Learning to Reconstruct and Segment 3D Objects

Bo Yang

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2020

This thesis is dedicated to my parents
for their indispensable, altruistic support.

Acknowledgements

Looking back to the past three and a half years at Oxford, I feel grateful to many people who have helped me go through my DPhil journey.

First and foremost, I would like to thank my supervisors Niki and Andrew. Their remarkable supervision has reshaped my attitude towards meaningful research, trained my skills for solid work, and created my vision for novel ideas. These are indeed essential for all the achievements in this thesis.

Second, I would like to thank all my peer collaborators, especially the seniors Dr. Hongkai Wen, Dr. Stefano Rosa, Dr. Sen Wang, and Dr. Ronald Clark. Their advice is extremely timely and beneficial to shape my research ideas and experiments.

Third, I want to thank the labmates of Cyber-Physical System Group and my friends in the department of computer science and Exeter college. It is always enjoyable and memorable to go for tennis, formal dinners, punting, balls, drinks and so many others.

At last, I would like to express my sincere thanks to my family. The support and love from my parents and brothers are everlasting. The sweetness from my beloved girlfriend always make my day.

Abstract

To endow machines with the ability to perceive the real-world in a three dimensional representation as we do as humans is a fundamental and long-standing topic in Artificial Intelligence. Given different types of visual inputs such as images or point clouds acquired by 2D/3D sensors, one important goal is to understand the geometric structure and semantics of the 3D environment. Traditional approaches usually leverage hand-crafted features to estimate the shape and semantics of objects or scenes. However, they are difficult to generalize to novel objects and scenarios, and struggle to overcome critical issues caused by visual occlusions. By contrast, we aim to understand scenes and the objects within them by learning general and robust representations using deep neural networks, trained on large-scale real-world 3D data. To achieve these aims, this thesis makes three core contributions from object-level 3D shape estimation from single or multiple views to scene-level semantic understanding.

In Chapter 3, we start by estimating the full 3D shape of an object from a single image. To recover a dense 3D shape with geometric details, a powerful encoder-decoder architecture together with adversarial learning is proposed to learn feasible geometric priors from large-scale 3D object repositories. In Chapter 4, we build a more general framework to estimate accurate 3D shapes of objects from an arbitrary number of images. By introducing a novel attention-based aggregation module together with a two-stage training algorithm, our framework is able to integrate a variable number of input views, predicting robust and consistent 3D shapes for objects. In Chapter 5, we extend our study to 3D scenes which are generally a complex collection of individual objects. Real-world 3D scenes such as point clouds are usually cluttered, unstructured, occluded and incomplete. By drawing on previous work in point-based networks, we introduce a brand-new end-to-end pipeline to recognize, detect and segment all objects simultaneously in a 3D point cloud.

Overall, this thesis develops a series of novel data-driven algorithms to allow machines to perceive our real-world 3D environment, arguably pushing the boundaries of Artificial Intelligence and machine understanding.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Humans and other animals rely heavily on vision as a primary sensing modality for perceiving the world around them. Fundamentally, the brain makes sense of input from 2D retinal projections of the 3D physical world. These are sparse and incomplete, requiring the use of prior knowledge to infer scene structure and composition, and to recognize objects. As illustrated in Figure 1.1, just by taking a single glance at a sofa, we can imagine what its likely 3D shape is, guiding how we could interact with it such as sitting on it or moving it closer to the table. In fact, we not only focus on

Figure 1.1: Motivating example. We humans can effortlessly perceive the individual objects and understand the 3D scene from visual inputs, guiding how we interact with it.

the sofa in isolation, but simultaneously perceive the complex scene. For example, we can quickly identify the total number of seats available for our guests and localize the tables where we could serve the tea or coffee.

A long-standing goal in computer vision is to build intelligent systems which have similar capabilities to infer the underlying 3D structure of individual objects as well as understand the composition of multiple objects within complex 3D scenes. These systems would inspire a wide range of applications in robotics and augmented reality (AR). For example, every family is likely to be equipped with an intelligent robot to provide daily services for people in the future. Given a single/few snapshot(s) of the kitchen from a camera or depth scanner, a robot is able to estimate the 3D shape of a mug, where the handle is, and then accurately pour hot coffee without spilling it or overfilling. Being able to understand the complex structure of the living room, the robot can naturally identify and localize all chairs, tables, couches, etc., and smoothly navigate itself to deliver the coffee into our hands.

However, building these intelligent systems is highly challenging for two fundamental reasons. Firstly, 2D visual projections theoretically have infinite possible 3D geometries, especially when the 2D projections are sparse (*e.g.*, given a single or only a few views). Secondly, since real-world scenarios are usually a complex composition of objects and structures, many parts of the objects are occluded by one another, resulting in the sparsely observed scenes being incomplete and the individual objects fragmented. Overcoming these challenges requires the system to be able to effectively learn plausible geometric priors from visual inputs.

Early methods to recover the 3D shape of an object mainly leveraged hand-crafted features or explicit priors [101, 253, 179, 256]. However, these predefined geometric regularities are only applicable to limited shapes and also unable to estimate fine-grained geometric details. Prior attempts towards the more ambitious goal of understanding complex 3D scenes primarily focused on recovering a sparse 3D point cloud to represent the structure of scenes. These systems include the classic structure from motion (SfM) [1, 248, 226, 194] and simultaneous localization and mapping (SLAM) [42, 189, 188, 18, 227] pipelines, but they are unable to recognize and localize individual objects in the 3D space.

The recent advances in the area of deep neural networks has yielded impressive results for a wide variety of tasks on 2D images, such as object recognition [117], detection [54] and segmentation [77], thanks in part to the availability of large-scale 2D datasets, *e.g.*, ImageNet [222] and COCO [146]. In essence, these methods consist of multiple processing layers to automatically discover valuable representations from

the raw data for classification or detection [130]. Applying this powerful data driven approach to tackle core tasks in 3D space has emerged as a promising direction, especially since the introduction of many large-scale real-world or synthetic datasets for 3D objects and scenes. For example, Wu *et al.* introduce the ModelNet dataset in [287], Chang *et al.* present the ShapeNet dataset in [21] and Koch *et al.* introduce the ABC dataset in [113]. These are richly-annotated and large-scale repositories of 3D CAD models of objects. In addition to single objects, a variety of 3D indoor scenes with dense geometry and high dynamic range textures are collected in ScanNet [39], S3DIS [6], SceneNN [87], SceneNet RGB-D [169] and Replica [245], whereas large-scale 3D outdoor scenes are scanned by LiDAR in Semantic3D [73], SemanticKITTI [10] and SemanticPOSS [197].

However, the ability to unleash the full power of deep neural nets to learn rich 3D representations is still in its infancy, in spite of the availability of large datasets. This is primarily because 3D data are usually high-dimensional, irregular and incomplete. These issues serve as the main motivation of this thesis - to design novel neural networks to address core tasks for 3D perception such as object reconstruction and segmentation.

1.2 Research Challenges and Objectives

This thesis aims to design a vision system that is able to understand the geometric structure and semantics of the 3D visual world, from single or multiple scans of common sensors such as a camera, Kinect device or LiDAR. Instead of solving the entire task in one go, we instead approach it from a single object level to a more complex scene level. In particular, this thesis firstly aims to reconstruct the 3D shape of a single object from a sparse number of 2D images, and then focuses on interpreting more complex 3D scenes. However, learning to infer the object-level 3D shape and the scene-level semantics is non-trivial. More specifically, the challenges are three fold as discussed below.

- **How to estimate the 3D full shape of an object when there is only a single view.** This is the extreme case where the sensed information is limited and serves as a fundamental proof-of-principle for the use of deep neural networks. The fundamental challenge is how best to integrate the prior knowledge from the available datasets into the deep network, since the single view itself is unable to recover a full 3D shape.

- **How to infer a better 3D shape when there are multiple views available.** Theoretically, given more input images, the 3D shape can be estimated more accurately because more object parts are observed from various angles and perspectives. However, to effectively aggregate the useful information from different views is not easy.

- **How to identify individual objects within complex 3D scenes.** To localize and recognize all 3D objects within a real-world scene is a necessity for understanding the surrounding environment. Nevertheless, 3D scenes such as point clouds are usually visually incomplete and unordered, resulting in the existing neural architectures being ineffective and inefficient.

Motivated by these challenges, the thesis aims to achieve the corresponding three primary research objectives.

- The first objective is to recover the accurate 3D structure of individual objects from a single view. Particularly, we aim at estimating a dense and full 3D shape of an object from only one depth image acquired by a Kinect scanner. Initially, a single depth view together with camera parameters captures the partial shape of a 3D model. However, most object parts are occluded by the object itself. In order to recover the full shape, the main objective is to learn the prior geometric knowledge of possible object shapes, so as to complete the occluded parts. This objective is achieved in Chapter 3.

- The second objective is to extend the single-view reconstruction method to multi-view scenarios. Traditionally, the SfM and visual SLAM pipelines usually fail when the multiple views are separated by large baselines, because feature registration across views is prone to failure. Ideally, the useful visual features across different input views should be aggregated automatically, steadily improving the estimated 3D shape as supplied with increasing information. This objective is studied in Chapter 4.

- The third objective is to identify all object instances from complex 3D scenes. In particular, given real-world 3D point clouds, obtained from multiple images or LiDAR scans, we aim to precisely recognize and segment all objects at the point level. This objective is studied in Chapter 5.

1.3 Contributions

In this section, the main contributions of each chapter in this thesis are summarized as follows.

- In Chapter 3, we propose a deep neural architecture based on the generative adversarial network (GAN) to learn a dense 3D shape of an object from a *single* depth view. Compared with existing approaches, our architecture is able to reconstruct a more compelling 3D shape with fine-grained geometric details. In particular, the estimated 3D shape is represented with a high resolution 256^3 voxel grid, thanks to our capable encoder-decoder and stable discriminator, outperforming state-of-the-art techniques. Extensive experiments on both synthetic and real-world datasets show the high performance of our approach. The work is published in:

  Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham and Niki Trigoni. 3D Object Reconstruction from a Single Depth View with Adversarial Learning. International Conference on Computer Vision Workshops (ICCVW), 2017 [301].

  Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni and Hongkai Wen. Dense 3D Object Reconstruction from a Single Depth View. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018 [298].

- In Chapter 4, we propose a new neural module based on an attention mechanism to infer better 3D shapes of objects from multiple views. Compared with existing methods, our approach learns to attentively aggregate useful information from different images. We also introduce a two-stage training algorithm to guarantee the estimated 3D shapes being robust given an arbitrary number of input images. Experiments on multiple datasets demonstrate the superiority of our approach to recover accurate 3D shapes of objects. The work is published in:

  Bo Yang, Sen Wang, Andrew Markham and Niki Trigoni. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. International Journal of Computer Vision (IJCV), 2019 [300].

- In Chapter 5, we introduce a new framework to identify all individual 3D objects within large-scale 3D scenes. Compared with existing works, our framework is able to directly and simultaneously detect, segment and recognize all object

instances, without requiring any heavy pre/post processing steps. We demonstrate significant improvements over baselines on multiple large-scale real-world datasets. The work is published in:

Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham and Niki Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. Advances in Neural Information Processing Systems (NeurIPS Spotlight), 2019 [299].

In addition to the above contributions which address the research challenges discussed in Section 1.2, I have also made contributions to the following coauthored publications, which are related to my thesis, but will not be included.

- Bo Yang*, Zihang Lai*, Xiaoxuan Lu, Shuyu Lin, Hongkai Wen, Andrew Markham and Niki Trigoni. Learning 3D Scene Semantics and Structure from a Single Depth Image. Computer Vision and Pattern Recognition Workshops (CVPRW), 2018 [297].

- Zhihua Wang, Stefano Rosa, Linhai Xie, Bo Yang, Niki Trigoni and Andrew Markham. *Defo-Net: Learning body deformation using generative adversarial networks.* In International Conference on Robotics and Automation (ICRA), 2018 [274].

- Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, Niki Trigoni and Andrew Markham. *3D-PhysNet: Learning the intuitive physics of non-rigid object deformations.* In International Joint Conference on Artificial Intelligence (IJCAI), 2018 [275].

- Shuyu Lin, Bo Yang, Robert Birke and Ronald Clark. *Learning Semantically Meaningful Embeddings Using Linear Constraints.* In Computer Vision and Pattern Recognition Workshops (CVPRW), 2019 [144].

- Wei Wang, Muhamad Risqi U Saputra, Peijun Zhao, Pedro Gusmao, Bo Yang, Changhao Chen, Andrew Markham and Niki Trigoni. *DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network.* In International Conference on Robotics and Automation (ICRA),2019 [268].

- Qingyong Hu, Bo Yang*, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni and Andrew Markham. *RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds.* Computer Vision and Pattern Recognition (CVPR), 2020 [85].

1.4 Thesis Structure

The remainder of this thesis is organised as follows:

- Chapter 2 presents a comprehensive review of 3D perception of objects and scenes.
- Chapter 3 introduces our 3D-RecGAN++ approach, a generative adversarial network based method that predicts a dense 3D shape of an object from a single view.
- Chapter 4 presents our AttSets module and FASet algorithm, an attention based pipeline which steadily improves the estimated 3D shapes given more input views.
- Chapter 5 presents our 3D-BoNet framework that simultaneously recognizes, detects and segments all individual 3D objects in a complex scene.
- The last Chapter 6 concludes the entire thesis and identifies a number of future directions.

Chapter 2

Literature Review

In this chapter, I discuss previous work related to 3D object reconstruction and segmentation. In particular, I will start reviewing existing efforts on 3D shape recovery from single and multiple views in Sections 2.1 and 2.2, followed by deep neural algorithms designed for 3D point clouds in Section 2.3. Since the framework presented in Chapter 3 utilizes generative adversarial networks and the proposed neural module in Chapter 4 involves the attention mechanism and deep learning on sets, I therefore review generative adversarial frameworks in Section 2.4, attention mechanisms in Section 2.5 and deep neural networks for sets in Section 2.6. Lastly, Section 2.7 clarifies the relation and novelty of this thesis with regards to previous work.

2.1 Single View 3D Object Reconstruction

In this section, different pipelines for single-view 3D reconstruction or shape completion are reviewed. Both conventional geometry based techniques and state-of-the-art deep learning approaches are covered.

(1) 3D Model/Shape Completion. Monszpart *et al.* use plane fitting to complete small missing regions in [177], while shape symmetry is applied in [175, 200, 236, 242, 256] to fill in voids. Although these methods show good results, relying on predefined geometric regularities fundamentally limits the structure space to hand-crafted shapes. Besides, these approaches are likely to fail when the missing or occluded regions are relatively large. Another similar fitting pipeline is to leverage database priors. Given a partial shape input, an identical or most likely 3D model is retrieved and aligned with the partial scan [109, 140, 184, 228, 232, 219]. However, these approaches explicitly assume the database contains identical or very similar shapes, thus being unable to generalize to novel objects or categories.

(2) Single RGB Image Reconstruction. Predicting a complete 3D object model from a single view is a long-standing and extremely challenging task. When reconstructing a specific object category, model templates can be used. For example, morphable 3D models are exploited for face recovery [14, 47]. This concept was extended to reconstruct simple objects in [103]. For general and complex object reconstruction from a single RGB image, recent works [72, 258, 295] aim to infer 3D shapes using multiple RGB images for weak supervision. Shape prior knowledge is utilized in [116, 123, 182] for shape estimation. To recover high resolution 3D shapes, Octree representation is introduced in [250, 217, 33] to reduce computational burden, while an inverse discrete cosine transform (IDCT) technique is proposed in [100] along similar lines. Lin *et al.* [142] designed a pseudo-renderer to predict dense 3D shapes, whilst 2.5D sketches and dense 3D shapes are sequentially estimated from a single RGB image in [283].

(3) Single Depth View Reconstruction. The task of reconstruction from a single depth view is to complete the 3D structure, where visible parts occlude other parts of the object. 3D ShapeNets [287] is amongst some of the earliest work using deep neural nets to estimate 3D shapes from a single depth view. Firman *et al.* [56] trained a random decision forest to infer unknown voxels. Originally designed for shape denoising, VConv-DAE [229] can also be used for shape completion. To facilitate robotic grasping, Varley *et al.* proposed a neural network to infer the full 3D shape from a single depth view in [260]. However, all these approaches are only able to generate low resolution voxel grids which are less than 40^3 and unlikely to capture fine geometric details. Subsequent works [41, 241, 74, 269] can infer higher resolution 3D shapes. However, the pipeline in [41] relies on a shape database to synthesize a higher resolution shape after learning a small 32^3 voxel grid from a depth view, while SSCNet [241] requires voxel-level annotations for supervised scene completion and semantic label prediction. Both [74] and [269] were originally designed for shape inpainting instead of directly reconstructing the complete 3D structure from a partial depth view.

(4) Different 3D Shape Representations. The works discussed above usually aim to reconstruct a 3D voxel grid to represent the object shape. Being concurrent to these approaches, the proposed neural algorithm in Chapter 3 is also based on voxel grids. Since 3D voxels are memory inefficient, more recent pipelines tend to learn point clouds, meshes, implicit surfaces and other intermediate representations for 3D shape reconstruction. In particular, PointSet [55] is amongst the first works to learn a **point-based** 3D shape from a single image. A number of recent works

further improve its performance using adversarial learning [97], shape priors [138] and Silhouettes [326], and some other works aim to recover denser point clouds as in [142, 163, 165, 201, 154]. A number of unsupervised/weakly-supervised frameworks [94, 20, 124] are also proposed for point-based 3D reconstruction. To recover a **3D mesh** from a single image, early works learn to deform a template mesh in [105, 267, 277, 249, 156, 196, 68, 44], while a number of recent works learn to generate polygon meshes directly from images in [29, 186]. Instead of recovering explicit 3D shapes, a number of recent works learn **implicit surfaces** in [198, 171, 30]. This pipeline is further improved by [157, 191, 237] in which the 3D supervision is no longer required. Some other **intermediate representations**, such as shape primitives [327] and multi-layer depth [233, 190], are also studied.

2.2 Multi-view 3D Object Reconstruction

In this section, both the classical SfM/SLAM pipelines and learning based approaches are reviewed, with a stronger focus towards recent work in learning based methods, for the purpose of multi-view 3D object recontruction.

(1) Traditional SfM/SLAM. To estimate the underlying 3D shape from multiple images, classic SfM [194] and SLAM [18] algorithms firstly extract and match hand-crafted geometric features [76] and then apply bundle adjustment [257] for both shape and camera motion estimation. Existing SfM strategies include incremental [1, 57, 282, 226], hierarchical [60], and global approaches [36, 248]. Classic SLAM systems usually consist of feature-based [42, 180, 181, 40] and direct approaches [189, 188, 81, 224, 43, 64, 238, 227]. These systems are recently integrated with deep neural networks in [15, 168, 38, 290, 322]. Although they can reconstruct visually satisfactory 3D models, the recovered shapes are usually sparse point clouds and the occluded regions are unable to be estimated.

(2) Learning to Integrate Multi-views. Recent deep neural net based approaches tend to recover dense 3D shapes through learnt features from multiple color images and achieve compelling results. To fuse the deep features from multiple images, 3D-R2N2 [32], LSM [102] and 3D2SeqViews [75] apply the recurrent unit GRU, resulting in the networks being permutation variant and inefficient for aggregating a long sequence of images. SilNet [279, 280], DeepMVS [91] and 3DensiNet [266] simply use max pooling to preserve the first order information of multiple images, while RayNet [199] and [243] apply average pooling to retain the first moment information of multiple deep features. MVSNet [306] proposes a variance-based approach

to capture the second moment information for multiple feature aggregation. These pooling techniques only capture partial information, ignoring the majority of the deep features. In addition, the geometric consistency is not explicitly considered. To overcome this, recent works [143, 277, 61, 24, 288] learn multi-view stereo by applying the multi-view consistency or taking the depth prior into account.

Object shapes can also be recovered from multiple depth scans - the traditional volumetric fusion method [37, 19] integrates multiple viewpoint information by averaging truncated signed distance functions (TSDF). The recent learning based OctNetFusion [217] also adopts a similar strategy to integrate multiple depth information. However, this integration might result in information loss since TSDF values are averaged [217]. PSDF [46] was recently proposed to learn a probabilistic distribution through Bayesian updating in order to fuse multiple depth images, but it is not straightforward to include the module into existing encoder-decoder networks.

(3) Learning Two-view Stereos. SurfaceNet [96], SuperPixel Soup [122] and Stereo2Voxel [289] learn to reconstruct 3D shapes from two images. Although demonstrating the viability of recovering the 3D models, they are unable to process an arbitrary number of input images.

2.3 Segmentation for 3D Point Clouds

To extract features from 3D point clouds, traditional approaches usually crafted features manually [34, 223, 254]. However, these features are unable to adapt to more complex shapes and scenes. Recent learning based approaches mainly include projection-based, voxel-based [223, 90, 251, 218, 129, 216, 152, 170] and point-based schemes [206, 112, 82, 88, 246], which are widely employed for the core tasks of 3D point cloud perception, such as object recognition, semantic segmentation, object detection and instance segmentation. Basically, these tasks are similar to that of 2D images. In particular, the task of object recognition aims to estimate the category of a small set of 3D points, whereas the semantic segmentation aims to predict the category of each 3D point of a large-scale point cloud. More than simply classifying individual points, both the tasks of object detection and instance segmentation seek to localize each object, but the object detection only infers a 3D bounding box for an object whereas the instance segmentation needs to precisely identify an object instance that each 3D point belongs to.

(1) 3D Semantic Segmentation. Point clouds can be voxelized into 3D grids and then powerful 3D CNNs are applied as in [66, 129, 31, 170, 28]. Although they

achieve leading results on semantic segmentation, their primary limitation is the heavy computation cost, especially when processing large-scale point clouds.

The recent point-based method PointNet [206] shows leading results on classification and semantic segmentation, but it does not capture context features. To overcome this, many recent works introduced sophisticated neural modules to learn per-point local features. These modules can be generally classified as 1) neighbouring feature pooling [208, 137, 319, 318, 48], 2) graph message passing [273, 230, 264, 265, 22, 98, 153, 252, 293, 272], 3) kernel-based convolution [246, 294, 139, 88, 286, 133, 115, 126, 255, 166, 211, 161, 16, 80, 150], 4) attention-based aggregation [155, 317, 302, 195, 272] and 5) recurrent-based learning [160, 92, 307, 285].

To further enable the networks to consume large-scale point clouds, the multi-scale methods [52, 70] and graph-based SPG [127] are introduced to preprocess the large point clouds to learn per super-point semantics. The recent FCPN [216] and PCT [25] apply both voxel-based and point-based networks to process the massive point clouds. Although achieving promising results, these approaches also require extremely high computation and memory resources. To overcome this, the recent RandLA-Net [85] is able to efficiently and effectively process large-scale point clouds by introducing a novel local feature aggregation module together with an efficient random point feature down-sampling strategy.

(2) 3D Object Detection. The most common way to detect objects in 3D point clouds is to project points onto 2D images to regress bounding boxes [135, 7, 259, 27, 2, 296, 314, 281, 11, 173, 234, 84], by using existing 2D detectors. Detection performance is further improved by fusing RGB images in [27, 291, 118, 205, 276, 172, 212, 203], which require the 2D images to be well aligned with the 3D point clouds within the field of view. Point clouds can be also divided into voxels for object detection [51, 134, 323, 304, 28, 128, 23]. The detection strategies usually follow the mature frameworks for object detection in 2D images. However, most of these approaches rely on predefined anchors and the two-stage region proposal network [215]. It is inefficient to extend them to 3D point clouds. Without relying on anchors, the recent PointRCNN [231] learns to detect via foreground point segmentation, and VoteNet [204] detects objects via point feature grouping, sampling and voting.

(3) 3D Instance Segmentation. SGPN [270] is the first neural algorithm to segment instances on 3D point clouds by grouping the point-level embeddings. The subsequent ASIS [271], JSIS3D [202], MASC [151], 3D-BEVIS [49], MTML [125], JSNet [320], MPNet [79] and [141, 3] use the same strategy to group point-level features for instance segmentation. Mo *et al.* introduce a segmentation algorithm in

PartNet [176] by classifying point features. However, the learnt segments of these proposal-free methods do not have high objectness as they do not explicitly detect object boundaries. In addition, these methods usually require a post-processing algorithm, *e.g.*, mean shift [35], to cluster the learnt per-point features to obtain the final object instance labels, thereby resulting in an extremely heavy computation burden.

Another set of approaches to learn the object instances from 3D point clouds are proposal-based methods. By drawing on the successful 2D RPN [215] and RoI [77], GSPN [308] and 3D-SIS [83] learn a large number of candidate object bounding boxes followed by per-point mask prediction. However, these approaches usually rely on two-stage training and a post-processing step for dense proposal pruning.

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [65] are a novel framework to model complex, real-world data distributions. Inspired by game theory, GANs consist of a generator and a discriminator, which compete with each other. By transforming a source data distribution, the generator learns to synthesize a new data distribution to mimic the target distribution, while the discriminator learns to distinguish between the synthesized and the real target samples. GANs have achieved impressive success in image generation [209, 104], natural language [311], and time-series synthesis [45].

GANs can be extended to a conditional model if either the generator or the discriminator is conditioned on some extra data distribution [174]. Based on conditional GANs, images can be generated conditioned on class labels [193], text [213], and other information [214]. Conditional GANs are also used for photo-realistic image synthesis [316], image super-resolution [131], and image translation between domains [324].

GANs and conditional GANs are also applied in [58, 239, 240, 284] to generate low resolution 3D structures from noise or images. However, incorporating generative adversarial learning to estimate high resolution 3D shapes is not straightforward, as it is difficult to generate samples for high dimensional and complex data distributions. Fundamentally, this is because GANs are notoriously hard to train, suffering from instability issues [5, 4].

2.5 Attention Mechanisms

The attention mechanism was originally proposed for natural language processing in [8]. In a nutshell, it learns to weight deep features by importance scores for a specific

task, and then uses this weighting mechanism to improve that task. Compared with the traditional encoder-decoder RNN models, the attention-based approach is able to learn a more complicated dependency that ranges across a long input sequence. This dependency is not only important for the sequential domain of language processing, but also the spatial domain of many visual tasks. Being coupled with RNNs, the attention mechanism achieves compelling results in neural machine translation [8], image captioning [292], image question answering [305], *etc.*. However, all these RNN-based attention approaches are permutation variant to the order of input sequences. In addition, they are computationally time-consuming when the input sequence is long due to the recurrent processing.

Dispensing with recurrence and convolutions entirely and solely relying on attention mechanism, Transformer [261] achieves superior performance in machine translation tasks. Similarly, being decoupled from RNNs, attention mechanisms are also applied for visual recognition [99, 220, 159, 225, 325, 183, 63], semantic segmentation [136], long sequence learning [210], multi-task learning [158], and image generation [315]. Although the above decoupled attention modules can be used to aggregate variable sized deep feature sets, they are literally designed to operate on fixed sized features for tasks such as image recognition and generation. The robustness of attention modules in the context of dynamic deep feature sets has not been investigated yet.

2.6 Deep Learning on Sets

In contrast to traditional approaches operating on fixed dimensional vectors or matrices, deep learning tasks defined on sets usually require learning functions to be permutation invariant and able to process an arbitrary number of elements in a set.

Zaheer *et al.* introduce general permutation invariant and equivariant models in [313], and they end up with a **sum pooling** for permutation invariant tasks such as population statistics estimation and point cloud classification. In the recent GQN [53], sum pooling is also used to aggregate an arbitrary number of orderless images for 3D scene representation. Gardner *et al.* [59] use **average pooling** to integrate an unordered deep feature set for a classification task. Su *et al.* [247] use **max pooling** to fuse the deep feature set of multiple views for 3D shape recognition. Similarly, PointNet [206] also uses max pooling to aggregate the set of features learnt from point clouds for 3D classification and segmentation. In addition, the higher-order statistics based pooling approaches are widely used for 3D object recognition from

multiple images. Vanilla **bilinear pooling** is applied for fine-grained recognition in [149] and is further improved in [147]. Concurrently, **log-covariance pooling** is proposed in [95], and is recently generalized by **harmonized bilinear pooling** in [312]. Bilinear pooling techniques are further improved in recent work [310, 148]. However, both first-order and higher-order pooling operations ignore a majority of the information of a set. In addition, the first-order poolings do not have trainable parameters, while the higher-order poolings have only a few parameters available for the network to learn. These limitations lead to the pooling based neural networks to be optimized with regards to the specific statistics of data batches during training, and therefore unable to be robust and generalize well to variable sized deep feature sets during testing.

2.7 Novelty with Respect to State of the Art

This section hightlights the novel aspects of this thesis compared with state of the art in the context of single/multi view 3D reconstruction and segmentation of 3D point clouds.

(1) Single View 3D Reconstruction. To recover the 3D shape of an object from a single depth view, the 3D-RecGAN++ is proposed in Chapter 3. The neural architecture consists of a generator which synthesizes a 3D shape conditioned on an input partial depth view, and a discriminator to distinguish whether the input 3D shape is synthesized or real. The overall pipeline extends from conditional GANs [174] as discussed in Section 2.4. However the existing adversarial loss functions, such as the original GAN [65], WGAN [5] and WGAN-GP [4], are unable to converge to synthesize a high dimensional 3D shape. To overcome this, the proposed 3D-RecGAN++ introduces a mean feature layer for the discriminator to stabilize the entire framework.

There are many other neural algorithms to estimate the 3D shape from a single view as discussed in Section 2.1. The most similar works to 3D-RecGAN++ are 3D-EPN [41], 3D-GAN [284], Varley *et al.* [260] and Han *et al.* [74]. However, all of them are only able to generate a low resolution 3D voxel grid, less than 128^3, to represent the shape of an object. By contrast, the proposed 3D-RecGAN++ directly generates a 3D shape within a 256^3 voxel grid, which is able to recover fine-grained geometric details. Since 3D-RecGAN++ uses a voxel grid to represent 3D shape, its memory consumption is generally less efficient than the latest approaches (published after

3D-RecGAN++) based on point clouds, meshes and implicit surfaces as discussed in Section 2.1.

(2) Multi-view 3D Reconstruction. In Chapter 4, we propose an AttSets module together with a FASet algorithm to integrate a variable number of views for more accurate shape estimation. The AttSets module extends the general idea of attention mechanism discussed in Section 2.5. Similar works include Transformer [261], SENet [99] and [210, 63]. However, the existing attention mechanisms are only able to be applied to a fixed number of input elements. To integrate an arbitrary number of input images, we formulate this multi-view 3D reconstruction as an aggregation process and propose AttSets with a series of carefully designed neural functions. Fundamentally, our AttSets involves deep learning techniques for sets. In this regard, the recent works [313, 132, 263, 93] are similar to AttSets. However, these existing approaches neglect an important issue. In particular, the existing neural algorithms are not robust to a variable number of input elements, and their performance drops if the cardinality of testing sets is significantly different from that of training sets. To overcome this, we further propose the FASet algorithm to separately optimize the AttSets module and the base feature extractor, guaranteeing the robustness of the entire network with regards to a variable number of input images.

As discussed in Section 2.2, the common way to fuse multiple views for 3D shape estimation leverages RNNs [32, 102] or heuristic poolings such as max/mean/sum poolings [279, 91]. However, the RNN approaches formulate the multiple views as an ordered sequence and the reconstructed shape varies given a different order of the same image set. The heuristic poolings usually discard the majority of the information, thereby being unable to obtain better 3D shapes even if more images are given. By contrast, our AttSets and FASet are able to attentively aggregate useful information from an arbitrary number of views and guarantee the final recovered shape is robust to the number of input images.

(3) Segmentation of 3D Point Clouds. Beyond recovering the 3D shape of a single object, we aim to identify all individual 3D objects from large-scale real-world point clouds in Chapter 5. The proposed 3D-BoNet is a general framework to recognize, detect and segment all object instances simultaneously. The backbone of the framework extends the basic idea of shared MLPs invented by PointNet/PointNet++ [206, 208].

To recognize 3D objects, 3D-BoNet simply leverages any of the front-ends of existing point-based networks such as PointNet++ [208] and SparseConv [66]. To detect and segment individual objects, existing works either follow the idea of RPN [215] to

extensively localize objects in 3D space, or learn to directly cluster per-point features as discussed in Section 2.3. However, these methods have a number of limitations. First, they usually require post-processing steps such as non-maximum suppression or mean shift clustering, which are extremely computationally heavy. Second, the learnt instances do not have high objectness as they do not explicitly learn object boundaries and the low-level point features are very likely to be incorrectly clustered. To overcome these shortcomings, our 3D-BoNet framework offers the following unique aspects: 1) the object bonding boxes are directly learnt from the global features of a point cloud via a carefully designed neural optimal association layer, guaranteeing high objectness of all detected instances; 2) each object is further precisely segmented within a bounding box via a simple binary classifier, without requiring any post-processing steps. The aspects above make 3D-BoNet simpler and more efficient than existing competing pipelines.

Chapter 3

Learning to Reconstruct 3D Object from a Single View

3.1 Introduction

To reconstruct the complete and precise 3D geometry of an object is essential for many graphics and robotics applications, from augmented reality (AR)/virtual reality (VR) [229] and semantic understanding, to object deformation [274], robot grasping [260] and obstacle avoidance. Classic approaches use off-the-shelf low-cost depth sensing devices such as Kinect and RealSense cameras to recover the 3D shape of an object from captured depth images. These approaches typically require multiple depth images from different viewing angles of an object to estimate the complete 3D structure [188][192][244]. However, in practice it is not always feasible to scan all surfaces of an object before reconstruction, which leads to incomplete 3D shapes with occluded regions and large holes. In addition, acquiring and processing multiple depth views require more computing power, which is not ideal in many applications that require real-time performance.

We aim to tackle the problem of estimating the complete 3D structure of an object using a single depth view. This is a very challenging task, since the partial observation of the object (*i.e.*, a depth image from one viewing angle) can be theoretically associated with an infinite number of possible 3D models. Traditional reconstruction approaches typically use interpolation techniques such as plane fitting, Laplacian hole filling [187][321], or Poisson surface estimation [106][107] to infer the underlying 3D structure. However, they can only recover very limited occluded or missing regions, *e.g.*, small holes or gaps due to quantization artifacts, sensor noise and insufficient geometry information.

Interestingly, humans are surprisingly good at solving such ambiguity by implicitly leveraging prior knowledge. For example, given a view of a chair with two rear legs occluded by front legs, humans are easily able to guess the most likely shape behind the visible parts. Recent advances in deep neural networks and data driven approaches show promising results in dealing with such a task.

In this chapter, we aim to acquire the complete and high-resolution 3D shape of an object given a single depth view. By leveraging the high performance of 3D convolutional neural nets and large open datasets of 3D models, our approach learns a smooth function that maps a 2.5D view to a complete and dense 3D shape. In particular, we train an end-to-end model which estimates full volumetric occupancy from a single 2.5D depth view of an object.

While state-of-the-art deep learning approaches [287][41][260] for 3D shape reconstruction from a single depth view achieve encouraging results, they are limited to very small resolutions, typically at the scale of 32^3 voxel grids. As a result, the learnt 3D structure tends to be coarse and inaccurate. In order to generate higher resolution 3D objects with efficient computation, Octree representation has been recently introduced in [250][217][33]. However, increasing the density of output 3D shapes would also inevitably pose a great challenge to learn the geometric details for high resolution 3D structures, which has yet to be explored.

Recently, deep generative models achieve impressive success in modeling complex high-dimensional data distributions, among which Generative Adversarial Networks (GANs) [65] and Variational Autoencoders (VAEs) [111] emerge as two powerful frameworks for generative learning, including image and text generation [86][104], and latent space learning [26][121]. In the past few years, a number of works [67][62][89][284] applied such generative models to learn latent space to represent 3D object shapes, in order to solve tasks such as new image generation, object classification, recognition and shape retrieval.

In this chapter, we propose 3D-RecGAN++, a simple yet effective model that combines a skip-connected 3D encoder-decoder with adversarial learning to generate a complete and fine-grained 3D structure conditioned on a single 2.5D view. In particular, our model firstly encodes the 2.5D view to a compressed latent representation which implicitly represents general 3D geometric structures, then decodes it back to the most likely full 3D shape. Skip-connections are applied between the encoder and decoder to preserve high frequency information. The rough 3D shape is then fed into a conditional discriminator which is adversarially trained to distinguish whether the coarse 3D structure is plausible or not. The encoder-decoder is able to approximate

the corresponding shape, while the adversarial training tends to add fine details to the estimated shape. To ensure the final generated 3D shape corresponds to the input single partial 2.5D view, adversarial training of our model is based on a conditional GAN [174] instead of random guessing. The above network excels the competing approaches [260][41][74], which either use a single fully connected layer [260], a low capacity decoder without adversarial learning [41], or the multi-stage and ineffective LSTMs [74] to estimate the full 3D shapes.

