DEEP LEARNING METHODS FOR 360 MONOCULAR DEPTH ESTIMATION AND POINT CLOUD SEMANTIC SEGMENTATION

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

Yuyan Li

Dr. Ye Duan, Thesis Supervisor

May 2022

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

DEEP LEARNING METHODS FOR
360 MONOCULAR DEPTH ESTIMATION
AND POINT CLOUD SEMANTIC SEGMENTATION

presented by Yuyan Li,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Ye Duan

Dr. Filiz Bunyak

Dr. Jeffery Uhlmann

Dr. Gang Yao

ACKNOWLEDGMENTS

I would like to thank my research advisor Dr. Ye Duan for his invaluable advice, continuous support, immense knowledge and plentiful experience during my PhD study. My gratitude extends to all my committee members, Dr. Jeffery Uhlmann, Dr. Filiz Bunyak, Dr. Gary Yao. It is their kind help and great suggestions that have made my comprehensive exam and my final dissertation wonderful experiences.

I would like to thank my parents, without their understanding and encouragement through the years, it would be impossible for me to finish my study. Additionally, I would like to thank my dear friend Weitian Sheng, who has offered great support and consolidation. I also appreciate Qing Lei, a good listener and a true friend. I would like to express my gratitude to my boyfriend, Imad Toubal, who has been the best companion, my inspiration and my best friend. I would like to thank my cat, Huang Da Meow, who provides companionship and helps get through hard times.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Monocular depth estimation and point cloud segmentation are essential tasks for 3D scene understanding in computer vision.

Depth estimation for omnidirectional images is challenging due to the spherical distortion issue and the availability of large-scale labeled datasets. We propose two separate works for 360 monocular depth estimation tasks. In the first work, we propose a novel, model-agnostic, two-stage pipeline for omnidirectional monocular depth estimation. Our proposed framework *PanoDepth* takes one 360 image as input, produces one or more synthesized views in the first stage, and feeds the original image and the synthesized images into the subsequent stereo matching stage. Utilizing the explicit stereo-based geometric constraints, *PanoDepth* can generate dense high-quality depth. In the second work, we propose a 360 monocular depth estimation pipeline, *OmniFusion*, to tackle the spherical distortion issue. Our pipeline transforms a 360 image into less-distorted perspective patches (i.e. tangent images) to obtain patch-wise predictions via CNN, and then merge the patch-wise results for final output. To handle the discrepancy between patch-wise predictions which is a major issue affecting the merging quality, we propose a new framework with (i) a geometry-aware feature fusion mechanism that combines 3D geometric features with 2D image features. (ii) the self-attention-based transformer architecture to conduct a global aggregation of patch-wise information. (iii) an iterative depth refinement mechanism to further refine the estimated depth based on the more accurate geometric features. Experiments show that both PanoDepth and OmniFusion achieve state-of-the-art performances on several 360 monocular depth estimation benchmark datasets.

For point cloud analysis, we mainly focus on defining effective local point convolution

operators. We propose two approaches, SPNet and Point-Voxel CNN respectively. For the former, we propose a novel point convolution operator named Shell Point Convolution (SPConv) as the building block for shape encoding and local context learning. Specifically, SPConv splits 3D neighborhood space into shells, aggregates local features on manually designed kernel points, and performs convolution on the shells. For the latter, we present a novel lightweight convolutional neural network which uses point voxel convolution (PVC) layer as building block. Each PVC layer has two parallel branches, namely the voxel branch and the point branch. For the voxel branch, we aggregate local features on non-empty voxel centers to reduce geometric information loss caused by voxelization, then apply volumetric convolutions to enhance local neighborhood geometry encoding. For the point branch, we use Multi-Layer Perceptron (MLP) to extract fine-detailed point-wise features. Outputs from these two branches are adaptively fused via a feature selection module. Experimental results show that SPConv and PVC layers are effective in local shape encoding, and our proposed networks perform well in semantic segmentation tasks.

Chapter 1

Introduction

1.1 360 Monocular Depth estimation

Depth estimation has been extensively studied in computer vision, which is an essential components for many applications such as autonomous driving, robot navigation, virtual reality, etc. Upon the advances of deep learning technologies and the availability of large scale datasets, deep learning-based monocular depth estimation shows great potential in achieving efficiency and robustness. However, since monocular depth estimation is by nature an ill-posed problem, how to better estimate the true scale of the depth in novel scenes and how to recover the structural details are the major challenges.

360 images give a comprehensive view of the entire scene, which is of great advantage in understanding the scene holistically. Particularly, due to a large Field of View (FoV), a 360 image-based method can rely on large context to infer depth, which is valuable when estimating the true scale in the monocular depth estimation. Meanwhile, the 360 perception

is free from the object truncation issue and independent of the intrinsic camera parameters, which are both understood as major challenges towards learning a generalizable model of the perspective image. However, 360 images projected in the equirectangular projection (ERP) coordinate system introduces irregular distortion, which varies with latitude. Such distortion issue significantly degrades the monocular depth estimation of 360 images if not handled carefully. Moreover, inferring depth directly from the monocular RGB input is to build a mapping from RGB pixel to real-valued depth. To build an accurate mapping requires a large amount of training data with ground truth label [17]. For 360 vision, such large-scale labeled dataset is difficult and expensive to acquire. To this end, we introduce two frameworks, PanoDepth and OmniFusion, to address the distortion problem and to improve depth estimation performance with limited training data.

We first propose a two-stage framework, *PanoDepth*, that takes one equirectangular projection (ERP) image as input, produces one or more synthesized views in the first stage, and feeds the original image and the synthesized images to the subsequent stereo matching stage to predict the final depth map. In the stereo matching stage, we propose a novel differentiable Spherical Warping Layer to handle omnidirectional stereo geometry efficiently and effectively. We conducted extensive experiments and ablation studies to evaluate *PanoDepth* with both the full pipeline and the individual networks in each stage on several public benchmark datasets. Our results demonstrated that our model-agnostic approach *PanoDepth* outperforms the one-stage method by a large margin despite the combinations of coarse estimation and stereo matching networks. Moreover, by adjusting these networks, *PanoDepth* can be adapted to the target computation constraints and performance requirements.

Second, we propose a 360 monocular depth estimation framework, *OmniFusion*, with

geometry-aware fusion that uses tangent image representation. It is advantageous to use tangent images [13] as it has less distortion, and can make good use of the large pool of pre-trained CNNs developed for perspective imaging. However, the vanilla pipeline [13] has some limitations. First, severe discrepancies occur between perspective views since the same object may appear differently from multiple views. This issue is problematic for the depth regression task, since the inconsistent depth scale estimated from individual tangent images creates undesired artifacts during merging. Second, the advantage of estimating depth from holistic 360 image is unfortunately lost, because of the decomposition of the global scene into local tangent images. The predictions from the tangent images are independent of each other and there is no information exchange between tangent images. We proposed the following three key components to solve the discrepancy issue and merge the depth results of tangent images seamlessly. First, we use a geometric embedding module to provide additional features to compensate for the discrepancy between 2D features from patch to patch. For each patch, we calculate the 3D points located on the spherical surface that correspond to the patch pixels, encode them and the patch center coordinate through shared Multi-layer Perceptron (MLP), and add the geometric features to the corresponding 2D features. Second, to regain the holistic power in understanding the entire scene, we incorporate a self-attention-based transformer in our pipeline. With the transformer, patch-wise information is globally aggregated to enhance the estimation of the global scale of depth, and to improve the consistency between patch-wise results. Third, we introduce an iterative refining mechanism, where more accurate 3D information from the predicted depth maps is fed back to the geometric embedding module to further improve the depth quality in an iterative manner. We test *OmniFusion* on three benchmark datasets: Stanford2D3D [1], Matterport3D [9], and 360D [2]. Experimental results show

that our method outperforms state-of-the-art methods by a significant margin on all of these datasets.

1.2 Point Cloud Semantic Segmentation

Deep learning based point cloud analysis is appealing and challenging. Increasing interest in applications such as robotics, autonomous driving, etc. has brought more attention to 3D point cloud learning. Point cloud segmentation involves dividing 3D point clouds into homogeneous regions, such that points in the same isolated and meaningful region have similar properties. 3D point cloud segmentation is a challenging task because of high redundancy, non-uniform sampling density, and lack of explicit structure of point cloud data. Segmenting objects in 3D point clouds is not a trivial task. The point cloud data is usually noisy, sparse, and unordered. Apart from that, the sampling density of points is uneven and the surface shape can be arbitrary with no statistical distribution pattern in data. Moreover, due to limitations in 3D sensors, the background is entangled with the foreground. Additionally, it is difficult to have a deep learning model that is computationally efficient and has a low memory footprint to perform segmentation. To effectively process the point cloud data, encode shape and learn context information, we introduce two different approaches, SPNet and Point-voxel CNN respectively, for point cloud semantic segmentation.

SPNet is inspired by the work KPConv [16], which manually designed a set of either rigid or deformable kernel points and used them to aggregate local neighbor information. Our SPConv operator partitions local 3D space into shells, each shell contains a set of rigid kernel points which aggregate local supporting point features. We perform kernel point convolution on each shell individually, then integrate the output features by an additional

1D convolution operation. The last convolution learns the contributions from shells and enhances shell correlation. Comparing to deformable KPConv [16], our method has an enhanced structure learning module, and does not require additional regularization during training. Furthermore, we find that incorporating low-level features such as color, normal, etc. in all layers for local feature re-weighting can be very effective for improving network performance. We propose two different approaches to accomplish the task, (1) Gaussian function based and (2) learning based. The first approach is hand-crafted and does not necessarily add GPU computation. The second approach has learnable weights and shows more robustness. Using SPConv as our building block, we build a deep architecture SPNet. Similar to standard CNNs which utilize downsampling and upsampling strategy to reduce computation cost as well as to enlarge receptive field, we use Poisson disk sampling (PDS) for downsampling, and feature propagation (FP) for upsampling.

The point-voxel CNN is constructed using point-voxel layer that takes advantages of both sparse point representation and volumetric grid representation. Our point-voxel layer consists of two parallel branches, a voxel-based branch which aggregates local neighboring features, and a point-based branch which maintains fine-grained point-wise features. During discretization in voxel branch, we aggregate neighboring features on non-empty voxel centers and use standard 3D convolutions to enhance local feature encoding. Voxel features are propagated back to point domain through devoxelization. Outputs from point and voxel branches are fused self-adaptively via a feature selection module(FSM), which learns channel-wise attention for both branches.

1.3 Dissertation Organization

The rest of the dissertation is organized as follows. In **Chapter 2**, we introduce the *PanoDepth* framwork. **Chapter 3** is the main body of *OmniFusion*. Both **Chapter 2** and **3** focus on monocular depth estimation for 360 images. In **Chapter 4** and **Chapter 5**, we include SPNet and Point-voxel CNN respectively. SPNet and Point-voxel CNN are evaluated for point cloud semantic segmentation task. In **Chapter 6**, we summarize our works and conclude the remarks.

Chapter 2

PanoDepth: A Two Stage Approach for Monocular Omnidirectional Depth Estimation

2.1 Introduction

Omnidirectional 3D information is essential for a wide range of applications (e.g. Virtual Reality [18], Augmented Reality [19], Autonomous Driving [20], and Robotics [21]). Quick and reliable acquisition of omnidirectional 3D data can facilitate many use cases, such as user interaction with the digital environment, robot navigation, and object detection for autonomous vehicles. Another relevant application is remote working/shopping/education [22], which has become ubiquitous due to the pandemic. To obtain high-quality omnidirectional 3D information, devices such as omnidirectional LiDARs are widely used in Autonomous Driving and indoor 3D scans. However, LiDARs are either very expensive, or can only produce sparse 3D scans. Compared with LiDARs, cameras are much cheaper

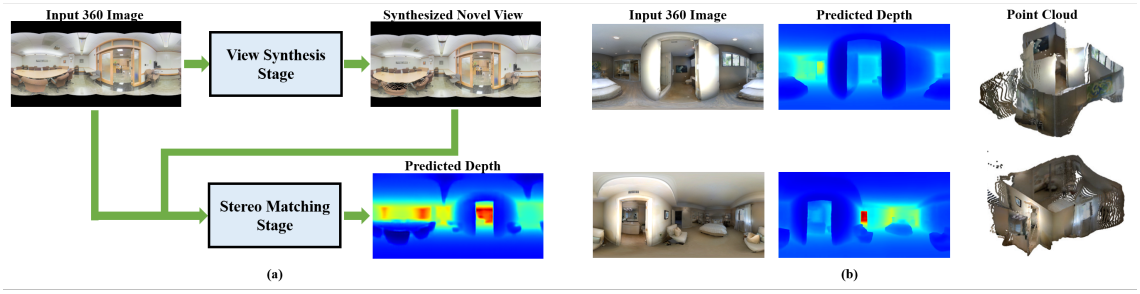


Figure 2.1: (a) Illustration of our *PanoDepth* framework. *PanoDepth* takes one 360 image as input to generate one or more novel views in the first view synthesis stage. The original and synthesized 360 images are then fed into the second multi-view stereo matching stage to predict final dense depth map. (b) Two examples (top and bottom row) of *PanoDepth* on 360D dataset [2] with 360 image (left) as input, and output depth (middle) and point cloud (right).

and already frequently used for capturing the visual appearance of the scenes. The cost can be significantly reduced if high-quality omnidirectional 3D can be generated directly from camera images.

Deep learning techniques, coupled with a growing accessibility of large-scale datasets, have largely improved the performance of many computer vision tasks including depth estimation [23]. Depth estimation often uses either a monocular input or a stereo pair. For the monocular methods, a common practice is to train a single network to map RGB pixel to real-value depth [24, 25, 26, 27], mostly by learning from various monocular cues such as shape, lighting, shading, object type, etc. Stereo matching methods [28, 29, 7], on the other hand, learn the disparity by matching image patches from stereo pairs and later convert disparity to depth. Despite the significant improvement in monocular estimation methods, there is still a large gap between monocular and stereo depth accuracy [30].

To apply stereo matching for monocular depth estimation, Luo et al. [17] proposed a two-stage pipeline that decomposes monocular depth estimation into two stages, view synthesis and stereo matching respectively. In their approach, a second view is first syn-

thesized and fed together with the original view to the stereo matching stage to compute the disparity. Stereo matching can leverage more geometric constraints into the network training, thus reducing the demand for ground truth depth. More recent studies [31, 32] improved upon this idea and achieved promising performances. These two-stage methods [17, 31, 32], are mainly designed for perspective images. In 360 domain, most of the recent studies [2, 33, 34, 3, 35] still follow the same single-stage monocular estimation procedure with adaptation to 360 [1] geometry.

In this paper, we propose a novel, model-agnostic, two-stage pipeline (see Figure 3.2) for solving the problem of 360 monocular depth estimation. Our proposed framework *PanoDepth* takes one equirectangular projection (ERP) image as input, produces one or more synthesized views in the first stage, and feeds the original image and the synthesized images to the subsequent stereo matching stage to predict the final depth map. In the stereo matching stage, we propose a novel differentiable Spherical Warping Layer to handle omnidirectional stereo geometry efficiently and effectively. We conducted extensive experiments and ablation studies to evaluate *PanoDepth* with both the full pipeline and the individual networks in each stage on several public benchmark datasets. Our results demonstrated that our model-agnostic approach *PanoDepth* outperforms the one-stage method by a large margin despite the combinations of coarse estimation and stereo matching networks. Moreover, by adjusting these networks, *PanoDepth* can be adapted to the target computation constraints and performance requirements.

Our contributions can be summarized as follows:

- We propose a novel, model-agnostic, two-stage framework *PanoDepth*, including view synthesis and stereo matching, to fully exploit the synthesized 360 views and

[1] We use the terms 360, omnidirectional, equirectangular, spherical interchangeably in this paper

spherical stereo constraints.

- *PanoDepth* outperforms the state-of-the-art monocular omnidirectional depth estimation approaches by a large margin.

- We propose a novel differentiable Spherical Warping Layer (SWL) which adapts regular stereo matching networks to 360 stereo geometry, and enables advanced features such as multi-view stereo and cascade mechanism for stereo performance boost.

2.2 Motivation

In this section, we explain the motivation of formulating the 360 monocular depth estimation problem as two separate stages, namely, a view synthesis stage based on coarse depth estimation, and a multi-view stereo matching stage for final depth output.

2.2.1 Why Two-Stage?

One main advantage of monocular depth estimation is its potential in dramatically reducing the hardware cost for 3D depth acquisition. Motivated by this, many studies have been proposed to solve this problem. The basic idea of supervised monocular estimation methods is to train a network that directly learns the mapping from the input RGB pixels to the real-value output depth in a single stage. For example, Laina et al. [36] proposed FCRN which uses ResNet-50 [37] as backbone, followed by multiple up-projection modules. Hu et al. [26] leveraged SENet-154 [38] as encoder together with multi-scale fusion module.

