted from TSDFs, the vertices and faces are relatively regular. This will enable easy and impartial evaluation of various approaches, and we hope it will encourage further work into various scene understanding directions.

Bibliography

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`, 2013.

[2] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[4] Christopher Batty. SDFGen. `https://github.com/christopherbatty/SDFGen`.

[5] Paul J Besl and Neil D McKay. A method for registration of 3-D shapes. *IEEE Trans. PAMI*, 14(2):239–256, 1992.

[6] Maros Blaha, Christoph Vogel, Audrey Richard, Jan D Wegner, Thomas Pock, and Konrad Schindler. Large-scale semantic 3d reconstruction: an adaptive multi-resolution model for multi-class volumetric labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3176–3184, 2016.

[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, 2015.

[9] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(4):113, 2013.

[10] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[11] Ian Cherabier, Christian Häne, Martin R Oswald, and Marc Pollefeys. Multi-label semantic 3d reconstruction using voxel blocks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 601–610. IEEE, 2016.

[12] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. June 2015.

[13] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565. IEEE, 2015.

[14] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016.

[15] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[16] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[17] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

[19] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[21] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[22] Zachary DeVito, Michael Mara, Michael Zollöfer, Gilbert Bernstein, Christian Theobalt, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. Opt: A domain specific language for non-linear least squares optimization in graphics and imaging. *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[23] Maurilio Di Cicco, Luca Iocchi, and Giorgio Grisetti. Non-parametric calibration for depth sensors. *Robotics and Autonomous Systems*, 74:309–317, 2015.

[24] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[25] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, September 2014.

[26] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. ICCV*, pages 1449–1456. IEEE, 2013.

[27] 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.

[28] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[29] Nicola Fioraio, Jonathan Taylor, Andrew Fitzgibbon, Luigi Di Stefano, and Shahram Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. June 2015.

[30] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Structured prediction of unobserved voxels from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5431–5440, 2016.

[31] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Proc. ICRA*, pages 15–22. IEEE, 2014.

[32] Daniel Girardeau-Montaut. CloudCompare3D point cloud and mesh processing software. *OpenSource Project*, 2011.

[33] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *TVCG*, 21(5):571–583, 2015.

[34] 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.

[35] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[36] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4077–4085, 2016.

[37] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *3D Vision (3DV), 2017 International Conference on*, pages 412–420. IEEE, 2017.

[38] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Robotics Research*, 31:647–663, April 2012.

[39] 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 *International Conference on 3D Vision (3DV)*, volume 1, 2016.

[40] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[41] Olaf Kähler, Victor A Prisacariu, and David W Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *European Conference on Computer Vision*, pages 500–516. Springer, 2016.

[42] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE transactions on visualization and computer graphics*, 21(11):1241–1250, 2015.

[43] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3D scenes via shape analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2088–2095. IEEE, 2013.

[44] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

[45] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

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

[47] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *Proc. 3DV*, pages 1–8. IEEE, 2013.

[48] Byung-soo Kim, Pushmeet Kohli, and Silvio Savarese. 3d scene understanding by voxel-crf. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1425–1432, 2013.

[49] Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):138, 2012.

[50] D Kinga and J Ba Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.

[51] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. ISMAR*, Nara, Japan, November 2007.

[52] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems*, 2015.

[53] 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.

[54] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Proc. ICRA*, pages 3607–3613. IEEE, 2011.

[55] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2479–2486, 2016.

[56] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[57] Hao Li, Etienne Vouga, Anton Gudym, Linjie Luo, Jonathan T Barron, and Gleb Gusev. 3d self-portraits. *ACM TOG*, 32(6):187, 2013.

[58] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015.

[59] 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.

[60] 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.

[61] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[62] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[63] Robert Maier, Jürgen Sturm, and Daniel Cremers. Submap-based bundle adjustment for 3d reconstruction from rgb-d data. In *Proc. GCPR*, Münster, Germany, September 2014.

[64] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21. Wiley Online Library, 2014.

[65] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J.Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. 2016.