Our contributions are as follows:

- We propose a simple yet effective model to reconstruct the complete and accurate 3D structure using a single arbitrary depth view. Particularly, our model takes a simple occupancy grid map as input without requiring object class labels or any annotations, while predicting a compelling shape within a high resolution of 256^3 voxel grid. By drawing on both 3D encoder-decoder and adversarial learning, our approach is end-to-end trainable with high level of generality.

- We exploit conditional adversarial training to refine the 3D shape estimated by the encoder-decoder. Our contribution here is that we use the mean value of a latent vector feature, instead of a single scalar, as the output of the discriminator to stabilize GAN training.

- We conduct extensive experiments for single category and multi-category object reconstruction, outperforming the state of the art. Importantly, our approach is also able to generalize to previously unseen object categories. Lastly, our model also is shown to perform robustly on real-world data, after being trained purely on synthetic datasets.

- To the best of our knowledge, there are no good open datasets which have the ground truth for occluded/missing parts and holes for each 2.5D view in real-world scenarios. We therefore collect and release our real-world testing dataset to the community.

Our code and data are available at: *https://github.com/Yang7879/3D-RecGAN-extended*

3.2 Method Overview

Our method aims to estimate a complete and dense 3D structure of an object, which only takes an arbitrary single 2.5D depth view as input. The output 3D shape is

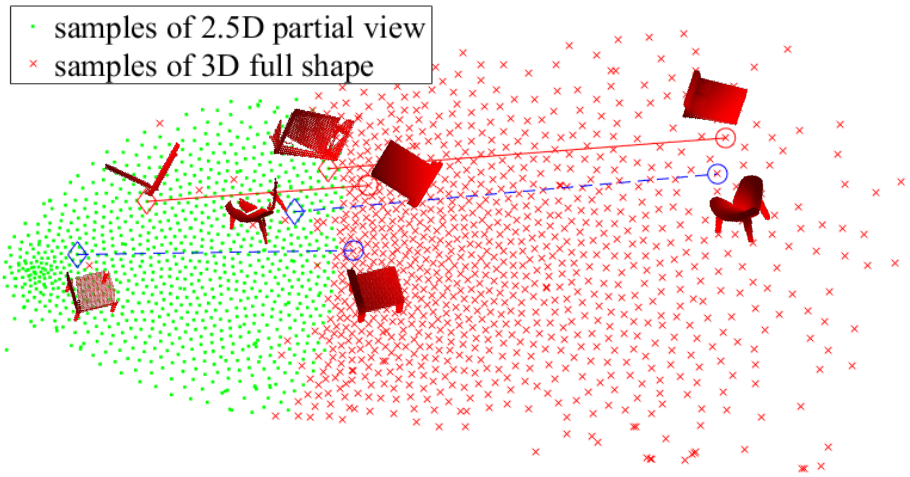


Figure 3.1: t-SNE embeddings of 2.5D partial views and 3D complete shapes of multiple object categories.

automatically aligned with the corresponding 2.5D partial view. To achieve this task, each object model is represented by a high resolution 3D voxel grid. We use the simple occupancy grid for shape encoding, where 1 represents an occupied cell and 0 an empty cell. Specifically, the input 2.5D partial view, denoted as $\boldsymbol{x}$, is a 64^3 occupancy grid, while the output 3D shape, denoted as $\boldsymbol{y}$, is a high resolution 256^3 probabilistic voxel grid. The input partial shape is directly calculated from a single depth image given camera parameters. We use the ground truth dense 3D shape with aligned orientation as same as the input partial 2.5D depth view to supervise our network.

To generate ground truth training and evaluation pairs, we virtually scan 3D objects from ShapeNet [21]. Figure 3.1 is the t-SNE visualization [164] of partial 2.5D views and the corresponding full 3D shapes for multiple general chair and bed models. Each green dot represents the t-SNE embedding of a 2.5D view, whilst a red dot is the embedding of the corresponding 3D shape. It can be seen that multiple categories inherently have similar 2.5D to 3D mapping relationships. Essentially, our neural network is to learn a smooth function, denoted as f, which maps green dots to red dots as close as possible in high dimensional space as shown in Equation 3.1. The function f is parametrized by neural layers in general.

$$\boldsymbol{y} = f(\boldsymbol{x}) \quad \left(\boldsymbol{x} \in Z^{64^3}, where\ Z = \{0, 1\}\right) \tag{3.1}$$

After generating training pairs, we feed them into our network. The first part of our network loosely follows the idea of a 3D encoder-decoder with the U-net connections [221]. The skip-connected encoder-decoder serves as an initial coarse generator

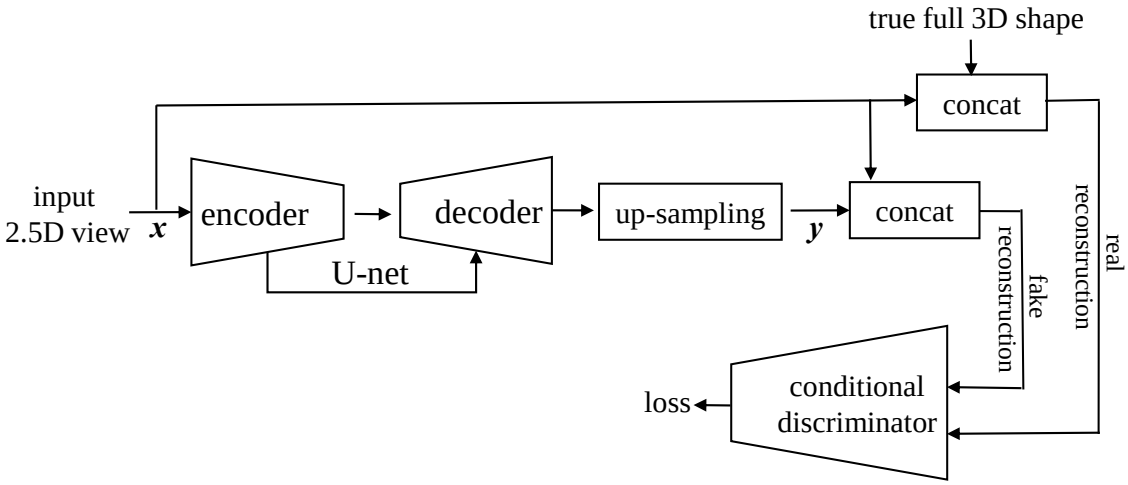


Figure 3.2: Overview of the network architecture for training.

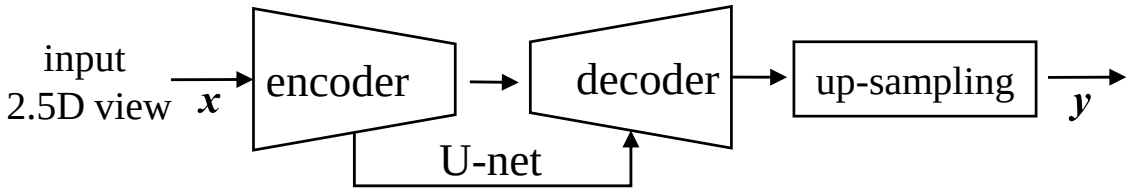


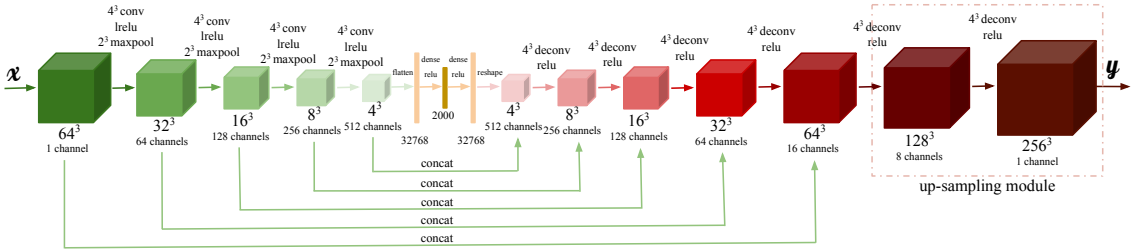
Figure 3.3: Overview of the network architecture for testing.

which is followed by an up-sampling module to further generate a higher resolution 3D shape within a 256^3 voxel grid. This whole generator learns a correlation between partial and complete 3D structures. With the supervision of complete 3D labels, the generator is able to learn a function f and infer a reasonable 3D shape given a brand new partial 2.5D view. During testing, however, the results tend to be grainy and without fine details.

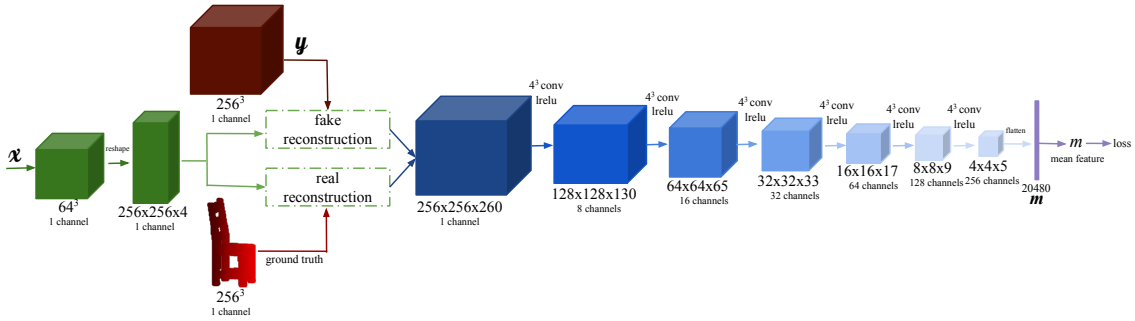
To address this issue, in the training phase, the reconstructed 3D shape from the generator is further fed into a conditional discriminator to verify its plausibility. In particular, a partial 2.5D input view is paired with its corresponding complete 3D shape, which is called the 'real reconstruction', while the partial 2.5D view is paired with its corresponding output 3D shape from generator, which is called the 'fake reconstruction'. The discriminator aims to discriminate all 'fake reconstruction' from 'real reconstruction'. In the original GAN framework [65], the task of the discriminator is to simply classify real and fake inputs, but its Jensen-Shannon divergence-based loss function is difficult to converge. The recent WGAN [5] leverages Wasserstein distance with weight clipping as a loss function to stabilize the training procedure, whilst the extended work WGAN-GP [71] further improves the training process using a gra-

dient penalty with respect to its input. In our 3D-RecGAN++, we apply WGAN-GP as the loss function on top of the mean feature of our conditional discriminator, which guarantees fast and stable convergence. The overall network architecture for training is shown in Figure 3.2, while the testing phase only needs the well trained generator as shown in Figure 3.3.

3.3 Method Details



(a) Generator for 3D shape estimation from a single depth view.



(b) Discriminator for 3D shape refinement.

Figure 3.4: Detailed architecture of 3D-RecGAN++, showing the two main building blocks. Note that, although these are shown as two separate modules, they are trained end-to-end.

3.3.1 Network Architecture

Figure 3.4 shows the detailed architecture of our proposed 3D-RecGAN++. It consists of two main networks: the generator as in Figure 3.4a and the discriminator as in Figure 3.4b.

The generator consists of a skip-connected encoder-decoder and an up-sampling module. Unlike the vanilla GAN generator which generates data from arbitrary latent distributions, our 3D-RecGAN++ generator synthesizes data from 2.5D views. Particularly, the encoder has five 3D convolutional layers, each of which has a bank of

$4 \times 4 \times 4$ filters with strides of $1 \times 1 \times 1$, followed by a leaky ReLU activation function and a max pooling layer with $2 \times 2 \times 2$ filters and strides of $2 \times 2 \times 2$. The number of output channels of max pooling layer starts with 64, doubling at each subsequent layer and ends up with 512. The encoder is lastly followed by two fully-connected layers to embed semantic information into a latent space. The decoder is composed of five symmetric up-convolutional layers which are followed by ReLU activations. Skip-connections between encoder and decoder guarantee propagation of local structures of the input 2.5D view. The skip-connected encoder-decoder is followed by the up-sampling module which simply consists of two layers of up-convolutional layers as detailed in Figure 3.4a. This simple up-sampling module directly upgrades the output 3D shape to a higher resolution of 256^3 without requiring complex network design and operations. It should be noted that without the two fully connected layers and skip-connections, the vanilla encoder-decoder would be unable to learn reasonable complete 3D structures as the latent space is limited and the local structure is not preserved. The loss function and optimization methods are described in Section **??**.

The discriminator aims to distinguish whether the estimated 3D shapes are plausible or not. Based on the conditional GAN, the discriminator takes both real reconstruction pairs and fake reconstruction pairs as input. In particular, it consists of six 3D convolutional layers, the first of which concatenates the generated 3D shape (*i.e.*, a 256^3 voxel grid) and the input 2.5D partial view (*i.e.*, a 64^3 voxel grid), reshaped as a $256 \times 256 \times 4$ tensor. The reshaping process is done straightforwardly using Tensorflow 'tf.reshape()'. Basically, this is to inject the condition information with a matched tensor dimension, and then leave the network itself to learn useful features from this condition input. Each convolutional layer has a bank of $4 \times 4 \times 4$ filters with strides of $2 \times 2 \times 2$, followed by a ReLU activation function except for the last layer which is followed by a sigmoid activation function. The number of output channels of the convolutional layers starts with 8, doubling at each subsequent layer and ends up with 256. The output of the last neural layer is reshaped as a latent vector which is the latent feature of discriminator, denoted as $\boldsymbol{m}$.

3.3.2 Mean Feature for Discriminator

At the early training stage of GAN, as the high dimensional real and fake distributions may not overlap, the discriminator can separate them perfectly using a single scalar output, which is analyzed in [4]. In our experiments, due to the extremely high dimensionality (*i.e.*, $256^3 + 64^3$ dimensions) of the input data pair, the WGAN-GP

always crashes in the early 3 epochs if we use a standard fully-connected layer followed by a single scalar as the final output for the discriminator.

To stabilize training, we propose to use the mean feature m (*i.e.*, mean of a vector feature $\boldsymbol{m}$) for discrimination. As the mean vector feature tends to capture more information from the input overall, it is more difficult for the discriminator to easily distinguish between fake or real inputs. This enables useful information to back-propagate to the generator. The final output of the discriminator $D()$ is defined as:

$$m = \mathbf{E}(\boldsymbol{m}) \tag{3.2}$$

Mean feature matching is also studied and applied in [9][178] to stabilize GAN. However, Bao *et al.* [9] minimize the L_2 loss of the mean feature, as well as the original Jensen-Shannon divergence-based loss [65], requiring hyper-parameter tuning to balance the two losses. By comparison, in our 3D-RecGAN++ setting, the mean feature of discriminator is directly followed by the existing WGAN-GP loss, which is simple yet effective to stabilize the adversarial training.

Overall, our discriminator learns to distinguish the distributions of mean feature of fake and real reconstructions, while the generator is trained to make the two mean feature distributions as similar as possible.

3.3.3 Loss Functions

The objective function of 3D-RecGAN++ includes two main parts: an object reconstruction loss ℓ_{en} for the generator; the objective function ℓ_{gan} for the conditional GAN.

(1) ℓ_{en} For the generator, inspired by [17], we use modified binary cross-entropy loss function instead of the standard version. The standard binary cross-entropy weights both false positive and false negative results equally. However, most of the voxel grid tends to be empty, so the network easily gets a false positive estimation. In this regard, we impose a higher penalty on false positive results than on false negatives. Particularly, a weight hyper-parameter α is assigned to false positives, with (1-α) for false negative results, as shown in Equation 3.3.

$$\ell_{en} = \frac{1}{N}\sum_{i=1}^{N}\Big[-\alpha\bar{y_i}\log(y_i) - (1-\alpha)(1-\bar{y_i})\log(1-y_i)\Big] \tag{3.3}$$

where $\bar{y_i}$ is the target value {0,1} of a specific i^{th} voxel in the ground truth voxel grid $\bar{\boldsymbol{y}}$, and y_i is the corresponding estimated value (0,1) in the same voxel from the

generator output $\boldsymbol{y}$. We calculate the mean loss over the total N voxels in the whole voxel grid.

(2) ℓ_{gan} For the discriminator, we leverage the state of the art WGAN-GP loss functions. Unlike the original GAN loss function which presents an overall loss for both real and fake inputs, we separately represent the loss function ℓ^g_{gan} in Equation 3.4 for generating fake reconstruction pairs and ℓ^d_{gan} in Equation 3.5 for discriminating fake and real reconstruction pairs. Detailed definitions and derivation of the loss functions can be found in [5][71], but we modify them for our conditional GAN settings.

$$\ell^g_{gan} = -\mathbf{E}\left[D(\boldsymbol{y}|\boldsymbol{x})\right] \tag{3.4}$$

$$\ell^d_{gan} = \mathbf{E}\left[D(\boldsymbol{y}|\boldsymbol{x})\right] - \mathbf{E}\left[D(\bar{\boldsymbol{y}}|\boldsymbol{x})\right] + \lambda\mathbf{E}\left[\left(\left\|\nabla_{\hat{\boldsymbol{y}}}D(\hat{\boldsymbol{y}}|\boldsymbol{x})\right\|_2 - 1\right)^2\right] \tag{3.5}$$

where $\hat{\boldsymbol{y}} = \epsilon\bar{\boldsymbol{y}} + (1-\epsilon)\boldsymbol{y}, \epsilon \sim U[0,1]$, $\boldsymbol{x}$ is the input partial depth view, $\boldsymbol{y}$ is the corresponding output of the generator, $\bar{\boldsymbol{y}}$ is the corresponding ground truth. λ controls the trade-off between optimizing the gradient penalty and the original objective in WGAN.

For the generator in our 3D-RecGAN++ network, there are two loss functions, ℓ_{en} and ℓ^g_{gan}, to optimize. As we discussed in Section 3.2, minimizing ℓ_{en} tends to learn the overall 3D shapes, whilst minimizing ℓ^g_{gan} estimates more plausible 3D structures conditioned on input 2.5D views. To minimize ℓ^d_{gan} is to improve the performance of discriminator to distinguish fake and real reconstruction pairs. To jointly optimize the generator, we assign weights β to ℓ_{en} and $(1-\beta)$ to ℓ^g_{gan}. Overall, the loss functions for generator and discriminator are as follows:

$$\ell_g = \beta\ell_{en} + (1-\beta)\ell^g_{gan} \tag{3.6}$$

$$\ell_d = \ell^d_{gan} \tag{3.7}$$

3.3.4 Implementation

We adopt an end-to-end training procedure for the whole network. To simultaneously optimize both the generator and discriminator, we alternate between one gradient descent step on the discriminator and then one step on the generator. For the WGAN-GP, λ is set as 10 for gradient penalty as in [71]. α ends up as 0.85 for our modified cross entropy loss function, while β is 0.2 for the joint loss function ℓ_g.

The Adam solver[110] is used for both discriminator and generator with a batch size of 4. The other three Adam parameters are set to default values. Learning rate

is set to $5e^{-5}$ for the discriminator and $1e^{-4}$ for the generator in all epochs. As we do not use dropout or batch normalization, the testing phase is exactly the same as the training stage. The whole network is trained on a single Titan X GPU from scratch.

3.3.5 Data Synthesis

For the task of 3D dense reconstruction from a single depth view, obtaining a large amount of training data is an obstacle. Existing real RGB-D datasets for surface reconstruction suffer from occlusions and missing data and there is no ground truth of complete and high resolution 256^3 3D shapes for each view. The recent work [41] synthesizes data for 3D object completion, but the object resolution is only up to a resolution of 128^3.

To tackle this issue, we use the ShapeNet [21] database to generate a large amount of training and testing data with synthetically rendered depth images and the corresponding complete 3D shape ground truth. Interior parts of individual objects are set to be filled, *i.e.*, '1', while the exterior to be empty, *i.e.*, '0'. A subset of object categories and CAD models are selected for our experiments. As some CAD models in ShapeNet may not be watertight, in our ray tracing based voxelization algorithm, if a specific point is inside more than 5 faces along the X, Y and Z axes, that point is deemed to be the interior of the object and set as '1', otherwise set to '0'.

For each category, to generate **training data**, around 220 CAD models are randomly selected. For each CAD model, we create a virtual depth camera to scan it from 125 different viewing angles, 5 uniformly sampled views for each of roll, pitch and yaw space ranging from $0 \sim 2\pi$ individually. Note that, the viewing angles for all 3D models are the same for simplicity. For each virtual scan, both a depth image and the corresponding complete 3D voxelized structure are generated with regard to the same camera angle. That depth image is simultaneously transformed to a point cloud using virtual camera parameters [108] followed by voxelization which generates a partial 2.5D voxel grid. Then a pair of partial 2.5D view and the complete 3D shape is synthesized. Overall, around 26K training pairs are generated for each 3D object category.

For each category, to synthesize **testing data**, around 40 CAD models are randomly selected. For each CAD model, two groups of testing data are generated. **Group 1**, each model is virtually scanned from 125 viewing angles which are the same as used in training dataset. Around 4.5k testing pairs are generated in total. This group of testing dataset is denoted as **same viewing** (SV) angles testing dataset. **Group 2**, each model is virtually scanned from 216 different viewing angles,

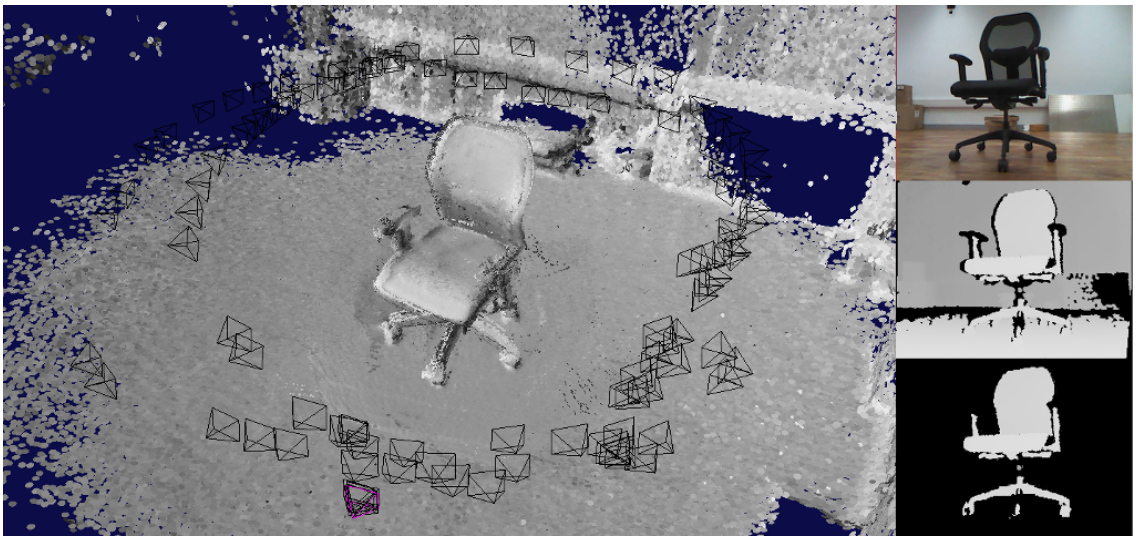


Figure 3.5: An example of ElasticFusion for generating real world data. Left: reconstructed object; sampled camera poses are shown in black. Right: Input RGB, depth image and segmented depth image.

6 uniformly sampled views from each of roll, pitch and yaw space ranging from $0 \sim 2\pi$ individually. Note that, these viewing angles for all testing 3D models are completely different from training pairs. Around 8k testing pairs are generated in total. This group of testing dataset is denoted as **cross viewing** (CV) angles testing dataset. Similarly, we also generate around 1.5k SV and 2.5k CV **validation data** split from another 12 CAD models, which are used for hyperparameter searching.

As our network is initially designed to predict an aligned full 3D model given a depth image from an arbitrary viewing angle, these two SV and CV testing datasets are generated separately to evaluate the viewing angle robustness and generality of our model.

Besides the large quantity of synthesized data, we also collect a **real-world dataset** in order to test the proposed network in a realistic scenario. We use a Microsoft Kinect camera to manually scan a total of 20 object instances belonging to 4 classes {bench, chair, couch, table}, with 5 instances per class from different environments, including offices, homes, and outdoor university parks. For each object we acquire RGB-D images of the object from multiple angles by moving the camera around the object. Then, we use the dense visual SLAM algorithm ElasticFusion [278] in order to reconstruct the full 3D shape of each object, as well as the camera pose in each scan.

We sample 50 random views from the camera trajectory, and for each one we obtain the depth image and the relative camera pose. In each depth image the 3D

object is segmented from the background, using a combination of floor removal and manual segmentation. We finally generate ground truth information by aligning the full 3D objects with the partial 2.5D views.

It should be noted that, due to noise and quantization artifacts of low-cost RGB-D sensors, and the inaccuracy of the SLAM algorithm, the full 3D ground truth is not 100% accurate, but can still be used as a reasonable approximation. The real-world dataset highlights the challenges related to shape reconstruction from realistic data: noisy depth estimates, missing depth information, depth quantization. In addition, some of the objects are acquired outdoors (*e.g.*, *bench*), which is challenging for the near-infrared depth sensor of the Micorsoft Kinect. However, we argue that a real-world benchmark for shape reconstruction is necessary for a thorough validation of future approaches. Figure 3.5 shows an example of the reconstructed object and camera poses in ElasticFusion.

3.4 Experiments

In this section, we evaluate our 3D-RecGAN++ with comparison to the state of the art approaches and an ablation study to fully investigate the proposed network.

3.4.1 Metrics

To evaluate the performance of 3D reconstruction, we consider two metrics. The first metric is the mean Intersection-over-Union (IoU) between predicted 3D voxel grids and their ground truth. The IoU for an individual voxel grid is formally defined as follows:

$$IoU = \frac{\sum_{i=1}^{N} \left[I(y_i > p) * I(\bar{y}_i) \right]}{\sum_{i=1}^{N} \left[I\left(I(y_i > p) + I(\bar{y}_i) \right) \right]}$$

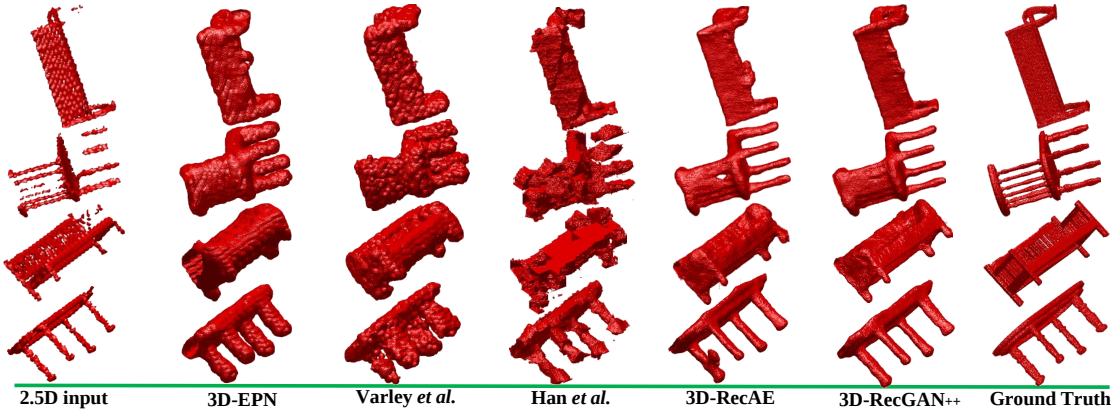
where $I(\cdot)$ is an indicator function, y_i is the predicted value for the i^{th} voxel, $\bar{y}_i$ is the corresponding ground truth, p is the threshold for voxelization, N is the total number of voxels in a whole voxel grid. In all our experiments, p is searched using the validation data split per category for each approach. Particularly, p is searched in the range $[0.1, 0.9]$ with a step size 0.05 using the validation datasets. The higher the IoU value, the better the reconstruction of a 3D model.

The second metric is the mean value of standard Cross-Entropy loss (CE) between a reconstructed shape and the ground truth 3D model. It is formally defined as:

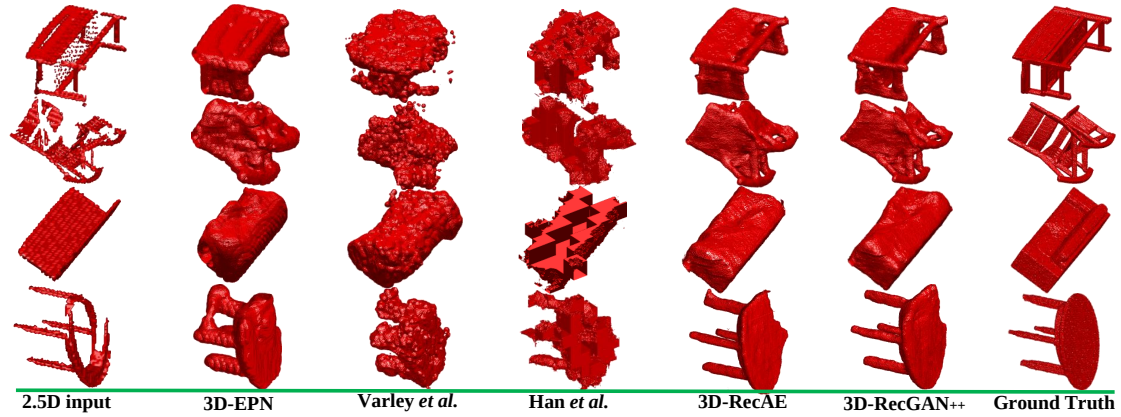
$$CE = -\frac{1}{N} \sum_{i=1}^{N} \left[\bar{y}_i \log(y_i) + (1 - \bar{y}_i) \log(1 - y_i) \right]$$

where y_i, $\bar{y}_i$ and N are the same as defined in above IoU. The lower CE value is, the closer the prediction to be either '1' or '0', the more robust and confident the 3D predictions are.

We also considered the Chamfer Distance (CD) or Earth Mover's Distance (EMD) as an additional metric. However, it is computationally heavy to calculate the distance between two high resolution voxel grids due to the large number of points. In our experiments, it takes nearly 2 minutes to calculate either CD or EMD between two 256^3 shapes on a single Titan X GPU. Although the 256^3 dense shapes can be downsampled to sparse point clouds on object surfaces to quickly compute CD or EMD, the geometric details are inevitably lost due to the extreme downsampling process. Therefore, we did not use CD or EMD for evaluation in our experiments.



(a) Qualitative results of single category reconstruction on testing datasets with same viewing angles.



(b) Qualitative results of single category reconstruction on testing datasets with cross viewing angles.

Figure 3.6: Qualitative results of single category reconstruction on testing datasets with same and cross viewing angles.

3.4.2 Competing Approaches

We compare against three state of the art deep learning based approaches for single depth view reconstruction. We also compare against the generator alone in our network, *i.e.*, without the GAN, named 3D-RecAE for short.

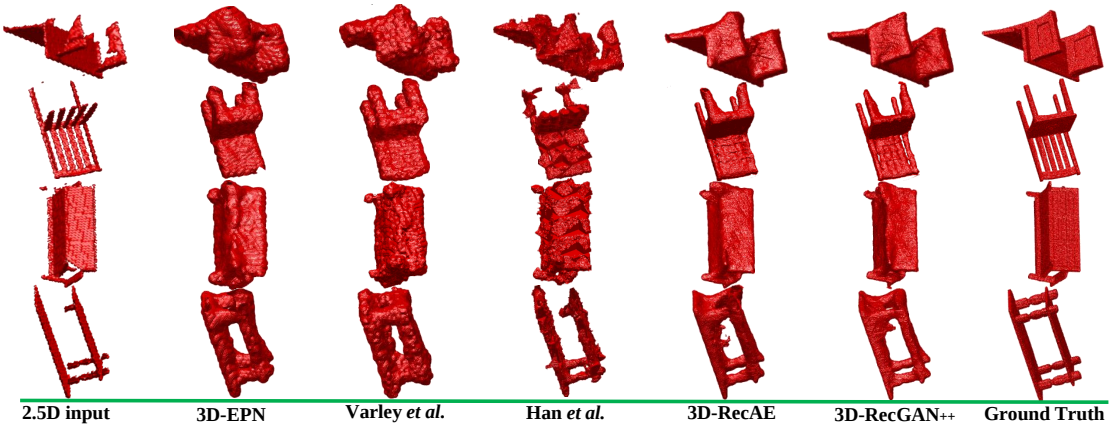
- **3D-EPN.** In [41], Dai *et al.* proposed a neural network, called "3D-EPN", to reconstruct the 3D shape up to a 32^3 voxel grid, after which a high resolution shape is retrieved from an existing 3D shape database, called "Shape Synthesis". In our experiment, we only compared with their neural network (*i.e.*, 3D-EPN) performance because we do not have an existing shape database for similar shape retrieval during testing. Besides, occupancy grid representation is used for the network training and testing.

- **Varley *et al.*** In [260], a network was designed to complete the 3D shape from a single 2.5D depth view for robot grasping. The output of their network is a 40^3 voxel grid.

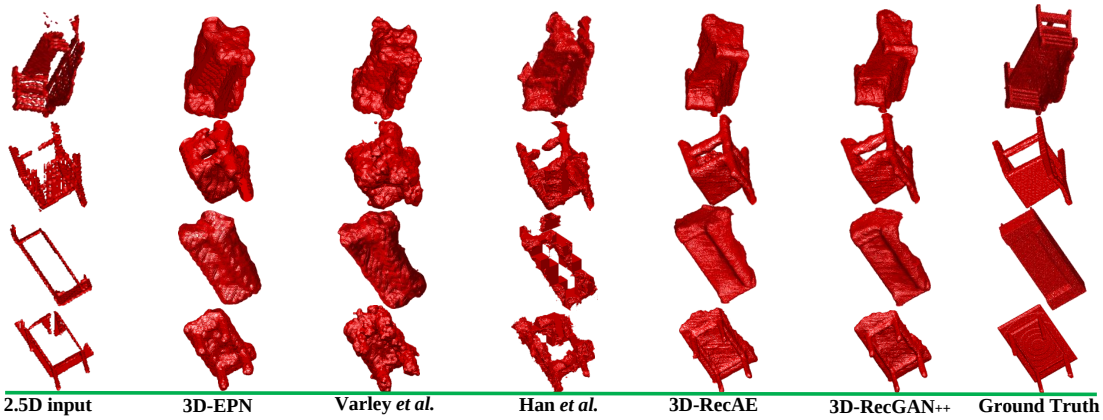
  Note that, the low resolution voxel grids generated by 3D-EPN and Varley *et al.* are all upsampled to 256^3 voxel grids using trilinear interpolation before calculating the IoU and CE metrics. The linear upsampling is a widely used post-processing technique for fair comparison in cases where the output resolution is not identical [250]. However, as both 3D-EPN and Varley *et al.* are trained using lower resolution voxel grids for supervision, while the below Han *et al.* and our 3D-RecGAN++ are trained using 256^3 shapes for supervision, it is not strictly fair comparison in this regard. Considering both 3D-EPN and Varley *et al.* are among the early works and also solid competing approaches regarding the single depth view reconstruction task, we therefore include them as baselines.

- **Han *et al.*** In [74], a global structure inference network and a local geometry refinement network are proposed to complete a high resolution shape from a noisy shape. The network is not originally designed for single depth view reconstruction, but its output shape is up to a 256^3 voxel grid and is comparable to our network. For fair comparison, the same occupancy grid representation is used for their network. It should be noted that [74] involves convoluted designs, thus the training procedure is slower and less efficient due to the many integrated LSTMs.

- **3D-RecAE.** As for our 3D-RecGAN++, we remove the discriminator and only keep the generator to infer the complete 3D shape from a single depth view. This comparison illustrates the benefits of adversarial learning.



(a) Qualitative results of multiple category reconstruction on testing datasets with the same viewing angles.



(b) Qualitative results of multiple category reconstruction on testing datasets with cross viewing angles.