On the other hand, deep learning based stereo matching networks [7, 29, 39] utilize the stereo constraints to improve efficiency and the output depth quality. These methods sim-

ulate the traditional stereo matching process by learning and optimizing the matching cost across the input image pairs in a deterministic manner. Unlike monocular depth methods which directly map RGB into depth by considering all the monocular cues, stereo matching methods focus on estimating disparity by developing image patch correspondence [24]. Given the predefined baseline and 1D search space along the epipolar line for image patch matching [40], stereo matching produces more accurate depth maps in comparison with monocular methods in general [30].

A typical stereo matching network requires at least two images as input, which is not directly applicable for a monocular input setting. However, if one or more novel views can be synthesized with high quality, these additional views can be utilized to train a stereo matching network. Luo et al. [17] first proposed the two-stage pipeline for perspective images where a novel right view is synthesized at the first stage and paired with the original view to the second stereo stage. Later, two-stage approaches [31] mostly followed this work to generate a coarse disparity/depth, and to synthesize novel views via image warping or Depth-Image-Based Rendering (DIBR). Taking the original and the synthesized views as input, the final depth generated from the later stereo matching stage of these approaches shows a significant improvement over the one-stage counterparts.

2.2.2 Can we successfully synthesize novel view 360 images?

Recent two-stage approaches [17, 31] have shown promising capabilities in improving depth quality on perspective images. However, it remains unclear whether the two-stage approach will be applicable for 360 images, as there are many fundamental differences between the perspective images and 360 images, such as camera projection model, image distortion, and field of view (FoV).

The difference in camera projection model, equirectangular projection (see Figure 2.3(a)) vs. perspective projection, can be resolved by integrating spherical geometry into the disparity calculation and cost volume fusion procedure. In this paper, we propose a novel Spherical Warping Layer specifically designed for spherical geometry as a solution (Section 2.4.2). Moreover, the distortion issue can be addressed by applying distortion-aware convolutions [34, 41, 42, 33, 43, 44, 45]. Hence, in this section we will mainly discuss the difference in FoV settings. Comparing with perspective images, 360 images have much larger FoVs (360° horizontally, 180° vertically). A 360 image encodes almost every piece of visual information of the scene except occluded areas, while perspective images suffer from information loss near the image boundaries in addition to occlusions. This could be a great advantage for novel view synthesis of 360 images.

To validate this observation, we conducted various experiments regarding the correlations between image FoV, baseline and synthesized view quality (more details in the Appendix). Our experiment confirms that i) with greater FoV, the synthesized views are less sensitive to large baselines, and ii) synthesized 360 images have the least error and artifacts. According to Gallup et al. [46], depth error of stereo matching comes from both the disparity error (proportional) and the baseline (inversely proportional). Thus, with higher quality synthesized novel views and larger baselines, we expect less error in the final depth output from stereo matching. This also indicates that the two-stage pipeline is well-suited for 360 monocular depth estimation.

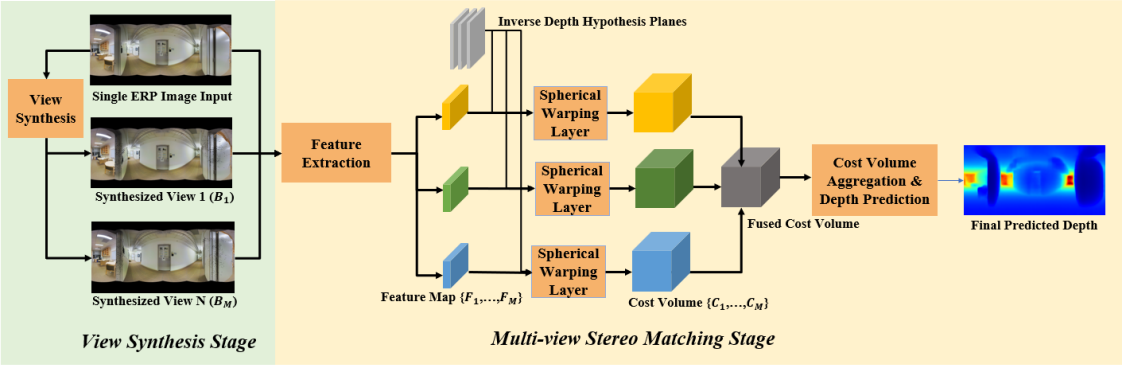


Figure 2.2: Illustration of our *PanoDepth* framework which consists of a view synthesis stage and a multi-view stereo matching stage. In the view synthesis stage, a total of M synthesized views are generated. In the multi-view stereo matching stage, the synthesized 360 views together with the original input view, are sent to the multi-view stereo matching network to produce the final depth estimation. To better adapt to 360 stereo geometry, we directly sample hypothesis plane on the inverse depth, and use a Spherical Warping Layer (SWL) to warp reference views to the target view (Section 2.4.2).

2.3 Related Work

2.3.1 Monocular Depth Estimation

Monocular depth estimation [47] has seen significant improvements [36, 26] since the first adoption of deep learning by Eigen et al. [24]. To further improve the performance, researchers explored many strategies such as multi-task learning with normal estimation [48] and semantic segmentation [49, 50] along with depth estimation, incorporating CRF [51, 52], integrating attention modules [27, 53], utilizing planar constraints [54, 27], conducting unsupervised learning using constrainsts such as left-right consistency [55, 56, 57], as well as two-stage approaches where stereo constraints are leveraged [17, 31, 32].

As 360 cameras become more affordable, researchers start to explore the possibility to use 360 images for depth estimation [34, 58, 4, 2, 8, 6, 5]. To advance the performance of

360 monocular depth estimation, Eder et al. [35] proposed joint training of surface normal, boundary, and depth. Zeng et al. [5] trained a network which combines 3D layout and depth. Jin et al. [59] took advantage of the correlation between depth map and geometric structure of 360 indoor images. Cheng et al. [4] proposed a low-cost sensing system which combines an omnidirectional camera with a calibrated projective depth camera. The 360 image and the limited FoV depth are used together as input to a CNN. Meanwhile, distortion-aware convolution filters [58, 44, 34, 45] are designed to handle spherical geometric distortion.

2.3.2 Multi-View Stereo Matching

Besides monocular depth estimation, Multi-View Stereo (MVS) is another group of methods for predicting depth. Given a set of images with known camera poses, MVS approaches [60, 61] can produce highly accurate depth estimates with multi-view geometric constraints. One example is MVSNet [61], in which the variance-based cost volume is presented to fuse multiple features maps from source images into one unified cost volume. Stereo matching can be treated as a special case of MVS. Conventionally, stereo-based depth estimation methods [28, 62] relied on matching pixels across stereo images. Many recent stereo matching approaches [29, 7] leveraged CNNs for feature extraction, cost matching, and aggregation. For example, PSMNet [7] incorporated spatial pyramid pooling (SPP) module and multi-scale 3D hourglass modules to further boost the performance. To improve the efficiency and accelerate training on high-resolution images, Gu et al. [39] presented a cascade cost volume design to gradually retrieve finer hypothesis plane ranging over multiple steps.

In 360 stereo domain, SweepNet [63] and OmniMVS [64] estimated depth from multi-

ple wide-baseline fisheye cameras. Another recent work is 360SD-Net [8] which predicted disparity/depth from a pair of ERP images that are taken by a top-bottom camera pair. They [8] incorporated polar angles to solve distortions and proposed learnable shifting filters to adjust the step size in disparity cost volume construction. However, the learnable shifting filters create extra overhead during training. In this paper, we propose a closed-form solution, Spherical Warping Layer (SWL), that does not require additional training overhead. Our experiments (Table 2.2) show that SWL can significantly improve the performance of 360 stereo matching.

2.4 Approach

We propose an end-to-end framework, *PanoDepth*, that takes a single ERP image as input and produces a high-quality omnidirectional depth map. *PanoDepth* consists of two stages: i) a view synthesis stage that conducts coarse depth estimation followed by a differentiable DIBR module for novel view synthesis, and ii) a stereo matching stage with a customized Spherical Warping Layer for efficient and high-quality 360 depth estimation. A full framework of *PanoDepth* is illustrated in Figure 3.2.

2.4.1 View Synthesis Stage

To synthesize high-quality novel views, the coarse depth map is usually estimated first followed by a Depth-Image-Based Rendering (DIBR) module [17, 65]. Based on our empirical observations (see the Appendix for details), such a procedure also works well for 360 novel view synthesis with different configurations of coarse depth estimation networks. Considering both performance and computation cost, in this paper we suggest using

a lightweight network: CoordNet [6] for doing the task. CoordNet utilizes coordinate convolution [66] to enforce 360 awareness. We append an atrous spatial pyramid pooling module (ASPP) [67] to the end of the encoder to better aggregate multi-scale context information. Note that *PanoDepth* is model-agnostic, thus any depth estimation network can be used here to fulfill specific requirement. The estimated coarse depth map and the original ERP image are then used to render multiple synthesized views of predefined baselines via a differentiable DIBR operation [68]. In this paper, we choose to use vertical baselines instead of horizontal ones. The analysis of this choice can be found in the Appendix.

2.4.2 Stereo Matching Stage

The second stage of our *PanoDepth* framework is stereo matching. Again, as *PanoDepth* is model-agnostic, any stereo matching network can be plugged in here. Experimental results that show the performance of different stereo matching network settings is discussed in Section 2.5.4.

The stereo matching network we used in this paper follows a similar pipeline as PSMNet [7], with several key modifications. The network consists of five main modules: feature extraction, spherical warping layer, cost volume construction, cost aggregation, and depth prediction. Comparing with the original PSMNet [7], our unique contribution is the Spherical Warping Layer (SWL) which is specifically designed for the 360 stereo geometry.

Feature Extraction After the view synthesis stage, the input image along with all the M synthesized novel views will be passed to a weight-sharing neural network to extract features. We use the same layer setting and keep the SPP module as the original PSMNet [7].

Spherical Warping Layer (SWL) The extracted feature maps of all views are then

used to build a cost volume at multiple depth hypothesis planes for cost matching. An essential step of cost volume construction is to determine the coordinate mapping, which is reflected as disparity, that warps reference view to the target view. Unlike perspective images where the disparity is proportional to the inverse depth [69, 7, 70], disparity of 360 stereo pairs is related to both inverse depth and spherical latitudes. Our SWL performs direct depth sampling instead of disparity sampling which is commonly used in perspective image pair stereo matching. Compared with the learnable shifting filters proposed in [8], our SWL makes the disparity computation adjustable to the pixel latitudinal value, without introducing additional computation overhead in training. Moreover, SWL directly samples on the absolute depth domain, thereby enabling horizontal 360 stereo (which has both horizontal and vertical disparity, more details in the Appendix), the adaptation of 360 multi-view, and the usage of cascade mechanism.

Figure 2.3 shows an illustration of the Spherical Warping Layer. We first sample the inverse depth to cover the whole depth range:

$$\frac{1}{d_j} = \frac{1}{d_{max}} + (\frac{1}{d_{min}} - \frac{1}{d_{max}})\frac{v \times j}{D-1}, j = 0, 1, ..., D-1 \tag{2.1}$$

where D is the total number of hypothesis planes, d_j is the j^{th} depth plane, d_{min} and d_{max} are the minimum and maximum of the depth image, v is the plane interval. The Spherical Warping Layer then transforms depth hypothesis d_j to displacement in spherical domain C_j, to map pixels from the reference synthesized view to the target view. The displacement C_j is defined as:

$$C_{x,j} = 0,\ C_{y,j} = \frac{cos(\theta) \times b}{d_j} \times \frac{H_f}{\pi} \tag{2.2}$$

where θ refers to the pixel-wise latitudinal values, b represents the baseline, and H_f is the

height of the feature map.

Cost Volume Construction The SWL transforms reference view feature maps into the target view domain at the individual hypothesis plane, and then a total of $M + 1$ feature volumes are generated. The variance-based cost volume formation method from MVSNet [61] is used for the fusion of these feature volumes into a compact one. Moreover, we adopt a cascade design from Gu et al. [39] to further improve the final depth quality. Specifically, at level $l(l > 1)$, d_{min} and d_{max} is recalculated based on the prediction of level $l - 1$, then the new depth range and the new number of planes D_l is used to determine the new intervals. Depth hypothesis for level l is then updated using equ (2.1). The corresponding displacements are calculated via the same spherical coordinate mapping procedure.

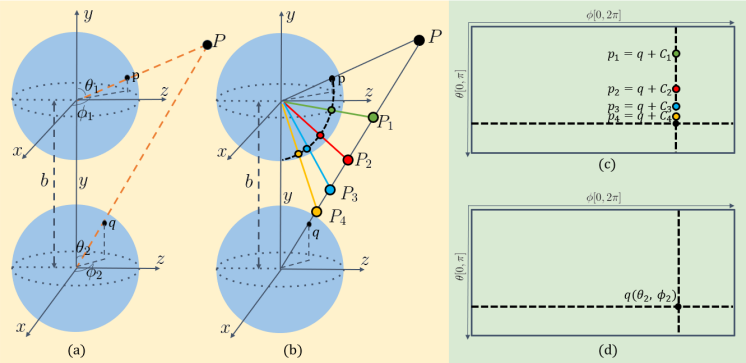


Figure 2.3: Visualization of our spherical warping method. (a) Vertical 360 stereo model. b is the baseline displacement of two cameras. P is a real-world point. The projection of P on two camera space is represented as $p(\phi_1, \theta_1)$ and $q(\phi_2, \theta_2)$. (b) Spherical epipolar geometry. P_i is the points sampled at different depths. (c) Projection of sampled inverse depth on the ERP image. p_i is the projection of P on the top view at sampled points P_i. C_i is the vertical disparity, it equals to C_y in Equ (2.2). (d) Projection on the reference image. q is the projection of P at bottom view.

Cost Aggregation and Depth Prediction After the construction of the cost volume, multi-scale 3D CNN is used to aggregate different levels of spatial context information through the hourglass-shape encoding and decoding network. It has been shown this kind

of cost aggregation module helps to regularize noises in ambiguous regions caused by occlusions, textureless surfaces, and to improve final prediction quality [29, 7, 71]. Finally, we regress the depth value at each level l:

$$\frac{1}{d_{pred,l}} = \frac{1}{d_{min,l}} + (\frac{1}{d_{min,l}} - \frac{1}{d_{max,l}})\frac{k_l}{D_l - 1} \tag{2.3}$$

$$k_l = \sum_{j=0}^{D_l-1} \sigma(p_j) \times (v_l \times j) \tag{2.4}$$

where k_l is the sum of each plane level weighted by its normalized probability, $\sigma(\cdot)$ represents softmax function, p_j denotes the probability of j^{th} plane value, v_l is the interval for level l.

2.4.3 Loss Function

PanoDepth is trained in an end-to-end fashion, supervision is applied on both stages. The final loss function is defined as follows,

$$\mathcal{L}_{total} = \omega_1 \mathcal{L}_{coarse} + \omega_2 \mathcal{L}_{stereo} \tag{2.5}$$

where ω_1 and ω_2 are the weights of coarse depth estimation loss and stereo matching loss respectively. For the optimization on the first stage coarse estimation, we use inverse Huber (berHu) loss as proposed in [36]:

$$\mathcal{L}_{coarse} = \frac{1}{\Omega} \sum_{i \in \Omega} \mathcal{L}_{berHu}(d_i, \quad d_i^*) \tag{2.6}$$

where Ω is a binary mask that is used to mask out missing regions (pixels that have depth values smaller than d_{min} or greater than d_{max}), d_i and d_i^* are the ground truth and the predicted depth value of a valid pixel i respectively. For stereo matching, we calculate berHu loss [36] on all outputs from each level l and then compute the weighted summation. The stereo matching loss is defined as:

$$\mathcal{L}_{stereo} = \frac{1}{\Omega} \sum_{i \in \Omega} \sum_{l=1}^{N} \lambda_l \mathcal{L}_{berHu}(d_i, \ \ d_i^*) \tag{2.7}$$

where λ_l is the level l stereo loss weight.

2.5 Experiments

2.5.1 Datasets

We train and evaluate our network on three panorama RGBD benchmark datasets including, Stanford2D3D [1], 360D [2] and the omnidirectional stereo dataset [6].

Stanford2D3D Stanford2D3D [1] dataset consists of 1413 real-world panorama images from six large-scale indoor areas. We follow the official train-test split which uses the fifth area for testing, and other areas for training. We resize the images to 256×512 to reduce computation time.

360D 360D [2] is a RGBD panorama benchmark provided by Zioulis et al. [2]. It is composed of two other synthetic datasets (SunCG and SceneNet), and two real-world datasets (Stanford2D3D and Matterport3D). There are 35,977 panorama RGBD images in the 360D that are rendered from the aforementioned four datasets. We again follow the default train-test splits.