[66] Maxime Meilland, A Comport, Patrick Rives, and INRIA Sophia Antipolis Méditerranée. Real-time dense visual tracking under large lighting variations. In *Proc. BMVC*, volume 29, 2011.

[67] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 560–568. ACM, 2006.

[68] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[69] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012.

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

[71] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389. ACM, 2006.

[72] 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 *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[73] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung. A robust 3D-2D interactive tool for scene segmentation and annotation. *arXiv preprint arXiv:1610.05883*, 2016.

[74] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 2013.

[75] 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.

[76] Mark Pauly, Niloy J Mitra, Joachim Giesen, Markus H Gross, and Leonidas J Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, number EPFL-CONF-149337, pages 23–32, 2005.

[77] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural regularity in 3d geometry. In *ACM transactions on graphics (TOG)*, volume 27, page 43. ACM, 2008.

[78] Krhenbühl Phillip and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2011.

[79] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[80] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[81] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[82] Scott E. Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. In *Proceedings of The 34th International Conference on Machine Learning (ICML)*, 2017.

[83] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *3D Vision (3DV), 2017 International Conference on*, pages 57–66. IEEE, 2017.

[84] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[85] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2484–2493, 2015.

[86] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[87] Henry Roth and Marsette Vona. Moving volume KinectFusion. In *Proc. BMVC*, 2012.

[88] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. *ACM TOG*, 21(3):438–446, 2002.

[89] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proc. 3DIM*, pages 145–152, 2001.

[90] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[91] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[92] Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. PiGraphs: Learning interaction snapshots from observations. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[93] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012.

[94] Yifei Shi, Pinxin Long, Kai Xu, Hui Huang, and Yueshan Xiong. Data-driven contextual modeling for 3d scene understanding. *Computers & Graphics*, 55:55–67, 2016.

[95] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, pages 167–178. IEEE, 2004.

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

[97] Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014.

[98] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015.

[99] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[100] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[101] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Shape Modeling Applications, 2004. Proceedings*, pages 191–199. IEEE, 2004.

[102] Pablo Speciale, Martin R Oswald, Andrea Cohen, and Marc Pollefeys. A symmetry prior for convex variational 3d reconstruction. In *European Conference on Computer Vision*, pages 313–328. Springer, 2016.

[103] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proc. ICCV*, Sydney, Australia, 2013.

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

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

[106] Minhyuk Sung, Vladimir G Kim, Roland Angst, and Leonidas Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 34(6):175, 2015.

[107] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, volume 2, page 8, 2017.

[108] Alex Teichman, Stephen Miller, and Sebastian Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and Systems*, volume 248, 2013.

[109] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1824–1831. IEEE, 2005.

[110] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment, a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[111] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proc. CVPR*, pages 4400–4408, 2015.

[112] Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi, and Philip Torr. SemanticPaint: Interactive 3D labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)*, 34(5):154, 2015.

[113] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[114] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *Neural Information Processing Systems (NIPS)*, 2016.

[115] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[116] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. In-hand scanning with online loop closure. In *Proc. ICCV Workshops*, pages 1630–1637, 2009.

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

[118] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially extended KinectFusion. Technical report, CSAIL, MIT, 2012.

[119] Thomas Whelan, Michael Kaess, John J Leonard, and John McDonald. Deformation-based loop closure for large scale dense rgb-d slam. In *Proc. IROS*, pages 548–555. IEEE, 2013.

[120] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. *Proc. Robotics: Science and Systems, Rome, Italy*, 2015.

[121] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proc. RSS*, Rome, Italy, July 2015.

[122] Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(6), 2014.

[123] 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.

[124] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proc. ICCV*, pages 1625–1632. IEEE, 2013.

[125] Raymond A Yeh, Chen Chen, Teck-Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, volume 2, page 4, 2017.

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

[127] Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, 2012.

[128] Wei Zhao, Shuming Gao, and Hongwei Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007.

[129] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (TOG)*, 32(4):112, 2013.

[130] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014.

[131] Qian-Yi Zhou, Steven Miller, and Vladlen Koltun. Elastic fragments for dense scene reconstruction. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 473–480. IEEE, 2013.

[132] Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 2015.