Figure 3.7: Qualitative results of multiple category reconstruction on testing datasets with same and cross viewing angles.

3.4.3 Single-category Results

(1) **Results.** All networks are separately trained and tested on four different categories, {bench, chair, couch, table}, with the same network configuration. Table 3.1 shows the IoU and CE loss of all methods on the testing dataset with same viewing angles on 256^3 voxel grids, while Table 3.2 shows the IoU and CE loss comparison on testing dataset with cross viewing angles. Figure 3.6 shows the qualitative results of

single category reconstruction on testing datasets with same and cross viewing angles. The meshgrid function in Matlab is used to plot all 3D shapes for better visualization.

Table 3.1: Per-category IoU and CE loss on testing dataset with same viewing angles (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
3D-EPN [41]	0.423	0.488	0.631	0.508	0.087	0.105	0.144	0.101
Varley *et al.* [260]	0.227	0.317	0.544	0.233	0.111	0.157	0.195	0.191
Han *et al.* [74]	0.441	0.426	0.446	0.499	0.045	0.081	0.165	0.058
3D-RecAE (ours)	0.577	0.641	0.749	0.675	0.036	0.063	0.067	0.043
3D-RecGAN++ (ours)	**0.580**	**0.647**	**0.753**	**0.679**	**0.034**	**0.060**	**0.066**	**0.040**

Table 3.2: Per-category IoU and CE loss on testing dataset with cross viewing angles (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
3D-EPN [41]	0.408	0.446	0.572	0.482	0.086	0.112	0.163	0.103
Varley *et al.* [260]	0.185	0.278	0.475	0.187	0.108	0.171	0.210	0.186
Han *et al.* [74]	0.439	0.426	0.455	0.482	0.047	0.090	0.163	0.060
3D-RecAE (ours)	0.524	0.588	0.639	0.610	0.045	0.079	0.117	0.058
3D-RecGAN++ (ours)	**0.531**	**0.594**	**0.646**	**0.618**	**0.041**	**0.074**	**0.111**	**0.053**

(2) **Analysis.** Both 3D-RecGAN++ and 3D-RecAE significantly outperform the competing approaches in terms of IoU and CE loss on both the SV and CV testing datasets for dense 3D shape reconstruction (256^3 voxel grids). Although our approach is trained on depth input with a limited set of viewing angles, it still performs well to predict aligned 3D shapes from novel viewing angles. The 3D shapes generated by 3D-RecGAN++ and 3D-RecAE are much more visually compelling than others.

Compared with 3D-RecAE, 3D-RecGAN++ achieves better IoU scores and smaller CE loss. Basically, adversarial learning of the discriminator serves as a regularizer for fine-grained 3D shape estimation, which enables the output of 3D-RecGAN++ to be more robust and confident. We also notice that the increase of 3D-RecGAN++ in IoU and CE scores is not dramatic compared with 3D-RecAE. This is primarily because the main object shape can be reasonably predicted by 3D-RecAE, while the finer geometric details estimated by 3D-RecGAN++ are usually smaller parts of the whole object shape. Therefore, 3D-RecGAN++ only obtains a reasonable better IoU and CE scores than 3D-RecAE. The 4^{th} row of Figure 3.6a shows a good example

in terms of finer geometric details prediction of 3D-RecGAN++. In fact, in all the remaining experiments, 3D-RecGAN++ is constantly, but not significantly, better than 3D-RecAE.

Table 3.3: Multi-category IoU and CE loss on testing dataset with same viewing angles (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
3D-EPN [41]	0.428	0.484	0.634	0.506	0.087	0.107	0.138	0.102
Varley *et al.* [260]	0.234	0.317	0.543	0.236	0.103	0.132	0.197	0.170
Han *et al.* [74]	0.425	0.454	0.440	0.470	0.045	0.087	0.172	0.065
3D-RecAE (ours)	0.576	0.632	0.740	0.661	0.037	0.060	0.069	0.044
3D-RecGAN++ (ours)	**0.581**	**0.640**	**0.745**	**0.667**	**0.030**	**0.051**	**0.063**	**0.039**

Table 3.4: Multi-category IoU and CE loss on testing dataset with cross viewing angles (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
3D-EPN [41]	0.415	0.452	0.531	0.477	0.091	0.115	0.147	0.111
Varley *et al.* [260]	0.201	0.283	0.480	0.199	0.105	0.143	0.207	0.174
Han *et al.* [74]	0.429	0.444	0.447	0.474	0.045	0.089	0.172	0.063
3D-RecAE (ours)	0.530	0.587	0.640	0.610	0.043	0.068	0.096	0.055
3D-RecGAN++ (ours)	**0.540**	**0.594**	**0.643**	**0.621**	**0.038**	**0.061**	**0.091**	**0.048**

3.4.4 Multi-category Results

(1) **Results.** All networks are also trained and tested on multiple categories without being given any class labels. The networks are trained on four categories: {bench, chair, couch, table}; and then tested separately on individual categories. Table 3.3 shows the IoU and CE loss comparison of all methods on testing dataset with same viewing angles for dense shape reconstruction, while Table 3.4 shows the IoU and CE loss comparison on testing dataset with cross viewing angles. Figure 3.7 shows the qualitative results of all approaches on testing datasets of multiple categories with same and cross viewing angles.

(2) **Analysis.** Both 3D-RecGAN++ and 3D-RecAE significantly outperforms the state of the art by a large margin in all categories which are trained together on a single model. Besides, the performance of our network trained on multiple categories, does not notably degrade compared with training the network on individual categories

as shown in previous Table 3.1 and 3.2. This confirms that our network has enough capacity and capability to learn diverse features from multiple categories.

Table 3.5: Cross-category IoU on testing dataset with the same viewing angles (256^3 voxel grids).

	car	faucet	firearm	guitar	monitor	plane
3D-EPN [41]	0.450	0.442	0.339	0.351	0.444	0.314
Varley *et al.* [260]	0.484	0.260	0.280	0.255	0.341	0.295
Han *et al.* [74]	0.360	0.402	0.333	0.353	0.450	0.306
3D-RecAE (ours)	**0.557**	0.530	0.422	0.440	0.556	0.390
3D-RecGAN++ (ours)	0.555	**0.536**	**0.426**	**0.442**	**0.562**	**0.394**

Table 3.6: Cross-category CE loss on testing dataset with the same viewing angles (256^3 voxel grids).

	car	faucet	firearm	guitar	monitor	plane
3D-EPN [41]	0.170	0.088	0.036	0.036	0.123	0.066
Varley *et al.* [260]	0.173	0.122	0.029	0.030	0.130	0.042
Han *et al.* [74]	0.167	0.077	0.018	0.015	0.088	0.031
3D-RecAE (ours)	0.110	0.057	0.018	0.016	0.072	0.036
3D-RecGAN++ (ours)	**0.102**	**0.053**	**0.016**	**0.014**	**0.067**	**0.031**

3.4.5 Cross-category Results

(1) **Results.** To further investigate the generality of networks, we train all networks on {bench, chair, couch, table}, and then test them on another 6 totally different categories: {car, faucet, firearm, guitar, monitor, plane}. For each of the 6 categories, we generate the same amount of testing datasets with same and cross viewing angles, which is similar to the previous {bench, chair, couch, table}. Table 3.5 and 3.6 shows the IoU and CE loss comparison of all approaches on the testing dataset with same viewing angles, while Table 3.7 and 3.8 shows the IoU and CE loss comparison on the testing dataset with cross viewing angles. Figure 3.8 shows the qualitative results of all methods on 6 unseen categories with same and cross viewing angles.

We further evaluate the generality of our 3D-RecGAN++ on a specific category. Particularly, we conduct four groups of experiments. In the first group, we train our 3D-RecGAN++ on bench, then separately test it on the remaining 3 categories: {chair, couch, table}. In the second group, the network is trained on chair and separately tested on {bench, couch, table}. Similarly, another two groups of experiments are conducted. Basically, this experiment is to investigate how well our approach

Table 3.7: Cross-category IoU on testing dataset with cross viewing angles (256^3 voxel grids).

	car	faucet	firearm	guitar	monitor	plane
3D-EPN [41]	0.446	0.439	0.324	0.359	0.448	0.309
Varley *et al.* [260]	0.489	0.260	0.274	0.255	0.334	0.283
Han *et al.* [74]	0.349	0.402	0.321	0.363	0.455	0.299
3D-RecAE (ours)	0.550	0.521	0.411	0.441	0.550	0.382
3D-RecGAN++ (ours)	**0.553**	**0.529**	**0.416**	**0.449**	**0.555**	**0.390**

Table 3.8: Cross-category CE loss on testing dataset with cross viewing angles (256^3 voxel grids).

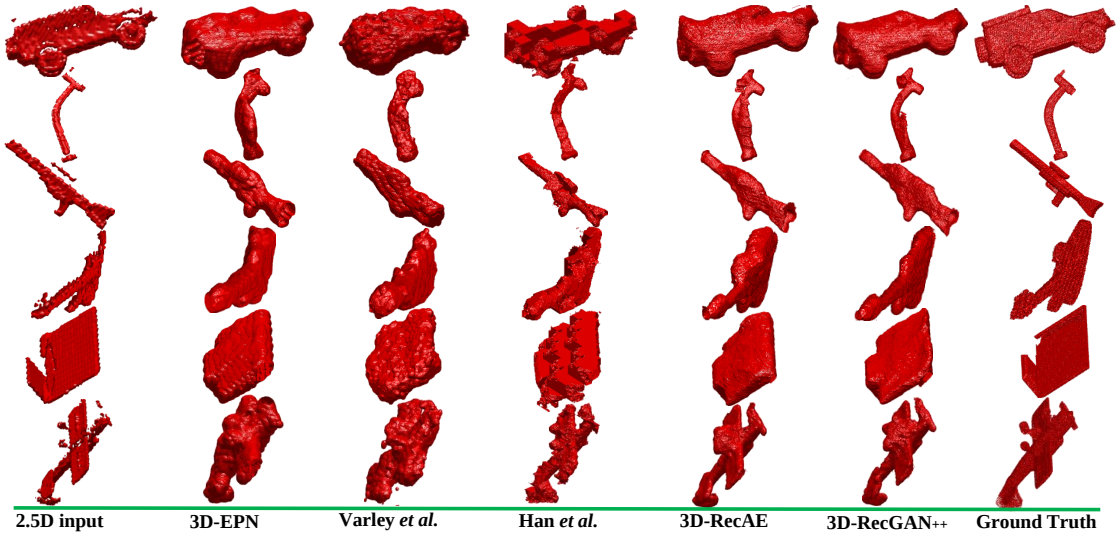
	car	faucet	firearm	guitar	monitor	plane
3D-EPN [41]	0.160	0.087	0.033	0.036	0.127	0.065
Varley *et al.* [260]	0.171	0.123	0.028	0.030	0.136	0.043
Han *et al.* [74]	0.171	0.076	0.018	0.016	0.088	0.031
3D-RecAE (ours)	0.101	0.059	0.017	0.017	0.079	0.036
3D-RecGAN++ (ours)	**0.100**	**0.055**	**0.014**	**0.015**	**0.074**	**0.031**

learns features from one category and then generalizes to a different category, and vice versa. Table 3.9 shows the cross-category IoU and CE loss of our 3D-RecGAN++ trained on individual category and then tested on the testing dataset with same viewing angles over 256^3 voxel grids.

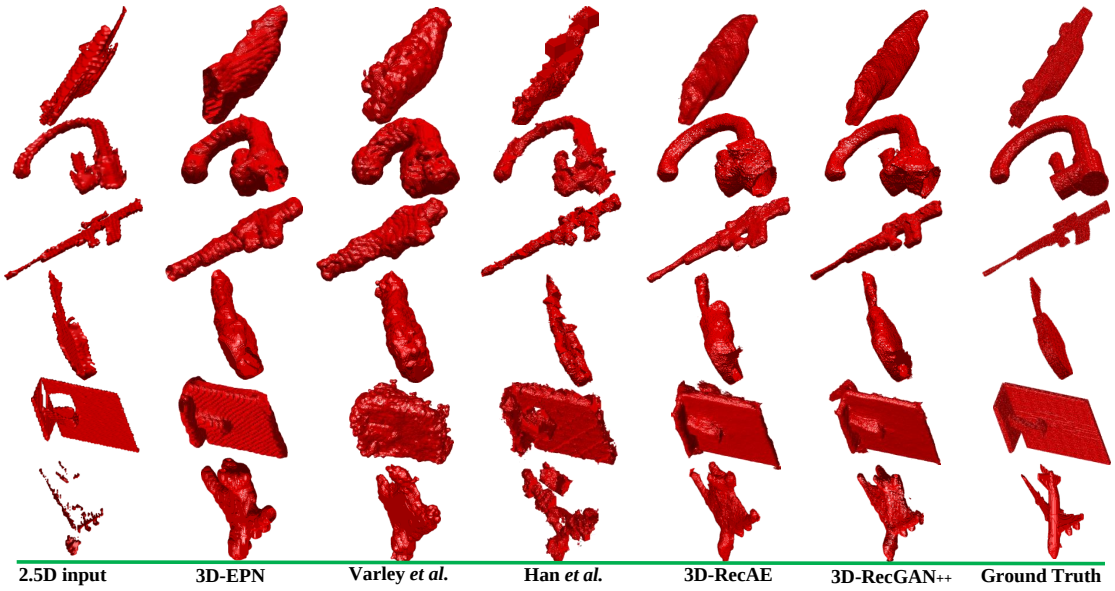
(2) **Analysis.** The proposed 3D-RecGAN++ achieves much higher IoU and smaller CE loss across the unseen categories than competing approaches. Our network not only learns rich features from different object categories, but also is able to generalize well to completely new types of categories. Our intuition is that the network may learn geometric features such as lines, planes, curves which are common across various object categories. As our network involves skip-connections between intermediate neural layers, it is not straightforward to visualize and analyze the learnt latent features.

It can be also observed that our model trained on *bench* tends to be more general than others. Intuitively, the bench category tends to have general features such as four legs, seats, and/or a back, which are also common among other categories {chair, couch, table}. However, not all chairs or couches consist of such general features that are shared across different categories.

Overall, we may safely conclude that the more similar features two categories share, including both the low-level lines/planes/curves and the high-level shape components, the better generalization of our model achieves cross those categories.



(a) Qualitative results of cross category reconstruction on testing datasets with same viewing angles.



(b) Qualitative results of cross category reconstruction on testing datasets with cross viewing angles.

Figure 3.8: Qualitative results of cross category reconstruction on testing datasets with same and cross viewing angles.

Table 3.9: Cross-category IoU and CE loss of 3D-RecGAN++ trained on individual category and then tested on the testing dataset with the same viewing angles (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
Group 1 (trained on bench)	**0.580**	0.510	0.507	0.599	**0.034**	0.110	0.164	0.062
Group 2 (trained on chair)	0.508	**0.647**	0.469	0.564	0.048	**0.060**	0.184	0.069
Group 3 (trained on couch)	0.429	0.504	**0.753**	0.437	0.070	0.105	**0.066**	0.126
Group 4 (trained on table)	0.510	0.509	0.402	**0.679**	0.049	0.111	0.260	**0.040**

Table 3.10: Multi-category IoU and CE loss on real-world dataset (256^3 voxel grids).

	IoU				CE Loss			
	bench	chair	couch	table	bench	chair	couch	table
3D-EPN [41]	0.162	0.190	0.508	0.140	0.090	0.158	0.413	0.187
Varley *et al.* [260]	0.118	0.152	0.433	0.075	0.073	0.155	0.436	0.191
Han *et al.* [74]	0.166	0.164	0.235	0.146	0.083	0.167	0.352	0.194
3D-RecAE (ours)	0.173	0.203	0.538	0.151	0.065	0.156	0.318	0.180
3D-RecGAN++ (ours)	**0.177**	**0.208**	**0.540**	**0.156**	**0.061**	**0.153**	**0.314**	**0.177**

3.4.6 Real-world Experiment Results

(1) **Results.** Lastly, in order to evaluate the domain adaptation capability of the networks, we train all networks on synthesized data of categories {bench, chair, couch, table}, and then test them on real-world data collected by a Microsoft Kinect camera. Table 3.10 compares the IoU and CE loss of all approaches on the real-world dataset. Figure 3.9 shows some qualitative results for all methods.

(2) **Analysis.** There are two reasons why the IoU is significantly lower compared with testing on the synthetic dataset. First, the ground truth objects obtained from ElasticFusion are not as solid as the synthesized datasets. However, all networks predict dense and solid voxel grids, so the interior parts may not match though the overall object shapes are satisfactorily recovered as shown in Figure 3.9. Secondly, the input 2.5D depth view from real-world dataset is noisy and incomplete, due to the limitation of the RGB-D sensor (*e.g.*, reflective surfaces, outdoor lighting). In some cases, the input 2.5D view does not capture the whole object and only contains a part of the object, which also leads to inferior reconstruction results (*e.g.*, the 5^{th}

row in Figure 3.9) and a lower IoU scores overall. However, our proposed network is still able to reconstruct reasonable 3D dense shapes given the noisy and incomplete 2.5D input depth views, while the competing algorithms (*e.g.*, Varley *et al.*) are not robust to real-world noise and unable to generate compelling results.

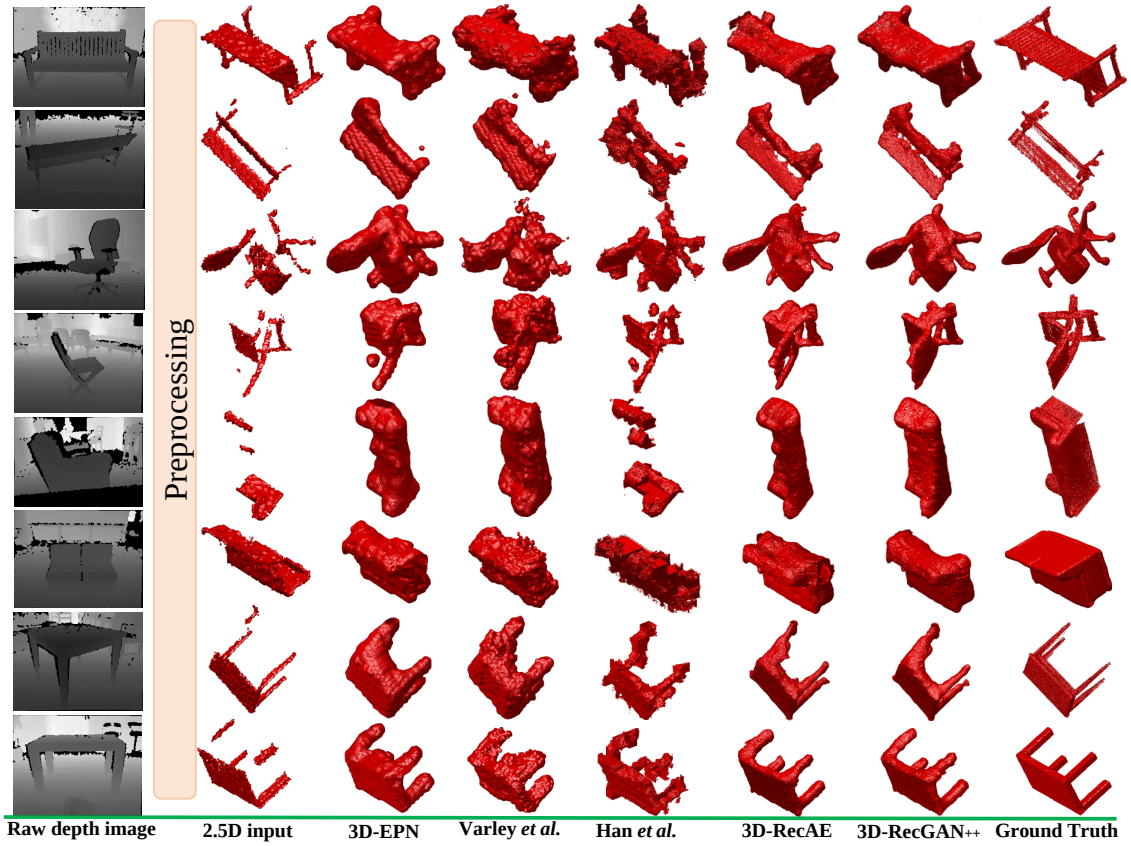


Figure 3.9: Qualitative results of real-world objects reconstruction from different approaches. The object instance is segmented from the raw depth image in preprocessing step.

3.4.7 Impact of Adversarial Learning

(1) **Results.** In all above experiments, the proposed 3D-RecGAN++ tends to outperform the ablated network 3D-RecAE which does not include the adversarial learning of GAN part. In all visualization of experiment results, the 3D shapes from 3D-RecGAN++ are also more compelling than 3D-RecAE. To further quantitatively investigate how the adversarial learning improves the final 3D results comparing with 3D-RecAE, we calculate the mean precision and recall from the previous multi-category experiment results in Section 3.4.4. Table 3.11 compares the mean precision

and recall of 3D-RecGAN++ and 3D-RecAE on individual categories using the network trained on multiple categories.

(2) **Analysis.** It can be seen that the results of 3D-RecGAN++ tend to consistently have higher precision scores than 3D-RecAE, which means 3D-RecGAN++ has less false positive estimations. Therefore, the estimated 3D shapes from 3D-RecAE are likely to be 'fatter' and 'bigger', while 3D-RecGAN++ tends to predict 'thinner' shapes with much more shape details being exposed. Both 3D-RecGAN++ and 3D-RecAE can achieve high recall scores (*i.e.*, above 0.8), which means both 3D-RecGAN++ and 3D-RecAE are capable of estimating the major object shapes without too many false negatives. In other words, the ground truth 3D shape tends to be a subset of the estimated shape result.

Overall, with regard to experiments on per-category, multi-category, and cross-category experiments, our 3D-RecGAN++ outperforms others by a large margin, although all other approaches can reconstruct reasonable shapes. In terms of the generality, Varley *et al.* [260] and Han *et al.* [74] are inferior because Varley *et al.* [260] use a single fully connected layers, instead of 3D ConvNets, for shape generation which is unlikely to be general for various shapes, and Han *et al.* [74] apply LSTMs for shape blocks generation which is inefficient and unable to learn general 3D structures. However, our 3D-RecGAN++ is superior thanks to the generality of the 3D encoder-decoder and the adversarial discriminator. Besides, the 3D-RecAE tends to over estimate the 3D shape, while the adversarial learning of 3D-RecGAN++ is likely to remove the over-estimated parts, so as to leave the estimated shape to be clearer with more shape details.

Table 3.11: Multi-category mean precision and recall on testing dataset with the same viewing angles (256^3 voxel grids).

	mean precision				mean recall			
	bench	chair	couch	table	bench	chair	couch	table
3D-RecAE	0.668	0.740	0.800	0.750	**0.808**	0.818	0.907	0.845
3D-RecGAN++	**0.680**	**0.747**	**0.804**	**0.754**	0.804	**0.820**	**0.910**	**0.853**

3.4.8 Computation Analysis

Table 3.12 compares the computation efficiency of all approaches regarding the total number of model parameters and the average time consumption to recover a single object.

The model proposed by Han *et al.* [74] has the least number of parameters because most of the parameters are shared to predict different blocks of an object. Our 3D-RecGAN++ has reasonable 167.1 millions parameters, which is on a similar scale to VGG-19 (*i.e.*, 144 millions) [235].

To evaluate the average time consumption for a single object reconstruction, we implement all networks in Tensorflow 1.2 and Python 2.7 with CUDA 8.0 and cuDNN 7.1 as the back-end driver and library. All models are tested on a single Titan X GPU in the same hardware and software environments. 3D-EPN [41] takes the shortest time to predict a 32^3 object on GPU, while our 3D-RecGAN++ only needs around 40 milliseconds to recover a dense 256^3 object. Comparatively, Han *et al.* takes the longest GPU time to generate a dense object because of the time-consuming sequential processing of LSTMs. The low resolution objects predicted by 3D-EPN and Varley *et al.* are further upsampled to 256^3 using existing SciPy library on a CPU server (Intel E5-2620 v4, 32 cores). It takes around 7 seconds to finish the upsampling for a single object.

Table 3.12: Comparison of model parameters and average time consumption to reconstruction a single object.

	parameters (millions)	**GPU time** (milliseconds)	**predicted 3D shapes** (resolution)
3D-EPN [41]	52.4	**15.8**	32^3
Varley *et al.* [260]	430.3	16.1	40^3
Han *et al.* [74]	**7.5**	276.4	$\mathbf{256^3}$
3D-RecGAN++	167.1	38.9	$\mathbf{256^3}$

3.5 Conclusion

In this chapter, we proposed a framework 3D-RecGAN++ that reconstructs the full 3D structure of an object from an arbitrary depth view. By leveraging the generalization capabilities of 3D encoder-decoder and generative adversarial networks, our 3D-RecGAN++ predicts dense and accurate 3D structures with fine details, outperforming the state of the art in single-view shape completion for individual object category. We further tested our network's ability to reconstruct multiple categories without providing any object class labels during training or testing, and it showed that our network is still able to predict precise 3D shapes. Furthermore, we investigated the network's reconstruction performance on unseen categories, showing that

our proposed approach can also predict satisfactory 3D structures. Finally, our model is robust to real-world noisy data and can infer accurate 3D shapes although the model is purely trained on synthesized data. This confirms that our network has the capability of learning general 3D latent features of the objects, rather than simply fitting a function for the training datasets, and the adversarial learning of 3D-RecGAN++ learns to add geometric details for estimated 3D shapes. In summary, our network only requires a single depth view to recover a dense and complete 3D shape with fine details.

Although our 3D-RecGAN++ achieves the state of the art performance in 3D object reconstruction from a single depth view, it has limitations. Firstly, our network takes the volumetric representation of a single depth view as input, instead of taking a raw depth image. Therefore, a preprocessing of raw depth images is required for our network. However, in many application scenarios such as robot grasping, such preprocessing would be trivial and straightforward given the depth camera parameters. Secondly, the input depth view of our network only contains a clean object information without cluttered background. One possible solution is to leverage an existing segmentation algorithm such as Mask-RCNN [77] to clearly segment the target object instance from the raw depth view.

Chapter 4

Learning to Reconstruct 3D Objects from Multiple Views

4.1 Introduction

The problem of recovering a geometric representation of the 3D world given a set of images is classically defined as multi-view 3D reconstruction in computer vision. Traditional pipelines such as Structure from Motion (SfM) [194] and visual Simultaneous Localization and Mapping (vSLAM) [18] typically rely on hand-crafted feature extraction and matching across multiple views to reconstruct the underlying 3D model. However, if the multiple viewpoints are separated by large baselines, it can be extremely challenging for the feature matching approach due to significant changes of appearance or self occlusions [162]. Furthermore, the reconstructed 3D shape is usually a sparse point cloud without geometric details.

Recently, a number of deep learning approaches, such as 3D-R2N2 [32], LSM [102], DeepMVS [91] and RayNet [199] have been proposed to estimate the 3D dense shape from multiple images and have shown encouraging results. Both 3D-R2N2 [32] and LSM [102] formulate multi-view reconstruction as a sequence learning problem, and leverage recurrent neural networks (RNNs), particularly GRUs, to fuse the multiple deep features extracted by a shared encoder from input images. However, there are three limitations. First, the recurrent network is permutation variant, *i.e.*, different permutations or orderings of the input image sequence give different reconstruction results [262]. Therefore, inconsistent 3D shapes are estimated from the same image set with different permutations. Second, it is difficult to capture long-term dependencies in the sequence because of the gradient vanishing or exploding [12, 114], so the estimated 3D shapes are unlikely to be refined even if more images are given during training and testing. Third, the RNN unit is inefficient as each element of the

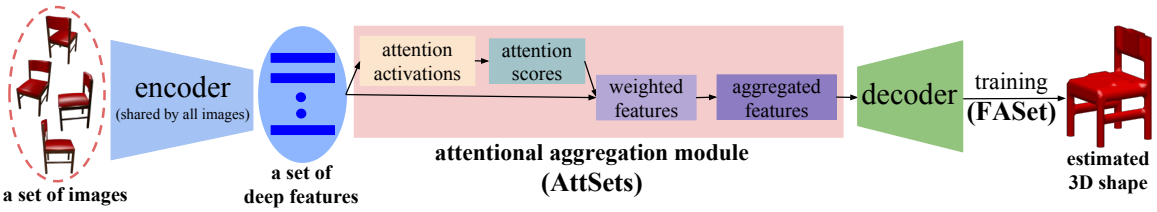


Figure 4.1: Overview of our attentional aggregation module for multi-view 3D reconstruction. A set of N images is passed through a common encoder to derive a set of deep features, one element for each image. The network is trained with our FASet algorithm.

input sequence must be sequentially processed without parallelization [167], so it is time-consuming to generate the final 3D shape given a sequence of images.

The recent DeepMVS [91] applies max pooling to aggregate deep features across a set of unordered images for multi-view stereo reconstruction, while RayNet [199] adopts average pooling to aggregate the deep features corresponding to the same voxel from multiple images to recover a dense 3D model. The very recent GQN [53] uses sum pooling to aggregate an arbitrary number of orderless images for 3D scene representation. Although max, average and summation poolings do not suffer from the above limitations of the RNN, they tend to be 'hard attentive', since they only capture the max/mean values or the summation without learning to attentively preserve the useful information. In addition, the above pooling based neural nets are usually optimized with a specific number of input images during training, therefore they are not robust or general to a dynamic number of input images during testing. This critical issue is also observed in GQN [53].

In this paper, we introduce a simple yet efficient attentional aggregation module, named **AttSets**[1]. It can be easily included in an existing multi-view 3D reconstruction network to aggregate an arbitrary number of elements of a deep feature set. Inspired by the attention mechanism, which shows great success in natural language processing [8, 210], image captioning [292], *etc.*, we design a feed-forward neural module that can automatically learn to aggregate each element of the input deep feature set. In particular, as shown in Figure 4.1, given a variable sized deep feature set, which encodes view-invariant visual representations from a shared encoder [199], our AttSets module firstly learns an **attention activation** for each latent feature through a standard neural layer (*e.g.*, a fully connected layer, a 2D or 3D convolutional layer), after which an **attention score** is computed for the corresponding feature. Subsequently, the attention scores are simply multiplied by the original elements of the deep

[1]Code is available at *https://github.com/Yang7879/AttSets*

feature set, generating a set of **weighted features**. Lastly, the weighted features are summed across different elements of the deep feature set, producing a fixed size vector of **aggregated features** which are then fed into a decoder to estimate 3D shapes. Basically, this AttSets module can be seen as a natural extension of sum pooling into a "weighted" sum pooling with learnt feature-specific weights. AttSets shares similar concepts with the concurrent work [93], but it does not require the additional gating mechanism in [93]. Notably, our simple feed-forward design allows the attention module to be separately trainable according to the property of its gradients.

In addition, we propose a new **Feature-Attention Separate training (FASet)** algorithm that elegantly decouples the base encoder-decoder (to learn deep features) from the AttSets module (to learn attention scores for features). This allows the AttSets module to learn desired attention scores for deep feature sets and forces the AttSets based neural networks to be robust and general to dynamic sized deep feature sets. Basically, in the proposed training algorithm, the base encoder-decoder neural layers are only optimized when the number of input images is **1**, while the AttSets module is only optimized where there are more than **1** input images. Eventually, the whole optimized AttSets based neural network achieves superior performance with a large number of input images, while simultaneously being extremely robust and able to generalize to a small number of input images, even to a single image in the most extreme case. Comparing with the widely used feed-forward attention mechanisms for visual recognition [99, 220, 159, 225, 63], our FASet algorithm is the first to investigate and improve the robustness of attention modules to dynamically sized input feature sets, whilst existing works are only applicable to fixed sized input data.

Overall, our novel AttSets module and FASet algorithm are distinguished from all existing aggregation approaches in three ways. 1) Compared with RNN approaches, AttSets is permutation invariant and computationally efficient. 2) Compared with the widely used pooling operations, AttSets learns to attentively select and weigh important deep features, thereby being more effective to aggregate useful information for better 3D reconstruction. 3) Compared with existing visual attention mechanisms, our FASet algorithm enables the whole network to be general to variable sized sets, being more robust and suitable for realistic multi-view 3D reconstruction scenarios where the number of input images usually varies dramatically.

Our key contributions are:

- We propose an efficient feed-forward attention module, AttSets, to effectively aggregate deep feature sets. Our design allows the attention module to be separately optimizable according to the property of the gradients of AttSets.

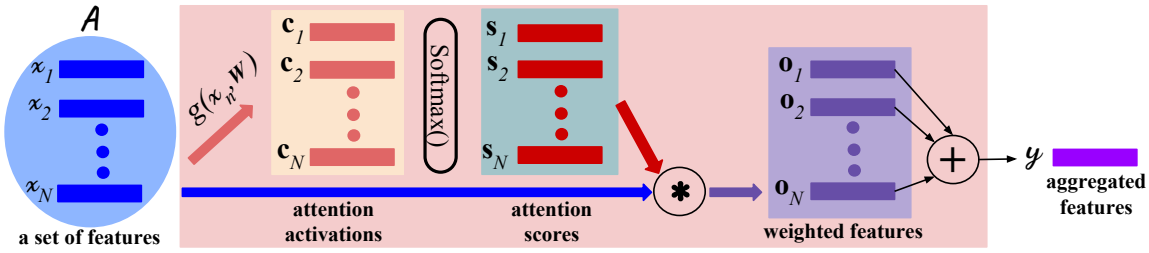


Figure 4.2: Attentional aggregation module on sets. This module learns an attention score for each individual deep feature.

- We propose a new two-stage training algorithm, FASet, to decouple the base encoder/decoder and the attention module, driving the whole network to be robust and general to an arbitrary number of input images.
- We conduct extensive experiments on multiple public datasets, demonstrating consistent improvement over existing aggregation approaches for 3D object reconstruction from either single or multiple views.

4.2 Attentional Aggregation Module

4.2.1 Problem Definition

We consider the problem of aggregating an arbitrary number of elements of a set $\mathcal{A}$ into a fixed single output $\boldsymbol{y}$. Each element of set $\mathcal{A}$ is a feature vector extracted from a shared encoder, and the fixed dimension output $\boldsymbol{y}$ is fed into a subsequent decoder, such that the whole network can process an arbitrary number of input elements.

Given N elements in the input deep feature set $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^{1\times D}$, where N is an arbitrary value, while D is fixed for a specific encoder, and the output $\boldsymbol{y} \in \mathbb{R}^{1\times D}$, which is then fed into the subsequent decoder, our task is to design an aggregation function f with learnable weights $\boldsymbol{W}$: $\boldsymbol{y} = f(\mathcal{A}, \boldsymbol{W})$, which should be permutation invariant, *i.e.*, for any permutation π:

$$f(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}, \boldsymbol{W}) = f(\{\boldsymbol{x}_{\pi(1)}, \cdots, \boldsymbol{x}_{\pi(N)}\}, \boldsymbol{W}) \tag{4.1}$$

The common pooling operations, *e.g.*, max/mean/sum, are the simplest instantiations of function f where $\boldsymbol{W} \in \emptyset$. However, these pooling operations are predefined and only capture partial information.

4.2.2 Attentional Aggregation

The key idea of our AttSets module is to learn an attention score for each latent feature of the whole deep feature set. In this chapter, 'each latent feature' refers to each entry of an individual element of the feature set, with an individual element usually represented by a latent vector, *i.e.*, $\boldsymbol{x}_n$. The learnt scores can be regarded as a mask that automatically selects useful latent features across the set. The selected features are then summed across multiple elements of the set.

As shown in Figure 4.2, given a set of features $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^{1\times D}$, AttSets aims to fuse it into a fixed dimensional output $\boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{1\times D}$.