Omnidirectional Stereo Dataset. The omnidirectional stereo dataset [6] consists of 7964 stereo pairs of panorama RGBD images rendered from two real-world datasets, Matterport3D [9] and Stanford2D3D [1]. We use the train-test split that removes 3 complete buildings from Matterport3D [9] and 1 complete area from Stanford2D3D [1] for test. Each set of data consists of left-down, right, and up view 360 RGBD images in a triangular fashion with size 256×512. We only use images from the left-down view in our single view depth estimation experiments. Up-down stereo pairs are only used for the ablation study of the stereo matching network.

To investigate the impact of baselines and FoV on the quality of view synthesis, we create a new dataset that is rendered from the mesh of Stanford2D3D [1]. More details of this new dataset as well as the experiments with various baseline configurations are included in the Appendix.

2.5.2 Implementation Details and Metrics

For parameter settings, we use a default of $N = 2$ levels, with $D_1 = 48$ and $D_2 = 24$ hypothesis planes respectively. The minimum and maximum depth d_{min} and d_{max} for the first level is set to be $0.2m$ and $8m$. We use a default of $M = 3$ synthesized views rendered at vertical baseline placements $-0.24m, +0.24m, +0.4m$. The loss weights ω_1 and ω_2 are set to 1 and 0.02. We train our framework from scratch using Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with a batch size of 8. Initial learning rates for the first and the second stage are set to 0.0002 and 0.0005. We separately train the coarse network for 10 epochs and then train the entire framework end-to-end for 200 epochs. Both of the learning rates decay by a factor of 0.5 every 30 epochs. Performances are evaluated based on commonly used depth quality measures [24]: absolute relative error (Abs Rel),

square relative error (Sq Rel), linear root mean square error(RMSE) and its natural log scale (RMSE log) and inlier ratios ($\delta_i < 1.25^i$, $i \in \{1, 2, 3\}$).

2.5.3 Overall Performance Comparison with the State-of-the-art Algorithms

Table 3.1 lists quantitative comparison between *PanoDepth* and other state-of-the-art omnidirectional monocular depth estimation methods [36, 2, 3, 4] on both Stanford2D3D [1] and 360D [2] datasets. As shown in Table 3.1, our method is able to reduce Abs Rel error by 19.60% on Stanford2D3D and 25.85% on 360D compare to the current leading 360 monocular depth estimation approach BiFuse [3]. Note that BiFuse [3] uses a network architecture with more than 200M parameters and has a large computation complexity for sharing information between CubeMap [72] and ERP formats. The framework with distortion-aware module proposed in [45] outperforms ours but it has more than 60M parameters. Our framework has only around 16M parameters with a smaller computation overhead. Comparing the performance with ODE-CNN [4] on 360D, our approach achieves comparable results while ODE-CNN requires additional depth sensor input. Figure 2.4 shows the qualitative comparison with the state-of-the-art approaches. As we can see, our method generates high-quality depth with a detailed surface, sharp edges, and precise range.

2.5.4 Ablation Studies

Spherical Warping Layer In order to evaluate the performance of the *PanoDepth* stereo matching module with the novel Spherical Warping Layer (SWL), we compare it to the state-of-the-art stereo matching approaches, PSMNet [7] and 360-SD Net [8], with ground truth up-down 360 stereo pair as input on the Omnidirectional Stereo Dataset [6]. In the

Datasets	Methods	Abs Rel↓	RMSE↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
Stanford2D3D [1]	FCRN [36]	0.1837	0.5774	0.7230	0.9207	0.9731
	RectNet [2]	0.1409	0.4568	0.8326	0.9518	0.9822
	BiFuse with fusion [3]	0.1209	0.4142	0.8660	0.9580	0.9860
	Joint wth layout and semantics [5]	0.0680	0.2640	0.9540	0.9920	0.9980
	PanoDepth(Ours)	0.0972	0.3747	0.9001	0.9701	0.9900
360D [2]	FCRN [36]	0.0699	0.2833	0.9532	0.9905	0.9966
	RectNet [2]	0.0702	0.2911	0.9574	0.9933	0.9979
	Mapped Convolution [34]	0.0965	0.2966	0.9068	0.9854	0.9967
	Distortion-aware [45]	0.0406	0.1769	0.9865	0.9966	0.9987
	BiFuse with fusion [3]	0.0615	0.2440	0.9699	0.9927	0.9969
	ODE-CNN [4]	0.0467	0.1728	0.9814	0.9967	0.9989
	PanoDepth(Ours)	0.0456	0.1955	0.9830	0.9957	0.9984

Table 2.1: A quantitative comparison with the state-of-the-art approaches on Stanford2D3D [1] dataset and 360D [2] dataset (↓ represents lower the better, ↑ represents higher the better). We report the results based on the original papers [2, 3, 4] using the same evaluation metrics. Note that ODE-CNN [4] requires additional depth sensor input besides the 360 image used by other methods listed in the table. Additional supervision signals including layout and semantics are used in [5]. For our *PanoDepth*, we use the default stereo network setting with three synthesized views and a two-level cascade design.

Methods	Abs Rel↓	Sq Rel↓	RMSE↓	RMSElog↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
(1) PSMNet [7], sample on disparity, $D = 64$	0.0433	0.0252	0.2541	0.1340	0.9722	0.9833	0.9900
(2) 360SD-Net [8], sample on disparity, $D = 64$	0.0387	0.0198	0.2286	0.0955	0.9776	0.9900	0.9940
(3) PanoDepth (ours), one-level, $D = 32$	0.0253	0.0222	0.2268	0.0686	0.9756	0.9874	0.9976
(4) PanoDepth (ours), one-level, $D = 64$	0.0229	0.0087	0.1731	0.0606	0.9900	0.9969	0.9987
(5) PanoDepth (ours), two-level, $D_1 = 48, D_2 = 24$	**0.0178**	**0.0064**	**0.1415**	**0.0519**	**0.9928**	**0.9976**	**0.9990**

Table 2.2: A quantitative comparison between the PanoDepth stereo matching and existing stereo matching networks on the Omnidirectional stereo dataset [6] where up-down stereo pairs are used as input and output the depth of bottom view. Our proposed stereo matching module (3,4,5) outperforms both (1) PSMNet [7] and (2) 360SD-Net [8]. The two cascade level setting achieves the best performance.

experiment, we use the officially released code of both approaches, and convert the output disparity into depth for evaluation. We set the number of depth hypothesis planes to 64 to ensure fair and consistent comparison. We can see from Table 2.2 that our proposed stereo matching method with SWL outperforms PSMNet [7] and 360-SD Net [8], even with one-

	1st Stage	2nd Stage w/ 1 synthesize view	#params	Abs Rel↓	Sq Rel↓	RMSE↓	RMSElog↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
(1)	CoordNet	N/A	6.1M	0.1264	0.0888	0.4456	0.2084	0.8533	0.9588	0.9813
(2)	RectNet	N/A	10.8M	0.1409	0.0859	0.4568	0.2124	0.8326	0.9518	0.9822
(3)	CoordNet	PSMNet [7], D=32	13.0M	0.1206	0.0833	0.4293	0.2150	0.8671	0.9548	0.9790
(4)	CoordNet	w/ SWL, one-level, D=32	13.0M	0.1132	0.0686	0.4077	0.1869	0.8757	0.9652	0.9863
(5)	RectNet	w/ SWL, one-level, D=32	17.6M	0.1192	0.0775	0.4202	0.1960	0.8655	0.9607	0.9846
(6)	CoordNet	w/ SWL, two-level, D_1=32, D_2=16	16.6M	**0.1040**	**0.0645**	**0.3918**	**0.1827**	**0.8865**	**0.9676**	**0.9875**
(7)	RectNet	w/ SWL, two-level, D_1=32, D_2=16	21.3M	0.1138	0.0761	0.4274	0.1961	0.8711	0.9577	0.9837

Table 2.3: An ablation study of the impact of various combinations of coarse estimation network and stereo matching network on the final performance. The experiments are trained on Stanford2D3D [1]. We use two types of coarse estimation networks, (1) CoordNet, and (2) RectNet [2]. We can see that even with one synthesize view, our proposed two-stage *PanoDepth* pipeline (3,4,5,6,7) is able to outperform the one-stage-only methods (1,2). Adding Spherical Warping Layer (SWL) (4,5,6,7) and two cascade levels (6,7) further improves the performance. The experimental results indicate that our two-stage pipeline is model-agnostic under various network settings.

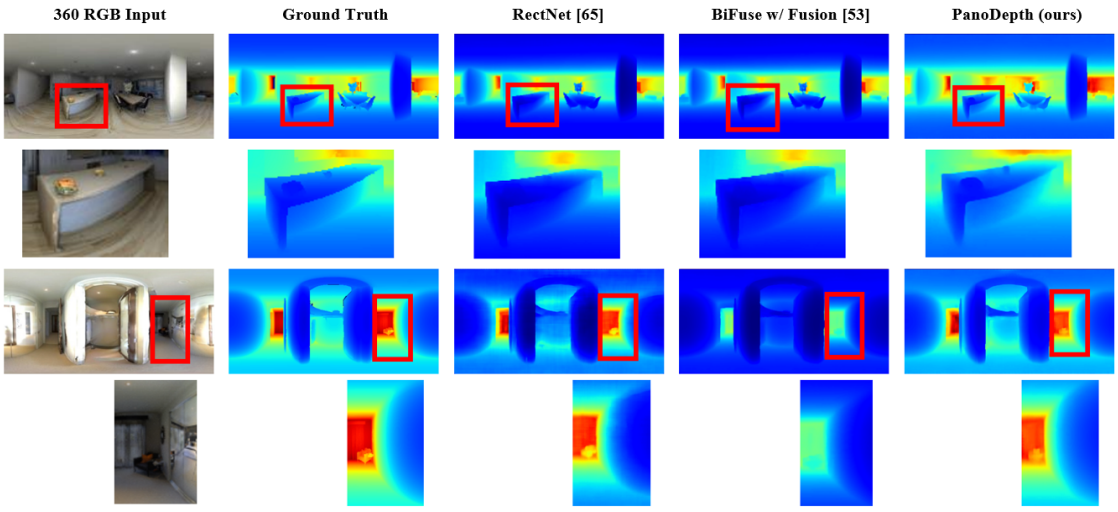


Figure 2.4: A qualitative comparison between RectNet [2] (3rd column), BiFuse [3] (4th column), and our method (5th column) on 360D [2]. We highlight and zoom in some areas that distinguish the performance of three methods. We can see that our *PanoDepth* is able to produce sharp edges, predict depth range accurately, and recover surface detail.

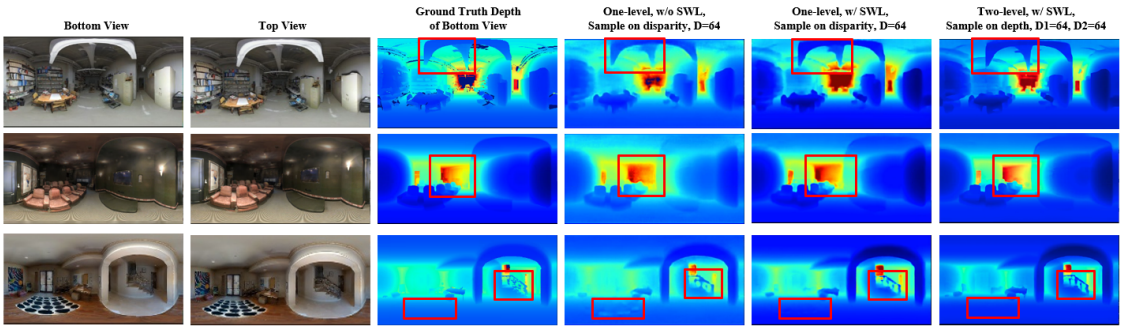


Figure 2.5: A qualitative comparison to show the effectiveness of SWL in the PanoDepth stereo matching module. We compare between one-level stereo matching method without SWL(4th column), one-level with SWL (5th column), and two-level with SWL (last column). The experiments are trained on Omnidirectional Stereo Dataset [6]. Our stereo matching module with SWL recovers clear details and shows fewer artifacts than the one without SWL (see highlighted areas).

level setting. Moreover, by adding SWL, our one-level setting outperforms the one without SWL (identical to PSMNet[7]) by 47% in terms of Abs Rel. with the same 64 sampling planes.

Qualitative illustrations of the effectiveness of SWL is shown in Figure 2.5.

Model-agnostic Evaluations In Table 2.3, we further test the performance of the full *PanoDepth* pipeline given different variations of stereo matching networks on Stanford2D3D dataset [1]. Comparing with the single-stage coarse depth estimation, all two-stage configurations show better performances. By adding a light-weight one-level stereo matching network with 32 depth plane in the second stage, *PanoDepth* can already reach comparable performance to BiFuse [3]. The performance can be further improved by introducing SWL, adding more cascade levels, and using more sophisticated coarse depth estimation.

In addition, by comparing the performance of two-stage approaches with two different backbones, CoordNet and RectNet, we can also postulate that coarse depth, which has an impact on the synthetic image quality, is positively correlated with the final depth predic-

tion.

More ablation studies regarding the number of synthesized views, the number of hypothesis depth planes, and the comparison between one-stage alternatives (e.g., multi-tasking and adding depth refinement) and our two-stage method can be found in the Appendix.

2.6 Conclusion and Future Work

In this paper, we demonstrate a technique that leverages view synthesis and stereo constraints to advance monocular depth estimation performance that can be applied on 360 images. We propose a novel model agnostic two-stage framework *PanoDepth* for generating dense high-quality depth from a monocular 360 input. Extensive experiments show that *PanoDepth* outperforms state-of-the-art approaches by a large margin. Our stereo matching sub-network in the later stage adapts to the 360 geometry and achieves top-ranking performance in 360 stereo matching. We believe the good performance of PanoDepth could draw more interests from both the industry and academia to 360 images for its still under-explored capability in tasks such as depth estimation. We hope our work can motivate more research and applications in 360 images. There are several research venues we would like to further explore in the future, such as alternative view synthesis methods like [73, 74], and 360 depth estimation in outdoor scenarios for applications like autonomous driving.

Chapter 3

OmniFusion: 360 Monocular Depth Estimation via Geometry-Aware Fusion

3.1 Introduction

A 360 image provides a comprehensive view of the scene with its wide field of view (FoV), which is beneficial in understanding the scene holistically. However, commonly used 360 image representation format such as the equirectangular projection (ERP) image can introduce geometric distortions. The distortion factor varies in the vertical direction and may degrade the performance of regular convolutional layers designed for non-distorted perspective images. Many studies have been proposed to address the distortion issue. [42, 45, 44] proposed distortion-aware convolutions or spherical customized kernels. However, it remains unclear how effective such spherical convolutions are, especially in deeper layers [3, 42]. Some spherical CNNs [75, 76] defined convolution in the spectral domain, with potentially heavier computation overhead. Attempts have also been made to tackle the

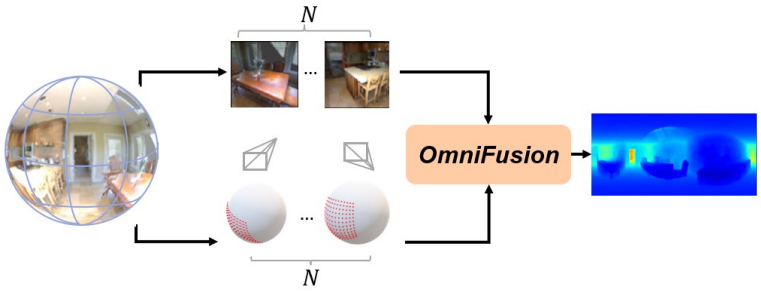


Figure 3.1: Our method, *Omnifusion*, produces high-quality dense depth (shown as the image on the right) from a monocular ERP input (shown as an image wrapped on a unit sphere on the left). Our method uses a set of N perspective patches (i.e. tangent images) to represent the ERP image (top branch), and fuse the image features with 3D geometric features (bottom branch) to improve the estimation of the merged depth map. The corresponding camera poses of the tangent images are shown in the middle row.

ERP distortion via other less-distorted formats. BiFuse [3] and UniFuse [15] took complementary properties from ERP and cubemap. Several works [77, 78] applied regular CNNs repeatedly to multiple perspective projections of the 360 image. Recently, Eder et al. [13] proposed to use a set of subdivided icosahedron tangent images, and demonstrated that using tangent image representation can facilitate the network transfer between perspective and 360 images.

It is advantageous to use tangent images [13] as it has less distortion, and can make good use of the large pool of pre-trained CNNs developed for perspective imaging. Additionally, the tangent image representation even inherit a superior scalability to handle high resolution inputs compared to those holistically method. However, the vanilla pipeline [13] has some limitations. First, severe discrepancies occur between perspective views since the same object may appear differently from multiple views (an example is shown in Figure 3.3). This issue is especially problematic for the depth regression task, since the inconsistent depth scale estimated from individual tangent images creates undesired artifacts during merging. Second, the advantage of estimating depth from holistic 360 image is unfortu-

nately lost, because of the decomposition of the global scene into local tangent images. The predictions from the tangent images are independent of each other and there is no information exchange between tangent images.