To build the AttSets module, we first feed each element of the feature set $\mathcal{A}$ into a shared function g which can be a standard neural layer, *i.e.*, a linear transformation layer without any non-linear activation functions. Here we use a fully connected layer as an example, and the bias term is dropped for simplicity. The output of function g when applied to all inputs is a set of learnt attention activations $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_N\}$, where

$$\begin{gathered} \boldsymbol{c}_n = g(\boldsymbol{x}_n, \boldsymbol{W}) = \boldsymbol{x}_n \boldsymbol{W}, \\ (\boldsymbol{x}_n \in \mathbb{R}^{1\times D}, \quad \boldsymbol{W} \in \mathbb{R}^{D\times D}, \quad \boldsymbol{c}_n \in \mathbb{R}^{1\times D}) \end{gathered} \tag{4.2}$$

Secondly, the learnt attention activations are normalized across the N elements of the set, computing a set of attention scores $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_N\}$. We choose $softmax$ as the normalization operation, so the attention scores for the n^{th} feature element are

$$\begin{gathered} \boldsymbol{s}_n = [s_n^1, s_n^2, \cdots, s_n^d, \cdots, s_n^D], \\ s_n^d = \frac{e^{c_n^d}}{\sum_{j=1}^{N} e^{c_j^d}}, \qquad \textit{where } c_n^d,\ c_j^d \textit{ are the } d^{th} \textit{ entry of } \boldsymbol{c}_n,\ \boldsymbol{c}_j. \end{gathered} \tag{4.3}$$

Thirdly, the computed attention scores $\mathcal{S}$ are multiplied by their corresponding original feature set $\mathcal{A}$, generating a new set of deep features, denoted as weighted features $\mathcal{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_N\}$, where

$$\boldsymbol{o}_n = \boldsymbol{x}_n * \boldsymbol{s}_n \tag{4.4}$$

Lastly, the set of weighted features $\mathcal{O}$ are summed up across the total N elements to get a fixed size feature vector, denoted as $\boldsymbol{y}$, where

$$\begin{gathered} \boldsymbol{y} = [y^1, y^2, \cdots, y^d, \cdots, y^D], \\ y^d = \sum_{n=1}^{N} o_n^d, \qquad \textit{where } o_n^d \textit{ is the } d^{th} \textit{ entry of } \boldsymbol{o}_n. \end{gathered} \tag{4.5}$$

In the above formulation, we show how AttSets gradually aggregates a set of N feature vectors $\mathcal{A}$ into a single vector $\boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{1\times D}$.

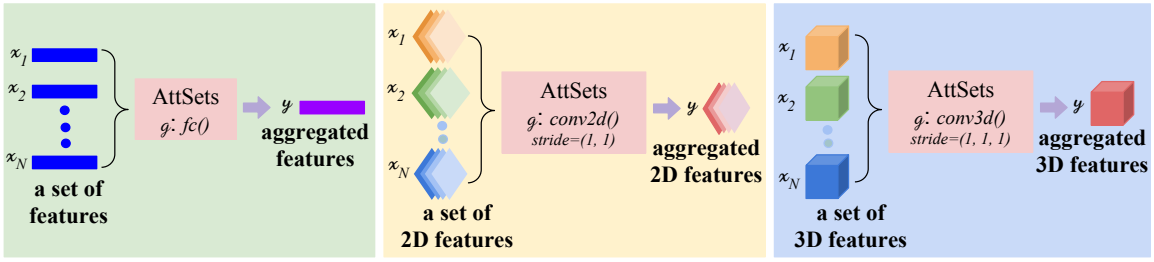


Figure 4.3: Different implementations of AttSets with a fully connected layer, 2D ConvNet, and 3D ConvNet. These three variants of AttSets can be flexibly plugged into different locations of an existing encoder-decoder network.

4.2.3 Permutation Invariance

The output of AttSets module $\boldsymbol{y}$ is permutation invariant with regard to the input deep feature set $\mathcal{A}$. Here is the simple proof.

$$[y^1, \cdots y^d \cdots, y^D] = f(\{\boldsymbol{x}_1, \cdots \boldsymbol{x}_n \cdots, \boldsymbol{x}_N\}, \boldsymbol{W}) \tag{4.6}$$

In Equation 4.6, the d^{th} entry of the output $\boldsymbol{y}$ is computed as follows:

$$\begin{aligned} y^d &= \sum_{n=1}^{N} o_n^d = \sum_{n=1}^{N} (x_n^d * s_n^d) \\ &= \sum_{n=1}^{N} \left(x_n^d * \frac{e^{c_n^d}}{\sum_{j=1}^{N} e^{c_j^d}} \right) \\ &= \sum_{n=1}^{N} \left(x_n^d * \frac{e^{(\boldsymbol{x}_n \boldsymbol{w}^d)}}{\sum_{j=1}^{N} e^{(\boldsymbol{x}_j \boldsymbol{w}^d)}} \right) \\ &= \frac{\sum_{n=1}^{N} \left(x_n^d * e^{(\boldsymbol{x}_n \boldsymbol{w}^d)} \right)}{\sum_{j=1}^{N} e^{(\boldsymbol{x}_j \boldsymbol{w}^d)}}, \end{aligned} \tag{4.7}$$

where $\boldsymbol{w}^d$ is the d^{th} column of the weights $\boldsymbol{W}$. In Equation 4.7, both the denominator and numerator are a summation of a permutation equivariant term. Therefore the value y^d, and also the full vector $\boldsymbol{y}$, is also invariant to different permutations of the deep feature set $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \cdots, \boldsymbol{x}_N\}$ [313].

4.2.4 Implementation

In Section 4.2.2, we described how our AttSets aggregates an arbitrary number of vector features into a single vector, where the attention activation learning function g embeds a fully connected (fc) layer. AttSets can also be easily implemented with

both 2D and 3D convolutional neural layers to aggregate both 2D and 3D deep feature sets, thus being flexible to be easily included into a 2D encoder/decoder or 3D encoder/decoder. Particularly, as shown in Figure 4.3, to aggregate a set of 2D features, *i.e.*, a tensor of $(width \times height \times channels)$, the attention activation learning function g embeds a standard $conv2d$ layer with a stride of (1×1). Similarly, to fuse a set of 3D features, *i.e.*, a tensor of $(width \times height \times depth \times channels)$, the function g embeds a standard $conv3d$ layer with a stride of $(1 \times 1 \times 1)$. For the above $conv2d/conv3d$ layer, the filter size can be 1, 3 or many. The larger the filter size, the larger the local spatial area the learnt attention score is considered to be correlated with.

Instead of embedding a single neural layer, the function g is also flexible to include multiple layers, with the restriction that the tensor shape of the output of function g is required to be consistent with the input element $\boldsymbol{x}_n$. This guarantees each individual feature of the input set $\mathcal{A}$ will be associated with a learnt and unique weight. For example, a standard 2-layer or 3-layer ResNet module [78] could be a candidate of the function g. The more the layers that g embeds, the greater the capability of the AttSets module is expected to be.

Compared with fc enabled AttSets, the $conv2d$ or $conv3d$ based AttSets variants tend to have fewer learnable parameters. Note that both the $conv2d$ and $conv3d$ based AttSets are still permutation invariant, as the function g is shared across all elements of the deep feature set and it does not depend on the order of the elements [313].

4.3 Feature Attention Separate Training

4.3.1 Motivation

Our AttSets module can be included in an existing encoder-decoder multi-view 3D reconstruction network, replacing the RNN units or pooling operations. Essentially, in an AttSets enabled encoder-decoder net, the encoder-decoder serves as the base architecture to learn visual features for shape estimation, while the AttSets module learns to assign different attention scores to combine those features. As such, the base network tends to have robustness and generality with regard to different input image content, while the AttSets module tends to be generally applicable to an arbitrary number of input images.

However, to achieve this robustness is not straightforward. The standard end-to-end joint optimization approach is unable to force that the base encoder-decoder

and AttSets are able to learn visual features and the corresponding scores separately, because there are no explicit feature score labels available to directly supervise the AttSets module.

Let us revisit Equation 4.7 below and draw insights from it.

$$y^d = \frac{\sum_{n=1}^{N} \left(x_n^d * e^{(\boldsymbol{x}_n \boldsymbol{w}^d)} \right)}{\sum_{j=1}^{N} e^{(\boldsymbol{x}_j \boldsymbol{w}^d)}} \tag{4.8}$$

where N is the size of an arbitrary input set and $\boldsymbol{w}^d$ are the AttSets parameters to be optimized. If N is 1, then the equation can be simplified as

$$y^d = x_n^d \tag{4.9}$$

$$\frac{\partial y^d}{\partial x_n^d} = 1, \qquad \frac{\partial y^d}{\partial \boldsymbol{w}^d} = \mathbf{0}, \qquad N = 1 \tag{4.10}$$

This shows that none of the parameters, *i.e.*, $\boldsymbol{w}^d$, of the AttSets module will be optimized when the size of the input feature set is 1.

However, if $N > 1$, Equation 4.8 is unable to be simplified to Equation 4.9. Therefore,

$$\frac{\partial y^d}{\partial x_n^d} \neq 1, \qquad \frac{\partial y^d}{\partial \boldsymbol{w}^d} \neq \mathbf{0}, \qquad N > 1 \tag{4.11}$$

This shows that both the parameters of AttSets and the base encoder-decoder layers will be optimized simultaneously, if the whole network is trained in the standard end-to-end fashion.

Here arises the critical issue. When $N > 1$, all derivatives of the parameters in the **encoder** are different from the derivatives when $N = 1$ due to the chain rule of differentiation applied backwards from $\frac{\partial y^d}{\partial x_n^d}$. Put simply, the derivatives of encoder are *N-dependent*. As a consequence, the encoded visual features and the learnt attention scores would be *N-biased* if the whole network is jointly trained. This biased network is unable to generalize to an arbitrary value of N during testing.

To illustrate the above issue, assuming the base encoder-decoder and the AttSets module are jointly trained given 5 images to reconstruct every object, the base encoder will be eventually optimized towards 5-view object reconstruction during training. The trained network can indeed perform well given 5 views during testing, but it is unable to predict a satisfactory object shape given only 1 image.

To alleviate the above problem, a naive approach is to enumerate various values of N during the joint training, such that the final optimized network can be somewhat

robust and general to arbitrary N during testing. However, this approach would inevitably optimize the encoder to learn the *mean* features of input data for varying N. The overall performance will hence not be optimal. In addition, it is impractical and also time-consuming to enumerate all values of N during training.

Algorithm 1 Feature-Attention Separate training of an AttSets enabled network. M is batch size, N is image number.

Stage 1:
for number of training iterations **do**

- Sample M sets of images $\{\mathcal{I}_1, \cdots, \mathcal{I}_m, \cdots, \mathcal{I}_M\}$ and sample N images for each set, *i.e.*, $\mathcal{I}_m = \{\boldsymbol{i}_m^1, \cdots, \boldsymbol{i}_m^n, \cdots, \boldsymbol{i}_m^N\}$. Sample M 3D shape labels $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m, \cdots, \boldsymbol{v}_M\}$.
- Update the base network by descending its stochastic gradient:

$$\nabla_{\Theta_{base}} \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[\ell(\hat{\boldsymbol{v}}_m^n, \boldsymbol{v}_m)\right], \ \textit{where } \hat{\boldsymbol{v}}_m^n \text{ is the estimated 3D shape of image } \boldsymbol{i}_m^n.$$

Stage 2:
for number of training iterations **do**

- Sample M sets of images $\{\mathcal{I}_1, \cdots, \mathcal{I}_m, \cdots, \mathcal{I}_M\}$ and sample N images for each set, *i.e.*, $\mathcal{I}_m = \{\boldsymbol{i}_m^1, \cdots, \boldsymbol{i}_m^n, \cdots, \boldsymbol{i}_m^N\}$. Sample M 3D shape labels $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m, \cdots, \boldsymbol{v}_M\}$.
- Update the AttSets module by descending its stochastic gradient:

$$\nabla_{\Theta_{att}} \frac{1}{M} \sum_{m=1}^{M} \left[\ell(\hat{\boldsymbol{v}}_m, \boldsymbol{v}_m)\right], \textit{where } \hat{\boldsymbol{v}}_m \text{ is the estimated 3D shape of the image set } \mathcal{I}_m.$$

The gradient-based updates can use any gradient optimization algorithm.

4.3.2 Algorithm

To resolve the critical issue discussed in Section 4.3.1, we propose a **Feature-Attention Separate training (FASet)** algorithm that decouples the base encoder-decoder and the AttSets module, enabling the base encoder-decoder to learn robust deep features and the AttSets module to learn the desired attention scores for the feature sets.

In particular, the base encoder-decoder neural layers are only optimized when a single input image is supplied, while the AttSets module is only optimized where there are more than one input images. In this regard, the parameters of the base encoding

layers have consistent derivatives over the whole training stage, thus being steadily optimized. In the meantime, the AttSets module would be optimized solely based on multiple elements of learnt visual features from the shared encoder.

The trainable parameters of the base encoder-decoder are denoted as $\mathbf{\Theta}_{base}$, and the trainable parameters of AttSets module are denoted as $\mathbf{\Theta}_{att}$, and the loss function of the whole network is represented by ℓ which is determined by the specific supervision signal of the base network. Our FASet is shown in Algorithm 1. It can be seen that $\mathbf{\Theta}_{base}$ and $\mathbf{\Theta}_{att}$ are completely decoupled from one another, thus being separately optimized in two stages. In stage 1, the $\mathbf{\Theta}_{base}$ is firstly well optimized until convergence, which guarantees the base encoder-decoder is able to learn robust and general visual features. In stage 2, the $\mathbf{\Theta}_{att}$ is optimized to learn attention scores for individual visual features. Basically, this module learns to select and weigh important deep features automatically.

In FASet algorithm, once the $\mathbf{\Theta}_{base}$ is well optimized in stage 1, it is not necessary to train it again, since the two-stage algorithm guarantees that optimizing $\mathbf{\Theta}_{base}$ is agnostic to the attention module. The FASet algorithm is a crucial component to maintain the superior robustness of the AttSets module, as is shown in Section 4.4.13. Without it, the feed-forward attention mechanism is ineffective with respect to dynamically sized input sets.

4.4 Experiments

4.4.1 Base Networks

To evaluate the performance and various properties of AttSets, we choose the encoder-decoders of 3D-R2N2 [32] and SilNet [279] as two base networks.

- Encoder-decoder of 3D-R2N2. The original 3D-R2N2 consists of (1) a shared ResNet-based 2D encoder which encodes a size of $127 \times 127 \times 3$ images into 1024 dimensional latent vectors, (2) a GRU module which fuses N 1024 dimensional latent vectors into a single $4 \times 4 \times 4 \times 128$ tensor, and (3) a ResNet-based 3D decoder which decodes the single tensor into a $32 \times 32 \times 32$ voxel grid representing the 3D shape. Figure 4.4 shows the architecture of AttSets based multi-view 3D reconstruction network where the only difference is that the original GRU module is replaced by AttSets in the middle. This network is called $\text{Base}_{\text{r2n2}}$-AttSets.

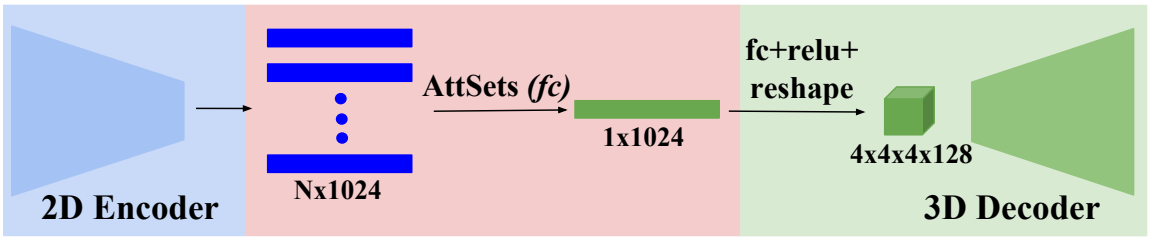


Figure 4.4: The architecture of $\text{Base}_{\text{r2n2}}$-AttSets for multi-view 3D reconstruction network. The base encoder-decoder is the same as 3D-R2N2.

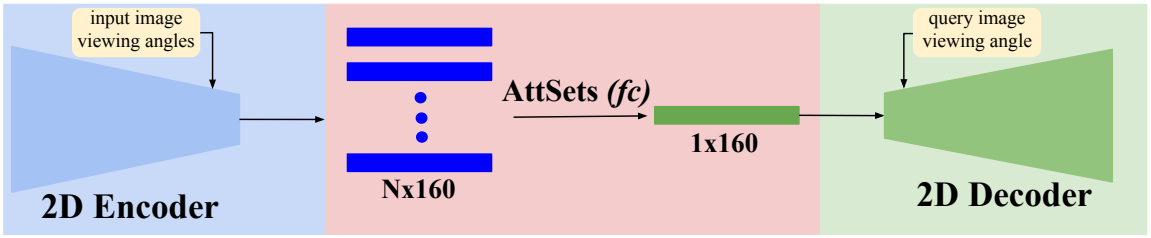


Figure 4.5: The architecture of $\text{Base}_{\text{silnet}}$-AttSets for multi-view 3D shape learning. The base encoder-decoder is the same as SilNet.

- Encoder-decoder of SilNet. The original SilNet consists of (1) a shared 2D encoder which encodes a size of $127 \times 127 \times 3$ images together with image viewing angles into 160 dimensional latent vectors, (2) a max pooling module which aggregates N latent vectors into a single vector, and (3) a 2D decoder which estimates an object silhouette (57×57) from the single latent vector and a new viewing angle. Instead of being explicitly supervised by 3D shape labels, SilNet aims to implicitly learn a 3D shape representation from multiple images via the supervision of 2D silhouettes. Figure 4.5 shows the architecture of AttSets based SilNet where the only difference is that the original max pooling is replaced by AttSets in the middle. This network is called $\text{Base}_{\text{silnet}}$-AttSets.

4.4.2 Competing Approaches

We compare our AttSets and FASet with three groups of competing approaches. Note that all the following competing approaches are connected at the same location of the base encoder-decoder shown in the pink block of Figures 4.4 and 4.5, with the same network configurations and training/testing settings.

- RNNs. The original 3D-R2N2 makes use of the **GRU** [32, 102] unit for feature aggregation and serves as a solid baseline.

- First-order poolings. The widely used **max**/**mean**/ **sum** pooling operations [91, 199, 53] are all implemented for comparison.

- Higher-order poolings. We also compare with the state-of-the-art higher-order pooling approaches, including bilinear pooling (**BP**) [149], and the very recent **MHBN** [312] and **SMSO** poolings [310].

4.4.3 Datasets

All approaches are evaluated on four large open datasets.

- $\text{ShapeNet}_{\text{r2n2}}$ Dataset [32]. The released 3D-R2N2 dataset consists of 13 categories of $43,783$ common objects with synthesized RGB images from the large scale ShapeNet 3D repository [21]. For each 3D object, 24 images are rendered from different viewing angles circling around each object. The train/test dataset split is $0.8 : 0.2$.

- $\text{ShapeNet}_{\text{lsm}}$ Dataset [102]. Both LSM and 3D-R2N2 datasets are generated from the same 3D ShapeNet repository [21], *i.e.*, they have the same ground truth labels regarding the same object. However, the $\text{ShapeNet}_{\text{lsm}}$ dataset has totally different camera viewing angles and lighting sources for the rendered RGB images. Therefore, we use the $\text{ShapeNet}_{\text{lsm}}$ dataset to evaluate the robustness and generality of all approaches. All images of $\text{ShapeNet}_{\text{lsm}}$ dataset are resized from 224×224 to 127×127 through linear interpolation.

- ModelNet40 Dataset. ModelNet40 [287] consists of $12,311$ objects belonging to 40 categories. The 3D models are split into $9,843$ training samples and $2,468$ testing samples. For each 3D model, it is voxelized into a $30 \times 30 \times 30$ shape in [207], and 12 RGB images are rendered from different viewing angles.All 3D shapes are zero-padded to be $32 \times 32 \times 32$, and the images are linearly resized from 224×224 to 127×127 for training and testing.

- Blobby Dataset [279]. It contains $11,706$ blobby objects. Each object has 5 RGB images paired with viewing angles and the corresponding silhouettes, which are generated from Cycles in Blender under different lighting sources and texture models.

Table 4.1: Group 1: mean IoU for multi-view reconstruction of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **2 images** per object in Stage 2, while other competing approaches are fine-tuned given **2 images** per object in Stage 2.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	0.574	0.608	0.622	0.629	0.633	0.639	0.642	0.642	0.641	0.640
$\text{Base}_{\text{r2n2}}$-max pooling	0.620	0.651	0.660	0.665	0.666	0.671	0.672	0.674	0.673	0.673
$\text{Base}_{\text{r2n2}}$-mean pooling	0.632	0.654	0.661	0.666	0.667	0.674	0.676	0.680	0.680	0.681
$\text{Base}_{\text{r2n2}}$-sum pooling	0.633	0.657	0.665	0.669	0.669	0.670	0.666	0.667	0.666	0.665
$\text{Base}_{\text{r2n2}}$-BP pooling	0.588	0.608	0.615	0.620	0.621	0.627	0.628	0.632	0.633	0.633
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.578	0.599	0.606	0.611	0.612	0.618	0.620	0.623	0.624	0.624
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.623	0.654	0.664	0.670	0.672	0.679	0.679	0.682	0.680	0.678
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.642**	**0.665**	**0.672**	**0.677**	**0.678**	**0.684**	**0.686**	**0.690**	**0.690**	**0.690**

4.4.4 Metrics

The explicit 3D voxel reconstruction performance of $\text{Base}_{\text{r2n2}}$-AttSets and the competing approaches is evaluated on three datasets: $\text{ShapeNet}_{\text{r2n2}}$, $\text{ShapeNet}_{\text{lsm}}$ and ModelNet40. We use the mean Intersection-over-Union (IoU) [32] between predicted 3D voxel grids and their ground truth as the metric. The IoU for an individual voxel grid is formally defined as follows:

$$IoU = \frac{\sum_{i=1}^{L} \left[I(h_i > p) * I(\bar{h_i}) \right]}{\sum_{i=1}^{L} \left[I\left(I(h_i > p) + I(\bar{h_i}) \right) \right]}$$

where $I(\cdot)$ is an indicator function, h_i is the predicted value for the i^{th} voxel, $\bar{h_i}$ is the corresponding ground truth, p is the threshold for voxelization, L is the total number of voxels in a whole voxel grid. As there is no validation split in the above three datasets, to calculate the IoU scores, we independently search the optimal binarization threshold value from $0.2 \sim 0.8$ with a step 0.05 for all approaches for fair comparison. In our experiments, we found that all optimal thresholds of different approaches end up with 0.3 or 0.35.

The implicit 3D shape learning performance of $\text{Base}_{\text{silnet}}$-AttSets and the competing approaches is evaluated on the Blobby dataset. The mean IoU between predicted 2D silhouettes and the ground truth is used as the metric [279].

4.4.5 Evaluation on $\text{ShapeNet}_{\text{r2n2}}$ Dataset

To fully evaluate the aggregation performance and robustness, we train the $\text{Base}_{\text{r2n2}}$-AttSets and its competing approaches on $\text{ShapeNet}_{\text{r2n2}}$ dataset. For fair comparison,

Table 4.2: Group 2: mean IoU for multi-view reconstruction of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **8 images** per object in Stage 2, while other competing approaches are fine-tuned given **8 images** per object in Stage 2.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	0.580	0.616	0.629	0.637	0.641	0.649	0.652	0.652	0.652	0.652
$\text{Base}_{\text{r2n2}}$-max pooling	0.524	0.615	0.641	0.655	0.661	0.674	0.678	0.683	0.684	0.684
$\text{Base}_{\text{r2n2}}$-mean pooling	0.632	0.657	0.665	0.670	0.672	0.679	0.681	0.685	0.686	0.686
$\text{Base}_{\text{r2n2}}$-sum pooling	0.580	0.628	0.644	0.656	0.660	0.672	0.677	0.682	0.684	0.685
$\text{Base}_{\text{r2n2}}$-BP pooling	0.544	0.599	0.618	0.628	0.632	0.644	0.648	0.654	0.655	0.656
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.570	0.596	0.606	0.612	0.614	0.621	0.624	0.628	0.629	0.629
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.570	0.621	0.641	0.652	0.656	0.668	0.673	0.679	0.680	0.681
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.642**	**0.662**	**0.671**	**0.676**	**0.678**	**0.686**	**0.688**	**0.693**	**0.694**	**0.694**

Table 4.3: Group 3: mean IoU for multi-view reconstruction of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **16 images** per object in Stage 2, while other competing approaches are fine-tuned given **16 images** per object in Stage 2.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	0.579	0.614	0.628	0.636	0.640	0.647	0.651	0.652	0.653	0.653
$\text{Base}_{\text{r2n2}}$-max pooling	0.511	0.604	0.633	0.649	0.656	0.671	0.678	0.684	0.686	0.686
$\text{Base}_{\text{r2n2}}$-mean pooling	0.594	0.637	0.652	0.662	0.667	0.677	0.682	0.687	0.688	0.689
$\text{Base}_{\text{r2n2}}$-sum pooling	0.570	0.629	0.647	0.657	0.664	0.678	0.684	0.690	0.692	0.692
$\text{Base}_{\text{r2n2}}$-BP pooling	0.545	0.593	0.611	0.621	0.627	0.637	0.642	0.647	0.648	0.649
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.570	0.596	0.606	0.612	0.614	0.621	0.624	0.627	0.628	0.629
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.580	0.627	0.643	0.652	0.656	0.668	0.673	0.679	0.680	0.681
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.642**	**0.660**	**0.668**	**0.673**	**0.676**	**0.683**	**0.687**	**0.691**	**0.692**	**0.693**

all networks (the pooling/GRU/AttSets based approaches) are trained according to the proposed two-stage training algorithm.

Training Stage 1. All networks are trained given only 1 image for each object, *i.e.*, $N = 1$ in all training iterations, until convergence. Basically, this is to guarantee all networks are well optimized for the extreme case where there is only one input image.

Training Stage 2. To enable these networks to be more robust for multiple input images, all networks are further trained given more images per object. Particularly, we conduct the following five parallel groups of training experiments.

- Group 1. All networks are further trained given only 2 images for each object, *i.e.*, $N = 2$ in all iterations. As with our $\text{Base}_{\text{r2n2}}$-AttSets, the well-trained

Table 4.4: Group 4: mean IoU for multi-view reconstruction of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **24 images** per object in Stage 2, while other competing approaches are fine-tuned given **24 images** per object in Stage 2.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	0.578	0.613	0.627	0.635	0.639	0.647	0.651	0.653	0.653	0.654
$\text{Base}_{\text{r2n2}}$-max pooling	0.504	0.600	0.631	0.648	0.655	0.671	0.679	0.685	0.688	0.689
$\text{Base}_{\text{r2n2}}$-mean pooling	0.593	0.634	0.649	0.659	0.663	0.673	0.677	0.683	0.684	0.685
$\text{Base}_{\text{r2n2}}$-sum pooling	0.580	0.634	0.650	0.658	0.660	0.678	0.682	0.689	0.690	0.691
$\text{Base}_{\text{r2n2}}$-BP pooling	0.524	0.585	0.609	0.623	0.630	0.644	0.650	0.656	0.659	0.660
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.566	0.595	0.606	0.613	0.616	0.624	0.627	0.631	0.632	0.632
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.556	0.613	0.635	0.647	0.653	0.668	0.674	0.681	0.682	0.684
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.642**	**0.660**	**0.668**	**0.674**	**0.676**	**0.684**	**0.688**	**0.693**	**0.694**	**0.695**

Table 4.5: Group 5: mean IoU for multi-view reconstruction of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given random number of images per object in Stage 2, *i.e.*, N is uniformly sampled from **[1, 24]**, while other competing approaches are fine-tuned given random number of views per object in Stage 2.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	0.580	0.615	0.629	0.637	0.641	0.648	0.651	0.651	0.651	0.651
$\text{Base}_{\text{r2n2}}$-max pooling	0.601	0.638	0.652	0.660	0.663	0.673	0.677	0.682	0.683	0.684
$\text{Base}_{\text{r2n2}}$-mean pooling	0.598	0.637	0.652	0.660	0.664	0.675	0.679	0.684	0.685	0.686
$\text{Base}_{\text{r2n2}}$-sum pooling	0.587	0.631	0.646	0.656	0.660	0.672	0.678	0.683	0.684	0.685
$\text{Base}_{\text{r2n2}}$-BP pooling	0.582	0.610	0.620	0.626	0.628	0.635	0.638	0.641	0.642	0.643
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.575	0.599	0.608	0.613	0.615	0.622	0.624	0.628	0.629	0.629
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.580	0.624	0.641	0.652	0.657	0.669	0.674	0.679	0.681	0.682
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.642**	**0.662**	**0.670**	**0.675**	**0.677**	**0.685**	**0.688**	**0.692**	**0.693**	**0.694**

encoder-decoder in previous stage 1 is frozen, and we only optimize the AttSets module according to our FASet algorithm 1. For the competing approaches, *e.g.*, GRU and all poolings, we fine-tune the whole networks because they do not have separate parameters suitable for special training. To be specific, we use smaller learning rate (1e-5) to carefully train these networks to achieve better performance where $N = 2$ until convergence.

- Group 2/3/4. Similarly, in these three groups of second-stage training experiments, N is set to 8, 16 and 24 separately.

- Group 5. All networks are further trained until convergence, but N is uniformly and randomly sampled from $[1, 24]$ for each object during training. In the above

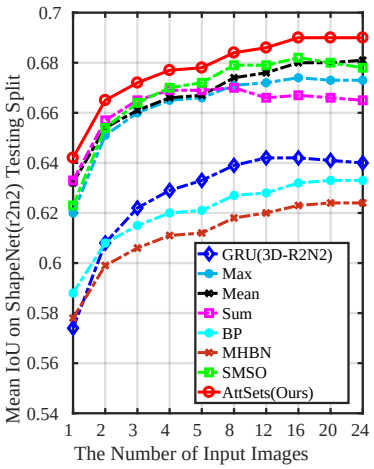


Figure 4.6: IoUs: Group 1.

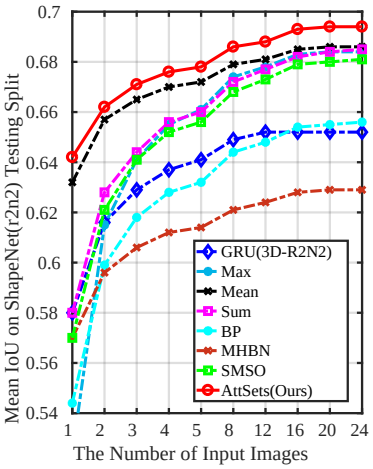


Figure 4.7: IoUs: Group 2.

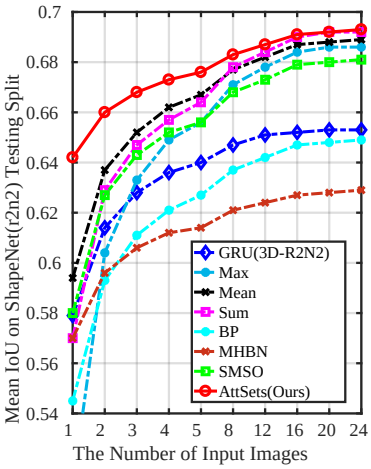


Figure 4.8: IoUs: Group 3.

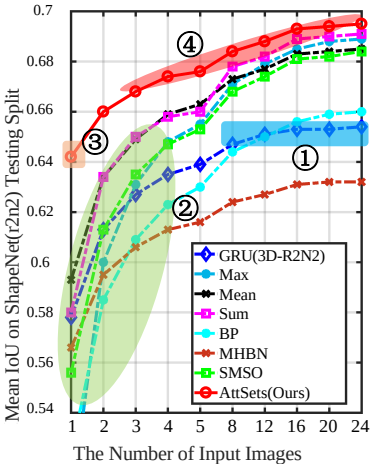


Figure 4.9: IoUs: Group 4.

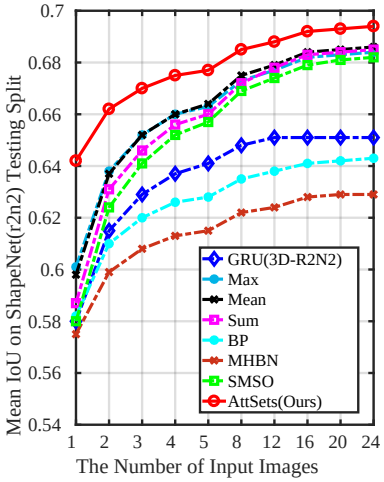


Figure 4.10: IoUs: Group 5.

Group 1/2/3/4, N is fixed for each object, while N is dynamic for each object in Group 5.

In the above experiment, Groups 1/2/3/4 are designed to investigate how all competing approaches would be further optimized towards the statistics of a fixed N during training, thus resulting in different levels of robustness given an arbitrary number of N during testing. By contrast, the paradigm in Group 5 aims at enumerating all possible N values during training. Therefore the overall performance might be more robust with an arbitrary number of input images during testing, compared with the above Group 1/2/3/4 experiments.

Testing Stage. All networks trained in the above five groups of experiments are separately tested given $N = \{1, 2, 3, 4, 5, 8, 12, 16, 20, 24\}$. The permutations of input images are the same for all different approaches for fair comparison. Note that, we do

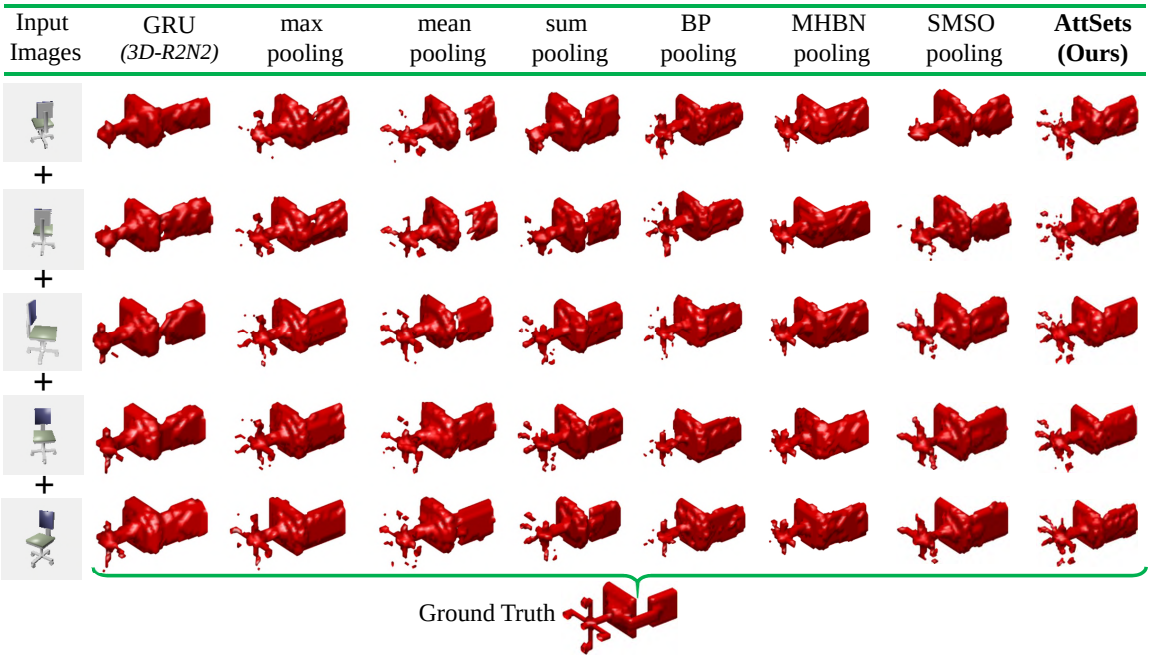


Figure 4.11: Qualitative results of multi-view reconstruction achieved by different approaches in experiment Group 5.

not test the networks which are only trained in Stage 1, because the AttSets module is not optimized and the corresponding $\text{Base}_{\text{r2n2}}$-AttSets is unable to generalize to multiple input images during testing. Therefore, it is meaningless to compare the performance when the network is solely trained on a single image.

Results. Tables 4.1 $\sim$ 4.5 show the mean IoU scores of all 13 categories for experiments of Group 1 $\sim$ 5, while Figures 4.6 $\sim$ 4.10 show the trends of mean IoU changes in different Groups. Figure 4.11 shows the estimated 3D shapes in experiment Group 5, with an increasing number of images from 1 to 5 for different approaches.

We notice that the reported IoU scores of ShapeNet data repository in the original LSM [102] are higher than our scores. However, the experimental settings in LSM [102] are quite different from ours in the following two ways. 1) The original LSM requires both RGB images and the corresponding viewing angles as input, while all our experiments do not. 2) The original LSM dataset has different styles of rendered color images and different train/test splits compared with our experimental settings. Therefore the reported IoU scores in LSM are not directly comparable with ours and we do not include the results in this paper to avoid confusion. Note that, the aggregation module of LSM [102], *i.e.*, GRU, is the same as used in 3D-R2N2 [32], and is indeed fully evaluated throughout our experiments.

To highlight the performance of single view 3D reconstruction, Table 4.6 shows the

Table 4.6: Per-category mean IoU for single view reconstruction on $\text{ShapeNet}_{\text{r2n2}}$ testing split.

	plane	bench	cabinet	car	chair	monitor	lamp	spk.	firearm	couch	table	phone	w.craft	mean
$\text{Base}_{\text{r2n2}}$-GRU	0.530	0.449	0.730	0.807	0.487	0.497	0.391	0.671	0.553	0.631	0.515	0.683	0.535	0.580
$\text{Base}_{\text{r2n2}}$-max pooling	0.573	0.521	0.755	0.835	0.533	0.544	0.423	0.695	0.587	0.678	0.562	0.710	0.582	0.620
$\text{Base}_{\text{r2n2}}$-mean pooling	0.582	0.540	0.773	0.837	0.547	0.550	0.440	0.713	0.595	0.695	0.576	0.718	0.593	0.632
$\text{Base}_{\text{r2n2}}$-sum pooling	0.588	0.536	0.771	0.838	0.554	0.547	0.442	0.710	0.598	0.690	0.575	0.728	0.598	0.633
$\text{Base}_{\text{r2n2}}$-BP pooling	0.536	0.469	0.747	0.816	0.484	0.499	0.398	0.678	0.556	0.646	0.528	0.681	0.550	0.588
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.528	0.451	0.742	0.812	0.471	0.487	0.386	0.677	0.548	0.637	0.515	0.674	0.546	0.578
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.572	0.521	0.763	0.833	0.541	0.548	0.433	0.704	0.581	0.682	0.566	0.721	0.581	0.623
OGN	0.587	0.481	0.729	0.816	0.483	0.502	0.398	0.637	0.593	0.646	0.536	0.702	**0.632**	0.596
AORM	**0.605**	0.498	0.715	0.757	0.532	0.524	0.415	0.623	**0.618**	0.679	0.547	0.738	0.552	0.600
PointSet	0.601	0.550	0.771	0.831	0.544	0.552	**0.462**	**0.737**	0.604	**0.708**	**0.606**	**0.749**	0.611	0.640
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	0.594	**0.552**	**0.783**	**0.844**	**0.559**	**0.565**	0.445	0.721	0.601	0.703	0.590	0.743	0.601	**0.642**

optimal per-category IoU scores for different competing approaches from experiments Group $1 \sim 5$. In addition, we also compare with the state-of-the-art dedicated single view reconstruction approaches including OGN [250], AORM [303] and PointSet [55] in Table 4.6. Overall, our AttSets based approach outperforms all others by a large margin for either single view or multi view reconstruction, and generates much more compelling 3D shapes.

Analysis. We investigate the results as follows:

- The GRU based approach can generate reasonable 3D shapes in all experiments Group $1 \sim 5$ given either few or multiple images during testing, but the performance saturates quickly after being given more images, *e.g.*, 8 views, because the recurrent unit is not particularly capable of capturing features from longer image sequences, as illustrated in Figure 4.9 ①.

- In Group $1 \sim 4$, all pooling based approaches are able to estimate satisfactory 3D shapes when given a similar number of images in testing as in training, but they are unlikely to predict reasonable shapes given an arbitrary number of images. For example, in experiment Group 4, all pooling based approaches have inferior IoU scores given only few images as shown in Table 4.4 and Figure 4.9 ②, because the pooled features from fewer images during testing are unlikely to be as general and representative as pooled features from more images during training. Therefore, those models trained on 24 images fail to generalize well to only one image during testing.

- In Group 5, as shown in Table 4.5 and Figure 4.10, all pooling based approaches are much more robust compared with Group 1∼4, because the networks are generally optimized according to an arbitrary number of images during training.

However, these networks tend to have the performance *in the middle*. Compared with Group 4, all approaches in Group 5 tend to have better performance when $N = 1$, while being worse when $N = 24$. Compared with Group 1, all approaches in Group 5 are likely to be better when $N = 24$, while being worse when $N = 1$. Basically, these networks tend to be optimized to learn the *mean* features overall.

- In all experiments Group $1 \sim 5$, all approaches tend to have better performance when given sufficiently many input images, *i.e.*, $N = 24$, because more images are able to provide enough information for reconstruction.

- In all experiments Group $1 \sim 5$, our AttSets based approach clearly outperforms all others in either single or multiple view 3D reconstruction and it is more robust to a variable number of input images. Our FASet algorithm completely decouples the base network to learn visual features for accurate single view reconstruction as illustrated in Figure 4.9 ③, while the trainable parameters of AttSets module are separately responsible for learning attention scores for better multi-view reconstruction as shown in Figure 4.9 ④. Therefore, the whole network does not suffer from limitations of GRU or pooling approaches, and can achieve better performance for either fewer or more image reconstruction.

Table 4.7: Mean IoU for multi-view reconstruction of all 13 categories from $\text{ShapeNet}_{\text{lsm}}$ dataset. All networks are well trained in previous experiment Group 5 of Section 4.4.5.

number of views	1	2	3	4	5	8	12	16	20
$\text{Base}_{\text{r2n2}}$-GRU	0.390	0.428	0.444	0.454	0.459	0.467	0.470	0.471	0.472
$\text{Base}_{\text{r2n2}}$-max pooling	0.276	0.388	0.433	0.459	0.473	0.497	0.510	0.515	0.518
$\text{Base}_{\text{r2n2}}$-mean pooling	0.365	0.426	0.452	0.468	0.477	0.493	0.503	0.508	0.511
$\text{Base}_{\text{r2n2}}$-sum pooling	0.363	0.421	0.445	0.459	0.466	0.481	0.492	0.499	0.503
$\text{Base}_{\text{r2n2}}$-BP pooling	0.359	0.407	0.426	0.436	0.442	0.453	0.459	0.462	0.463
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.365	0.403	0.418	0.427	0.431	0.441	0.446	0.449	0.450
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.364	0.419	0.445	0.460	0.469	0.488	0.500	0.506	0.510
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.404**	**0.452**	**0.475**	**0.490**	**0.498**	**0.514**	**0.522**	**0.528**	**0.531**

4.4.6 Evaluation on $\text{ShapeNet}_{\text{lsm}}$ Dataset

To further investigate how well the learnt visual features and attention scores generalize across different style of images, we use the well trained networks of previous Group 5 of Section 4.4.5 to test on the large $\text{ShapeNet}_{\text{lsm}}$ dataset. Note that, we

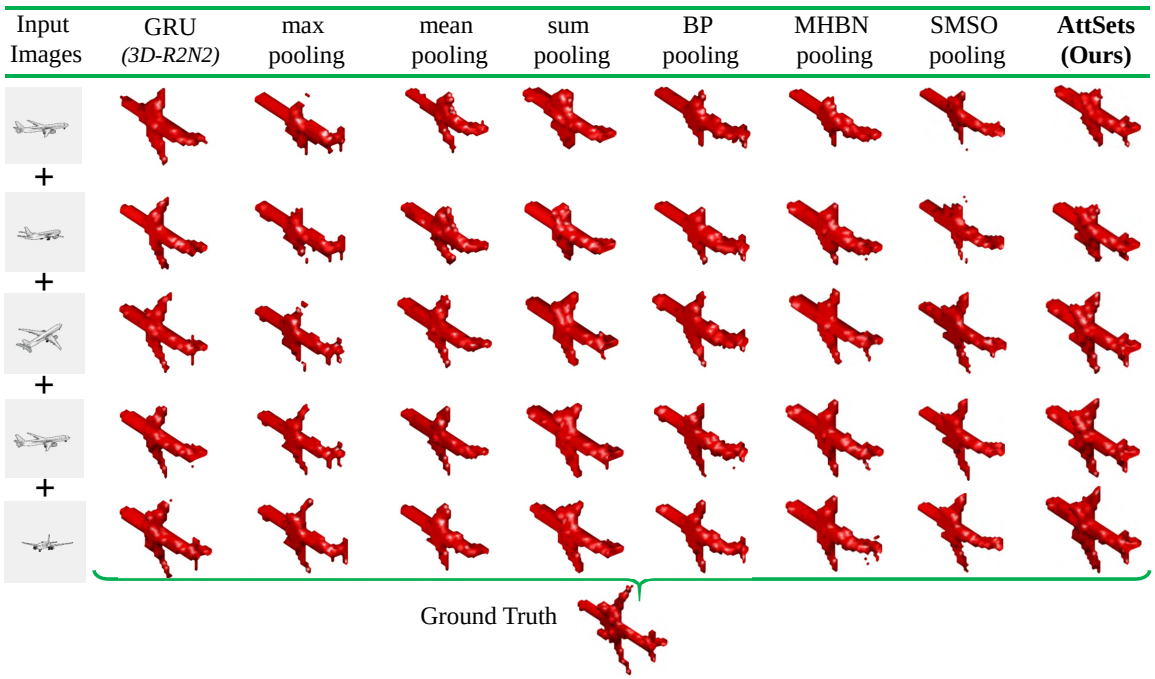


Figure 4.12: Qualitative results of multi-view reconstruction from different approaches in $\text{ShapeNet}_{\text{lsm}}$ testing split.

only borrow the synthesized images from $\text{ShapeNet}_{\text{lsm}}$ dataset corresponding to the objects in $\text{ShapeNet}_{\text{r2n2}}$ testing split. This guarantees that all the trained models have never seen either the style of LSM rendered images or the 3D object labels before. The image viewing angles from the original $\text{ShapeNet}_{\text{lsm}}$ dataset are not used in our experiments, since the $\text{Base}_{\text{r2n2}}$ network does not require image viewing angles as input. Table 4.7 shows the mean IoU scores of all approaches, while Figure 4.12 shows the qualitative results.

Our AttSets based approach outperforms all others given either few or multiple input images. This demonstrates that our $\text{Base}_{\text{r2n2}}$-AttSets approach does not overfit the training data, but has better generality and robustness over new styles of rendered color images compared with other approaches.

4.4.7 Evaluation on ModelNet40 Dataset

We train the $\text{Base}_{\text{r2n2}}$-AttSets and its competing approaches on ModelNet40 dataset from scratch. For fair comparison, all networks (the pooling/GRU/AttSets based approaches) are trained according to the proposed FASet algorithm, which is similar to the two-stage training strategy of Section 4.4.5.

Training Stage 1. All networks are trained given only 1 image for each object,

Table 4.8: Group 1: mean IoU for multi-view reconstruction of all 40 categories in ModelNet40 testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **12 images** per object in Stage 2, while other competing approaches are fine-tuned given **12 images** per object in Stage 2.

number of views	1	2	3	4	5	8	12
$\text{Base}_{\text{r2n2}}$-GRU	0.344	0.390	0.414	0.430	0.440	0.454	0.464
$\text{Base}_{\text{r2n2}}$-max pooling	0.393	0.468	0.490	0.504	0.511	0.523	0.525
$\text{Base}_{\text{r2n2}}$-mean pooling	0.415	0.464	0.481	0.495	0.502	0.515	0.520
$\text{Base}_{\text{r2n2}}$-sum pooling	0.332	0.441	0.473	0.492	0.500	0.514	0.520
$\text{Base}_{\text{r2n2}}$-BP pooling	0.431	0.466	0.479	0.492	0.497	0.509	0.515
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.423	0.462	0.478	0.491	0.497	0.509	0.515
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.441	0.476	0.490	0.500	0.506	0.517	0.520
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.487**	**0.505**	**0.511**	**0.517**	**0.521**	**0.527**	**0.529**

Table 4.9: Group 2: mean IoU for multi-view reconstruction of all 40 categories in ModelNet40 testing split. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given random number of images per object in Stage 2, *i.e.*, N is uniformly sampled from [**1, 12**], while other competing approaches are fine-tuned given random number of views per object in Stage 2.

number of views	1	2	3	4	5	8	12
$\text{Base}_{\text{r2n2}}$-GRU	0.388	0.421	0.434	0.440	0.444	0.449	0.452
$\text{Base}_{\text{r2n2}}$-max pooling	0.461	0.489	0.498	0.506	0.509	0.515	0.517
$\text{Base}_{\text{r2n2}}$-mean pooling	0.455	0.487	0.498	0.507	0.512	0.520	0.523
$\text{Base}_{\text{r2n2}}$-sum pooling	0.453	0.484	0.494	0.503	0.506	0.514	0.517
$\text{Base}_{\text{r2n2}}$-BP pooling	0.454	0.479	0.487	0.496	0.499	0.507	0.510
$\text{Base}_{\text{r2n2}}$-MHBN pooling	0.453	0.480	0.488	0.497	0.500	0.507	0.509
$\text{Base}_{\text{r2n2}}$-SMSO pooling	0.462	0.488	0.497	0.505	0.509	0.516	0.519
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	**0.487**	**0.505**	**0.511**	**0.518**	**0.520**	**0.525**	**0.527**

i.e., $N = 1$ in all training iterations, until convergence. This guarantees all networks are well optimized for single view 3D reconstruction.

Training Stage 2. We further conduct the following two parallel groups of training experiments to optimize the networks for multi-view reconstruction.

- Group 1. All networks are further trained given all 12 images for each object, *i.e.*, $N = 12$ in all iterations, until convergence. As with our $\text{Base}_{\text{r2n2}}$-AttSets, the well-trained encoder-decoder in the previous Stage 1 is frozen, and only the AttSets module is trained. All other competing approaches are fine-tuned using a smaller learning rate (1e-5) in this stage.

- Group 2. All networks are further trained until convergence, but N is uniformly

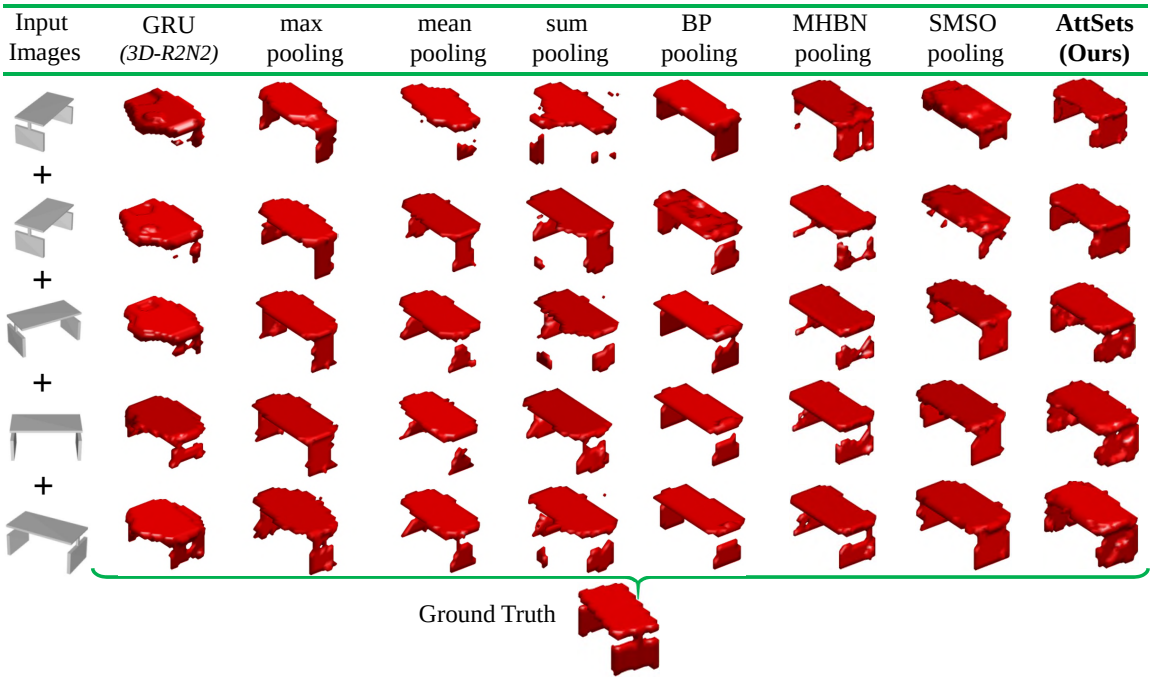


Figure 4.13: Qualitative results of multi-view reconstruction from different approaches in ModelNet40 testing split.

and randomly sampled from $[1, 12]$ for each object during training. Only the AttSets module is trained, while all other competing approaches are fine-tuned in Stage 2.

Testing Stage. All networks trained in the above two groups are separately tested given $N = [1, 2, 3, 4, 5, 8, 12]$. The permutations of input images are the same for all different approaches for fair comparison.

Results. Tables 4.8 and 4.9 show the mean IoU scores of Groups 1 and 2 respectively, and Figure 4.13 shows qualitative results of Group 2. The $\text{Base}_{\text{r2n2}}$-AttSets surpasses all competing approaches by a large margin for both single and multiple view 3D reconstruction, and all the results are consistent with previous experimental results on both $\text{ShapeNet}_{\text{r2n2}}$ and $\text{ShapeNet}_{\text{lsm}}$ datasets.

4.4.8 Evaluation on Blobby Dataset

In this section, we evaluate the $\text{Base}_{\text{silnet}}$-AttSets and the competing approaches on the Blobby dataset. For fair comparison, the GRU module is implemented with a single fully connected layer of 160 hidden units, which has similar network capacity with our AttSets based network. All networks (the pooling/GRU/AttSets based approaches) are trained with the proposed two-stage FASet algorithm as follows:

Table 4.10: Group 1: mean IoU for silhouettes prediction on the Blobby dataset. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **2 images** per object, *i.e.*, N =2, while other competing approaches are fine-tuned given 2 views per object in Stage 2.

	1 view	2 views	3 views	4 views
$\text{Base}_{\text{silnet}}$-GRU	0.857	0.860	0.860	0.860
$\text{Base}_{\text{silnet}}$-max pooling	0.922	0.923	0.924	0.924
$\text{Base}_{\text{silnet}}$-mean pooling	0.920	0.922	0.923	0.924
$\text{Base}_{\text{silnet}}$-sum pooling	0.913	0.918	0.917	0.916
$\text{Base}_{\text{silnet}}$-BP pooling	0.908	0.912	0.914	0.914
$\text{Base}_{\text{silnet}}$-MHBN pooling	0.901	0.904	0.906	0.906
$\text{Base}_{\text{silnet}}$-SMSO pooling	0.860	0.865	0.865	0.865
$\text{Base}_{\text{silnet}}$-AttSets(Ours)	**0.924**	**0.931**	**0.933**	**0.935**

Table 4.11: Group 2: mean IoU for silhouettes prediction on the Blobby dataset. All networks are firstly trained given only 1 image for each object in Stage 1. The AttSets module is further trained given **4 images** per object, *i.e.*, N=4, while other competing approaches are fine-tuned given 4 views per object in Stage 2.

	1 view	2 views	3 views	4 views
$\text{Base}_{\text{silnet}}$-GRU	0.863	0.865	0.865	0.865
$\text{Base}_{\text{silnet}}$-max pooling	0.923	0.927	0.929	0.929
$\text{Base}_{\text{silnet}}$-mean pooling	0.923	0.925	0.927	0.927
$\text{Base}_{\text{silnet}}$-sum pooling	0.902	0.917	0.921	0.924
$\text{Base}_{\text{silnet}}$-BP pooling	0.911	0.916	0.919	0.920
$\text{Base}_{\text{silnet}}$-MHBN pooling	0.904	0.908	0.911	0.911
$\text{Base}_{\text{silnet}}$-SMSO pooling	0.863	0.865	0.865	0.865
$\text{Base}_{\text{silnet}}$-AttSets(Ours)	**0.924**	**0.932**	**0.936**	**0.937**

Training Stage 1. All networks are trained given only 1 image together with the viewing angle for each object, *i.e.*, N=1 in all training iterations, until convergence. This guarantees the performance of single view shape learning.

Training Stage 2. Another two parallel groups of training experiments are conducted to further optimize the networks for multi-view shape learning.

- Group 1. All networks are further trained given only 2 images for each object, *i.e.*, N=2 in all iterations. As to $\text{Base}_{\text{silnet}}$-AttSets, only the AttSets module is optimized with the well-trained base encoder-decoder being frozen. For fair comparison, all competing approaches are fine-tuned given 2 images per object for better performance where N =2 until convergence.

- Group 2. Similar to the above Group 1, all networks are further trained given

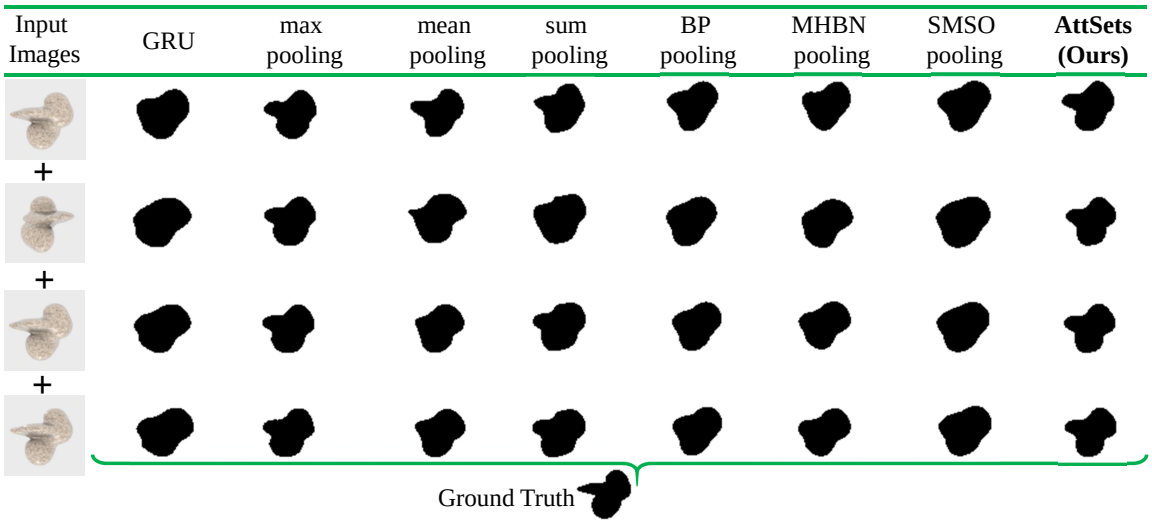


Figure 4.14: Qualitative results of silhouettes prediction from different approaches on the Blobby dataset.

all 4 images for each object, *i.e.*, N=4, until convergence.

Testing Stage. All networks trained in the above two groups are separately tested given N = [1,2,3,4]. The permutations of input images are the same for all different networks for fair comparison.

Results. Tables 4.10 and 4.11 show the mean IoUs of the above two groups of experiments and Figure 4.14 shows the qualitative results of Group 2. Note that, the IoUs are calculated on predicted 2D silhouettes instead of 3D voxels, so they are not numerically comparable with previous experiments on $\text{ShapeNet}_{\text{r2n2}}$, $\text{ShapeNet}_{\text{lsm}}$, and ModelNet40 datasets. We do not include the IoU scores of the original SilNet [279], because the original IoU scores are obtained from an end-to-end training strategy. In this paper, we uniformly apply the proposed two-stage FASet training paradigm on all approaches for fair comparison. Our $\text{Base}_{\text{silnet}}$-AttSets consistently outperforms all competing approaches for shape learning from either single or multiple views.

4.4.9 Qualitative Results on Real-world Images

To the best of our knowledge, there is no public real-world dataset for multi-view 3D object reconstruction. Therefore, we manually collect real world images from Amazon online shops to qualitatively demonstrate the generality of all networks which are trained on the synthetic $\text{ShapeNet}_{\text{r2n2}}$ dataset in experiment Group 4 of Section 4.4.5, as shown in Figure 4.15.

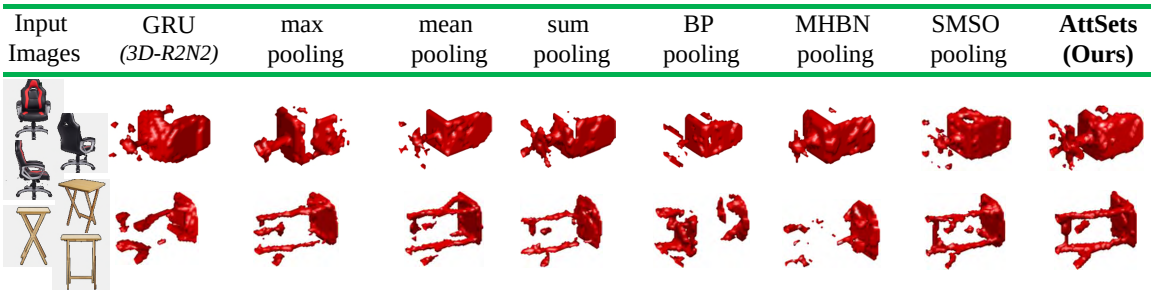


Figure 4.15: Qualitative results of multi-view 3D reconstruction from real-world images.

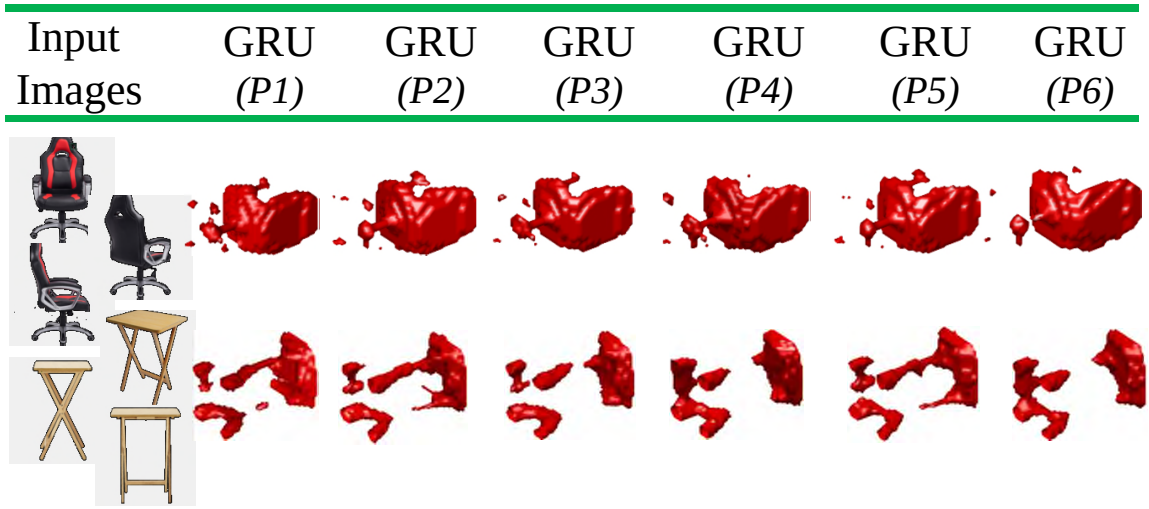


Figure 4.16: Qualitative results of inconsistent 3D reconstruction from the GRU based approach.

In the meantime, we use these real-world images to qualitatively show the permutation invariance of different approaches. In particular, for each object, we use 6 different permutations in total for testing. As shown in Figure 4.16, the GRU based approach generates inconsistent 3D shapes given different image permutations. For example, the arm of a chair and the leg of a table can be reconstructed in permutation 1, but fail to be recovered in another permutation. By comparison, all other approaches are permutation invariant, as shown in Figure 4.15.

4.4.10 Computational Efficiency

To evaluate the computation and memory cost of AttSets, we implement $\text{Base}_{\text{r2n2}}$-AttSets and the competing approaches in Python 2.7 and Tensorflow 1.2 with CUDA 9.0 and cuDNN 7.1 as the back-end driver and library. All approaches share the same $\text{Base}_{\text{r2n2}}$ network and run in the same Titan X and software environments. Table

Table 4.12: Mean time consumption for a single object (32^3 voxel grid) estimation from different number of images (milliseconds).

number of input images	1	4	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-GRU	6.9	11.2	17.0	22.8	28.8	34.7	40.7
$\text{Base}_{\text{r2n2}}$-max pooling	6.4	**10.0**	**15.1**	20.2	**25.3**	**30.2**	**35.4**
$\text{Base}_{\text{r2n2}}$-mean pooling	**6.3**	10.1	**15.1**	**20.1**	**25.3**	30.3	35.5
$\text{Base}_{\text{r2n2}}$-sum pooling	6.4	10.1	**15.1**	**20.1**	**25.3**	30.3	35.5
$\text{Base}_{\text{r2n2}}$-BP pooling	6.5	10.5	15.6	20.5	25.7	30.6	35.8
$\text{Base}_{\text{r2n2}}$-MHBN pooling	6.5	10.3	15.3	20.3	25.5	30.7	35.7
$\text{Base}_{\text{r2n2}}$-SMSO pooling	6.5	10.2	15.3	20.3	25.4	30.5	35.6
$\text{Base}_{\text{r2n2}}$-AttSets(Ours)	7.7	11.0	16.3	21.2	26.3	31.4	36.4

Table 4.13: Mean IoU of AttSets variants on all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split.

number of views	1	2	3	4	5	8	12	16	20	24
$\text{Base}_{\text{r2n2}}$-AttSets ($conv2d$)	**0.642**	0.648	0.651	0.655	0.657	0.664	0.668	0.674	0.675	0.676
$\text{Base}_{\text{r2n2}}$-AttSets ($conv3d$)	**0.642**	**0.663**	**0.671**	**0.676**	**0.677**	0.683	0.685	0.689	0.690	0.690
$\text{Base}_{\text{r2n2}}$-AttSets ($fc$)	**0.642**	0.660	0.668	0.674	0.676	**0.684**	**0.688**	**0.693**	**0.694**	**0.695**

5.4 shows the average time consumption to reconstruct a single 3D object given different number of images. Our AttSets based approach is as efficient as the pooling methods, while $\text{Base}_{\text{r2n2}}$-GRU (*i.e.*, 3D-R2N2) takes more time when processing an increasing number of images due to the sequential computation mechanism of its GRU module. In terms of the total trainable weights, the max/mean/sum pooling based approaches have 16.66 million, while AttSets based net has 17.71 million. By contrast, the original 3D-R2N2 has 34.78 million, the BP/MHBN/SMSO have $141.57, 60.78$ and 17.71 million respectively. Overall, our AttSets outperforms the recurrent unit and pooling operations without incurring notable computation and memory cost.

4.4.11 Comparison between Variants of AttSets

We further compare the aggregation performance of fc, $conv2d$ and $conv3d$ based AttSets variants in Figure 4.3 in Section 4.2.4. The fc based AttSets net is the same as in Section 4.4.5. The $conv2d$ based AttSets is inserted into the middle of the 2D encoder, fusing a $(N, 4, 4, 256)$ tensor into $(1, 4, 4, 256)$, where N is an arbitrary image number. The $conv3d$ based AttSets is inserted into the middle of the 3D decoder, integrating a $(N, 8, 8, 8, 128)$ tensor into $(1, 8, 8, 8, 128)$. All other layers of these variants are the same. Both the $conv2d$ and $conv3d$ based AttSets networks are trained using the paradigm of experiment Group 4 in Section 4.4.5. Table 4.13

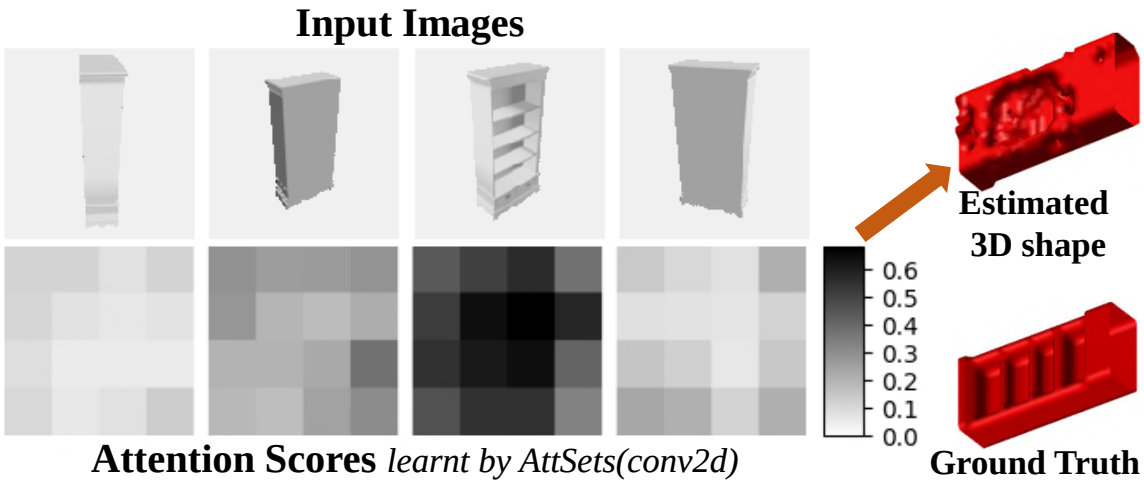


Figure 4.17: Learnt attention scores for deep feature sets via *conv*2*d* based AttSets.

Table 4.14: Mean IoU of all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split for feature-wise and element-wise attentional aggregation.

number of views	1	2	3	4	5	8	12	16	20	24
Base$_{\text{r2n2}}$-AttSets (element)	**0.642**	0.653	0.657	0.660	0.661	0.665	0.667	0.670	0.671	0.672
Base$_{\text{r2n2}}$-AttSets (feature)	**0.642**	**0.660**	**0.668**	**0.674**	**0.676**	**0.684**	**0.688**	**0.693**	**0.694**	**0.695**

shows the mean IoU scores of three variants on $\text{ShapeNet}_{\text{r2n2}}$ testing split. The fc and *conv*3*d* based variants achieve similar IoU scores for either single or multi view 3D reconstruction, demonstrating the superior aggregation capability of AttSets. In the meantime, we observe that the overall performance of *conv*2*d* based AttSets net is slightly decreased compared with the other two. One possible reason is that the 2D feature set has been aggregated at an early layer of the network, resulting in features being lost early. Figure 4.17 visualizes the learnt attention scores for a 2D feature set, *i.e.*, $(N, 4, 4, 256)$ features, via the *conv*2*d* based AttSets net. To visualize 2D feature scores, we average the scores along the channel axis and then roughly trace back the spatial locations of those scores corresponding to the original input. The more visual information the input image has, the higher the attention scores that are learnt by AttSets for the corresponding latent features. For example, the third image has richer visual information than the first image, so its attention scores are higher. Note that, for a specific base network, there are many potential locations to drop in AttSets and it is also possible to include multiple AttSets modules into the same net. How to fully evaluate these factors is suggested as an interesting direction for future work.

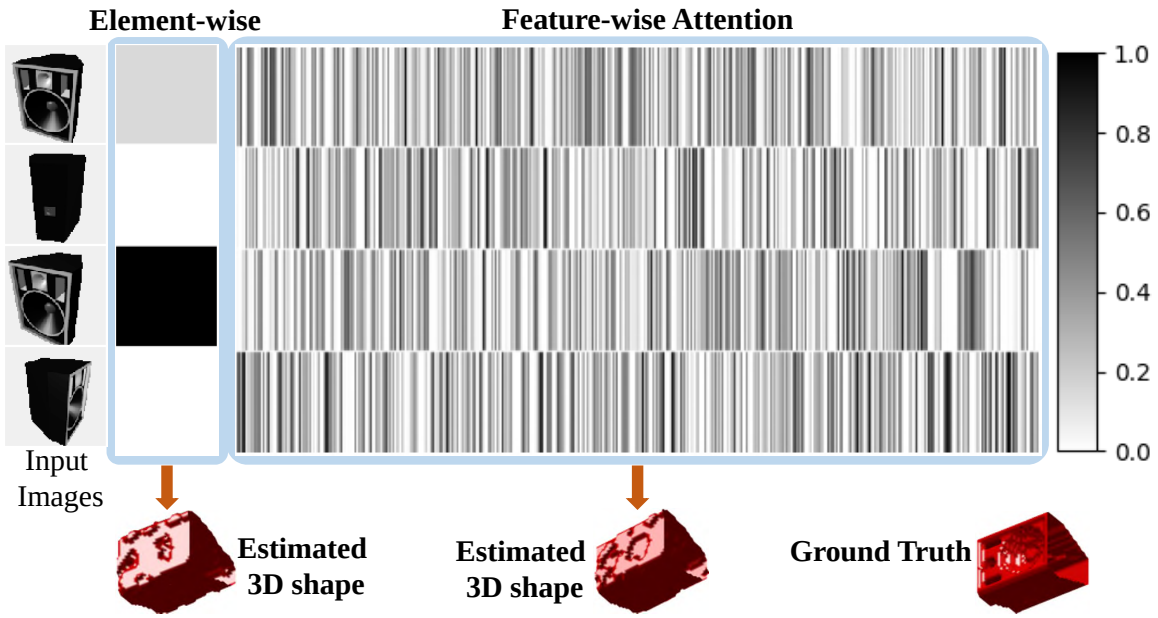


Figure 4.18: Learnt attention scores for deep feature sets via element-wise attention and feature-wise attention AttSets.