In this paper, we present *OmniFusion*, a 360 monocular depth estimation framework with geometry-aware fusion (see Figure 3.1). We proposed the following three key components to solve the aforementioned discrepancy issue and merge the depth results of tangent images seamlessly. First, we use a geometric embedding module to provide additional features to compensate for the discrepancy between 2D features from patch to patch. For each patch, we calculate the 3D points located on the spherical surface that correspond to the patch pixels, encode them and the patch center coordinate through shared Multi-layer Perceptron (MLP), and add the geometric features to the corresponding 2D features. Second, to regain the holistic power in understanding the entire scene, we incorporate a self-attention-based transformer in our pipeline. With the transformer, patch-wise information is globally aggregated to enhance the estimation of the global scale of depth, and to improve the consistency between patch-wise results. Third, we introduce an iterative refining mechanism, where more accurate 3D information from the predicted depth maps is fed back to the geometric embedding module to further improve the depth quality in an iterative manner.

We test *OmniFusion* on three benchmark datasets: Stanford2D3D [1], Matterport3D [9], and 360D [2]. Experimental results show that our method outperforms state-of-the-art methods by a significant margin on all of these datasets.

Our contributions can be summarized as follows:

- We present a 360 monocular depth prediction pipeline that addresses the distortion issue via geometry-aware fusion and achieves the state-of-the-art performance.

- We introduce a geometric embedding network to provide 3D geometric features to mitigate the discrepancy in patch-wise image features.
- We incorporate a self-attention-based transformer to globally aggregate patch-wise information which enhances the estimation of the physical scale of depth.
- We propose an iterative mechanism to further improve the depth estimation with structural details.

3.2 Related Works

3.2.1 Monocular depth estimation

Monocular depth estimation, which takes a single RGB image as input to predict pixel-wise depth value, has been extensively investigated due to its broad applications. Early works mainly focused on network architecture and supervision [24, 36, 37]. Recently, researchers has been investigating the use of unsupervised learning on stereo pairs [79, 80, 55] or monocular video streams [81, 82] to expand training data to unlabelled image sequences for broader applications. However, such approaches are still sensitive to many factors (e.g. camera intrinsic changes), and very challenging to be generalizable to new scenes. To improve the robustness and scalability, some methods utilize additional sensor input such as LiDAR and RGBD camera [83, 4]. However, the extra computation or power consumption are not welcomed in many practical scenarios.

3.2.2 360 depth estimation

Monocular depth estimation from 360 images has been investigated from a variety of perspectives. Zioulis et al. [6] explored the spherical stereo geometry and estimated depth from monocular ERP input via spherical view synthesis. Eder et al.[35] and Zeng et al.[5] explored joint learning from different modalities (e.g. layout, normal, semantics, etc.). HoHoNet [14] proposed to utilize latent horizontal feature representation to encode ERP image features. To handle the irregular distortion of ERP images, several distortion-aware convolutions [33, 84, 45, 44, 42] have been proposed. For example, Fernandez et al. [84] introduced EquiConv which applied deformable convolution to accommodate spherical geometry.

Tateno et al.[58] proposed to apply regular CNN to perspective images during training, and distortion-aware convolution during testing. Instead of directly tackling the distortion of ERP, several approaches proposed to use other representations with less distortion, such as cubemap [85, 86], fusion between ERP and cubemap [3, 15], and multiple perspective projections of 360 images[77, 78]. A recent work by Eder et al. [13] proposed to use tangent images, a set of oriented, low-distortion images rendered tangent to faces of the icosahedron, to represent a 360 image. It is advantageous to use tangent images since it has less distortion and can effectively leverage pre-trained CNN models developed for perspective imaging.

However, discrepancies between tangent images are not addressed in [13], which leads to a downgrade of the final merged result. In this work, we follows the paradigm proposed in [13] of using tangent images, but simplified and adapted it for depth estimation. In addition, we successfully address the discrepancy issue by incorporating geometry-aware fusion and the transformer.

3.2.3 Transformer

Originally proposed in natural language processing [87], the transformer architecture has since been widely used in computer vision tasks such as image classification [88], depth estimation [89], object detection [90], and semantic segmentation [91, 92]. The visual transformer has a natural fit with monocular depth estimation as long-range context can be explicitly exploited by the self-attention module. When applying transformer to 360 images, the distortion however, can decrease the power of the transformer in exploiting the pairwise correlation between patches. In this work, we feed the transformer with distortion-free, and geometry-aware input, so that the transformer can focus on the global aggregation of patch-wise information.

3.3 Method

Figure 3.2 shows an overview of the full pipeline of the proposed *OmniFusion* framework. First, an ERP input image is transformed into a set of tangent images via gnomonic projection (Figure 3.3). The projected distortion-free tangent images are then passed through an encoder-decoder network to produce patch-wise depth estimations, which are later fused into an ERP depth output. To ease the patch-wise discrepancy, we introduce a novel geometric embedding module that encodes the spherical coordinate associated with each tangent image pixel, providing additional geometric features to facilitate the integration of patch image features. To further improve the consistency between patch-wise predictions and to better estimate the global depth scale, the features from the deepest level of the encoder are globally aggregated through a self-attention-based transformer. Finally, an iterative refining mechanism is adopted to further improve the depth quality. We update the

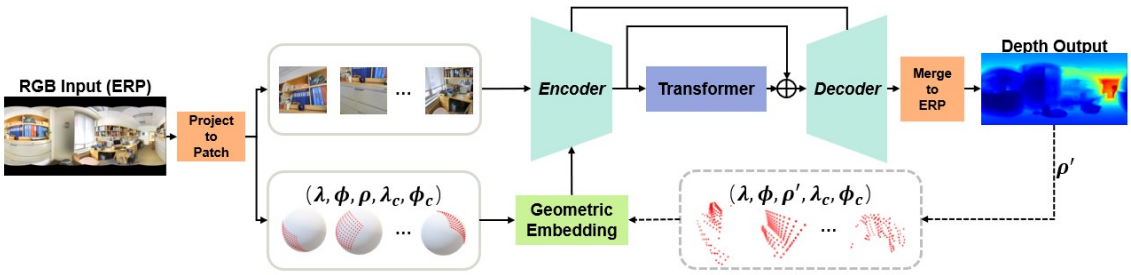


Figure 3.2: An overview of our proposed *OmniFusion*. Our method takes a monocular RGB image in ERP format as input, projects it onto multiple patches at multiple viewpoints, and processed each distortion-free patch with an encoder-decoder network to produce patch-wise depth maps (top-stream). The patch-wise outputs are merged into a final ERP depth map in the end. Meanwhile, the corresponding points located on the spherical surface are sampled and passed through a geometric embedding network to produce geometric features (bottom-stream). The geometric features are fused into the image encoder to compensate for the patch-wise discrepancy and to improve the quality of the merged result. For each sampled point, we use its spherical coordinates (λ, ϕ, ρ), together with the tangent plane center coordinates (λ_c, ϕ_c) as input attributes to the geometric embedding network which provides the necessary information to align 2D features. A transformer architecture is integrated to conduct global aggregation of the deep patch-wise feature which further improves the consistency of patch-wise outputs. Moreover, we incorporate an iterative refining mechanism (visualized in dashes), to further improve the depth recovery. In particular, ρ value is updated according to the depth estimated from the previous iteration.

spherical coordinates iteratively based on the more accurate estimation obtained from the previous iteration. We train our network in an end-to-end fashion, with the only supervision being the final merged depth compared to the ground truth.

3.3.1 Depth estimation from tangent images

We use the distortion-free tangent image representation to address the irregular 360 image distortion. Tangent image is the *gnomonic projection* of a sphere surface onto a flat, rectangular plane surface. The gnomonic projection [93] (Figure 3.3) is a map projection obtained by projecting points P_s on the surface of sphere from a sphere's center O to point

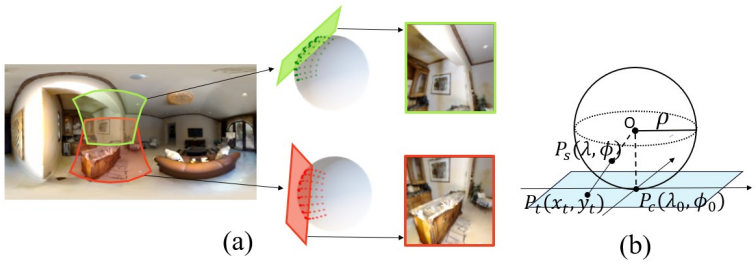


Figure 3.3: (a) An example of tangent image projection. Two tangent images are projected from two different viewpoints. The corresponding areas are highlighted with the same color in both ERP and tangent patches. As illustrated, there usually exist overlapping areas between two neighboring patches, and the same object may appear differently in different patches. (b) The illustration of the gnomonic projection. A point $P_s(\lambda, \phi)$ located on the spherical sphere is projected onto a point $P_t(x_t, y_t)$ on the flat plane which is tangent to a point $P_c(\lambda_c, \phi_c)$.

P_t in a plane that is tangent to a point P_c.

For a pixel on the ERP image $P_e(x_e, y_e)$, we first find its corresponding point $P_s(\lambda, \phi)$ locating on the unit sphere.

$$\lambda = \frac{2\pi x_e}{W}, \ \phi = \frac{\pi y_e}{H} \tag{3.1}$$

where H and W are height and width of the ERP image. The projection from $P_s(\lambda, \phi)$ to $P_t(x_t, y_t)$ is defined as:

$$\begin{aligned} x_t &= \frac{\mathbf{cos}(\phi)\mathbf{sin}(\lambda - \lambda_c)}{\mathbf{cos}(c)} \\ y_t &= \frac{\mathbf{cos}(\phi_c)\mathbf{sin}(\phi) - \mathbf{sin}(\phi_c)\mathbf{cos}(\phi)\mathbf{cos}(\lambda - \lambda_c)}{\mathbf{cos}(c)} \\ \mathbf{cos}(c) &= \mathbf{sin}(\phi_c)\mathbf{sin}(\phi) + \mathbf{cos}(\phi_c)\mathbf{cos}(\phi)\mathbf{cos}(\lambda - \lambda_c) \end{aligned} \tag{3.2}$$

where (λ_c, ϕ_c) are the spherical coordinates of the tangent plane center P_s.

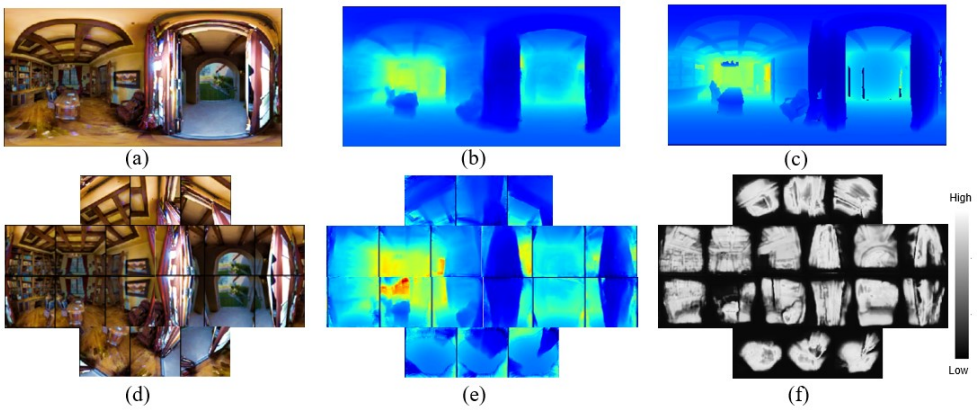


Figure 3.4: Visualization of representations involved in different pipeline components. First row: (a) An example of an RGB input image in ERP, (b) the final merged predicted depth map in ERP, (c) the ground truth depth map in ERP used for training. Second row: (d) RGB tangent image patches generated from the ERP input, (e) the patch-wise estimated depth maps, (f) the patch-wise estimated confidence maps that facilitates the depth merging where the final depth map (b) is calculated as the weighted average of all patches (e) given the confidence maps (f) as weights.

The inverse gnomonic transformations are:

$$\begin{aligned} \lambda &= \lambda_c + \mathbf{tan}^{-1}\left(\frac{x_t\ \mathbf{sin}(c)}{\gamma\ \mathbf{cos}(\phi_1)\mathbf{cos}(c) - y_t\ \mathbf{sin}(\phi_c)\mathbf{sin}(c)}\right) \\ \phi &= \mathbf{sin}^{-1}\left(\mathbf{cos}(c)\mathbf{sin}(\phi_c) + \frac{1}{\gamma} y_t \mathbf{sin}(c)\mathbf{cos}(\phi_c)\right) \end{aligned} \tag{3.3}$$

where $\gamma = \sqrt{x_t^2 + y_t^2}$ and $c = \mathbf{tan}^{-1}\gamma$.

With Equation 3.2 and 3.3, we can build one-to-one forward and inverse mapping functions between pixels on the ERP image and pixels on the tangent image.

In our experiments, we use a set of $N = 18$ tangent images for a balance of speed and performance (A related ablation study can be found in Section 3.4.4). Tangent images are sampled at four different latitudes: $-67.5°, -22.5°, 22.5°, 67.5°$, and we sample $3, 6, 6, 3$ patches on each of these latitudes, respectively (Figure 3.4). All tangent images

share the same resolution and FoV. We chose this non-uniform sampling based on the fact that tangent images of the same resolution can cover different ranges of longitude when the centered at different latitudes. To ensure the sampled patches near the poles do not overlap to an extreme extent, we take fewer samples to cover the near-pole area in the ERP space. Since the generated tangent images are distortion-free, we can easily apply regular encoder-decoder CNN architectures to predict a depth map from each tangent image. For better convergence and accuracy, we leverage high-performance pre-trained networks (e.g., ResNet [37]) when initializing our encoder. We pass all N tangent images simultaneously through the encoder, and obtain N feature maps that will be used as tokens later in the transformer. For the decoder, we use a stack of upsampling layers followed by 3×3 convolutions, with skip-connections from the encoder.

Without introducing additional modules, the baseline presented so far can be considered as a customized version of [13]. Specifically, we introduce a simplified paradigm in generating tangent images, and adopt a basic encoder-decoder network for the depth estimation task.

3.3.2 Geometry-aware feature fusion

The simplicity of predicting depth maps from tangent images nonetheless comes with a cost. As the depth estimation is now conducted independently, a globally consistent depth scale is no longer guaranteed. Furthermore, as shown in Figure 3.3 (a) and Figure 3.4 (d), an object (e.g. the painting on the wall in Figure 3.3 (a)) will be projected onto multiple tangent images from various angles and therefore will be encoded differently in different tangent images. Discrepancies between patch depth estimations, especially in overlapping areas, can result in significant artifacts in the final merged ERP depth map (Figure 3.5 (e)).

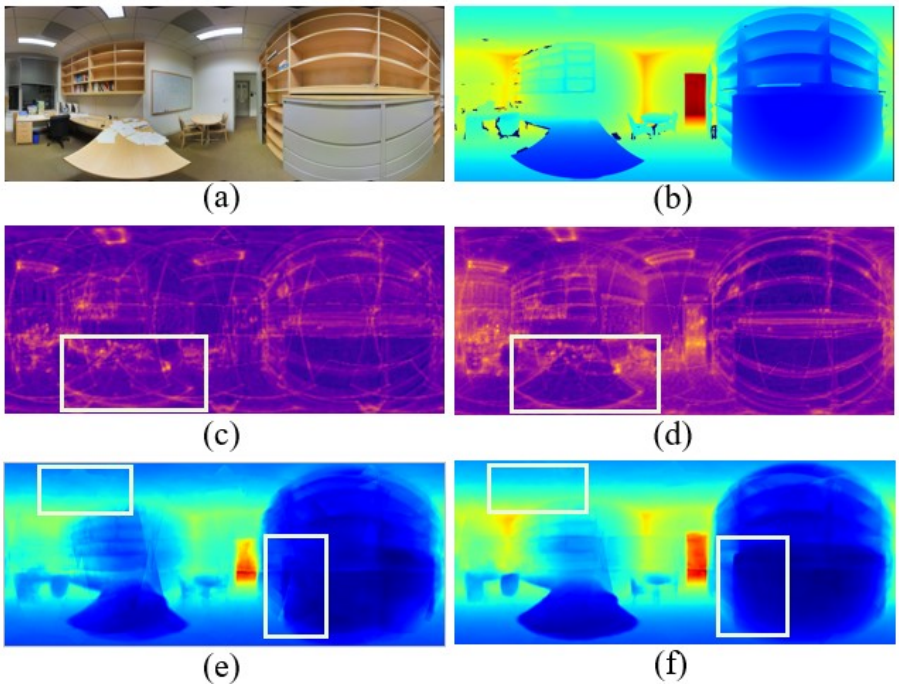


Figure 3.5: An illustration of the effectiveness of geometry-aware feature fusion. An ERP RGB image is shown in (a), the ground truth depth is shown in (b). Visualizations of the feature map and the final depth map from the baseline are shown in (c) and (e) respectively. For comparison, (d) and (f) show the feature map and the final depth map out of the proposed *OmniFusion*, where the image features are fused with geometric features. Observe that our method yields a more self-consistent feature map and a more structural depth map compared to the baseline, especially in regions highlighted in rectangles.