4.4.12 Feature-wise Attention *vs.* Element-wise Attention

Our AttSets module is initially designed to learn unique feature-wise attention scores for the whole input deep feature set, and we demonstrate that it significantly improves the aggregation performance over dynamic feature sets in Sections 4.4.5, 4.4.6, 4.4.7 and 4.4.8. In this section, we further investigate the advantage of this feature-wise attentive pooling over element-wise attentional aggregation.

For element-wise attentional aggregation, the AttSets module tends to learn a single attention score for each element of the feature set $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, followed by the $softmax$ normalization and weighted summation pooling. In particular, as shown in Figure 4.2, the shared function $g(\boldsymbol{x}_n, \boldsymbol{W})$ now learns a scalar, instead of a vector, as the attention activation for each input element. Eventually, all features within the same element are weighted by a learnt common attention score. Intuitively, the original feature-wise AttSets tends to be fine-grained aggregation, while the element-wise AttSets learns coarse aggregate features.

Following the same training settings of experiment Group 4 in Section 4.4.5, we conduct another group of experiments on the $\text{ShapeNet}_{\text{r2n2}}$ dataset for element-wise attentional aggregation. Table 4.14 compares the mean IoU for 3D object reconstruction through feature-wise and element-wise attentional aggregation. Figure 4.18 shows an example of the learnt attention scores and the predicted 3D shapes. As

Table 4.15: Mean IoU of different training algorithms on all 13 categories in $\text{ShapeNet}_{\text{r2n2}}$ testing split.

number of views	1	2	3	4	5	8	12	16	20	24
$\textbf{Base}_{\textbf{r2n2}}$**-AttSets** (JoinT)	0.307	0.437	0.516	0.563	0.595	0.639	0.659	0.673	0.677	0.680
$\textbf{Base}_{\textbf{r2n2}}$**-AttSets (FASet)**	**0.642**	**0.660**	**0.668**	**0.674**	**0.676**	**0.684**	**0.688**	**0.693**	**0.694**	**0.695**

expected, the feature-wise attention mechanism clearly achieves better aggregation performance compared with the coarse element-wise approach. As shown in Figure 4.18, the element-wise attention mechanism tends to focus on a few images, while completely ignoring others. By comparison, the feature-wise AttSets learns to fuse information across all images, thus achieving better aggregation performance.

4.4.13 Significance of FASet Algorithm

In this section, we investigate the impact of FASet algorithm by comparing it with the standard end-to-end joint training (JoinT). Particularly, in JoinT, all parameters Θ_{base} and Θ_{att} are jointly optimized with a single loss. Following the same training settings of experiment Group 4 in Section 4.4.5, we conduct another group of experiments on $\text{ShapeNet}_{\text{r2n2}}$ dataset under the JoinT training strategy. As IoU scores shown in Table 4.15, the JoinT training approach tends to optimize the whole net given the training multi-view batches, thus being unable to generalize well for fewer images during testing. Effectively, the network itself is unable to dedicate its base layers to learning visual features, decoupling the AttSets module to learning attention scores. The theoretical reason behind this has been discussed previously in Section 4.3.1. The FASet algorithm may also be applicable to other learning based aggregation approaches, as long as the aggregation module can be decoupled from the base encoder/decoder.

4.5 Conclusion

In this chapter, we present AttSets module and the FASet training algorithm to aggregate elements of deep feature sets. AttSets together with FASet has powerful permutation invariance, computation efficiency, robustness and flexible implementation properties, along with theoretical underpinnings and extensive experiments to support its prowess for multi-view 3D reconstruction. Both quantitative and qualitative results explicitly show that AttSets significantly outperforms other widely used

aggregation approaches. Nonetheless, all of our experiments are dedicated to multi-view 3D reconstruction. It would be interesting to explore the generality of AttSets and FASet over other set-based tasks such as point cloud classification and semantic segmentation [206], multi-view object recognition [247] and image tagging [313]. These tasks usually take an arbitrary number of elements as input.

Chapter 5

Learning to Segment 3D Objects from Point Clouds

5.1 Introduction

Imbuing machines with the ability to understand 3D scenes is a fundamental necessity for a number of key applications such as autonomous driving, augmented reality and robotics. Core problems over 3D geometric data, such as point clouds, include semantic segmentation, object detection and instance segmentation. Of these problems, instance segmentation has only started to be tackled in the literature. The primary obstacle is that point clouds are inherently unordered, unstructured and unevenly sampled. Widely used convolutional neural networks require the 3D point clouds to be voxelized into a regular grid, incurring high computational and memory costs.

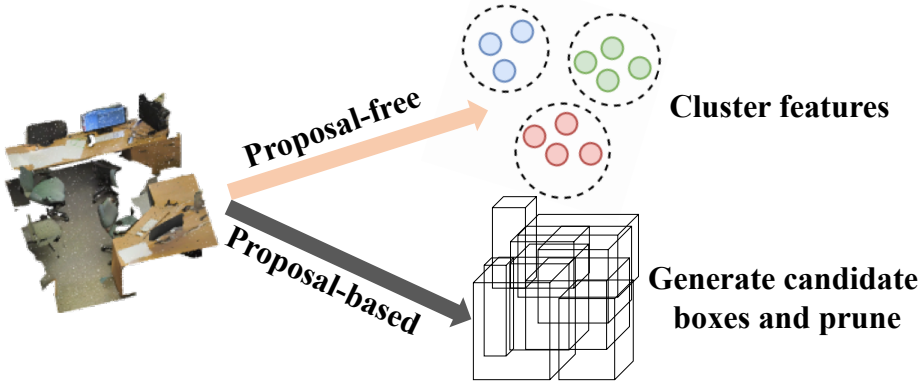


Figure 5.1: The existing pipelines for instance segmentation on 3D point clouds.

The first neural algorithm to directly tackle 3D instance segmentation is SGPN [270], which learns to group per-point features through a similarity matrix. Similarly, ASIS [271], JSIS3D [202], MASC [151], 3D-BEVIS [49] and [141] apply the same

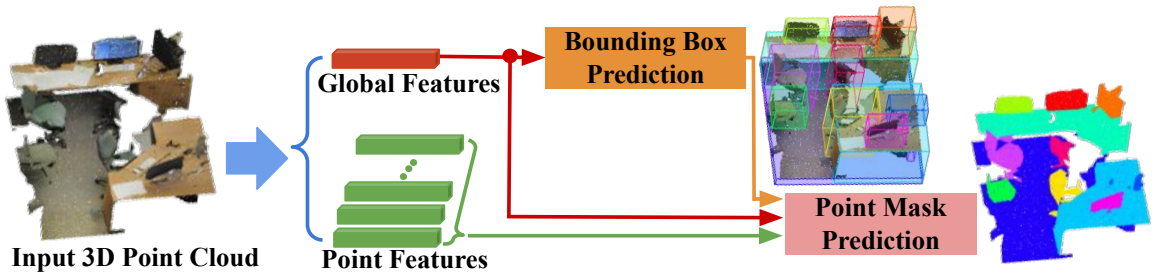


Figure 5.2: The 3D-BoNet framework for instance segmentation on 3D point clouds.

per-point feature grouping pipeline to segment 3D instances. Mo *et al.* formulate the instance segmentation as a per-point feature classification problem in PartNet [176]. However, the learnt segments of these proposal-free methods do not have high objectness as they do not explicitly detect the object boundaries. In addition, they inevitably require a post-processing step such as mean-shift clustering [35] to obtain the final instance labels, which is computationally heavy. Another pipeline is the proposal-based 3D-SIS [83] and GSPN [308], which usually rely on two-stage training and the expensive non-maximum suppression to prune dense object proposals.

Figure 5.1 illustrates the existing two pipelines for instance segmentation for 3D point clouds. Overall, the proposal-free methods learn to cluster point features, while the proposal-based approaches firstly generate candidate bounding boxes based on spatial anchors and then classify the points within each bounding box.

In this chapter, we present an elegant, efficient, and novel framework for 3D instance segmentation, where objects are loosely but uniquely detected through a single-forward stage using efficient MLPs, and then each instance is precisely segmented through a simple point-level binary classifier. To this end, we introduce a new bounding box prediction module together with a series of carefully designed loss functions to directly learn object boundaries. Our framework is significantly different from existing proposal-based and proposal-free approaches, since we are able to efficiently segment all instances with high objectness, but without relying on expensive and dense object proposals. Our code and data are available at *https://github.com/Yang7879/3D-BoNet*.

As shown in Figure 5.2, our framework, called **3D-BoNet**, is a single-stage, anchor-free, and end-to-end trainable neural architecture. It first uses an existing backbone network to extract a local feature vector for each point and a global feature vector for the whole input point cloud. The backbone is followed by two branches: 1) instance-level bounding box prediction, and 2) point-level mask prediction for instance segmentation.

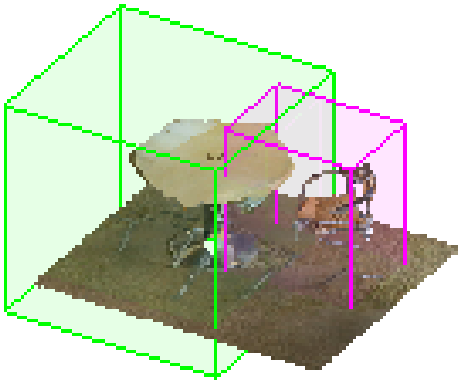


Figure 5.3: Rough instance boxes.

The **bounding box prediction branch** is the core of our framework. This branch aims to predict a unique, unoriented and rectangular bounding box for each instance in a single forward stage, without relying on predefined spatial anchors or a region proposal network [215]. As shown in Figure 5.3, we believe that roughly drawing a 3D bounding box for an instance is relatively achievable, because the input point clouds explicitly include 3D geometry information. Once the object bounding boxes are specified, it becomes easier to tackle point-level instance segmentation since reasonable bounding boxes can guarantee high objectness for learnt segments. However, learning good instance boxes involves two critical issues: 1) the number of total instances is variable, *i.e.*, from 1 to many, 2) there is no fixed order for all instances. These issues pose significant challenges for correctly optimizing the network, because there is no information to directly link predicted boxes with ground truth labels to supervise the network. However, we show how to elegantly solve these issues. The proposed box prediction branch simply takes the global feature vector as input and directly outputs a large and fixed number of bounding boxes together with confidence scores. These scores are used to indicate whether it is likely that the box contains a valid instance or not. To supervise the network, we design a novel *bounding box association layer* followed by a *multi-criteria loss function*. Given a set of ground-truth instances, we need to determine which of the predicted boxes best fit them. We formulate this association process as an optimal assignment problem with an existing solver. After the boxes have been optimally associated, our multi-criteria loss function not only minimizes the Euclidean distance of paired boxes, but also maximizes the coverage of valid points inside the predicted boxes.

The predicted boxes together with point and global features are then fed into the subsequent **point mask prediction branch**, in order to predict a point-level binary mask for each instance. The purpose of this branch is to classify whether each point inside a bounding box belongs to the valid instance or the background. Assuming the estimated instance box is reasonably good, it is very likely that an accurate point mask can be obtained, as the purpose of this branch is simply to reject points that are not part of the detected instance. As a degenerate example, randomly guessing could, on average, bring about 50% corrections.

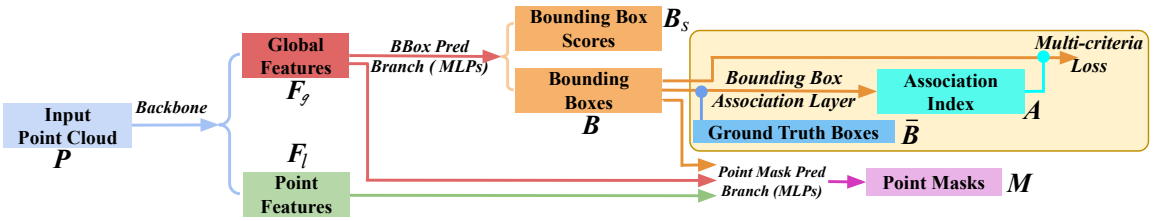


Figure 5.4: The general workflow of 3D-BoNet framework.

Overall, our framework is distinguished from all existing 3D instance segmentation approaches in three distinct ways: 1) Compared with the proposal-free pipeline, our method segments instances with high objectness by explicitly learning 3D object boundaries. 2) Compared with the widely-used proposal-based approaches, our framework does not require expensive and dense proposals. 3) Our framework is remarkably efficient, since the instance-level masks are learnt in a single-forward pass without requiring any post-processing steps. Our key contributions are:

- We propose a new framework for instance segmentation on 3D point clouds. The framework is single-stage, anchor-free and end-to-end trainable, without requiring any post-processing steps.
- We design a novel bounding box association layer followed by a multi-criteria loss function to supervise the box prediction branch.
- We demonstrate significant improvement over baselines and provide insight and intuition behind our design choices through extensive ablation studies.

5.2 Method Overview

As shown in Figure 5.4, our framework consists of two branches on top of the backbone network. Given an input point cloud $\boldsymbol{P}$ with N points in total, *i.e.*, $\boldsymbol{P} \in \mathbb{R}^{N \times k_0}$, where k_0 is the number of channels such as the location $\{x, y, z\}$ and color $\{r, g, b\}$ of each point, the **backbone network** extracts per-point local features, denoted as $\boldsymbol{F}_l \in \mathbb{R}^{N \times k}$, and aggregates a global point cloud feature vector, denoted as $\boldsymbol{F}_g \in \mathbb{R}^{1 \times k}$, where k is the length of feature vectors.

The **bounding box prediction branch** simply takes the global feature vector $\boldsymbol{F}_g$ as input, and directly regresses a predefined and fixed set of bounding boxes, denoted as $\boldsymbol{B}$, and the corresponding box scores, denoted as $\boldsymbol{B}_s$. We use ground truth bounding box information to supervise this branch. During training, the predicted

bounding boxes $\boldsymbol{B}$ and the ground truth boxes are fed into a *box association layer*. This layer aims to automatically associate a unique and the most similar predicted bounding box to each ground truth box. The output of the association layer is a list of association indices $\boldsymbol{A}$. The indices reorganize the predicted boxes, such that each ground truth box is paired with a unique predicted box for subsequent loss calculation. The predicted bounding box scores are also reordered accordingly before calculating loss. The reordered predicted bounding boxes are then fed into the *multi-criteria loss function*. At a high-level, this loss function aims to not only minimize the Euclidean distance between each ground truth box and the associated predicted box, but also maximize the coverage of valid points inside of each predicted box. Note that, both the bounding box association layer and multi-criteria loss function are only designed for network training. They are not used during testing. Eventually, this branch is able to predict a correct bounding box together with a box score for each instance directly.

In order to predict a point-level binary mask for each instance, every predicted box together with the previous derived local and global features, *i.e.*, $\boldsymbol{F}_l$ and $\boldsymbol{F}_g$, are further fed into the **point mask prediction branch**. This network branch is shared by all instances of different categories, and is therefore extremely light and compact. Such a class-agnostic approach inherently allows general segmentation across unseen categories.

5.3 Bounding Box Prediction

5.3.1 Bounding Box Encoding

In existing object detection networks, a bounding box is usually represented by the center location and the length of three dimensions [27], or the corresponding residuals [323] together with orientations. Instead, we completely parameterize the rectangular bounding box by specifying the two extreme vertices i.e. the minimum and maximum for simplicity:

$$\{[x_{min}\ y_{min}\ z_{min}], [x_{max}\ y_{max}\ z_{max}]\}$$

5.3.2 Neural Layers

As shown in Figure 5.5, the global feature vector $\boldsymbol{F}_g$ is fed through two fully connected layers with Leaky ReLU as the non-linear activation function. This is followed by another two parallel fully connected layers. One layer outputs a $6H$ dimensional

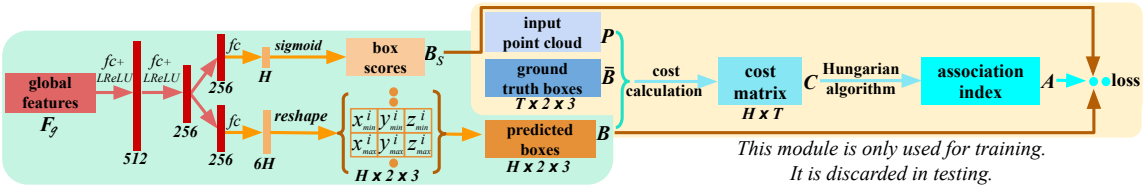


Figure 5.5: The architecture of the bounding box regression branch. The predicted H boxes are optimally associated with T ground truth boxes before calculating the multi-criteria loss.

vector, which is then reshaped as an $H \times 2 \times 3$ tensor. H is a predefined and fixed number of bounding boxes that the whole network is expected to predict as an upper limit. The other layer outputs an H dimensional vector followed by *sigmoid* function to represent the bounding box scores. The higher the score, the more likely it is that the predicted box contains an instance, indicating that the box is likely to be valid.

5.3.3 Bounding Box Association Layer

Given the previously predicted H bounding boxes, *i.e.*, $\boldsymbol{B} \in \mathbb{R}^{H\times 2\times 3}$, it is not straightforward to take use of the ground truth boxes, denoted as $\bar{\boldsymbol{B}} \in \mathbb{R}^{T\times 2\times 3}$, to supervise the network, because there are no predefined anchors to trace each predicted box back to a corresponding ground truth box in our framework. Besides, for each input point cloud $\boldsymbol{P}$, the number of ground truth boxes T varies and it is usually different from the predefined number H, although we can safely assume the predefined number $H \geq T$ for all input point clouds. In addition, there is no box order for either predicted or ground truth boxes.

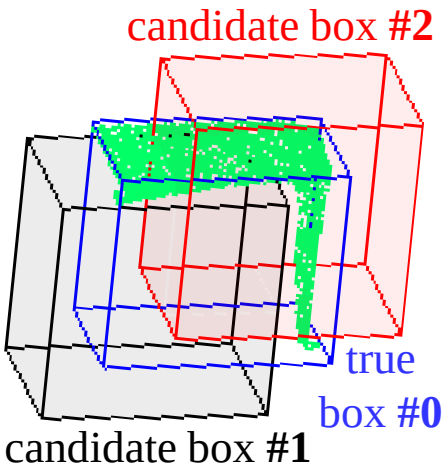


Figure 5.6: A sparse input point cloud.

(1) Optimal Association Formulation: To associate a unique predicted bounding box from $\boldsymbol{B}$ for each ground truth box of $\bar{\boldsymbol{B}}$, we formulate this association process as an optimal assignment problem. Formally, let $\boldsymbol{A}$ be a boolean association matrix where $\boldsymbol{A}_{i,j}$ =1 *iff* the i^{th} predicted box is assigned to the j^{th} ground truth box. $\boldsymbol{A}$ is also called association index in this paper. Let $\boldsymbol{C}$ be the association cost matrix where $\boldsymbol{C}_{i,j}$ represents the cost that the i^{th} predicted box is assigned to the j^{th} ground truth box. Basically, the cost $\boldsymbol{C}_{i,j}$ represents the similarity between two boxes; the smaller

the cost, the more similar the two boxes. Therefore, the bounding box association problem is to find the optimal assignment matrix $\boldsymbol{A}$ with the minimum overall cost:

$$\boldsymbol{A} = \arg\min_{\boldsymbol{A}} \sum_{i=1}^{H} \sum_{j=1}^{T} \boldsymbol{C}_{i,j} \boldsymbol{A}_{i,j}$$
$$\text{subject to } \sum_{i=1}^{H} \boldsymbol{A}_{i,j} = 1, \sum_{j=1}^{T} \boldsymbol{A}_{i,j} \leq 1, j \in \{1..T\}, i \in \{1..H\} \tag{5.1}$$

To solve the above optimal association problem, the existing Hungarian algorithm [119, 120] is applied.

(2) *Association Matrix Calculation:* To evaluate the similarity between the i^{th} predicted box and the j^{th} ground truth box, a simple and intuitive criterion is the Euclidean distance between two pairs of min-max vertices. However, it is not optimal. In essence, we want the predicted box to include as many valid points as possible. As illustrated in Figure 5.6, the input point cloud is usually sparse and distributed non-uniformly in 3D space. With regards to the ground truth box #0 (blue), the candidate box #2 (red) is believed to be much better than the candidate #1 (black), because the box #2 has more valid points overlapped with #0. Therefore, the coverage of valid points should also be included to calculate the cost matrix $\boldsymbol{C}$. In this chapter, we consider the following three criteria:

- Euclidean Distance between Vertices. Formally, the cost between the i^{th} predicted box $\boldsymbol{B}_i$ and the j^{th} ground truth box $\bar{\boldsymbol{B}}_j$ is calculated as follows:

$$\boldsymbol{C}_{i,j}^{ed} = \frac{1}{6} \sum (\boldsymbol{B}_i - \bar{\boldsymbol{B}}_j)^2 \tag{5.2}$$

- Soft Intersection-over-Union on Points. Given the input point cloud $\boldsymbol{P}$ and the j^{th} ground truth instance box $\bar{\boldsymbol{B}}_j$, it is able to directly obtain a hard-binary vector $\bar{\boldsymbol{q}}_j \in \mathbb{R}^N$ to represent whether each point of the whole input point cloud $\boldsymbol{P}$ is inside the box or not, where '1' indicates the point being inside and '0' outside. However, for a specific i^{th} predicted box of the same input point cloud $\boldsymbol{P}$, to directly obtain a similar hard-binary vector would result in the framework being non-differentiable, due to the discretization operation. Therefore, we introduce a differentiable yet simple Algorithm 2 to obtain a similar but soft-binary vector $\boldsymbol{q}_i$, called **point-in-pred-box-probability**, where all values are in the range $(0, 1)$. The deeper the corresponding point is inside of the box, the higher the value. The farther away the point is outside, the smaller the value. Formally,

the Soft Intersection-over-Union (sIoU) cost between the i^{th} predicted box and the j^{th} ground truth box is defined as follows:

$$\boldsymbol{C}_{i,j}^{sIoU} = \frac{-\sum_{n=1}^{N}(q_i^n * \bar{q}_j^n)}{\sum_{n=1}^{N} q_i^n + \sum_{n=1}^{N} \bar{q}_j^n - \sum_{n=1}^{N}(q_i^n * \bar{q}_j^n)} \tag{5.3}$$

where q_i^n and $\bar{q}_j^n$ are the n^{th} values of $\boldsymbol{q}_i$ and $\bar{\boldsymbol{q}}_j$, N is the total number of points in $\boldsymbol{P}$.

- Cross-Entropy Score. In addition, we also consider the cross-entropy score between $\boldsymbol{q}_i$ and $\bar{\boldsymbol{q}}_j$. Unlike sIoU cost which prefers tighter boxes, this score represents the confidence with which a predicted bounding box includes as many valid points as possible. It prefers larger and more inclusive boxes, and is formally defined as:

$$\boldsymbol{C}_{i,j}^{ces} = -\frac{1}{N}\sum_{n=1}^{N}\left[\bar{q}_j^n \log q_i^n + (1-\bar{q}_j^n)\log(1-q_i^n)\right] \tag{5.4}$$

where q_i^n and $\bar{q}_j^n$ are the n^{th} values of $\boldsymbol{q}_i$ and $\bar{\boldsymbol{q}}_j$, N is the total number of points in $\boldsymbol{P}$.

Overall, the criterion (1) guarantees the geometric boundaries for learnt boxes and criteria (2)(3) maximize the coverage of valid points and overcome the non-uniformity as illustrated in Figure 5.6. The final association cost between the i^{th} predicted box and the j^{th} ground truth box is defined as:

$$\boldsymbol{C}_{i,j} = \boldsymbol{C}_{i,j}^{ed} + \boldsymbol{C}_{i,j}^{sIoU} + \boldsymbol{C}_{i,j}^{ces} \tag{5.5}$$

Note that we assign an equal weight '1' to all the three criteria in Equation 5.5 for simplicity. Since different criteria tend to favor different densities of input point clouds, it would be ideal to learn a combination of the three criteria automatically. This is suggested as an interesting direction for future work.

Algorithm 2 An algorithm to calculate point-in-pred-box-probability. H is the number of predicted bounding boxes $\boldsymbol{B}$, N is the number of points in point cloud $\boldsymbol{P}$, θ_1 and θ_2 are hyperparameters for numerical stability. We use $\theta_1 = 100$, $\theta_2 = 20$ in all our implementation.

for $i \leftarrow 1$ to H **do**

- the i^{th} box min-vertex $\boldsymbol{B}^i_{min} = [x^i_{min}\ y^i_{min}\ z^i_{min}]$.
- the i^{th} box max-vertex $\boldsymbol{B}^i_{max} = [x^i_{max}\ y^i_{max}\ z^i_{max}]$.

for $n \leftarrow 1$ to N **do**

- the n^{th} point location $\boldsymbol{P}^n = [x^n\ y^n\ z^n]$.
- step 1: $\boldsymbol{\Delta}_{xyz} \leftarrow (\boldsymbol{B}^i_{min} - \boldsymbol{P}^n)(\boldsymbol{P}^n - \boldsymbol{B}^i_{max})$.
- step 2: $\boldsymbol{\Delta}_{xyz} \leftarrow max\left[min(\theta_1 \boldsymbol{\Delta}_{xyz}, \theta_2), -\theta_2\right]$.
- step 3: probability $\boldsymbol{p}_{xyz} = \frac{\mathbf{1}}{\mathbf{1}+\exp(-\boldsymbol{\Delta}_{xyz})}$.
- step 4: point probability $q^n_i = min(\boldsymbol{p}_{xyz})$.

- obtain the soft-binary vector $\boldsymbol{q}_i = [q^1_i \cdots q^N_i]$.

The above two loops are only for illustration. They are easily replaced by standard and efficient matrix operations.

5.3.4 Loss Functions

After the bounding box association layer, both the predicted boxes $\boldsymbol{B}$ and scores $\boldsymbol{B}_s$ are reordered using the association index $\boldsymbol{A}$, such that the first predicted T boxes and scores are well paired with the T ground truth boxes.

(1) Multi-criteria Loss for Box Prediction: The previous association layer finds the most similar predicted box for each ground truth box according to the minimal cost including: 1) vertex Euclidean distance, 2) sIoU cost on points, and 3) cross-entropy score. Therefore, the loss function for bounding box prediction is naturally designed to consistently minimize those cost. It is formally defined as follows:

$$\ell_{bbox} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}^{ed}_{t,t} + \boldsymbol{C}^{sIoU}_{t,t} + \boldsymbol{C}^{ces}_{t,t}) \tag{5.6}$$

where $\boldsymbol{C}^{ed}_{t,t}$, $\boldsymbol{C}^{sIoU}_{t,t}$ and $\boldsymbol{C}^{ces}_{t,t}$ are the cost of t^{th} paired boxes. Note that, we only minimize the cost of T paired boxes; the remaining $H - T$ predicted boxes are ignored because there is no corresponding ground truth for them. Therefore, this box prediction sub-branch is agnostic to the predefined value of H. Here however arises an issue. Since the $H - T$ negative predictions are not penalized, it might be possible that the network predicts multiple similar boxes for a single instance. Fortunately, the loss function for the parallel box score prediction is able to alleviate this problem.

(2) Loss for Box Score Prediction: The predicted box scores aim to indicate the validity of the corresponding predicted boxes. After being reordered by the association

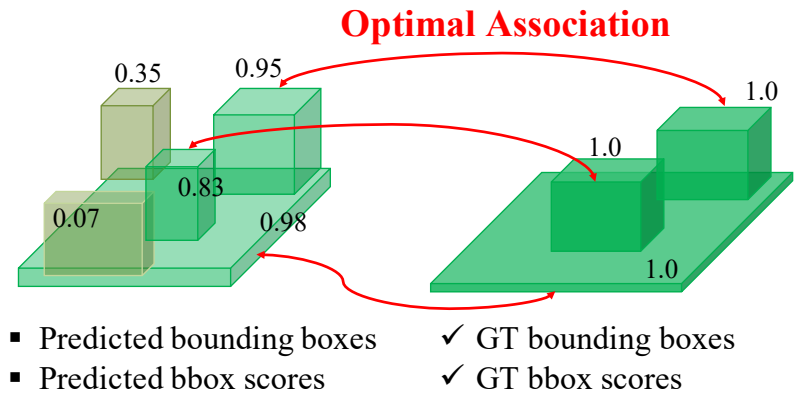


Figure 5.7: Illustration of the proposed bounding box association layer.

index $\boldsymbol{A}$, the ground truth scores for the first T scores are all '1', and '0' for the remaining invalid $H-T$ scores. We use cross-entropy loss for this binary classification task:

$$\ell_{bbs} = -\frac{1}{H}\left[\sum_{t=1}^{T} \log \boldsymbol{B}_s^t + \sum_{t=T+1}^{H} \log(1-\boldsymbol{B}_s^t)\right] \tag{5.7}$$

where $\boldsymbol{B}_s^t$ is the t^{th} predicted score after being associated. Basically, this loss function rewards the correctly predicted bounding boxes, while implicitly penalizing the cases where multiple similar boxes are regressed for a single instance.

Overall, Figure 5.7 illustrates the proposed bounding box association layer with the optimal assignment algorithm.

5.3.5 Gradient Estimation for Hungarian Algorithm

Given the predicted bounding boxes, $\mathbf{B}$, and ground-truth boxes, $\bar{\mathbf{B}}$, we compute the assignment cost matrix, $\mathbf{C}$. This matrix is converted to a permutation matrix, $\mathbf{A}$, using the Hungarian algorithm. Here we focus on the euclidean distance component of the loss, $\mathbf{C}^{ed}$. The derivative of our loss component w.r.t the network parameters, θ, in matrix form is:

$$\frac{\partial \mathbf{C}^{ed}}{\partial \theta} = -2(\mathbf{AB} - \bar{\mathbf{B}})\left[\mathbf{A} + \frac{\partial \mathbf{A}}{\partial \mathbf{C}}\frac{\partial \mathbf{C}}{\partial \mathbf{B}}\mathbf{B}\right]^T \frac{\partial \mathbf{B}}{\partial \theta} \tag{5.8}$$

The components are easily computable except for $\frac{\partial \mathbf{A}}{\partial \mathbf{C}}$ which is the gradient of the permutation w.r.t the assignment cost matrix which is zero nearly everywhere. In our implementation, we found that the network is able to converge when setting this term to zero.

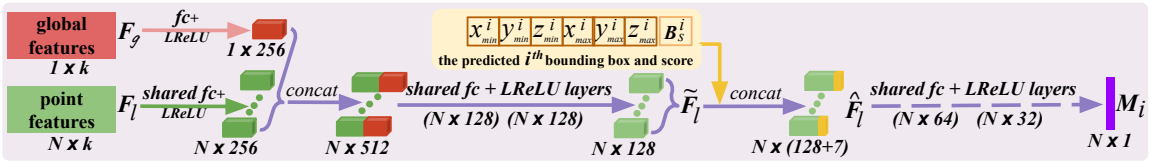


Figure 5.8: The architecture of point mask prediction branch. The point features are fused with each bounding box and score, after which a point-level binary mask is predicted for each instance.

However, convergence could be sped up using the straight-through-estimator [13], which assumes that the gradient of the rounding is identity (or a small constant), $\frac{\partial \mathbf{A}}{\partial \mathbf{C}} = \mathbb{1}$. This would speed up convergence as it allows both the error in the bounding box alignment (1st term of Eq. 5.8) to be backpropagated and the assignment to be reinforced (2nd term of Eq. 5.8). This approach has been shown to work well in practice for many problems including for differentiating through permutations for solving combinatorial optimization problems [50] and for training binary neural networks [309]. More complex approaches could also be used in our framework for computing the gradient of the assignment such as [69] which uses a Plackett-Luce distribution over permutations and a reparameterized gradient estimator.

5.4 Point Mask Prediction

Given the predicted bounding boxes $\boldsymbol{B}$, the learnt point features $\boldsymbol{F}_l$ and global features $\boldsymbol{F}_g$, the point mask prediction branch processes each bounding box individually with shared neural layers.

5.4.1 Neural Layers

As shown in Figure 5.8, both the point and global features are compressed to be 256 dimensional vectors through fully connected layers, before being concatenated and further compressed to be 128 dimensional mixed point features $\widetilde{\boldsymbol{F}}_l$. For the i^{th} predicted bounding box $\boldsymbol{B}_i$, the estimated vertices and score are fused with features $\widetilde{\boldsymbol{F}}_l$ through concatenation, producing box-aware features $\widehat{\boldsymbol{F}}_l$. These features are then fed through shared layers, predicting a point-level binary mask, denoted as $\boldsymbol{M}_i$. We use *sigmoid* as the final activation function. This simple box fusing approach is extremely computationally efficient, compared with the commonly used RoIAlign in prior art [308, 83, 77] which involves expensive point feature sampling and alignment.

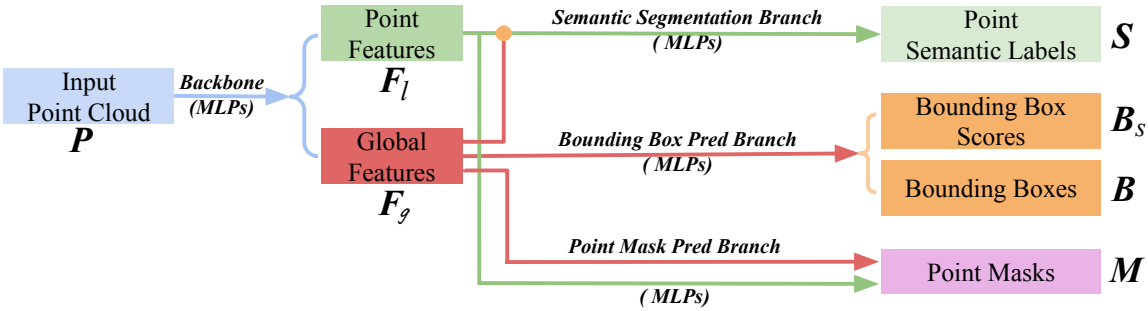


Figure 5.9: The end-to-end implementation for semantic segmentation, bounding box prediction and point mask prediction of 3D point clouds.

5.4.2 Loss Function

The predicted instance masks $\boldsymbol{M}$ are similarly associated with the ground truth masks according to the previous association index $\boldsymbol{A}$. Due to the imbalance of instance and background point numbers, we use focal loss [145] with default hyper-parameters instead of the standard cross-entropy loss to optimize this branch. Only the valid T paired masks are used for the loss ℓ_{pmask}. It is defined as follows:

$$\ell_{pmask} = -\frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}\left[\alpha(1-\boldsymbol{M}_t^n)^{\gamma}\bar{\boldsymbol{M}}_t^n \log \boldsymbol{M}_t^n + (1-\alpha)(\boldsymbol{M}_t^n)^{\gamma}(1-\bar{\boldsymbol{M}}_t^n)\log(1-\boldsymbol{M}_t^n)\right] \tag{5.9}$$

where $\boldsymbol{M}_t^n$, $\bar{\boldsymbol{M}}_t^n$ are the predicted and ground truth point-level mask values for the t^{th} paired instances. We use the default 0.75 and 2 for α and γ in all our experiments.

5.5 Implementation

While our framework is not restricted to a particular point cloud network, we adopt PointNet++ [208] as the backbone to learn the local and global features. In parallel, another separate branch is implemented to learn per-point semantics with the standard $softmax$ cross-entropy loss function ℓ_{sem}. Figure 5.9 shows the overall architecture for semantic segmentation, bounding box prediction and point mask prediction of 3D point clouds. The architecture of the backbone and semantic branch is the same as used in [270]. Note that our 3D-BoNet focuses on precisely segmenting individual objects and we do not aim to improve the recognition accuracy of 3D points and objects in this chapter. Therefore we leverage the existing semantic segmentation networks as a sub-branch in our end-to-end framework to estimate the categories

Table 5.1: Instance segmentation results on ScanNet(v2) benchmark (hidden test set). The metric is AP(%) with IoU threshold 0.5. Accessed on 2 June 2019.

	mean	btub	bed	bkshelf	cab	chair	counter	curt	desk	door	other	pic	refrig	shower	Cur	sink	sofa	table	toilet	window
MaskRCNN [77]	5.8	33.3	0.2	0.0	5.3	0.2	0.2	2.1	0.0	4.5	2.4	23.8	6.5	0.0	1.4	10.7	2.0	11.0	0.6	
SGPN [270]	14.3	20.8	39.0	16.9	6.5	27.5	2.9	6.9	0.0	8.7	4.3	1.4	2.7	0.0	11.2	35.1	16.8	43.8	13.8	
3D-BEVIS [49]	24.8	66.7	56.6	7.6	3.5	39.4	2.7	3.5	9.8	9.9	3.0	2.5	9.8	37.5	12.6	60.4	18.1	85.4	17.1	
R-PointNet [308]	30.6	50.0	40.5	31.1	34.8	58.9	5.4	6.8	12.6	28.3	29.0	2.8	21.9	21.4	33.1	39.6	27.5	82.1	24.5	
UNet-Backbone [141]	31.9	66.7	71.5	23.3	18.9	47.9	0.8	21.8	6.7	20.1	17.3	10.7	12.3	43.8	15.0	61.5	35.5	91.6	9.3	
3D-SIS (5 views) [83]	38.2	**100.0**	43.2	24.5	19.0	57.7	1.3	26.3	3.3	32.0	24.0	7.5	42.2	85.7	11.7	**69.9**	27.1	88.3	23.5	
MASC [151]	44.7	52.8	55.5	38.1	**38.2**	63.3	0.2	50.9	26.0	36.1	43.2	32.7	**45.1**	57.1	36.7	63.9	38.6	**98.0**	27.6	
ResNet-Backbone [141]	45.9	**100.0**	**73.7**	15.9	25.9	58.7	**13.8**	47.5	21.7	**41.6**	40.8	12.8	31.5	71.4	41.1	53.6	**59.0**	87.3	30.4	
PanopticFusion [185]	47.8	66.7	71.2	**59.5**	25.9	55.0	0.0	61.3	17.5	25.0	**43.4**	**43.7**	41.1	85.7	**48.5**	59.1	26.7	94.4	35.9	
MTML	48.1	**100.0**	66.6	37.7	27.2	**70.9**	0.1	57.9	25.4	36.1	31.8	9.5	43.2	**100.0**	18.4	60.1	48.7	93.8	38.4	
3D-BoNet(Ours)	**48.8**	**100.0**	67.2	59.0	30.1	48.4	9.8	**62.0**	**30.6**	34.1	25.9	12.5	43.4	79.6	40.2	49.9	51.3	90.9	**43.9**	

of segmented objects. Given an input point cloud $\boldsymbol{P}$, the above three branches are linked and end-to-end trained using a single combined multi-task loss:

$$\ell_{all} = \ell_{sem} + \ell_{bbox} + \ell_{bbs} + \ell_{pmask} \tag{5.10}$$

We use Adam solver [110] with its default hyper-parameters for optimization. Initial learning rate is set to $5e^{-4}$ and then divided by 2 every 20 epochs. The whole network is trained on a Titan X GPU from scratch. We use the same settings for all experiments, which guarantees the reproducibility of our framework.

5.6 Experiments

5.6.1 Evaluation on ScanNet

We first evaluate our approach on ScanNet(v2) 3D semantic instance segmentation benchmark [39]. ScanNet(v2) consists of 1613 complete 3D scenes acquired from real-world indoor spaces. The official split has 1201 training scenes, 312 validation scenes and 100 hidden testing scenes. The original large point clouds are divided into $1m \times 1m$ blocks with $0.5m$ overlapped between neighbouring blocks. This data preprocessing step is the same as the one used by PointNet [206] and SGPN [270] for the S3DIS dataset. We sample 4096 points from each block for training, but use all points of a block for testing followed by the BlockMerging algorithm [270] to assemble blocks into complete 3D scenes. Each point is represented by a 9D vector *(normalized xyz in the block, RGB, normalized xyz in the room)*. H is set as 20 in our experiments. In our experiment, we observe that the performance of the vanilla PointNet++ based semantic prediction sub-branch is limited and unable to provide satisfactory semantics. Thanks to the flexibility of our framework, we therefore easily train a parallel SCN network [66] to estimate more accurate per-point semantic labels for the predicted instances of our 3D-BoNet. The average precision (AP) with an IoU threshold 0.5 is used as the evaluation metric.

We compare with the leading approaches on 18 object categories in Table 5.1. Particularly, SGPN [270], 3D-BEVIS [49], MASC [151] and [141] are point feature clustering based approaches; R-PointNet [308] learns to generate dense object proposals followed by point-level segmentation; 3D-SIS [83] is a proposal-based approach using both point clouds and color images as input. PanopticFusion [185] learns to segment instances on multiple 2D images by Mask-RCNN [77] and then uses the SLAM system to reproject back to 3D space. Our approach surpasses them all using point clouds only. Remarkably, our framework offers satisfactory performance on all categories without preferring specific classes.

Figure 5.10 shows qualitative results of our 3D-BoNet for instance segmentation on ScanNet validation split. It can be seen that our approach tends to predict complete object instances, instead of inferring tiny and but invalid fragments. This demonstrates that our framework indeed guarantees high objectness for segmented instances. The red circles showcase the failure cases, where the very similar instances are unable to be well segmented by our approach.

5.6.2 Evaluation on S3DIS

We further evaluate the semantic instance segmentation of our framework on S3DIS [6], which consists of 3D complete scans from 271 rooms belonging to 6 large areas. Our data preprocessing and experimental settings strictly follow PointNet [206], SGPN [270], ASIS [271], and JSIS3D [202]. In particular, the original large point clouds are divided into $1m \times 1m$ blocks with $0.5m$ overlapped between neighbouring blocks. We sample 4096 points from each block for training, but use all points of a block for testing. Each point is represented by a 9D vector *(normalized xyz in the block, rgb, normalized xyz in the room)*. In our experiments, H is set as 24 and we follow the 6-fold evaluation [6, 271]. We train our 3D-BoNet to predict object bounding boxes and point-level masks, and in parallel train a vanilla PointNet++ based sub-branch to predict point-level semantic labels. Particularly, all the semantic, bounding box and point mask sub-branches share the same PointNet++ backbone to extract point features, and are end-to-end trained from scratch.

We compare the proposed approach with ASIS [271], the state of art on S3DIS, and the PartNet baseline [176]. For fair comparison, we carefully train the PartNet baseline with the same PointNet++ backbone and other settings as used in our framework. For evaluation, the classical metrics mean precision (mPrec) and mean recall (mRec) with IoU threshold 0.5 are reported. Note that, we use the same Block-Merging algorithm [270] to merge the instances from different blocks for both our

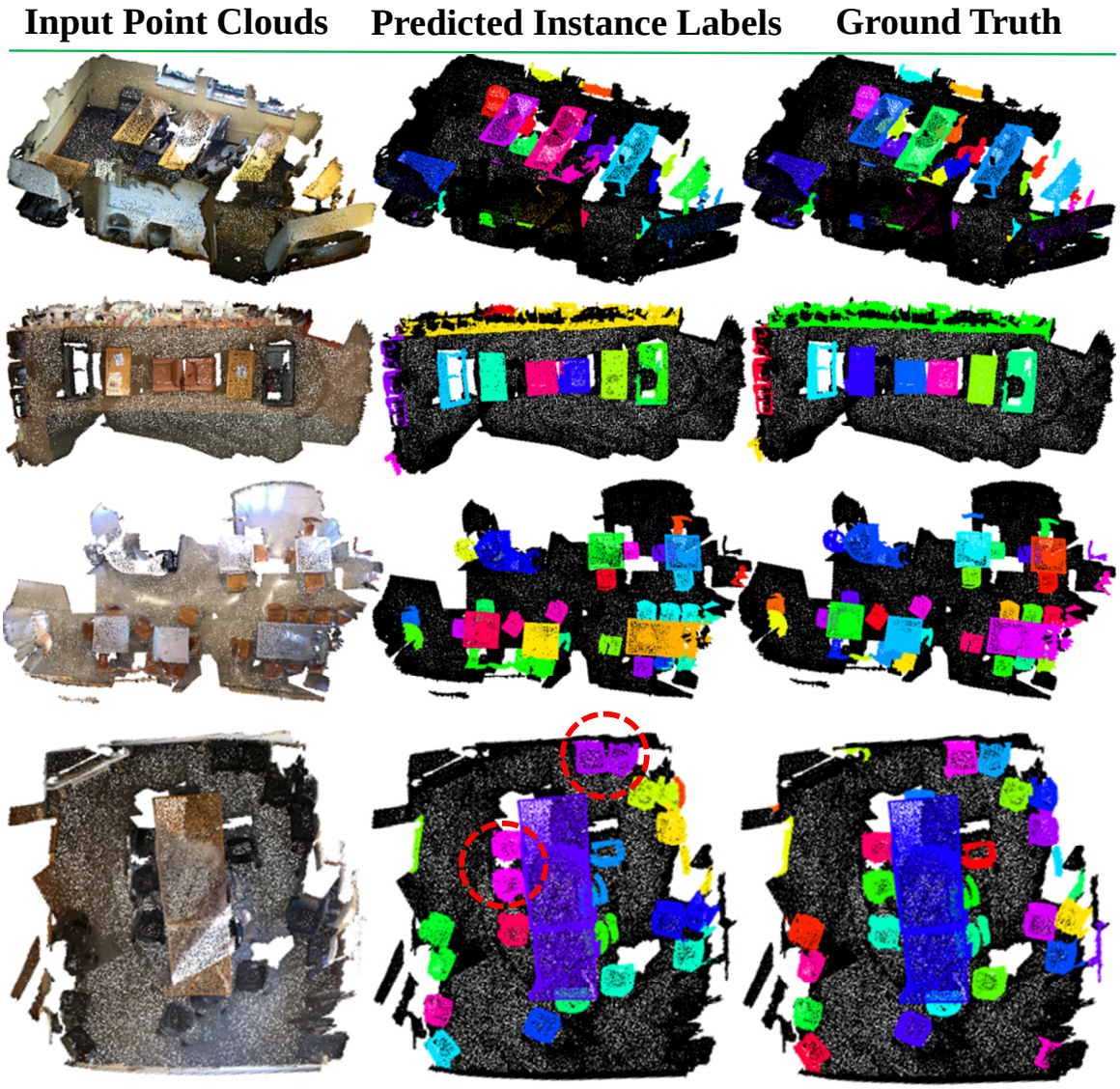


Figure 5.10: Qualitative results of our approach for instance segmentation on ScanNet(v2) validation split. Black points are not of interest as they do not belong to any of the 18 object categories.

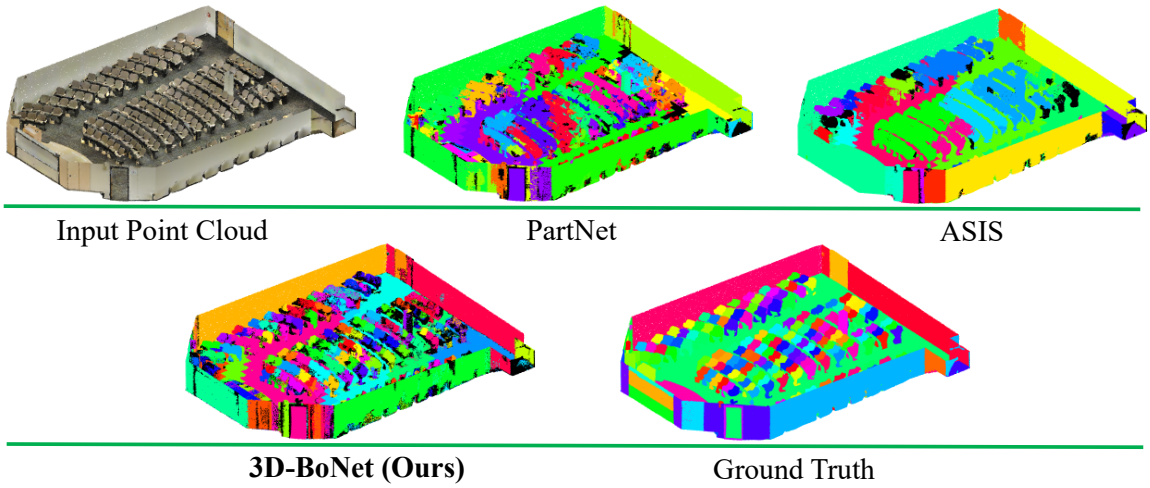


Figure 5.11: This shows a lecture room with hundreds of objects (*e.g.*, chairs, tables), highlighting the challenge of instance segmentation. Different colors indicates different instances. Our framework predicts more precise instance labels than other techniques.

Table 5.2: Instance segmentation results on S3DIS dataset.

	mPrec	mRec
PartNet [176]	56.4	43.4
ASIS [271]	63.6	47.5
3D-BoNet (Ours)	**65.6**	**47.6**

approach and the PartNet baseline. The final scores are averaged across the total 13 categories. Table 5.2 presents the mPrec/mRec scores and Figure 5.11 shows qualitative results. Our method surpasses PartNet baseline [176] by large margins, and also outperforms ASIS [271], but not significantly, mainly because our semantic prediction branch (vanilla PointNet++ based) is inferior to ASIS which tightly fuses semantic and instance features for mutual optimization. We leave the feature fusion as our future exploration.

Figure 5.12 shows the training curves of our proposed loss functions on Areas (1,2,3,4,6) of S3DIS dataset. It demonstrates that all the proposed loss functions are able to converge consistently, thus jointly optimizing the semantic segmentation, bounding box prediction, and point mask prediction branches in an end-to-end fashion.

Figure 5.13 presents the qualitative results of predicted bounding boxes and scores. It can be seen that the predicted boxes are not necessarily tight and precise. Instead, they tend to be inclusive but with high objectness. Fundamentally, this highlights the simple but effective concept of our bounding box prediction network. Given these

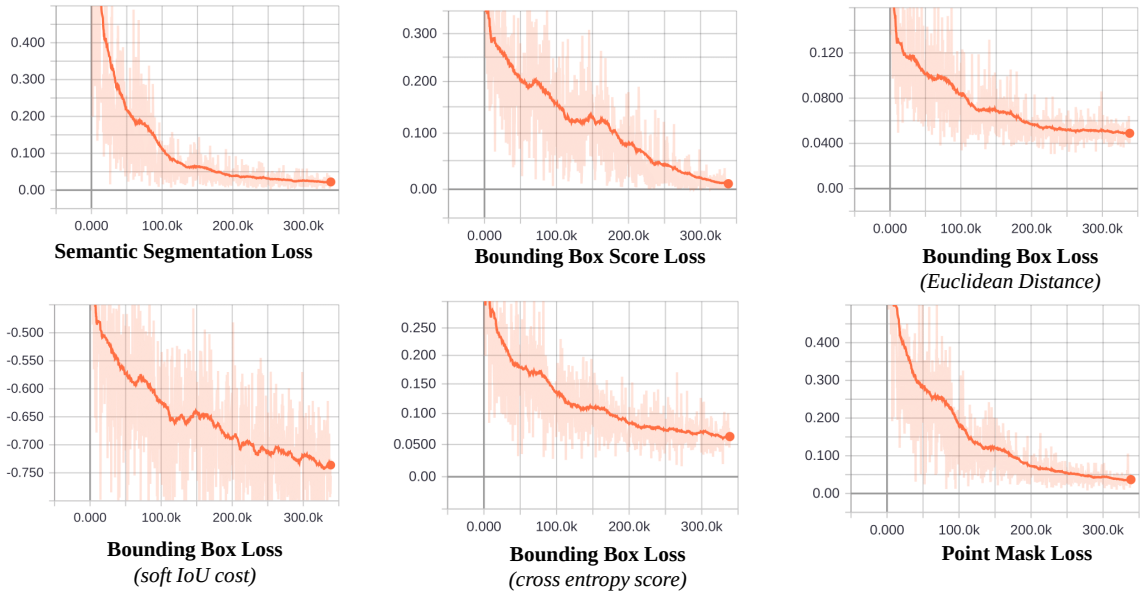


Figure 5.12: Training losses on S3DIS Areas (1,2,3,4,6).

bounded points, it is extremely easy to segment the instance inside.

Figure 5.14 visualizes the predicted instance masks, where the black points have ~ 0 probability and the brighter points have ~ 1 probability to be an instance within each predicted mask. In particular, the predicted masks #1 $\sim$ #4 show the results of point mask branch from 4 predicted bounding boxes. It can be seen that both the table (blue) and the floor (purple) are clearly segmented, while there is ambiguity between the predicted two chairs (green and red masks).

5.6.3 Generalization to Unseen Scenes and Categories

Our framework learns the object bounding boxes and point masks from raw point clouds without direct coupling to the semantic information, which inherently allows for generalization to new categories and scenes. We conduct further experiments to qualitatively demonstrate the generality of our framework. In particular, we use the well-trained model from S3DIS dataset (Areas 1/2/3/4/6) to directly test on the validation split of ScanNet(v2) dataset. Since the ScanNet dataset consists of many more object categories than S3DIS dataset, there are a number of categories (*e.g.*, toilet, desk, sink, bathtub) that the trained model has never seen before.

As shown in Figure 5.15, our model is still able to predict high-quality instance labels even though the scenes and some object categories have not been seen before. This shows that our model does not simply fit the training dataset. Instead, it tends

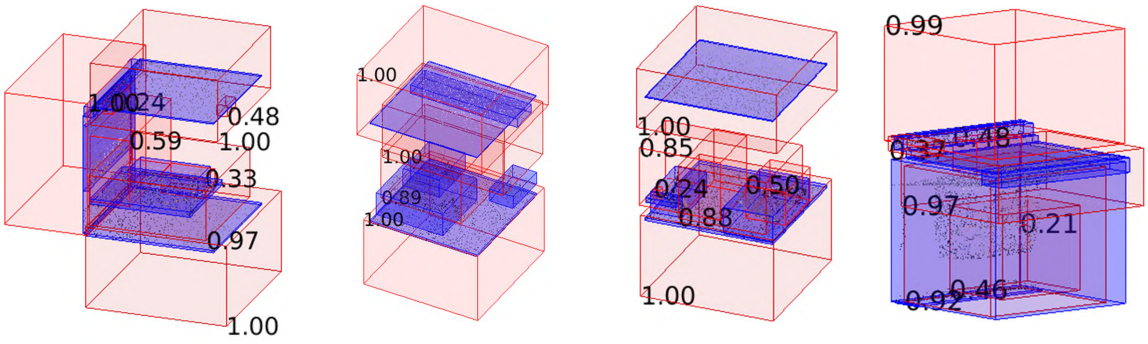


Figure 5.13: Qualitative results of predicted bounding boxes and scores on S3DIS Area 2. The point clouds inside of the blue boxes are fed into our framework which then estimates the red boxes to roughly detect instances. The tight blue boxes are the ground truth.

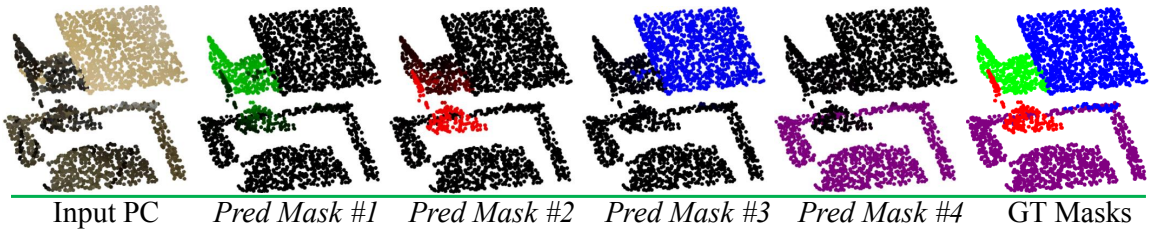


Figure 5.14: Qualitative results of predicted instance masks.

to learn the underlying geometric features which are able to be generalized across new objects and scenes.

5.6.4 Ablation Study

To evaluate the effectiveness of each component of our framework, we conduct 6 groups of ablation experiments on the largest Area 5 of S3DIS dataset.

(1) **Remove Box Score Prediction Sub-branch.** Basically, the box score

Table 5.3: Instance segmentation results of all ablation experiments on Area 5 of S3DIS.

	mPrec	mRec
(1) Remove Box Score Sub-branch	50.9	40.9
(2) Euclidean Distance Only	53.8	**41.1**
(3) Soft IoU Cost Only	55.2	40.6
(4) Cross-Entropy Score Only	51.8	37.8
(5) Do Not Supervise Box Prediction	37.3	28.5
(6) Remove Focal Loss	50.8	39.2
(7) The Full Framework	**57.5**	40.2

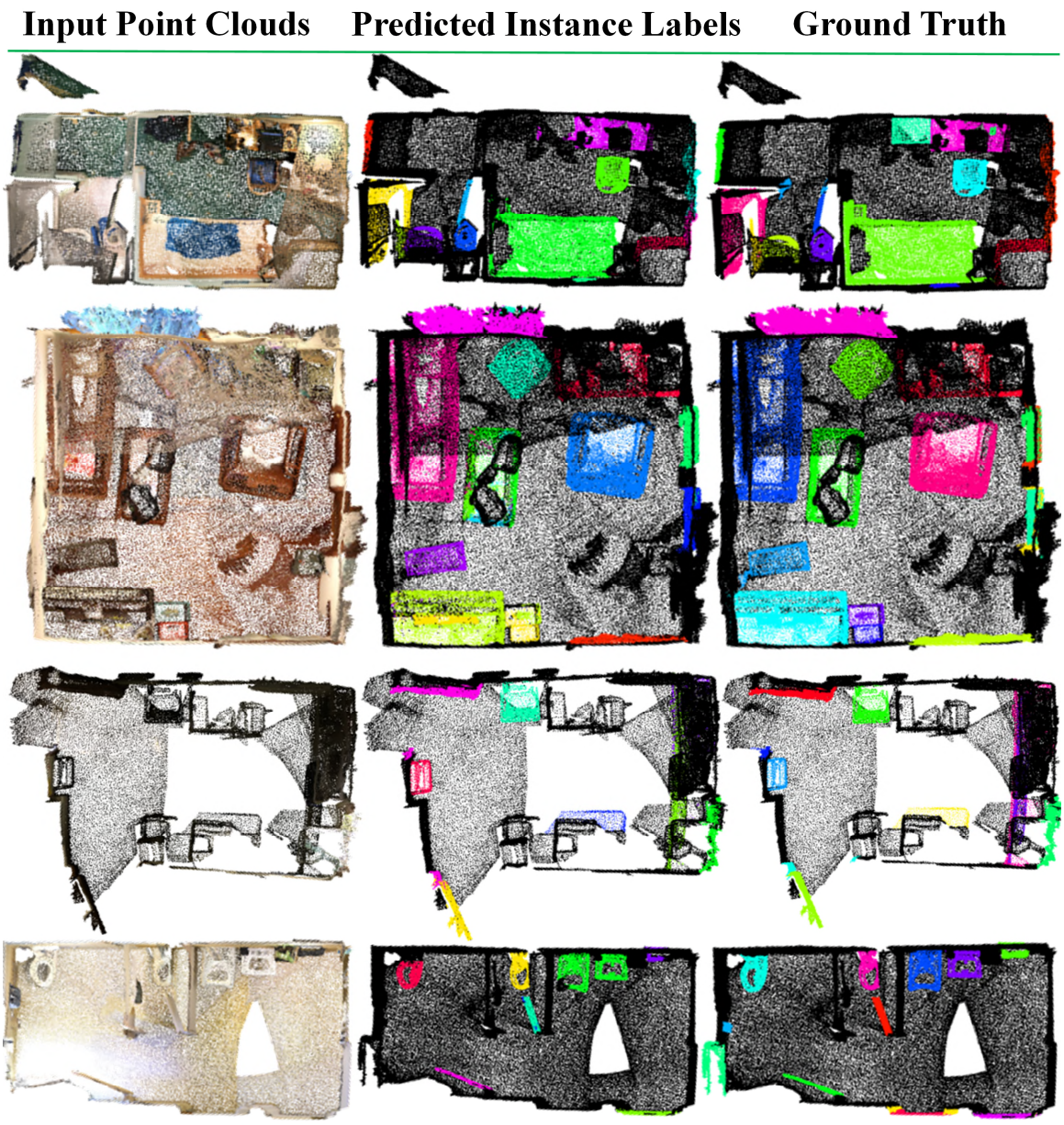


Figure 5.15: Qualitative results of instance segmentation on ScanNet dataset. Although the model is trained on S3DIS dataset and then directly tested on ScanNet validation split, it is still able to predict high-quality instance labels.

serves as an indicator and regularizer for valid bounding box prediction. After removing it, we train the network with:

$$\ell_{ab1} = \ell_{sem} + \ell_{bbox} + \ell_{pmask}$$

Initially, the multi-criteria loss function is a simple unweighted combination of the Euclidean distance, the soft IoU cost, and the cross-entropy score. However, this may not be optimal, because the density of input point clouds is usually inconsistent and tends to prefer different criteria. We conduct 3 groups of experiments on ablated bounding box loss functions.

(2) **Use Single Criterion: Euclidean Distance.** Only the Euclidean Distance criterion is used for the box association and loss ℓ_{bbox}.

$$\ell_{ab2} = \ell_{sem} + \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{C}_{t,t}^{ed} + \ell_{bbs} + \ell_{pmask}$$

(3) **Use Single Criterion: Soft IoU.** Only the Soft IoU criterion is used for the box association and loss ℓ_{bbox}.

$$\ell_{ab3} = \ell_{sem} + \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{C}_{t,t}^{sIoU} + \ell_{bbs} + \ell_{pmask}$$

(4) **Use Single Criterion: Cross-Entropy Score.** Only the Cross-Entropy Score criterion is used for the box association and loss ℓ_{bbox}.

$$\ell_{ab4} = \ell_{sem} + \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{C}_{t,t}^{ces} + \ell_{bbs} + \ell_{pmask}$$

(5) **Do Not Supervise Box Prediction.** The predicted boxes are still associated according to the three criteria, but we remove the box supervision signal. The framework is trained with:

$$\ell_{ab5} = \ell_{sem} + \ell_{bbs} + \ell_{pmask}$$

(6) **Remove Focal Loss for Point Mask Prediction.** In the point mask prediction branch, the focal loss is replaced by the standard cross-entropy loss for comparison.

Analysis. Table 5.3 shows the scores for ablation experiments.

- The box score sub-branch indeed benefits the overall instance segmentation performance, as it tends to penalize duplicated box predictions.

- Compared with Euclidean distance and cross-entropy score, the sIoU cost tends to be better for box association and supervision, thanks to our differentiable Algorithm 2. As the three individual criteria prefer different types of point structures, a simple combination of three criteria may not always be optimal on a specific dataset.

- Without the supervision for box prediction, the performance drops significantly, primarily because the network is unable to infer satisfactory instance 3D boundaries and the quality of predicted point masks deteriorates accordingly.

- Compared with focal loss, the standard cross entropy loss is less effective for point mask prediction due to the imbalance of instance and background point numbers.

5.6.5 Computation Analysis

The computation complexity of all existing approaches are analysed as follows:

(1) For point feature clustering based approaches including SGPN [270], ASIS [271], JSIS3D [202], 3D-BEVIS [49], MASC [151], and [141], the computation complexity of the post clustering algorithm such as Mean Shift [35] tends towards $\mathcal{O}(TN^2)$, where T is the number of instances and N is the number of input points.

(2) For dense proposal-based methods including GSPN [308], 3D-SIS [83] and PanopticFusion [185], region proposal network and non-maximum suppression are usually required to generate and prune the dense proposals, which is computationally expensive [185].

(3) Both PartNet baseline [176] and our 3D-BoNet have similar efficient computation complexity $\mathcal{O}(N)$. Empirically, our 3D-BoNet takes around 20 ms GPU time to process $4k$ points, while most approaches in (1)(2) need more than 200ms GPU/CPU time to process the same number of points.

Table 5.4 compares the time consumption of four existing approaches using their released codes on the validation split (312 scenes) of ScanNet(v2) dataset. Normally, each scene has dozens of objects with around 80k points spanning up to $10 \times 20 \times 5$ meters in 3D space. SGPN [270], ASIS [271], GSPN [308] and our 3D-BoNet are implemented by Tensorflow 1.4, 3D-SIS [83] by Pytorch 0.4. All approaches are running on a single Titan X GPU and the pre/post-processing steps on an i7 CPU core with a single thread. Note that 3D-SIS automatically uses CPU for computing when some large scenes are unable to be processed by the single GPU.

It can be seen that our approach is much more computationally efficient than existing methods, and is up to 20× faster than ASIS [271]. However, the majority of time spent by our 3D-BoNet is within the *block merging* step, which is used to assemble the partitioned blocks of point clouds back to large-scale scenes. Fundamentally, this is because the backbone network used in our framework, *i.e.*, PointNet++, is unable to take large-scale point clouds as input due to its expensive sampling operations. In this regard, the efficiency of our framework can be further improved if a more advanced backbone network is available to process large-scale point clouds. We leave it as our future exploration.

Table 5.4: Time consumption of different approaches on the validation split (312 scenes) of ScanNet(v2) (seconds).

	SGPN [270]	ASIS [271]	GSPN [308]	3D-SIS [83]	**3D-BoNet(Ours)**
	network(GPU): 650 group merge(CPU): 46562 block merge(CPU): 2221	network(GPU): 650 mean shift(CPU): 53886 block merge(CPU): 2221	network(GPU): 500 point sampling(GPU): 2995 neighbour search(CPU): 468	voxelization, projection, network.. (GPU+CPU): 38841	network(GPU): 650 *SCN (GPU parallel): 208* block merge(CPU): 2221
total	49433	56757	3963	38841	**2871**

5.7 Conclusion

In this chapter, we proposed a simple, effective and efficient framework for instance segmentation on 3D point clouds. The framework consists of a bounding box branch and a point mask prediction branch. The bounding box branch directly regresses a set of 3D bounding boxes to roughly detect all object instances, whereas the point mask branch focuses on each predicted bounding box to classify whether each 3D point belongs to the foreground instance or the background clutter. By training the branches in an end-to-end fashion, we obtain the instance segmentation results in a single forward pass. Compared with all existing works, our framework only consists of feed-forward MLPs without requiring any heavy post-processing steps such as non-maximum-suppression or feature clustering, therefore being lightweight and efficient.

However, it also has some limitations which open up new directions to future work. (1) Instead of using unweighted combination of three criteria, it could be better to design a module to automatically learn the weights, in order to be able to adapt to different types of input point clouds. (2) Instead of training a separate branch for semantic prediction, more advanced feature fusion modules can be introduced to mutually improve both semantic and instance segmentation. (3) Our framework follows the MLP design and is therefore agnostic to the number and order of input points. It is desirable to directly train and test on large-scale input point clouds instead of the divided small blocks, by drawing on recent work [52][127].

Chapter 6

Conclusion and Future Work

6.1 Summary of Key Contributions

The overarching purpose of this thesis has been to build intelligent systems that understand the geometric structure and semantics of the 3D real-world environments. To achieve this goal, we proposed solutions from the object level to the more complex scene level.

In Chapter 3, we introduced an encoder-decoder architecture together with adversarial training to accurately estimate dense 3D shape of objects from a single depth view. To enable the encoder-decoder to learn more fine-grained geometric details, we integrated an adversarial component conditioned on the input single view. However, the adversarial component is hard to train due to the extremely high-dimensional 3D data. To stabilise the end-to-end training procedure, we proposed a simple mean feature neural layer to discriminate the real-world 3D shapes and the predicted ones, allowing the entire network to recover more plausible 3D structures. Since the existing datasets are based on 3D CAD models and synthesized images, we therefore collected a reasonable amount of real-world datasets using the Kinect device, demonstrating strong generalization of our approach from simulated data to real-world noisy environments. Compared with existing methods, our approach is distinguished in the following ways: 1) it estimates dense 3D shapes within high-resolution voxel grids (*i.e.*, 256^3) which represent compelling geometric details. 2) Thanks to the adversarial training, the estimated 3D shapes tend to be more realistic because the priors of object shapes can be well learnt and propagated from our stable discriminator.

In Chapter 4, we aimed to reconstruct a better 3D shape of an object from multiple views instead of merely a single view. Multiple views naturally provide much more valuable information to infer the 3D structure. However, to effectively integrate this valuable information is not easy. In this chapter, we introduced an attentive

aggregation module to selectively fuse the deep visual features for precise 3D shape estimation from multiple views. By formulating the multi-view 3D reconstruction problem as an aggregation step for a set of deep features, our attention based method learns an attention score for each visual feature of all multiple input views. These scores are then used to weigh the corresponding features. After that, all the weighted features are summed and integrated across multiple views, generating the final 3D shapes. Fundamentally, the proposed module serves as a mask to filter out the less informative deep features, whereas preserving the relative important information according to the input multiple views. However, we observed that a naive end-to-end training strategy would result in the whole network not being robust to an arbitrary number of input images. To overcome this problem, we theoretically investigated the principle underlying this issue and presented a two-stage training algorithm. Our algorithm separately optimizes the base encoder-decoder and the proposed attention module, achieving superior performance at reconstructing 3D shapes. Compared with existing RNN based methods, our approach is permutation invariant and estimates consist of 3D shapes regardless of the different orderings of input views. In contrast to the max/mean/sum poolings which usually ignore the majority of information from multiple images, our module learns to attentively preserve useful features and infers better 3D shapes. Unlike the existing attention based methods which only operate on a fixed number of input images, our module uses a novel training strategy, forcing the robustness of predicted 3D shapes given an arbitrary number of input images.

In Chapter 5, we extended object-level perception to scene-level understanding. In real-world scenarios, the 3D scenes are usually cluttered and include multiple objects. In this chapter, we proposed an efficient neural pipeline to identify and segment all individual 3D objects from real-world and large-scale point clouds. In contrast to segmenting 2D images, to identify objects in 3D space is extremely challenging, due to the irregular and incomplete statistics of 3D point clouds. We solved this difficult task by directly regressing a set of bounding boxes to roughly detect individual 3D objects and then segmenting each object within its box. By leveraging pioneering work on neural networks for 3D point clouds processing, we designed a bounding box prediction branch and a point mask prediction branch based on the the learnt per-point features and global features. The bounding box prediction branch directly regresses a set of 3D boxes to detect all objects, whereas the point mask prediction branch serves to classify whether each 3D point belongs to the foreground object instance or the background clutter. To supervise the bounding box prediction branch, we drew on the seminal work on data association and carefully designed loss functions to train the

entire network in an end-to-end fashion. The proposed neural pipeline demonstrates accurate and efficient segmentation results on multiple large-scale real-world datasets. Compared with the existing proposal-free approaches, our framework does not require computationally expensive algorithms to cluster point features, and the explicitly predicted bounding boxes guarantee high objectness for the final segmented instances. In contrast to the existing proposal-based methods, our pipeline does not rely on spatial anchors to generate numerous candidate bounding boxes, and thus does not require computationally heavy post-processing steps such as non-maximum-suppression.

6.2 Limitations and Future Work

The work in this thesis has opened up a number of new directions for future work:

Using 2D Supervision. The proposed architecture in this thesis mainly relies on ground truth 3D data for supervision. However, it is labour-intensive to acquire large-scale 3D labels. In addition, the learnt representations tend to be fitted to specific 3D datasets and are unlikely to generalize to completely unseen scenarios due to domain gaps. An alternative approach is to leverage widely available 2D images as supervision signal. Unlike 3D supervision, 2D images inherently have weaker prior knowledge. Nevertheless, the explicit geometric consistency would provide valuable information to supervise the networks.

Better 3D Representation. The widely used voxel grid requires extremely heavy computation and memory if being used to represent large-scale 3D scenes. Point clouds are efficient to represent complex 3D structures, but they are unable to present the object surface which is useful for high-level applications such as rendering and grasping. By contrast, 3D meshes have been recently integrated into deep neural networks and emerge as an efficient and promising approach to shape representation.