To compensate for the differences between patch-wise image features, we introduce a *geometric embedding* network (see Figure 3.2) to provide additional geometric information. For a pixel $P_t(x_t, y_t)$ located on a tangent image, we use its corresponding spherical coordinates located on the unit sphere, $P_s(\lambda, \phi, \rho)$, together with the center of the tangent image $P_c(\lambda_c, \phi_c)$, as the input attributes of the geometric embedding network. P_s makes the embedding aware of the global position, e.g., to tell whether two image pixels from two patches relate to the same spherical coordinates. However, geometric features out of P_s alone can not align different 2D features. To this end, P_c is taken as additional attributes to

make the embedding able to differentiate from patch to patch, such that the learned geometric features can make the patch features tend to be consistent. Through the combination of the tangent image features and the geometric features as well as an end-to-end learned network, the adjusted features lead to a much cleaner merged depth. As observed in Figure 3.5 (d), the extracted image features with geometric embedding show much better consistency in the feature map merged in the ERP space, compared to features without geometric embedding as shown in Figure 3.5 (c). Consequently, the final depth map out of *OmniFusion* shown in Figure 3.5 (f) appears to be much cleaner compared to the baseline depth map shown in Figure 3.5 (e).

The geometric embedding network consists of two layers of MLPs, and encodes the 5-channel spherical attributes into 64-channel feature maps. We fuse this geometric embedding with image features at the same pixel location in the encoder via element-wise summation. In order to maintain more structural details, early fusion is adopted. Geometric features are added to the *layer1* of the ResNet encoder where we experimentally achieved the best performance. It is worth noting that the additional computational cost associated with the geometric embedding module is minimal compared to the original encoder-decoder (Table 3.2). The geometric features for the first iteration are even fixed once learned, since they are independent from image inputs. Only the second iteration requires to re-compute the geometric features.

3.3.3 Global aggregation with transformer

When decomposing the ERP into a sequence of tangent images, we no longer have the holistic view of the 3D environment. To make up for this loss, we leverage the transformer architecture to aggregate information from the patches in a global fashion. The global

aggregation is expected to improve the consistency of depth estimations from patches, and to better regress the global scale of depth out of a larger FoV.

Using the feature maps extracted from the encoder, we first apply a 1×1 convolution layer to reduce channel dimensions for better efficiency. Then we flatten the feature maps into N 1-D feature vectors $X_0 = [x^1, x^2, ..., x^N] \in R^{N \times d}$ which will be used as tokens in the transformer. The learnable positional embedding $E_{pos} \in R^{N \times d}$ are added to the feature tokens to retain positional information in a similar way as proposed in [88]. Through the self-attention architecture, the transformer learns to globally aggregate the information from all the patches to adjust the features from each patch, where the aggregation weights account for the pairwise correlation both from the visual features and the positional features. The architecture of the multi-head attention transformer follows [87].

3.3.4 Depth merging with learnable confidence map

The aforementioned geometric embedding and transformer modules significantly reduce discrepancies among different patch-wise depth estimations. Yet, the depth merging does not achieve a pixel-level seamless fusion. To further improve the merging (Figure 3.4 (b)), we ask the network to simultaneously predict a confidence map for each patch besides depth regression. The merged depth is then computed as a weighted average of all patch depth predictions with confidence scores used as weights. In detail, two separate regression layers are appended to the decoder, one for depth regression, the other for confidence score regression. Both the depth maps (Figure 3.4 (e)) and confidence maps (Figure 3.4 (f)) are mapped to ERP domain following the inverse gnomonic transformation before merging. (More details are included in the Appendix.)

3.3.5 Iterative depth refinement

The geometric embedding utilizes the spherical coordinates (λ, ϕ, ρ) corresponding to tangent image pixels for geometry-aware fusion. ρ is initially fixed as no depth information is available. The depth information will be available after one iteration, which can be used to update ρ and provide more accurate geometry information for the geometric embedding module. Based on this observation, we propose an iterative depth refinement scheme (see Figure 3.2).

In the first iteration (Section 3.3.2), the spherical coordinates (λ, ϕ, ρ) of points located on the unit sphere are used for geometric embedding. For the subsequent iterations, we update $\rho \rightarrow \rho'$, using the new depth value estimated from the previous iteration (the depth of ERP image is defined as the distance from the real-world point to the camera center). The updated attributes with more accurate geometry will be passed into the geometric embedding network in the next iteration. An ablation study is presented in section 3.4 to demonstrate the effectiveness of more accurate geometric embedding.

3.4 Experiments

3.4.1 Datasets

OmniFusion is tested on three well-known benchmark datasets: Stanford2D3D [1], Matterport3D [9], 360D [2].

Stanford2D3D [1] dataset consists of 1,413 real world panorama images from six large-scale indoor areas. We follow the official train-test split which uses the fifth area for testing, and others for training. We use resolution 512×1024.

Matterport3D [9] contains a total of 10,800 indoor panorama RGBD images. We follow the official split which takes 61 rooms for training and the rest for testing. We use resolution 512×1024 in our experiments.

360D [2] is a RGBD panorama benchmark provided by Zioulis et al. [2]. It is composed of two other synthetic datasets (SunCG and SceneNet), and two real world datasets (Stanford2D3D and Matterport3D). There are 35,977 photo-realistic panorama RGBD images in the 360D that are rendered from the aforementioned four datasets. We follow the default train-test splits and use resolution 256×512.

3.4.2 Implementation details

We adopt the same quantitative evaluation metrics as used in [36, 2], including Absolute Relative Error (Abs Rel), Root Mean Squared Error (RMSE), Root Mean Squared Error in logarithmic space (RMSE(log)) and accuracy with a threshold δ_t, where $t \in 1.25, 1.25^2, 1.25^3$. Arrows next to the metric indicate the direction of better performance in all tables. We implement our network using PyTorch and train it on two Nvidia RTX GPUs. We use the default setting of Adam optimizer [94] and a initial learning rate of 0.0001 with cosine annealing [95] learning rate policy. We train Stanford2D3D [1] for 80 epochs, and 60 epochs for Matterport3D [9] and 360D [2]. The default number of patches we use is 18. The default patch size we use for Stanford2D3D [1] and matterport [9] is 256×256, the patch FoV is 80°. For 360D [2], we use 128×128 as patch size. We leverage pre-trained ResNet [37] as image encoder in these experiments. The network is trained end-to-end, the same model is used for all iterations. For the loss function, following [58, 3], we adopt BerHu loss [36] for depth supervision. The final loss is the summation of depth losses from all iterations.

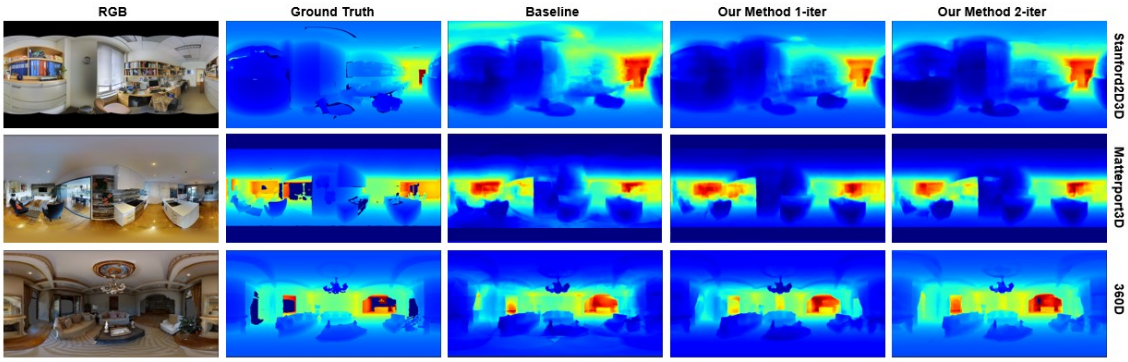


Figure 3.6: Qualitative results on Stanford2D3D [1], Matterport3D [9] and 360D [2]. From left to right: ERP image input, ground truth depth, depth output from the baseline, depth output from our method (1-iter), and our method (2-iter). In comparison to the baseline, which is directly tailored from [13], our method (1-iter, 2-iter) leads to more structural depth maps, which appear sharp along those object boundaries and smooth within surfaces.

3.4.3 Overall performance

We present our model performances and compare it to the existing methods in Table 3.1. We omit the methods that use supervision signals other than depth [35, 5] and the self-supervised approaches [6] for fair comparison. For all datasets, we show our results with 1-iteration (1-iter) and 2-iterations (2-iter). We demonstrate in Table 3.1 that even with 1-iter setting, our method is able to outperform all the existing methods on Matterport3D [9], and achieve on par performance with current state-of-the-arts on 360D. With 2-iter setting, our method outperforms BiFuse [3] by 21.4% (Abs Rel) on Stanford2D3D, 56.1% (Abs Rel) on Matterport3D, 30% (Abs Rel) on 360D. Comparing to UniFuse [15], our method improves by 6.3% (Abs Rel) on Stanford2D3D, 15.3% (Abs Rel) on Matterport3D, 7.7%(Abs Rel) on 360D. Note that compared to ODE-CNN [4] which used additional sensor input, our method reduces Abs Rel by 7.9%. Qualitative results of our method can be visualized in Figure 3.6. As observed, our method (1-iter and 2-iter) improves the baseline, a direct customization from [13], significantly in producing less erroneous depth maps with sharper

boundaries and smoother surfaces recovered.

Moreover, we compare our method with UniFuse [15] and HohoNet [14] qualitatively in Figure 3.7, 3.8, 3.9. We use the pretrained models downloaded from their official GitHub repositories, respectively. [1] [2]

Datasets	Methods	Abs Rel↓	Sq Rel ↓	RMSE↓	RMSE(log)↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Stanford2D3D [1]	FCRN [36]	0.1837	-	0.5774	-	0.7230	0.9207	0.9731
	RectNet [2]	0.1996	-	0.6152	-	0.6877	0.8891	0.9578
	BiFuse with fusion [3]	0.1209	-	0.4142		0.8660	0.9580	0.9860
	UniFuse with fusion [15]	0.1114	-	0.3691	-	0.8711	0.9664	0.9882
	HoHoNet [14]	0.1014	-	0.3834	-	**0.9054**	0.9693	0.9886
	OmniFusion, Ours (1-iter)	0.0961	0.0543	0.3715	0.1699	0.8940	0.9714	0.9900
	OmniFusion, Ours (2-iter)	**0.0950**	**0.0491**	**0.3474**	**0.1599**	0.8988	**0.9769**	**0.9924**
Matterport3D [9]	FCRN [36]	0.2409	-	0.6704	-	0.7703	0.9714	0.9617
	RectNet [2]	0.2901	-	0.7643	-	0.6830	0.8794	0.9429
	BiFuse with fusion [3]	0.2048	-	0.6259	-	0.8452	0.9319	0.9632
	UniFuse with fusion [3]	0.1063	-	0.4941	-	0.8897	0.9623	0.9831
	HoHoNet [14]	0.1488	-	0.5138	-	0.8786	0.9519	0.9771
	OmniFusion, Ours (1-iter)	0.0980	0.0611	0.4536	0.1587	0.9040	0.9757	0.9919
	OmniFusion, Ours (2-iter)	**0.0900**	**0.0552**	**0.4261**	**0.1483**	**0.9189**	**0.9797**	**0.9931**
360D [2]	FCRN [36]	0.0699	0.2833			0.9532	0.9905	0.9966
	RectNet [2]	0.0702	0.0297	0.2911	0.1017	0.9574	0.9933	0.9979
	Mapped Convolution [34]	0.0965	0.0371	0.2966	0.1413	0.9068	0.9854	0.9967
	BiFuse with fusion [3]	0.0615	-	0.2440	-	0.9699	0.9927	0.9969
	UniFuse with fusion [3]	0.0466	-	0.1968	-	0.9835	0.9965	0.9987
	ODE-CNN [4]	0.0467	0.0124	**0.1728**	0.0793	0.9814	0.9967	0.9989
	OmniFusion, Ours (1-iter)	0.0469	0.0127	0.1880	0.0792	0.9827	0.9963	0.9988
	OmniFusion, Ours (2-iter)	**0.0430**	**0.0114**	0.1808	**0.0735**	**0.9859**	**0.9969**	**0.9989**

Table 3.1: Quantitative Results for depth estimation on Stanford2D3d [1], Matterport3D [9], 360D [2] datasets. Notably, our method *OmniFusion* achieves state-of-the-art performances in all datasets, outperforming the existing works by a significant margin.

3.4.4 Ablation studies

Individual component study. We investigate the effectiveness of our method by adding one key component at a time (Table 3.2 and Figure 3.10). We form our baseline experiment with ResNet34 as encoder without the transformer or the geometric fusion. We experiment on Stanford2D3D, using the configuration of 18 patches, 256×256 patch size, 80° FoV. As observed from Table 3.2, the geometry-aware fusion, which only adds less than 2K parameters, is able to improve Abs Rel significantly by 9.7%. While being extremely light-

[1] https://github.com/sunset1995/HoHoNet

[2] https://github.com/alibaba/UniFuse-Unidirectional-Fusion

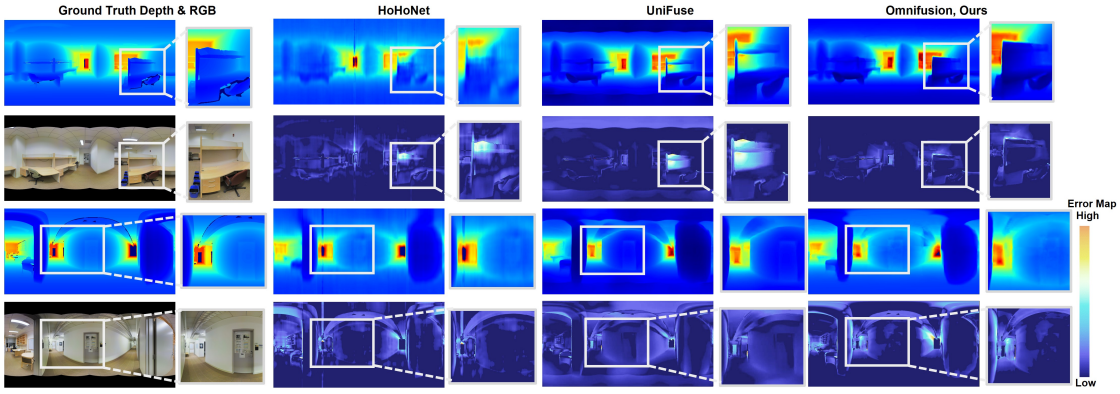


Figure 3.7: The qualitative comparisons with the current state-of-the-art works on the dataset Stanford2D3D [1]. We show the results of HoHoNet [14] (second column), UniFuse [15] (third column), and ours (last column). Both the depth maps and the error maps against the ground-truth are included for comparison. See the zoomed-in areas for detailed comparisons.

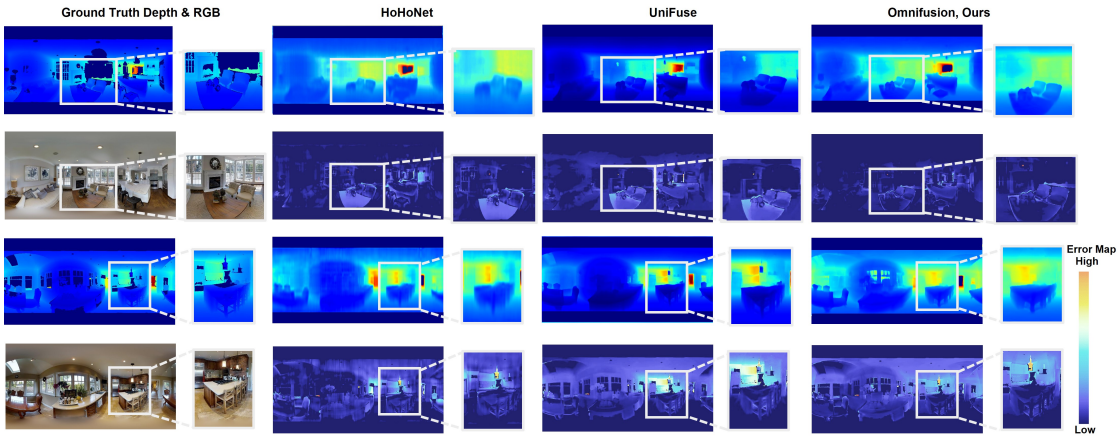


Figure 3.8: The qualitative comparisons with current state-of-the-art works on the dataset Matterport3D [9]. We show the results of HoHoNet [14] (second column), UniFuse [15] (third column), and ours (last column). Both the depth maps and the error maps against the ground-truth are included for comparison. See the zoomed-in areas for detailed comparisons.