Disentanglement of 3D Scenes. The proposed approaches in this thesis usually learn a singular representation for an object or a scene, without decomposing the shape components explicitly. However, learning to factorize the objects or scenes into more detailed elements would be desirable, as it allows more robustness and generalization across different scenarios.

In conclusion, this thesis has taken a step towards 3D scene understanding by learning to reconstruct and segment 3D objects. We hope it can inspire researchers to develop more advanced systems that would endow machines with the ability to perceive and interact with our environment, thus benefiting us humans in a variety of real-world applications.

Bibliography

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a Day. *IEEE International Conference on Computer Vision*, pages 105–112, 2009.

[2] Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab. YOLO3D : End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud. *European Conference on Computer Vision Workshops*, 2018.

[3] Kosuke Arase, Yusuke Mukuta, and Tatsuya Harada. Rethinking Task and Metrics of Instance Segmentation on 3D Point Clouds. *IEEE International Conference on Computer Vision Workshops*, 2019.

[4] Martin Arjovsky and Leon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *International Conference on Learning Representations*, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. *International Conference on Machine Learning*, pages 214–223, 2017.

[6] Iro Armeni, Ozan Sener, AR Zamir, and Helen Jiang. 3D Semantic Parsing of Large-Scale Indoor Spaces. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[7] Alireza Asvadi, Luis Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano J. Nunes. DepthCN: Vehicle detection using 3D-LIDAR and ConvNet. *IEEE International Conference on Intelligent Transportation Systems*, pages 1–6, 2017.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations*, 2015.

[9] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training. *IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[10] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. *IEEE International Conference on Computer Vision*, pages 9297–9307, 2019.

[11] Jorge Beltran, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo de la Escalera. BirdNet: a 3D Object Detection Framework from LiDAR Information. *IEEE International Conference on Intelligent Transportation Systems*, pages 3517–3523, 2018.

[12] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.

[14] V. Blanz and T.Vetter. Face Recognition based on Fitting a 3D Morphable Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, 2003.

[15] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. CodeSLAM - Learning a Compact, Optimisable Representation for DenseVisual SLAM. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2560–2568, 2018.

[16] Alexandre Boulch. ConvPoint: Continuous Convolutions for Point Cloud Processing. *Computers & Graphics*, 2020.

[17] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. *Conference on Neural Information Processing Systems Workshops*, 2016.

[18] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian D. Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[19] Yan-Pei Cao, Zheng-Ning Liu, Zheng-Fei Kuang, Leif Kobbelt, and Shi-Min Hu. Learning to Reconstruct High-quality 3D Shapes with Cascaded Fully Convolutional Networks. *European Conference on Computer Vision*, pages 616–633, 2018.

[20] Geonho Cha, Minsik Lee, and Songhwai Oh. Unsupervised 3D Reconstruction Networks. *International Conference on Computer Vision*, pages 3849–3858, 2019.

[21] 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. *arXiv preprint arXiv:1512.03012*, 2015.

[22] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. ClusterNet: Deep Hierarchical Cluster Network with Rigorously Rotation-Invariant Representation for Point Cloud Analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019.

[23] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots. *arXiv:1912.12791*, 2019.

[24] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-Based Multi-View Stereo Network. *IEEE International Conference on Computer Vision*, pages 1538–1547, 2019.

[25] Siheng Chen, Sufeng Niu, Tian Lan, and Baoan Liu. PCT: Large-Scale 3D Point Cloud Representations via Graph Inception Networks with Applications to Autonomous Driving. *IEEE International Conference on Image Processing*, pages 4395–4399, 2019.

[26] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[27] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-View 3D Object Detection Network for Autonomous Driving. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[28] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast Point R-CNN. *IEEE International Conference on Computer Vision*, pages 9775–9784, 2019.

[29] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[30] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[31] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[32] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *European Conference on Computer Vision*, pages 628–644, 2016.

[33] H Christian, Shubham Tulsiani, and Jitendra Malik. Hierarchical Surface Prediction for 3D Object Reconstruction. *International Conference on 3D Vision*, pages 412–420, 2017.

[34] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

[35] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[36] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous Optimization for Large-scale Structure from Motion. *IEEE Conference on Computer Vision and Pattern Recognition*, page 3001–3008, 2011.

[37] Brian Curless and Marc Levoy. A Volumetric Method for Building Complex Models from Range Images. *Conference on Computer Graphics and Interactive Techniques*, pages 303–312, 1996.

[38] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J. Davison. DeepFactors: Real-Time Probabilistic DenseMonocular SLAM. *IEEE Robotics and Automation Letters*, 2019.

[39] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[40] Angela Dai, Matthias Niessner, Michael Zollhofer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics*, 36(3), 2017.

[41] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[42] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Oliver Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[43] Amael Delaunoy and Marc Pollefeys. Photometric Bundle Adjustment for Dense Multi-View 3D Modeling. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1486–1493, 2014.

[44] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Learning Elementary Structures for 3D Shape Generation and Matching. *Advances in Neural Information Processing Systems*, pages 7433–7443, 2019.

[45] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing Audio with GANs. *International Conference on Learning Representations Workshops*, 2018.

[46] Wei Dong, Qiuyuan Wang, Xin Wang, and Hongbin Zha. PSDF Fusion: Probabilistic Signed Distance Function for On-the-fly 3D Data Fusion and Scene Reconstruction. *European Conference on Computer Vision*, pages 714–730, 2018.

[47] Pengfei Dou, Shishir K Shah, and Ioannis A Kakadiaris. End-to-end 3D Face Reconstruction with Deep Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5908–5917, 2017.

[48] Yueqi Duan, Yu Zheng, Jiwen Lu, Jie Zhou, and Qi Tian. Structural Relational Reasoning of Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 949–958, 2019.

[49] Cathrin Elich, Francis Engelmann, Jonas Schult, Theodora Kontogianni, and Bastian Leibe. 3D-BEVIS: Birds-Eye-View Instance Segmentation. *German Conference on Pattern Recognition*, pages 48–61, 2019.

[50] Patrick Emami and Sanjay Ranka. Learning Permutations with Sinkhorn Policy Gradient. *arXiv preprint arXiv:1805.07010*, 2018.

[51] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. *IEEE International Conference on Robotics and Automation*, pages 1355–1361, 2017.

[52] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds. *IEEE International Conference on Computer Vision Workshops*, pages 716–724, 2017.

[53] S.M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural Scene Representation and Rendering. *Science*, 360(6394):1204–1210, 2018.

[54] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, (88):303–338, 2010.

[55] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 605–613, 2017.

[56] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow. Structured Prediction of Unobserved Voxels From a Single Depth Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5431–5440, 2016.

[57] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building Rome on a Cloudless Day. *European Conference on Computer Vision*, pages 368–381, 2010.

[58] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3D Shape Induction from 2D Views of Multiple Objects. *International Conference on 3D Vision*, pages 402–411, 2017.

[59] Andrew Gardner, Jinko Kanno, Christian A. Duncan, and Rastko R. Selmic. Classifying Unordered Feature Sets with Convolutional Deep Averaging Networks. *IEEE International Conference on Systems, Man and Cybernetics*, pages 3447–3453, 2019.

[60] Riccardo Gherardi, Michela Farenzena, and Andrea Fusiello. Improving the Efficiency of Hierarchical Structure-and-motion. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1594–1600, 2010.

[61] Pallabi Ghosh, Larry S Davis, and Neel Joshi. Deep Depth Prior for Multi-View Stereo. *arXiv:2001.07791*, 2020.

[62] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a Predictable and Generative Vector Representation for Objects. *European Conference on Computer Vision*, pages 484–499, 2016.

[63] Rohit Girdhar and Deva Ramanan. Attentional Pooling for Action Recognition. *Advances in Neural Information Processing Systems*, pages 34–45, 2017.

[64] Bastian Goldlücke, Mathieu Aubry, Kalin Kolev, and Daniel Cremers. A Super-Resolution Framework for High-Accuracy Multiview Reconstruction. *International Journal of Computer Vision*, 106:172–191, 2014.

[65] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[66] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[67] Edward Grant, Pushmeet Kohli, and Marcel Van Gerven. Deep Disentangled Representations for Volumetric Reconstruction. *European Conference on Computer Vision Workshops*, pages 266–279, 2016.

[68] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A Papier-Mache Approach to Learning 3D Surface Generation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 216–224, 2018.

[69] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic Optimization of Sorting Networks via Continuous Relaxations. *International Conference on Learning Representations*, 2019.

[70] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNET: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum*, 37(2):75–85, 2018.

[71] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[72] JunYoung Gwak, Christopher B. Choy, Manmohan Chandraker, Animesh Garg, and Silvio Savarese. Weakly supervised 3D Reconstruction with Adversarial Constraint. *International Conference on 3D Vision*, pages 263–272, 2017.

[73] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A New Large-scale Point Cloud Classification Benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1-W1:91–98, 2017.

[74] 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. *IEEE International Conference on Computer Vision*, pages 85–93, 2017.

[75] Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi Man Vong, Yu Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. 3D2SeqViews: Aggregating Sequential Views for 3D Global Feature Learning by CNN with Hierarchical Attention Aggregation. *IEEE Transactions on Image Processing*, 28(8):3986–3999, 2019.

[76] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[77] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[79] Tong He, Dong Gong, Zhi Tian, and Chunhua Shen. Learning and Memorizing Representative Prototypes for 3D Point Cloud Semantic and Instance Segmentation. *arXiv:2001.01349*, 2020.

[80] Xingzhe He, Helen Lu Cao, and Bo Zhu. AdvectiveNet: An Eulerian-Lagrangian Fluidic reservoir for Point Cloud Processing. *International Conference on Learning Representations*, 2020.

[81] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments. *International Journal of Robotics Research*, 31(5):647–663, 2012.

[82] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vazquez, Alvar Vinacua, and Timo Ropinski. Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds. *ACM Transactions on Graphics*, 37(6):1–12, 2018.

[83] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4421–4430, 2019.

[84] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What You See is What You Get: Exploiting Visibility for 3D Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[85] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[86] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward Controlled Generation of Text. *International Conference on Machine Learning*, pages 1587–1596, 2017.

[87] 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. *International Conference on 3D Vision*, 2016.

[88] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise Convolutional Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[89] Haibin Huang, Evangelos Kalogerakis, and Benjamin Marlin. Analysis and Synthesis of 3D Shape Families via Deep-learned Generative Models of Surfaces. *Computer Graphics Forum*, 34(5):25–38, 2015.

[90] Jing Huang and Suya You. Point Cloud Labeling using 3D Convolutional Neural Network. *International Conference on Pattern Recognition*, pages 2670–2675, 2016.

[91] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning Multi-view Stereopsis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[92] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent Slice Networks for 3D Segmentation of Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018.

[93] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based Deep Multiple Instance Learning. *International Conference on Machine Learning*, pages 2127–2136, 2018.

[94] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018.

[95] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. *IEEE International Conference on Computer Vision*, pages 2965–2973, 2015.

[96] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis. *IEEE International Conference on Computer Vision*, pages 2326–2334, 2017.

[97] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. GAL: Geometric adversarial loss for single-view 3D-object reconstruction. *European Conference on Computer Vision*, pages 820–834, 2018.

[98] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical Point-Edge Interaction Network for Point Cloud Semantic Segmentation. *IEEE International Conference on Computer Vision*, pages 10433–10441, 2019.

[99] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[100] Adrian Johnston, Ravi Garg, Gustavo Carneiro, Ian Reid, and Anton van den Hengel. Scaling CNNs for High Resolution Volumetric Reconstruction from a Single Image. *IEEE International Conference on Computer Vision Workshops*, 2017.

[101] Takeo Kanade. Recovery of the Three-Dimensional Shape of an Object from a Single View. *Artificial Intelligence*, 17(1-3):409–460, 1981.

[102] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a Multi-View Stereo Machine. *International Conference on Neural Information Processing Systems*, pages 364–375, 2017.

[103] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific Object Reconstruction from a Single Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1966–1974, 2015.

[104] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations*, 2018.

[105] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[106] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. *Symposium on Geometry Processing*, 2006.

[107] Michael Kazhdan and Hugues Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013.

[108] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12:1437–1454, 2012.

[109] Young Min Kim, Niloy J. Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3D Indoor Environments with Variability and Repetition. *ACM Transactions on Graphics*, 31(6), 2012.

[110] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

[111] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014.

[112] Roman Klokov and Victor Lempitsky. Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models. *IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[113] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A Big CAD Model Dataset For Geometric Deep Learning. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

[114] John F. Kolen and Stefan C. Kremer. Gradient Flow in Recurrent Nets: The Difficulty of Learning Long Term Dependencies. *A Field Guide to Dynamical Recurrent Networks*, 2001.

[115] Artem Komarichev, Zichun Zhong, and Jing Hua. A-CNN: Annularly Convolutional Neural Networks on Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7421–7430, 2019.

[116] Chen Kong, Chen-Hsuan Lin, and Simon Lucey. Using Locally Corresponding CAD Models for Dense 3D Reconstructions from a Single Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4857–4865, 2017.

[117] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[118] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–8, 2018.

[119] Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[120] Harold W. Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, 1956.

[121] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Joshua B Tenenbaum. Deep Convolutional Inverse Graphics Network. *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[122] Suryansh Kumar, Yuchao Dai, and Hongdong Li. Monocular Dense 3D Reconstruction of a Complex Dynamic Scene from Two Perspective Frames. *IEEE International Conference on Computer Vision*, pages 4649–4657, 2017.

[123] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Choy, and Silvio Savarese. DeformNet: Free-Form Deformation Network for 3D Shape Reconstruction from a Single Image. *IEEE Winter Conference on Applications of Computer Vision*, pages 858–866, 2017.

[124] Navaneet K L, Priyanka Mandikal, Mayank Agarwal, and R. Venkatesh Babu. CAPNet: Continuous Approximation Projection For 3D Point Cloud Reconstruction Using 2D Supervision. *AAAI Conference on Artificial Intelligence*, 2019.

[125] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R. Oswald. 3D Instance Segmentation via Multi-Task Metric Learning. *IEEE International Conference on Computer Vision*, pages 9256–9266, 2019.

[126] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S. Davis. Modeling Local Geometric Structure of 3D Point Clouds using Geo-CNN. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1008, 2019.

[127] Loic Landrieu and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[128] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[129] Truc Le and Ye Duan. PointGrid: A Deep Network for 3D Shape Understanding. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9204–9214, 2018.

[130] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[131] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[132] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. *International Conference on Machine Learning*, pages 3744–3753, 2019.

[133] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided CNN with Spherical Kernels for 3D Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2019.

[134] Bo Li. 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1513–1518, 2017.

[135] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *Robotics: Science and Systems*, 2016.

[136] Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. Pyramid Attention Network for Semantic Segmentation. *arXiv*, 1805.10180, 2018.

[137] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. SO-Net: Self-Organizing Network for Point Cloud Analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.

[138] Kejie Li, Ravi Garg, Ming Cai, and Ian Reid. Single-view Object Shape Reconstruction Using Deep Shape Prior and Silhouette. *British Machine Vision Conference*, 2019.

[139] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN : Convolution On X -Transformed Points. *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[140] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-Assisted Object Retrieval for Real-Time 3D Reconstruction. *Computer Graphics Forum*, 34(2):435–446, 2015.

[141] Zhidong Liang, Ming Yang, and Chunxiang Wang. 3D Graph Embedding Learning with a Structure-aware Loss Function for Point Cloud Semantic Instance Segmentation. *arXiv preprint arXiv:1902.05247*, 2019.

[142] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction. *AAAI Conference on Artificial Intelligence*, 2018.

[143] Chen-Hsuan Lin, Oliver Wang, Bryan C. Russell, Eli Shechtman, Vladimir G. Kim, Matthew Fisher, and Simon Lucey. Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 969–978, 2019.

[144] Shuyu Lin, Bo Yang, Robert Birke, and Ronald Clark. Learning Semantically Meaningful Embeddings Using Linear Constraints. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 53–56, 2018.

[145] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. *ICCV*, 2017.

[146] 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. *European Conference on Computer Vision*, pages 740–755, 2014.

[147] Tsung-Yu Lin and Subhransu Maji. Improved Bilinear Pooling with CNNs. *British Machine Vision Conference*, 2017.

[148] Tsung-Yu Lin, Subhransu Maji, and Piotr Koniusz. Second-order Democratic Aggregation. *European Conference on Computer Vision*, pages 620–636, 2018.

[149] Tsung Yu Lin, Aruni Roychowdhury, and Subhransu Maji. Bilinear CNN Models for Fine-grained Visual Recognition. *IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[150] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. FPConv: Learning Local Flattening for Point Convolution. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[151] Chen Liu and Yasutaka Furukawa. MASC: Multi-scale Affinity with Sparse Convolution for 3D Instance Segmentation. *arXiv preprint arXiv:1902.04478*, 2019.

[152] Fangyu Liu, Shuaipeng Li, Liqiang Zhang, Chenghu Zhou, Rongtian Ye, Yuebin Wang, and Jiwen Lu. 3DCNN-DQN-RNN: A Deep Reinforcement Learning Framework for Semantic Parsing of Large-scale 3D Point Clouds. *IEEE International Conference on Computer Vision*, pages 5678–5687, 2017.

[153] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. Dynamic Points Agglomeration for Hierarchical Point Sets Learning. *IEEE International Conference on Computer Vision*, pages 7546–7555, 2019.

[154] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and Sampling Network for Dense Point Cloud Completion. *AAAI Conference on Artificial Intelligence*, 2020.

[155] Sainan Liu, Saining Xie, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for Point Cloud Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018.

[156] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[157] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to Infer Implicit Surfaces without 3D Supervision. *Advances in Neural Information Processing Systems*, pages 8293–8304, 2019.

[158] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-End Multi-Task Learning with Attention. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.

[159] Xiaofeng Liu, B.V.K Vijaya Kumar, Chao Yang, Qingming Tang, and Jane You. Dependency-aware Attention Control for Unconstrained Face Recognition with Image Sets. *European Conference on Computer Vision*, pages 548–565, 2018.

[160] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-based Sequence to Sequence Network. *AAAI Conference on Artificial Intelligence*, pages 8778–8785, 2019.

[161] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[162] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[163] Qiang Lu, Mingjie Xiao, Yiyang Lu, Xiaohui Yuan, and Ye Yu. Attention-Based Dense Point Cloud Reconstruction from a Single Image. *IEEE Access*, 7:137420–137431, 2019.

[164] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[165] Priyanka Mandikal and R. Venkatesh Babu. Dense 3D point cloud reconstruction using a deep pyramid network. *IEEE Winter Conference on Applications of Computer Vision*, pages 1052–1060, 2019.

[166] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated Convolutional Networks for 3D Point Cloud Understanding. *IEEE International Conference on Computer Vision*, pages 1578–1587, 2019.

[167] Eric Martin and Chris Cundy. Parallelizing Linear Recurrent Neural Nets over Sequence Length. *International Conference on Learning Representations*, 2018.

[168] John McCormac, Ronald Clark, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger. Fusion++: Volumetric Object-Level SLAM. *International Conference on 3D Vision*, pages 32–41, 2018.

[169] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth. *IEEE International Conference on Computer Vision*, pages 2697–2706, 2017.

[170] Hsien-Yu Meng, Gao Lin, Yu-Kun Lai, and Dinesh Manocha. VV-Net: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation. *IEEE International Conference on Computer Vision*, pages 8500–8508, 2019.

[171] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function

Space. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4455–4465, 2019.

[172] Gregory P. Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez. Sensor Fusion for Joint 3D Object Detection and Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[173] Kazuki Minemura, Hengfui Liau, Abraham Monrroy, and Shinpei Kato. LMNet: Real-time Multiclass Object Detection on CPU using 3D LiDAR. *Asia-Pacific Conference on Intelligent Robot Systems*, pages 28–34, 2018.

[174] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

[175] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and Approximate Symmetry Detection for 3D Geometry. *SIGGRAPH*, 25(3):560–568, 2006.

[176] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[177] Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow, and Niloy J. Mitra. RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes. *ACM Transactions on Graphics*, 34(4):1–12, 2015.

[178] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. McGAN: Mean and Covariance Feature Matching GAN. *International Conference on Machine Learning*, pages 2527–2535, 2017.

[179] Dipti Prasad Mukherjee, Andrew Zisserman, and Michael Brady. Shape from Symmetry: Detecting and Exploiting Symmetry in Affine Images. *Philosophical Transactions: Physical Sciences and Engineering*, 351(1695):77–106, 1995.

[180] Raul Mur-Artal, JMM M M Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans on Robotics*, 31(5):1147–1163, 2015.

[181] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255 – 1262, 2017.

[182] J Krishna Murthy, G V Sai Krishna, Falak Chhaya, and K Madhava Krishna. Reconstructing Vechicles from a Single Image: Shape Priors for Road Scene Understanding. *IEEE International Conference on Robotics and Automation*, pages 724–731, 2017.

[183] Krishna Kanth Nakka and Mathieu Salzmann. Deep Attentional Structured Representation Learning for Visual Recognition. *British Machine Vision Conference*, 2018.

[184] Liangliang Nan, Ke Xie, and Andrei Sharf. A Search-Classify Approach for Cluttered Indoor Scene Understanding. *ACM Transactions on Graphics*, 31(6):1–10, 2012.

[185] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[186] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. PolyGen: An Autoregressive Generative Model of 3D Meshes. *International Conference on Machine Learning*, 2020.

[187] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian Mesh Optimization. *International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, pages 381–389, 2006.

[188] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time Dense Surface Mapping and Tracking. *International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[189] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davision. DTAM: Dense Tracking and Mapping in Real-time. *IEEE International Conference on Computer Vision*, pages 2320–2327, 2011.

[190] Andrea Nicastro, Ronald Clark, and Stefan Leutenegger. X-Section: Cross-Section Prediction for Enhanced RGBD Fusion. *IEEE International Conference on Computer Vision*, pages 1517–1526, 2019.

[191] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[192] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. *ACM Transactions on Graphics*, 32(6):1–11, 2013.

[193] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. *International Conference on Machine Learning*, page 2642–2651, 2017.

[194] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A Survey of Structure from Motion. *Acta Numerica*, 26:305–364, 2017.

[195] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional PointNet for 3D-Object Detection in Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[196] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks. *IEEE International Conference on Computer Vision*, pages 9964–9973, 2019.

[197] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Liy, and Huijing Zhao. SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances. *arXiv:2002.09147*, 2020.

[198] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[199] Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. RayNet: Learning Volumetric 3D Reconstruction with Ray

Potentials. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2018.

[200] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. Discovering Structural Regularity in 3D Geometry. *ACM Transactions on Graphics*, 27(3):1, 2008.

[201] Yanjun Peng, Ming Chang, Qiong Wang, Yinling Qian, Yingkui Zhang, Mingqiang Wei, and Xiangyun Liao. Sparse-to-Dense Multi-Encoder Shape Completion of Unstructured Point Cloud. *IEEE Access*, 2020.

[202] Quang-Hieu Pham, Duc Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2019.

[203] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J. Guibas. ImVoteNet : Boosting 3D Object Detection in Point Clouds with Image Votes. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[204] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. *IEEE International Conference on Computer Vision*, pages 9277–9286, 2019.

[205] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[206] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[207] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[208] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[209] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *International Conference on Learning Representations*, 2016.

[210] Colin Raffel and Daniel P. W. Ellis. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. *International Conference on Learning Representations Workshops*, 2016.

[211] Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical Fractal Convolutional Neural Networks for Point Cloud Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 452–460, 2019.

[212] Hazem Rashed, Mohamed Ramzy, Victor Vaquero, Ahmad El Sallab, Ganesh Sistu, and Senthil Yogamani. FuseMODNet: Real-Time Camera and LiDAR based Moving Object Detection for robust low-light Autonomous Driving. *IEEE International Conference on Computer Vision Workshops*, 2019.

[213] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text to Image Synthesis. *International Conference on Machine Learning*, pages 1060–1069, 2016.

[214] Scott E. Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning What and Where to Draw. *Advances in Neural Information Processing Systems*, pages 217–225, 2016.

[215] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[216] Dario Rethage, Johanna Wald, Jürgen Sturm, Nassir Navab, and Federico Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. *European Conference on Computer Vision*, pages 596–611, 2018.

[217] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning Depth Fusion from Data. *International Conference on 3D Vision*, pages 57–66, 2017.

[218] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[219] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3D Object Shape from One Depth Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2484–2493, 2015.

[220] Pau Rodríguez, Josep M. Gonfaus, Guillem Cucurull, F. Xavier Roca, and Jordi Gonzàlez. Attend and Rectify: a Gated Attention Mechanism for Fine-Grained Recovery. *European Conference on Computer Vision*, pages 349–364, 2018.

[221] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net : Convolutional Networks for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[222] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, (115):211–252, 2015.

[223] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (fpfh) for 3D Registration. *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[224] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly, and Andrew J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[225] Nikolaos Sarafianos, Xiang Xu, and Ioannis A. Kakadiaris. Deep Imbalanced Attribute Classification using Visual Attention Aggregation. *European Conference on Computer Vision*, pages 680–697, 2018.

[226] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[227] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.

[228] 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*, 31(6):1–11, 2012.

[229] Abhishek Sharma, Oliver Grau, and Mario Fritz. VConv-DAE : Deep Volumetric Shape Learning Without Object Labels. *European Conference on Computer Vision*, pages 236–250, 2016.

[230] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4557, 2018.

[231] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[232] 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.

[233] Daeyun Shin, Zhile Ren, Erik B. Sudderth, and Charless C. Fowlkes. 3D Scene Reconstruction with Multi-layer Depth and Epipolar Transformers. *IEEE International Conference on Computer Vision*, pages 2172–2182, 2019.

[234] Martin Simon, Stefan Milz, Karl Amende, and Horst-michael Gross. Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds. *European Conference on Computer Vision Workshops*, 2018.

[235] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2015.

[236] Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate Symmetry Detection in Partial 3D Meshes. *Computer Graphics Forum*, 33(7):131–140, 2014.

[237] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019.

[238] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. SDF-2-SDF: Highly Accurate 3D Object Reconstruction. *European Conference on Computer Vision*, pages 680–696, 2016.

[239] Edward Smith and David Meger. Improved Adversarial Systems for 3D Object Generation and Reconstruction. *Conference on Robot Learning*, pages 87–96, 2017.

[240] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D. Kulkarni, and Joshua B. Tenenbaum. Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1511–1519, 2017.

[241] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.

[242] Pablo Speciale, Martin R. Oswald, Andrea Cohen, and Marc Pollefeys. A Symmetry Prior for Convex Variational 3D Reconstruction. *European Conference on Computer Vision*, pages 313–328, 2016.

[243] Srinath Sridhar, Davis Rempe, Julien Valentin, Sofien Bouaziz, and Leonidas J. Guibas. Multiview Aggregation for Learning Category-Specific Shape Reconstruction. *Advances in Neural Information Processing Systems*, pages 2348–2359, 2019.

[244] Frank Steinbrucker, Christian Kerl, Jurgen Sturm, and Daniel Cremers. Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences. *IEEE International Conference on Computer Vision*, pages 3264–3271, 2013.

[245] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, and Hauke M Strasdat. The Replica Dataset : A Digital Replica of Indoor Spaces. *arXiv:1906.05797*, 2019.

[246] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[247] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *IEEE International Conference on Computer Vision*, pages 945–953, 2015.

[248] Chris Sweeney, Torsten Sattler, Tobias Höllerer, Matthew Turk, and Marc Pollefeys. Optimizing the Viewing Graph for Structure-from-Motion. *IEEE International Conference on Computer Vision*, pages 801–809, 2015.

[249] Jiapeng Tang, Xiaoguang Han, Junyi Pan, Kui Jia, and Xin Tong. A Skeleton-bridged Deep Learning Approach for Generating Meshes of Complex Topologies from Single RGB Images. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4541–4550, 2019.

[250] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[251] Lyne P. Tchapmi, Christopher B Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. SEGCloud: Semantic Segmentation of 3D Point Clouds. *International Conference on 3D Vision*, pages 537–547, 2017.

[252] Gusi Te, Wei Hu, Zongming Guo, and Amin Zheng. RGCNN : Regularized Graph CNN for Point Cloud Segmentation. *ACM Multimedia Conference on Multimedia Conference*, pages 746–754, 2018.

[253] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1:211–221, 1988.

[254] Hugues Thomas, Jean-emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Yann Legall. Semantic Classification of 3D Point Clouds with Multiscale Geometric Neighborhoods. *International Conference on 3D Vision*, pages 390–398, 2018.

[255] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. *IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[256] Sebastian Thrun and Ben Wegbreit. Shape from Symmetry. *IEEE International Conference on Computer Vision*, pages 1824–1831, 2005.

[257] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. *International Workshop on Vision Algorithms*, 1999.

[258] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2634, 2017.

[259] Víctor Vaquero, Ivan Del Pino, Francesc Moreno-Noguer, Joan Soì, Alberto Sanfeliu, and Juan Andrade-Cetto. Deconvolutional Networks for Point-Cloud Vehicle Detection and Tracking in Driving Scenarios. *European Conference on Mobile Robots*, pages 1–7, 2017.

[260] Jacob Varley, Chad Dechant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape Completion Enabled Robotic Grasping. *International Conference on Intelligent Robots and Systems*, pages 2442–2447, 2017.

[261] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[262] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order Matters: Sequence to Sequence for Sets. *International Conference on Learning Representations*, 2015.

[263] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the Limitations of Representing Functions on Sets. *International Conference on Machine Learning*, pages 6487–6494, 2019.

[264] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local Spectral Graph Convolution for Point Set Feature Learning. *European Conference on Computer Vision*, pages 52–66, 2018.

[265] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph Attention Convolution for Point Cloud Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.

[266] Meng Wang, Lingjing Wang, and Yi Fang. 3DensiNet: A Robust Neural Network Architecture towards 3D Volumetric Object Prediction from 2D Image. *ACM international conference on Multimedia*, page 961–969, 2017.

[267] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *European Conference on Computer Vision*, pages 52–67, 2018.

[268] Wei Wang, Muhamad Risqi U Saputra, Peijun Zhao, Pedro Gusmao, Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network. *International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[269] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann. Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks. *IEEE International Conference on Computer Vision*, pages 2298–2306, 2017.

[270] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

[271] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively Segmenting Instances and Semantics in Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019.

[272] Xu Wang, Jingming He, and Lin Ma. Exploiting Local and Global Structure for Point Cloud Semantic Segmentation with Contextual Point Representations. *Advances in Neural Information Processing Systems*, pages 4573–4583, 2019.

[273] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 38(5), 2019.

[274] Zhihua Wang, Stefano Rosa, Linhai Xie, Bo Yang, Sen Wang, Niki Trigoni, and Andrew Markham. Defo-Net: Learning Body Deformation Using Generative Adversarial Networks. *IEEE International Conference on Robotics and Automation*, pages 2440–2447, 2018.

[275] Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, Niki Trigoni, and Andrew Markham. 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations. *International Joint Conference on Artificial Intelligence*, page 4958–4964, 2018.

[276] Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing Bird View LIDAR Point Cloud and Front View Camera Image for Deep Object Detection. *arXiv:1711.06703*, 2017.

[277] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation. *IEEE International Conference on Computer Vision*, pages 1042–1051, 2019.

[278] Thomas Whelan, Stefan Leutenegger, Renato F Salas-moreno, Ben Glocker, and Andrew J Davison. ElasticFusion : Dense SLAM Without A Pose Graph. *Robotics: Science and Systems*, 2015.

[279] Olivia Wiles and Andrew Zisserman. SilNet : Single- and Multi-View Reconstruction by Learning from Silhouettes. *British Machine Vision Conference*, 2017.

[280] Olivia Wiles and Andrew Zisserman. Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views. *International Journal of Computer Vision*, pages 1–21, 2018.

[281] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. *IEEE International Conference on Robotics and Automation*, pages 1887–1893, 2018.

[282] Changchang Wu. Towards Linear-Time Incremental Structure from Motion. *International Conference on 3D Vision*, 2013.

[283] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. *Advances in Neural Information Processing Systems*, pages 540–550, 2017.

[284] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[285] Pengxiang Wu, Chao Chen, Jingru Yi, and Dimitris Metaxas. Point Cloud Processing via Recurrent Set Encoding. *AAAI Conference on Artificial Intelligence*, pages 5441–5449, 2019.

[286] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[287] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[288] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, Shengping Zhang, and Xiaojun Tong. Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images. *IEEE International Conference on Computer Vision*, pages 2690–2698, 2019.

[289] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, Xiaoshuai Sun, and Wenxiu Sun. Toward 3D Object Reconstruction from Stereo Images. *arXiv:1910.08223*, 2019.

[290] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. *IEEE International Conference on Robotics and Automation*, pages 5231–5237, 2019.

[291] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

[292] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *International Conference on Machine Learning*, pages 2048–2057, 2015.

[293] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-GCN for Fast and Scalable Point Cloud Learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[294] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. *European Conference on Computer Vision*, pages 87–102, 2018.

[295] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision. *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016.

[296] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR : Real-time 3D Object Detection from Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[297] Bo Yang, Zihang Lai, Xiaoxuan Lu, Shuyu Lin, Hongkai Wen, Andrew Markham, and Niki Trigoni. Learning 3D Scene Semantics and Structure from a Single Depth Image. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 309–312, 2018.

[298] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3D Object Reconstruction from a Single Depth View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):2820 – 2834, 2019.

[299] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. *Advances in Neural Information Processing Systems*, pages 6737–6746, 2019.

[300] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *International Journal of Computer Vision*, 128:53–73, 2020.

[301] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3D Object Reconstruction from a Single Depth View with Adversarial Learning. *IEEE International Conference on Computer Vision Workshops*, pages 679–688, 2017.

[302] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling Point Clouds with Self-Attention and Gumbel Subset Sampling. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019.

[303] Xin Yang, Yuanbo Wang, Yaru Wang, Baocai Yin, Qiang Zhang, Xiaopeng Wei, and Hongbo Fu. Active Object Reconstruction Using a Guided View Planner. *International Joint Conference on Artificial Intelligence*, pages 4965–4971, 2018.

[304] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: Sparse-to-Dense 3D Object Detector for Point Cloud. *IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[305] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked Attention Networks for Image Question Answering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[306] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. *European Conference on Computer Vision*, pages 767–783, 2018.

[307] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation. *European Conference on Computer Vision*, pages 403–417, 2018.

[308] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019.

[309] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. *International Conference on Learning Representations*, 2019.

[310] Kaicheng Yu and Mathieu Salzmann. Statistically Motivated Second Order Pooling. *European Conference on Computer Vision*, pages 600–616, 2018.

[311] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *AAAI Conference on Artificial Intelligence*, 2017.

[312] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view Harmonized Bilinear Network for 3D Object Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–194, 2018.

[313] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. *International Conference on Neural Information Processing Systems*, pages 3391–3401, 2017.

[314] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. RT3D: Real-Time 3D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. *IEEE Robotics and Automation Letters*, 3(4):3434–3440, 2018.

[315] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. *International Conference on Machine Learning*, pages 7354–7363, 2019.

[316] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. *IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[317] Wenxiao Zhang and Chunxia Xiao. PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019.

[318] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. ShellNet: Efficient Point Cloud Convolutional Neural Networks using Concentric Shells Statistics. *IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.

[319] Hengshuang Zhao, Li Jiang, Chi-wing Fu Jiaya, and Jiaya Jia. PointWeb : Enhancing Local Neighborhood Features for Point Cloud Processing. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.

[320] Lin Zhao and Wenbing Tao. JSNet: Joint Instance and Semantic Segmentation of 3D Point Clouds. *AAAI Conference on Artificial Intelligence*, 2020.

[321] Wei Zhao, Shuming Gao, and Hongwei Lin. A Robust Hole-filling Algorithm for Triangular Mesh. *The Visual Computer*, 23(12):987–997, 2007.

[322] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J. Davison. SceneCode: Monocular Dense Semantic Reconstruction using Learned EncodedScene Representations. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11776–11785, 2019.

[323] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[324] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. *IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[325] Yingying Zhu, Jiong Wang, Lingxi Xie, and Liang Zheng. Attention-based Pyramid Aggregation Network for Visual Place Recognition. *ACM International Conference on Multimedia*, pages 99–107, 2018.

[326] Chuhang Zou and Derek Hoiem. Silhouette Guided Point Cloud Reconstruction beyond Occlusion. *Winter Conference on Applications of Computer Vision*, 2020.

[327] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. *IEEE International Conference on Computer Vision*, pages 900–909, 2017.