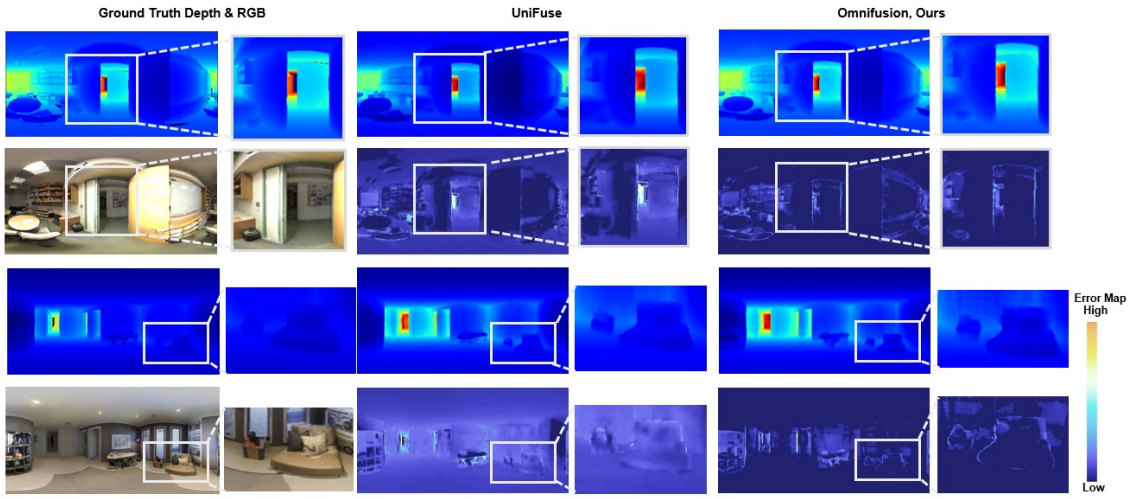


Figure 3.9: The qualitative comparisons with current state-of-the-art works on the dataset 360D [2], We show the results of UniFuse [15] (second column), and ours (last column). Both the depth maps and the error maps against the ground-truth are included for comparison. See the zoomed-in areas for detailed comparisons.

Methods	**#Params**	**FPS↑**	**Abs Rel↓**	**Sq Rel↓**	**RMSE↓**
Baseline	23.5M	9.4	0.1136	0.0638	0.3894
Baseline + geometric fusion (1-iter)	23.5M (+1.3K)	9.3	0.1026	0.5880	0.3812
Baseline + geometric fusion + transformer (1-iter)	42.3M (+18.8M)	9.2	0.0961	0.0543	0.3715
Baseline + geometric fusion + transformer (2-iter)	42.3M (+18.8M)	4.6	**0.0950**	**0.0491**	**0.3474**

Table 3.2: The ablation study for individual components. Starting from a baseline method with no geometric fusion or transformer, we add each component one at a time. We use ResNet34 for all the experiments.

weighted, the geometric fusion part proves to be quite beneficial. The incorporation of the transformer, which increases around 19M parameters, leads to another boost of performance by 5.7% (Abs Rel). Together with transformer and geometric fusion, the performance is significantly improved by 15.4% (Abs Rel) with 1-iter setting, and 16.4% (Abs Rel) with 2-iter setting. Qualitative results are shown in Figure 3.10. As observed, as we

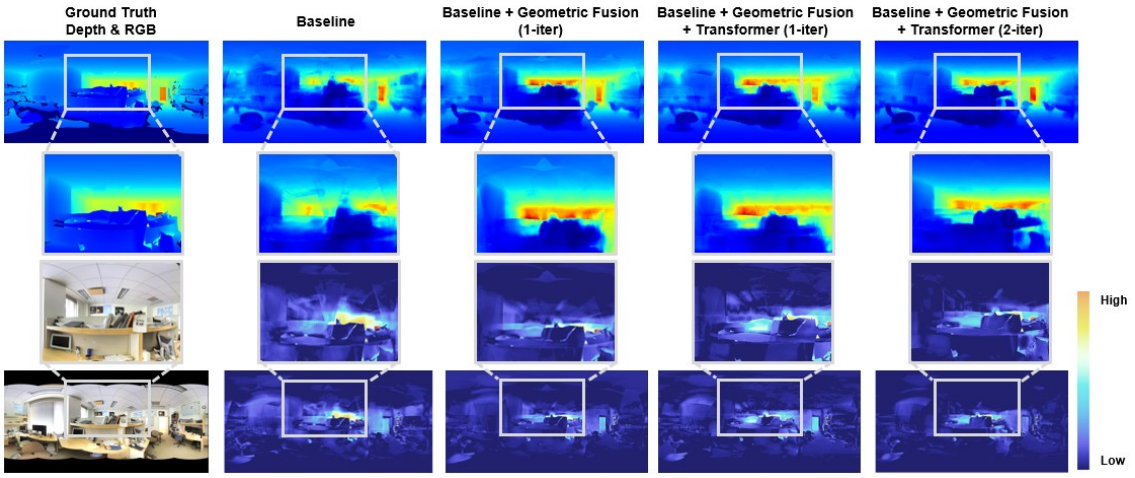


Figure 3.10: Qualitative comparisons regarding individual components. The top row shows the visual comparisons in depth maps with the ground truth depth maps shown in the left-most column, and the bottom row shows the visual comparisons of the corresponding error maps between the predicted depth maps and ground truth with the RGB input images shown in the leftmost column. The middle two rows show the close-up views of the highlighted areas in the top and bottom rows, respectively. As can be seen clearly (from left to right), as we add more modules into the pipeline (Figure 3.2), the depth estimation becomes more accurate with lower errors, sharper object boundaries and smoother surfaces. The trend of the change in errors can be directly observed from the error maps.

add more modules into our pipeline, the output depth map appears to show fewer artifacts and more structural details. In the meantime, the visualized error maps clearly show the decreasing trend of estimation errors.

Patch size and number of patches. Patch size and the number of patches affect both the accuracy and the efficiency of the method. In this study, we aim to find an optimal balance between efficiency and performance. Theoretically, neither a large patch size nor a large number of patches is desired since they both lead to higher computational complexity. However, table 3.3 also indicates the patch size can not be too small, since the monocular depth estimation requires large-enough FoV to hypothesis the depth scale. We also observe that keep increasing the number of patches (e.g., $>= 26$) can degrade the performance,

since a larger number of patches also increases the overlapping area, which in turn may intensify the discrepancy problem. As a result, we choose to use a relatively small number of patches $N = 18$ with a relatively large resolution 256×256 to balance between efficiency and performance.

#patch	Patch size	Patch FoV	Abs Rel↓	Sq Rel↓	RMSE↓
10	256x256	120	0.1067	0.0571	0.3788
18	128x128	80	0.1178	0.0666	0.4018
18	**256x256**	**80**	**0.1037**	**0.0589**	**0.3686**
26	256x256	60	0.1104	0.0679	0.3955
46	128x128	50	0.1181	0.0680	0.4101

Table 3.3: The ablation study for patch size and number of patches.

Image encoder and number of iterations. We compare the performance of leveraging different image encoders. As listed in Table 3.4, ResNet34 [37] outperforms ResNet18 with more complexity. This indicates the potential of our method, as one can incorporate a more sophisticated encoder network. We also study the influence of iterations.

We use the 2-iteration framework for the training since we expect the trained network to handle different types of 3D coordinates. While for testing, we compare 1-4 iterations respectively on the two backbones. As seen from Table 3.4, there is an evident improvement from 1-iter to 2-iter, a slighter improvement from 2-iter to 3-iter, and no gain from 3-iter to 4-iter. Considering the trade-off in performance and the speed, we opt to choose 1-iter or 2-iter settings.

3.5 Conclusion

In this work, we propose a novel pipeline, *OmniFusion*, for 360 monocular depth estimation. To address the spherical distortion presented in 360 images, as well as to improve

Encoder	#iters	FPS↑	Abs Rel↓	Sq Rel↓	RMSE↓
ResNet18	1	**9.8**	0.1037	0.0589	0.3686
ResNet18	2	4.6	0.0979	0.0539	0.3702
ResNet18	3	3.1	**0.0981**	0.0521	**0.3699**
ResNet18	4	1.5	0.0983	**0.0519**	0.3700
ResNet34	1	**9.2**	0.0961	0.0543	0.3715
ResNet34	2	4.6	0.0950	0.0491	**0.3474**
ResNet34	3	2.9	**0.0894**	**0.0482**	0.3498
ResNet34	4	1.4	0.0899	0.0485	0.3491

Table 3.4: The ablation study for different encoder models and different number of iterations.

the scalability to high-resolution inputs, we use gnomonic projection-based tangent image presentation. To alleviate the discrepancy between patches, we introduce a geometry-aware fusion mechanism which fuse 3D geometric features with the image features. A self-attention transformer is integrated into our pipeline to globally aggregate information from patches, which leads to more consistent patch-wise predictions. We further extend the geometry-aware fusion with an iterative refining scheme which further improves the depth estimation with more structural details. We show that using tangent images effectively mitigates the distortion issue, and the incorporation of the geometric features as well as transformer significantly improves the depth estimation performance. Our experiments show that our method achieves state-of-the-art performances on several datasets.

Chapter 4

SPNet: Multi-Shell Kernel Convolution for Point Cloud Semantic Segmentation

Deep learning has achieved great success in image classification [96, 37, 97], semantic segmentation [98, 99] and object detection [100, 101, 102]. However, deep learning based point cloud analysis is still a challenging topic. One major reason is that point clouds are non-uniformly sampled from a large, continuous 3D space, making them lack of regular grid structure. To tackle this, one straightforward approach is to voxelize point cloud into 3D regular grids and utilize standard 3D Convolutions [103, 104]. But the voxelization approach has a major limitation, the discretization step inevitably loses geometric information. To address this problem, many researchers have proposed approaches to directly process point clouds. One of the seminal works is PointNet proposed by Qi et al. [105]. It uses Multi Layer Perceptron (MLP) and global pooling to preserve permutation invariance and gather a combination of local and global feature presentation. This work is further improved in their follow-up work PointNet++ [106] which adds the local geometry

pooling/sampling over local neighborhood. Later works seek other ways to enhance local feature aggregation. For example, Pointweb [107] constructs a dense fully-linked web, ShellNet [108] conducts convolution based on statistics from concentric spherical shells. Besides point-based methods, several graph-based methods are proposed to capture 3D shape and structures of point clouds. Methods such as [109, 110] treat point clouds as nodes in a graph whose edges carry learnable affinitiy/similarity between adjacent points.

Recently, there has been another thread of research [111, 112, 113, 16, 114] that proposes learnable kernel functions which define convolutional kernels on a continuous space. One of these approaches is KPConv [16] introduced by Thomas et al. KPConv [16] proposes kernel point operator that consists of a set of local point filters which simulate 2D image convolution processes. Features from unordered point clouds are aggregated on either rigid or deformable kernel points. Using structural kernel points makes convolution feasible on a continuous space.

Following this line of work, we propose a novel multi-shell kernel point convolution named SPConv. Our SPConv operator partitions local 3D space into shells, each shell contains a set of rigid kernel points which aggregate local supporting point features. We perform kernel point convolution on each shell individually, then integrate the output features by an additional 1D convolution operation. The last convolution learns the contributions from shells and enhances shell correlation. An illustration of our SPConv is shown in Figure 4.1. Comparing to deformable KPConv [16], our method has an enhanced structure learning module, and does not require additional regularization during training. Furthermore, we find that incorporating low-level features such as color, normal, etc. in all layers for local feature re-weighting can be very effective for improving network performance. We propose two different approaches to accomplish the task, (1)Gaussian function based

and (2)learning based. The first approach is hand-crafted and does not necessarily add GPU computation. The second approach has learnable weights and shows more robustness.

Using SPConv as our building block, we build a deep architecture SPNet. Similar to standard CNNs which utilize downsampling and upsampling strategy to reduce computation cost as well as to enlarge receptive field, we use Poisson disk sampling (PDS) for downsampling, and feature propagation (FP) for upsampling. In section 4, we evaluate the effectiveness of our methods on the most competitive indoor segmentation datasets. Notably, experimental results show that we achieve top-ranking performances. Our main contribution is summarized as follows:

- We propose a multi-shell kernel convolution operator that shows powerful local shape encoding ability.
- We introduce a simple yet effective attention mechanism for local neighbor feature re-weighting. This attention module improves performance and speeds up convergence.
- We present a comprehensive architecture design which outperforms stage-of-the-arts on challenging large-scale indoor datasets.

4.1 Related Work

View-based and Voxel-based Methods. One classic category of point cloud representations is multi-view representation. MVCNN [115] renders 3D shape into images from various viewpoints and combines features from CNNs to predict point labels. However, these methods suffer from surface occlusion and density variation, which make it difficult

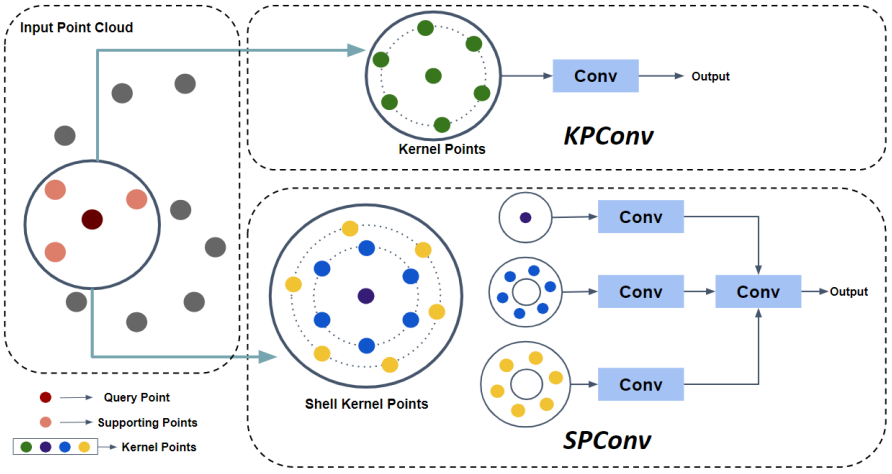


Figure 4.1: Comparison between SPConv and KPConv. For a query point, a range search is performed to locate supporting points. KPConv defines a set of kernel points to aggregate local features and performs point convolution. Our SPConv has multiple shells, each shell contains one set of kernel points. Point convolutions are conducted for shells individually to encode distinctive geometric information. An additional convolution layer is used to fuse shell outputs together, as an enhancement of structure correlation.

to capture the internal structure of the shape. Another strategy is to convert point cloud into a 3D voxel structure which can be processed by standard 3D convolutions. VoxNet [116] and subsequent work [11, 117] discretize point cloud into 3D volumetric grids. To improve efficiency on processing high resolution 3D voxels, recent researches [118, 119] process volumetric data only on non-empty voxels.

Point-wise MLP Methods. Point-based methods receive great attention since PointNet [105] was proposed. In PointNet [105], points go through shared MLPs to obtain high dimensional features followed by a global max-pooling layer. In order to capture local neighborhood context, PointNet++ [106] is developed by hierarchically applying pointnet in local regions. There are extensive works based on PointNet++. For example, PointWeb [107] builds a dense fully connected web to explore local context, and uses an Adaptive Feature Adjustment module for feature refinement. ShellNet [108] proposes a ShellConv

operator with concentric spherical shells to capture representative features.

Graph-based Methods. Graph-based approaches treat point cloud as a graph. Point or point sets formulate the vertex of the graph. The correlation between points or point sets is defined as edges of the graph. [120] applies kd-tree which is a special graph structure to represent the 3D scene and designs a Kd-network with learnable parameters to mimic CNN. [121, 109] are based on the typical graph structure $G = \{V, E\}$. [121] proposes spectral graph convolution. It follows the pointnet++ framework, while adopting a recursive cluster pooling strategy. [109] is able to handle the whole point cloud by partitioning point cloud into superpoints. Then the superpoints are fed to the graph convolution network proposed in [122]. It is able to model long-range interaction between vertices.

Point Convolution Methods. Some recent works define explicit kernels for point convolution. KCNet [123] develops a kernel correlation layer to compute affinities between each point's K nearest neighbors and a predefined set of kernel points. Local features are acquired by graph pooling layers. SpiderCNN [111] designs a family of Taylor polynomial kernels to aggregate neighbor features. PointCNN [124] introduces X-transformation to exploit the canonical order of points. PCNN [112] builds a network using parametric continuous convolutional layers. SPH3D [114] uses spherical harmonic kernels during convolution on quantized space to identify distinctive geometric features. Our work is most related to KPConv [16], which defines rigid and deformable kernel points for feature aggregation. This convolution operator resolves point cloud ambiguity, alleviates varying density, and shows superior performances. Compared to KPConv [16], our SPConv enhances local structure correlation by incorporating shell-structured kernel points and learning on a larger neighborhood context.

4.2 Method

4.2.1 Review on Kernel Point Convolution

KPConv [16] effectively resolves the point cloud ambiguity by placing manually designed kernel points in a local neighborhood. This convolution simulates image-based convolutions. A typical image based 2D convolution with a $(2m+1) \times (2m+1)$ kernel at location $i, j \in \mathbb{Z}$ is defined as:

$$F * W = \sum_{x=-m}^{m} \sum_{y=-m}^{m} F(i-x, j-y)W(i,j) \tag{4.1}$$

where $x, y \in \{-m, ..., m\}$, W is the learnable weight, $F(i,j)$ is the feature for pixel (i,j). Image based convolution describes a one-to-one relationship between single kernel and single image pixel. Similarly, for a 3D point $p \in \mathbb{R}^3$ with a local neighborhood of radius R, point convolution can be defined as:

$$F * W = \sum_{k}^{K} F(p_k, p)W(k) \tag{4.2}$$

where $F(p_k, p)$ is the aggregated features on kernel point p_k. p_k carries learnable matrix $W_k \in \mathbb{R}^{C_{in} \times C_{out}}$. C_{in} and C_{out} are input and output feature channels respectively. With proper aggregation approach, the structure of supporting points can be well captured and learned by weight W. There are two key components of this convolution, placements of kernel points and aggregation function.

For a 3D point $x \in \mathbb{R}^3$ surrounded by neighboring points $x_j \in \mathbb{R}^3$ within a ball radius R, kernel points are distributed on the surface of a sphere with radius r, plus one point

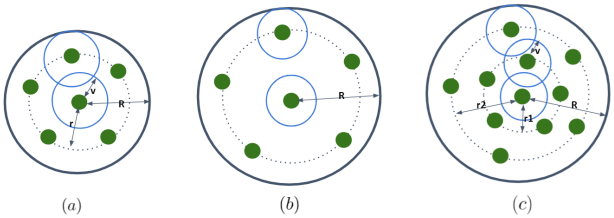


Figure 4.2: A KPConv [16] operator is defined in (a) with kernel point radius r, neighborhood size R, kernel influence radius v. Kernel points have overlapping influence regions. When enlarging neighborhood size R to capture a larger context as shown in (b), kernel points become sparse and this may cause a loss of information for complex scenes with objects of different scales. Our SPConv (c) has multiple shells and keeps overlapping influence regions. r_1 and r_2 are kernel point radius for the second and third shell.

placed at center. Aggregated features $F(p_k, p)$ for kernel point p_k are computed as the sum of the features carried by neighboring points that fall into the influenced radius v. These neighboring features are weighted based upon the Euclidean distance between p_k and p_j. An illustration of KPConv is shown in Figure 4.1.

$$F(p_k, p) = \sum_{p_j, \|p_j - p\| < R, \|p_k - p_j\| < v} F_{p_j} d(p_k, p_j) \tag{4.3}$$

where $d(p_k, p_j)$ denotes the correlation of kernel point p_k and a neighbor point p_j. This correlation can be calculated by a linear function:

$$d(p_k, p_j) = max(0, 1 - \frac{\|p_k - p_j\|}{v}) \tag{4.4}$$

4.2.2 SPConv

We extend the work of KPConv and propose a new point convolution operator, SPConv. SPConv divides the local 3D space into a total of N shells. Each shell has one set of kernel points. Specifically, the innermost shell contains one central kernel point p_0, for outter $n^{th}(n > 1)$ shell, a set of kernel points $p_{1,m}, m \in M_n$ scatter on the surface of a sphere with radius r_n. Central kernel point impacts on a spherical region, and the n^{th} shell forms a ring-shaped influence area. As a result, all kernel points cover a spherical space of radius $(r_N + v)$. We perform kernel point convolution on N shells respectively, then stack shell features together along a new dimension. Finally, we use an additional convolution layer to further correlate shell structure. This convolution operator can be defined as follows:

$$(F(x) * W_1) * W_2 = \sigma(\sum_{n}^{N} \sigma(F(x) * W_1)W_{2,n}) \tag{4.5}$$

where σ refers to non-linear activation function. $W_1 \in \mathbb{R}^{K \times C_{in} \times C_{out}/2}$ is the learnable weight matrix for kernel point convolution, $W_2 \in \mathbb{R}^{N \times C_{out}/2 \times C_{out}}$ is the weight matrix for shell correlation. To balance off efficiency and accuracy, we choose to use a total of 3 shells and 14 kernel points for the second and third shell respectively. An illustration of our SPConv is shown in Figure 4.1.

A detailed illustration of the influence area of SPConv kernel points is shown in Figure 4.2. The central kernel point encodes features from points that are spatially close to the query point. Kernel points located far from the center tend to encode more contextual information. Therefore, we learn the features by shells based on the distance from kernel point to center such that the encoded features can be representative for each shell. Furthermore, the 1D convolution layer applies weight matrix on the fused shell features, which

enhances structure learning across shells. Our method aims to improve descriptive power of kernel points, so does KPConv [16] deformable version. Although deformable kernels provide more flexibility, regularization imposed on offsets is mandatory to account for mis-shifts. By contrast, our kernels are rigid and do no require regularization. In section 4.4, we compare our evaluation results with deformable KPConv [16]. Our method outperforms deformable KPConv with even less parameters.

4.2.3 Feature Attention

To further improve local feature encoding, we propose a feature attention module using low level features such as RGB or surface normal.

We propose two approaches for local feature attention. First approach is to apply a pre-defined Gaussian function:

$$\omega_k = exp(-\frac{\|f(p) - f(s_k)\|}{2\sigma^2}) \tag{4.6}$$

where σ is a parameter that needs to be manually set. The second approach is to use sequential MLPs:

$$\omega_k = g(f(p) - f(s_k)), \tag{4.7}$$

where g is a sequence of MLPs activated by $ReLU$, except the last one which uses $sigmoid$. The final updated feature $f^{'}$ for point s_k can be calculated as follows with a residual connection:

$$f^{'}(s_k) = \omega_k f(s_k) + f(s_k) \tag{4.8}$$

Both of the approaches improve performances and accelerate convergence speed. One

issue for the first approach is that it is manually designed and not flexible. The second approach takes advantages of learnable weights and non-linear activations but adds more computation costs.

4.2.4 Network Architecture

We build a deep encoder-decoding network with point downsampling and upsampling to accomplish semantic segmentation task. A detailed SPNet architecture is shown in Figure 4.3(a).

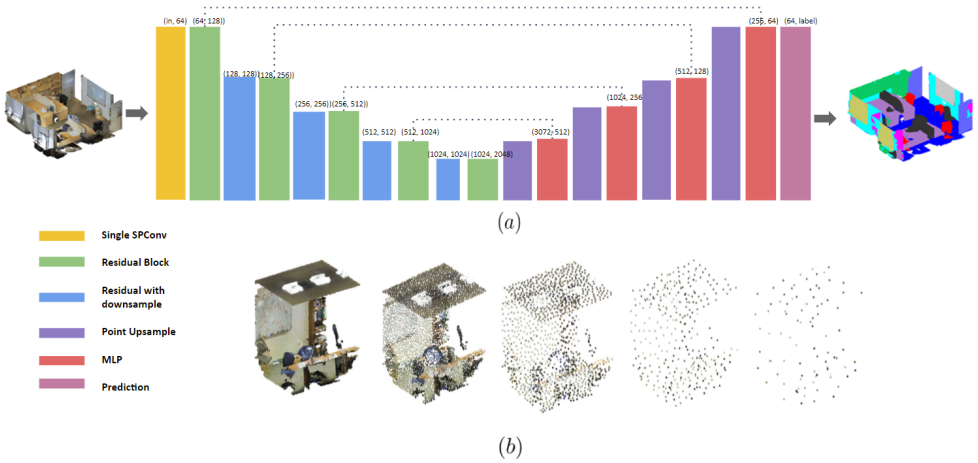
Downsampling Strategy Similar to works [16, 106], we adopt downsampling to reduce computation load as well as to increase receptive field. In our work, we favors Poisson disk sampling (PDS) strategy to deal with the varying density. PDS controls spatial uniformity by Poisson disk radius r_p, thus maintaining a minimal distance between points. Unlike grid sampling as used in [16] in which downsampled location are interpolated as the barycenter of a cell, PDS keeps the original locations of sub-sets and preserves shape patterns. Comparing to farthest point sampling (FPS) [106], PDS is faster when sampling large-scale points. A downsampling process by PDS is shown in Figure 4.3(b).

Upsampling Strategy With PDS, sampled points at each level are always a sub-set from input point sets. Therefore, we can accurately recover the point sampling patterns and gradually propagate features in decoder. We adopt feature propagation module proposed in [106]. For a point p_j at level j, its propagated features f are calculated as:

$$f = \sum_{k}^{K} w_k * f_k, \; w_k = \frac{d_k^2}{\sum_k^K d_k^2} \tag{4.9}$$

where d_k is the inverse Euclidean Distance between p_j and its k_{th} nearest neighbor at level

$j-1$.



(a)

(b)

Figure 4.3: (a) Illustration of the network architecture. (b) Downsampling process by PDS at each level.

4.3 Experiments

In this section, we evaluate the performance of our network on large-scale semantic segmentation datasets. We provide extensive ablation studies to justify the effectiveness of our proposed methods. A comparison of scene segmentation results between existing methods and our is shown in Table 4.1.

4.3.1 Datasets

Stanford Large-Scale 3D Indoor Spaces (S3DIS) The S3DIS dataset [1] is a benchmark for large-scale indoor scene semantic segmentation. It consists of point clouds of six floors from three different buildings. Following the convention [105, 124], We perform experiments on both 6-fold and Area 5 to evaluate our framework. For evaluation metrics, we use

Overall point-wise accuracy (OA), and mean intersection over union (mIoU). The detailed results for individual class are listed in Table 4.2 and Table 4.3. We can see that our method has the highest scores for several challenging classes, such as door, wall and board.

Scannet The Scannet [10] dataset contains more than 1500 scanned scenes annotated with 20 valid semantic classes. It provides a 1,201/312 data split for training and testing. The Scannet dataset is reconstructed from RGB-D scanner. We report the per-voxel accuracy (OA) as evaluation metrics. As shown in Table 4.1, our framework achieves state-of-the-art performance.

4.3.2 Overall Performance

4.3.3 Implementation Details

Parameter Setting SPNet uses residual block similar to [37]. Each block consists of one MLP for feature dimension reduction, one SPConv, and another MLP to increase feature dimension. SPNet consists of 5 encoding levels and 4 decoding levels, as shown in Figure 4.3. The kernel influence v_0 for the first encoding level is set to $0.04m$ for both S3DIS [1] and ScanNet [10]. For subsequent level l, kernel influence is increased to $v_l = 2^l v_0$. The rest of the parameters are adjusted according to v_l. For SPConv operator, we use a total of $K = 3$ shells. Kernel points of the second and the third shell are initialized on the surfaces of spheres with radius $r_2 = 1.5v_l, r_3 = 3v_l$ respectively. Query neighborhood radius R_l is set to $4v_l$, PDS radius is set to $0.75v$. For attention module, both color and normal are used to compute the attentional scores. For the Gaussian function, we set $\sigma = v_l$ at each level.

Network Training Our network is implemented using PyTorch [125] on a single Nvidia Titan RTX for all experiments. We use a batch size of 8, initial learning rate of 0.001.

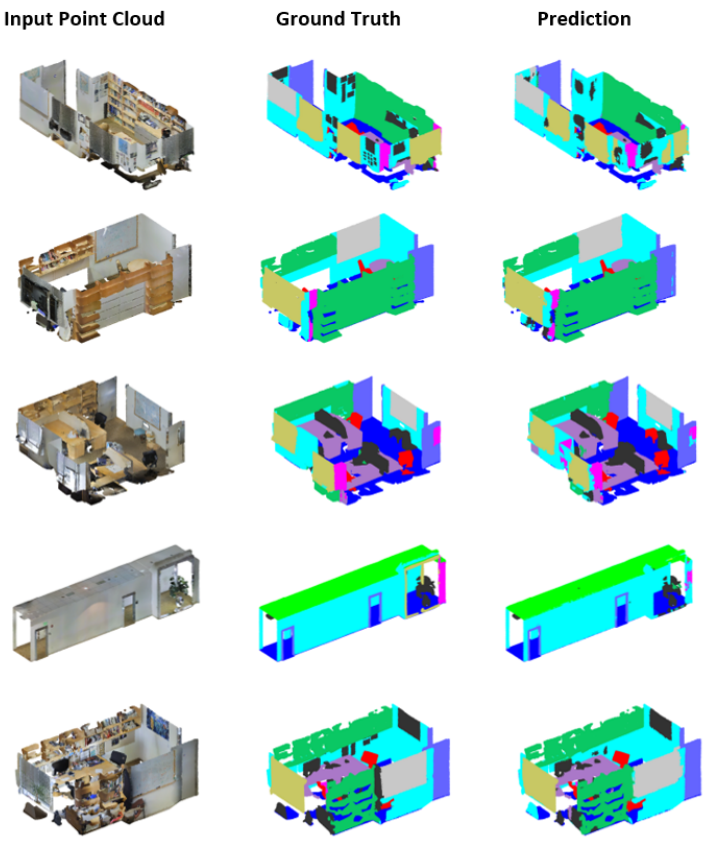


Figure 4.4: Qualitative results of semantic segmentation on S3DIS. Points above $2.5m$ are removed for better visualization purpose.

Optimization is done with Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) [126]. Learning rate decays by a factor of 0.3 every 50 epoch for S3DIS [1], and every 30 epoch for ScanNet [10].

4.3.4 Ablation Studies

To prove the effectiveness of our proposed method, we conduct a series of experiments on S3DIS [1], evaluate on Area 5. Our baseline employs kernel point convolution with 15 kernel points. As shown in Table 4.4, each time we add or replace a module while keep-

Table 4.1: Comparative 3D scene segmentation scores on S3DIS [1], ScanNet [10] datasets. S3DIS [1] scores are reported in metric of mean Intersection over Union(mIoU) including Area5 and 6-fold cross validation. ScanNet [10] scores are reported as Overall Accuracy(OA) and mIoU. The symbol $'-'$ means the results are not available.

Methods	S3DIS(mIoU) Area5	S3DIS(mIoU) 6-fold	ScanNet (OA)
PointNet [105]	41.1	47.6	-
PointNet++ [106]	-	54.5	84.5
DGCNN [122]	-	56.1	-
SPGraph [109]	58.0	62.1	-
ShellNet [108]	-	66.8	85.2
PointWeb [107]	60.3	66.7	85.9
GACNet [127]	62.9	-	-
RandLA-Net [128]	-	68.5	-
SPH3D-GCN [114]	59.5	68.9	-
Point2Node [129]	63.0	70.0	86.3
KPConv(R) [16]	65.4	69.6	-
KPConv(D) [16]	67.1	70.6	-
Minkowski [119]	65.4	-	-
Ours	**69.9**	**73.7**	**89.5**

Table 4.2: Semantic segmentation mIoU and OA scores on S3DIS [1] Area 5.

Method	mIoU	OA	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
PointNet [105]	41.1	49.0	88.8	97.3	69.8	0.1	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
PointWeb [107]	60.3	87.0	91.9	**98.5**	79.4	0.0	21.1	59.7	34.8	76.3	88.3	46.9	69.3	64.9	52.5
Point2Node [129]	62.9	88.8	93.8	98.3	83.3	0.0	35.6	55.3	58.8	79.5	84.7	44.1	71.1	58.7	55.2
KPConv(R) [16]	65.4	-	92.6	97.3	81.4	0.0	16.5	54.5	69.5	90.1	80.2	74.6	66.4	63.7	58.1
KPConv(D) [16]	67.1	-	92.8	97.3	82.4	0.0	23.9	58.0	69.0	**91.0**	**81.5**	**75.3**	75.4	66.7	58.9
Ours	**69.9**	**90.3**	**94.5**	98.3	**84.0**	0.0	24.0	59.7	**79.8**	89.6	81.0	75.2	**82.4**	**80.4**	**60.4**

Table 4.3: Semantic segmentation mIoU and OA scores on S3DIS [1] 6-fold.

Method	mIoU	OA	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
PointNet [105]	47.8	78.5	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
SPGraph [109]	62.1	85.5	89.9	95.1	76.4	62.8	47.1	55.3	68.4	69.2	73.5	45.9	63.2	8.7	52.9
PointCNN [124]	65.4	88.1	**94.8**	**97.3**	75.8	63.3	51.7	58.4	57.2	69.1	71.6	61.2	39.1	52.2	58.6
PointWeb [107]	66.7	87.3	93.5	94.2	80.8	52.4	41.3	64.9	68.1	71.4	67.1	50.3	62.7	62.2	58.5
KPConv(R) [16]	69.6	-	93.7	92.0	82.5	62.5	49.5	65.7	**77.3**	57.8	64.0	68.8	71.7	60.1	59.6
Point2Node [129]	70.0	89.0	94.1	97.3	83.4	62.7	52.3	72.3	64.3	75.8	70.8	65.7	49.8	60.3	60.9
KPConv(D) [16]	70.6	-	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	**74.9**	61.3	60.3
Ours	**73.7**	**90.9**	94.6	**97.3**	**85.0**	45.2	56.9	**82.1**	63.4	**73.1**	**83.4**	**71.5**	68.8	**68.6**	**67.8**

Table 4.4: Ablation studies evaluated on Area 5 of S3DIS [1].

	mIoU	GainΔ
Baseline + grid sampling	65.4	-
Baseline + grid sampling + FP	65.4	-
Baseline + PDS	66.0	+0.6
Baseline + PDS + FP	67.7	+2.3
SPConv + PDS + FP	68.8	+3.4
Baseline + Attention + PDS + FP	68.3	+2.9
SPConv + Attention + PDS + FP	69.9	+4.5

ing the rest unchanged. First, combining feature propagation(FP) with PDS produces a $+2.3\%$ boost. An explanation is that PDS preserves shape patterns in every downsampling level, FP correctly retrieves the shape patterns by interpolating features from neighborhood at upsampling level. Grid sampling loses geometric information, and this incorrectness accumulates through multiple downsampling layers. Moreover, adding local feature attention module produces $+2.9\%$ gain. Finally, SPConv improves the performance by $+3.4\%$, showing its great capability of local feature encoding. Our full pipeline exceeds baseline with grid sample by $+4.5\%$, which has the state-of-the-art performance on S3DIS dataset [1].

To illustrate the effectiveness of our proposed attention module, see Table 4.5. Learnable MLP-based method achieves better performance however it burdens the computation.

4.4 Conclusions

In this work, we propose an architecture named SPNet for 3D point cloud semantic segmentation. We introduce a SPConv operator to effectively learn point cloud geometry. We demonstrate that with Poisson disk sampling as well as feature propagation, our network

Table 4.5: Ablation studies with the proposed feature attention.

Approach	Input features	mIoU	Inference speed (iter/s)
Gaussian function	color	68.0	4.3
Gaussian function	normal	67.1	4.3
Gaussian function	color+normal	68.3	4.3
2layer MLP	color	68.7	4.1
2layer MLP	normal	68.2	4.1
2layer MLP	color+normal	68.7	3.0
3layer MLP	color+normal	69.9	3.0

can go deep without losing much inherent shape patterns. Our framework outperforms many competing approaches proved by experimental results on public large-scale datasets. We will experiment our method on outdoor Lidar datasets and investigate more effective attention methods.

Chapter 5

Fast Point Voxel Convolution Neural Network with Selective Feature Fusion for Point Cloud Semantic Segmentation

5.1 Introduction

Deep learning in 3D point cloud analysis has received increasing attention with the rising trend of Virtual Reality and 3D scene understanding applications, etc. Existing approaches have made great progresses in tasks such as point cloud classification [11] and point cloud semantic segmentation [12, 1]. One fundamental issue to be tackled with in point cloud analysis is the representation of unstructured point clouds. Some early methods discretize point clouds into regular volumetric grids which can be directly fed into standard 3D CNNs. However, two main problems coupled with this volumetric representation are information loss and huge memory consumption. A high resolution voxel grid leads to expensive computation cost, while a low resolution inevitably suffers from information loss during vox-

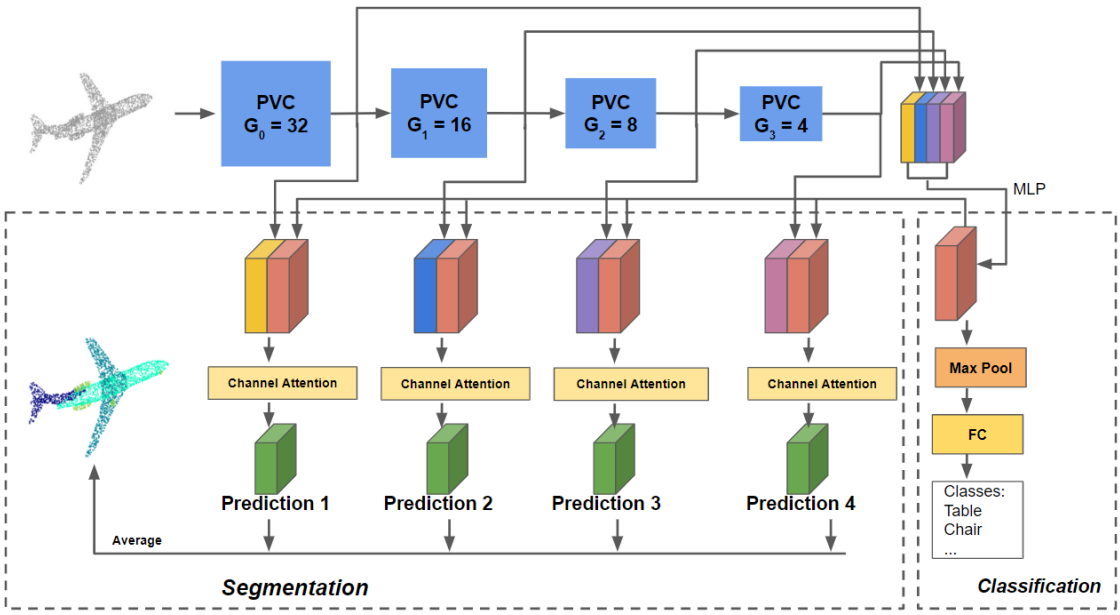


Figure 5.1: Illustration of our proposed network. For an input 3D data, we pass it through a sequence of point-voxel convolution layers(PVC). G denotes grid resolution. Outputs from each PVC layer are concatenated together to form a global feature. MLP is used for feature dimension reduction. This global feature is concatenated with output from each PVC layer and passed through a channel attention module, which re-weights the features of all channels and increases feature disciminability. The final prediction is the average of all auxiliary predictions.

elization procedure.

To address the problems mentioned above, another big stream is to directly consume sparse point clouds. The pioneer work is PointNet proposed by Qi et al. [105]. PointNet is able to process unordered point cloud inputs with permutation invariance using a sequence of multi-layer perceptron(MLP). The subsequent work PointNet++[106] achieves better performance by proposing a hierarchical network that encodes local neighborhood information. Based on PointNet++[106], a great number of networks[107, 108, 127] with more advanced local feature aggregation techniques are introduced. Apart from MLP-based methods, some recent works propose kernel-based approaches to mimic standard

convolution[16, 112, 111, 130]. In general, point-based approaches suffer from point sampling scalability, neighbor point searching efficiency, point density inconsistency issues. Most recently, Liu et al. [131] attempt to design a point-voxel CNN that represents 3D data as points to reduce memory footprint, and leverages voxel-based convolution to capture neighborhood features. This network is able to achieve reasonable performance with low memory usage and fast training/inference speed.

In this paper, we propose a novel CNN architecture that is well-balanced between efficiency and accuracy. Inspired by point-voxel CNN[131], we construct our network using point-voxel layer that takes advantages of both sparse point representation and volumetric convolution. Our point-voxel layer consists of two parallel branches, a voxel-based branch which aggregates local neighboring features, and a point-based branch which maintains fine-grained point-wise features. During discretization in voxel branch, we aggregate neighboring features on non-empty voxel centers and use standard 3D convolutions to enhance local feature encoding. Voxel features are propagated back to point domain through devoxelization. Outputs from point and voxel branches are fused self-adaptively via a feature selection module(FSM), which learns channel-wise attention for both branches.

Most of the existing studies rely heavily on point sampling strategy to avoid expensive computation cost as network goes deeper. However, point sampling cannot always retain the fine-detailed features for every point. Details of points are discarded as a trade-off for larger receptive field and processing speed. In our network, we only use a small number of point-voxel layers (default is 4 layers) that are carefully designed with effective feature encoding modules to facilitate processing efficiency. Supervision is applied on outputs from all layers to enforce semantic information learning. Though no point sampling is conducted, our network is able to remain lightweight, and effective to process large-scale

point clouds. A visualization of our proposed network is shown in Figure 5.1. We evaluate the performance and efficiency of our proposed model for object classification, object part segmentation and indoor scene semantic segmentation tasks (see Section 4).

5.2 Related Work

5.2.1 Volumentric Representation

Some early deep learning approaches transformed point clouds into 3D voxel structure and convolve it with standard 3D kernels. VoxNet [116] and subsequent works [11, 132, 103] discretized point cloud into a 3D binary occupancy grid. The occupancy grid is fed to a CNN for object proposal and classification. These voxel-based methods suffered from high memory consumption due to the waste of computation on empty spaces. OctNet[133, 134] proposed adaptive representation using octree structure to reduce memory consumption. Recent researches [118, 119] introduced approaches to process high dimensional data and apply sparse convolution only on non-empty voxels. In general, volumetric methods preserve neighborhood information of point clouds, enable regular 3D CNN applications, but suffer from significant discretization artifacts.

5.2.2 Point-based Representation

Point-wise models such as PointNet [105] and PointNet++ [106] directly operates on point clouds. The former used MLPs to extract point-wise features and permutation- invariant max pooling operation to obtain a global feature. The latter built a hierarchical architecture that incorporates point downsampling and local structure aggregation strategies. Inspired

by PointNet [105] and PointNet++ [106], many recent works propose advanced local feature learning modules. For example, PointWeb [107] built a dense fully connected web to explore local context, and used an Adaptive Feature Adjustment module for feature refinement. GACNet [127] proposed to selectively learn distinctive features by dynamically assigning attention weights to neighbouring points based on spatial positions and feature differences. ShellNet [108] built a model with several layers of ShellConv, and solved point ambiguity by constructing concentric shells and applying 1D convolution on ordered shells. Derived from point-based methods, some recent works define explicit kernels for point convolution. KCNet [123] developed a kernel correlation layer to compute affinities between each point's K nearest neighbors and a predefined set of kernel points. Local features are acquired by graph pooling layers. SpiderCNN [111] designed a family of Taylor polynomial kernels to aggregate neighbor features. PointCNN [124] introduced χ-transformation to exploit the canonical order of points. PCNN [112] built a network using parametric continuous convolutional layers. SPH3D [114] used spherical harmonic kernels during convolution on quantized space to identify distinctive geometric features. KPConv [16] defined rigid and deformable kernel points for local geometry encoding based on the Euclidean space relations between kernel point and neighborhood supporting points.

5.2.3 Efficiency of Current Models

When processing large-scale point clouds, efficiency is one of the fundamental measurements to evaluate models. Most of the point-based methods utilized point sampling to improve efficiency. However, it is non-trivial to choose an effective point sampling method. For example, Farthest Point Sampling(FPS) which is widely adopted in [106, 107, 108], has $O(NlogN)$ computation complexity, meaning it does not have good scalability. Ran-

dom point sampling as used in RandLA-Net[128], has $O(1)$ time complexity, but random point sampling cannot be invariant to point densities and key information might be discarded. Other approaches manage to incorporate hybrid representations to avoid the redundant computing and storing of more useful spatial information. A recent work Grid-GCN [135] proposed a novel method which facilitates grid space structuring and provides more complete coverage of the point cloud. This method is able to handle massive points with fast speed and good scalability. Point-Voxel CNN [131] is most related to our method. This work combines fine-grained point features with coarse-grained voxel features with speedup and low memory consumption. Compared to their work, our method builds PVC layers with multi-resolution voxels, and incorporates more accurate local feature aggregation which reduce information loss artifacts.

5.3 Method

We build a deep architecture with a sequence of point-voxel convolution(PVC) layers. In this section, we introduce the details of our PVC layer, including voxelization, local feature aggregation, devoxelization, selective feature fusion, and deep supervision.

5.3.1 Point Voxel Convolution

Voxelization and Local Aggregation The purpose of voxel branch is to encode contextual information through volumetric convolution. As aforementioned, information loss from the process of discretization is inevitable. Introducing large voxel grids reduces the loss, but burdens the network with huge computation overhead. In our design, we opt to use low-resolution volumetric grid, and mitigate information loss by effective local feature

aggregation.

As the scale of point clouds varies, we first normalize the input data into a bounding box. For voxelization, we quantize point cloud by calculating voxel coordinates $(u, v, w) \in \mathbb{N}$ from point coordinates$(x, y, z) \in \mathbb{R}$.

$$u = floor((x - x_{min})/g_x),\ v = floor((y - y_{min})/g_y),\ w = floor((z - z_{min})/g_z) \quad (5.1)$$

where g_x, g_y, g_z is the grid length of x, y, z axis respectively:

$$g_x = (x_{max} - x_{min})/G,\ g_y = (y_{max} - y_{min})/G,\ g_z = (z_{max} - z_{min})/G \quad (5.2)$$

where $G \in \mathbb{N}$ is the grid resolution of this PVC layer.

Given the voxelized point cloud, we calculate the center location P_c of every voxel. To accelerate processing speed, we locate non-empty voxels and aggregates local features only on these voxels. For a non-empty voxel (u, v, w), we use the cell center as query position and gather K neighbors through K-nearest-neighbor(KNN). We adopt the dilated point convolution strategy as proposed in [136], in which $K \cdot n$ nearest neighbors and every $n - th$ neighbor is selected. The feature f for voxel (u, v, w) is the weighted summation of all K neighbors.

$$f_{u,v,w} = \sum_{k}^{K} l_k * f_k \quad (5.3)$$

where $l_k \in \mathbb{R}^{1\times C}$ is the weight, $f_k \in \mathbb{R}^{1\times C}$ is the feature of k-th neighbor. Inspired by [127, 112], we use self-attention mechanism to learn weight of different neighboring points:

$$l_k = \sigma(\mathcal{G}(\Delta_{P_k}, f_k)) \quad (5.4)$$

where Δ_{P_k} is the normalized neighbor point coordinates, $\Delta_{P_k} = P_k - P_c$. $\mathcal{G}(*)$ takes the concatenation of Δ_{P_k} and f_k as input and models the attention weight. σ represents ReLU activation function. Through neighbor feature aggregation, we collect useful information and store it in the voxels. Next, we apply 3D convolutions on the voxel grid, as an enhancement of local neighborhood learning.

Devoxelization To allow feature fusion from two branches, we propagate voxel features back to point domain based on their voxel coordinates. Taking efficiency into account, we assign voxel feature $f_(u, v, w)$ to all points that fall into this voxel. We observe that in our experiment since point-wise features are carried all along by point branch, fusing voxel feature with point feature would be effective to discriminate individual point.

Point and Voxel Feature Fusion As illustrated in Figure 5.2, for point branch, we use MLPs to extract point-wise features. While voxel features encode local neighborhood, point branch is able to carry fine-detailed per-point features. Next, we incorporate a feature selection module to correlate features. First, we use element-wise summation to fuse features from point and voxel branches:

$$f' = f_p + f_v \tag{5.5}$$

where $f_p \in \mathbb{R}^{N \times C}$ and $f_v \in \mathbb{R}^{N \times C}$ are features from point and voxel branch respectively. Then a global average pooling is applied to squeeze N point to one compact point feature. Fully connected layer with non-linearity is used to provide guidance for feature selection:

$$S = \sigma(\mathcal{F}_{gp}(f') \cdot W_{fc}) \tag{5.6}$$

where σ is the ReLU activation function, $\mathcal{F}_{gp}$ is the global average pooling, $W_{fc} \in \mathbb{R}^{C \times d}(d =$

$C/4)$ is the learnable weight for fully connected layer. Two separate fully connected layers are applied to increase channel dimensions for S and produce soft attention vector $S_p \in \mathbb{R}^{1\times C}$ and $S_v \in \mathbb{R}^{1\times C}$.

$$S_p = S \cdot W_1, \quad S_v = S \cdot W_2 \tag{5.7}$$

where $W_1 \in \mathbb{R}^{d\times C}$ and $W_2 \in \mathbb{R}^{d\times C}$ are learnable weights. We adopt the softmax mechanism on S_p and S_v to adaptively select features.

$$S_{p,c} = \frac{e^{S_p,c}}{e^{S_p,c} + e^{S_v,c}}, \; S_{v,c} = \frac{e^{S_v,c}}{e^{S_p,c} + e^{S_v,c}} \tag{5.8}$$

where $S_{p,c}$ and $S_{v,c}$ are soft attention vector for point and voxel feature at c^{th} channel. The fused feature at c^{th} channel can be calculated as follows:

$$F_{fused,c} = S_{p,c} \odot F_{p,c} + S_{v,c} \odot F_{v,c} \tag{5.9}$$

Therefore, FSM adjusts channel-wise weight for different branches, and outputs the fused feature adaptively.

5.3.2 Deep Supervision

As shown in Figure 5.1, we build our network with several PVC layers sequentially. Grid resolution decreases while output feature channel increases from shallow to deep. Different PVC layers extract different levels of semantic information. We add supervision on each PVC layer output to enforce different levels of semantic feature learning. Similar strategy is adopted by [137] for multi-scale medical image segmentation. In detail, we concatenate

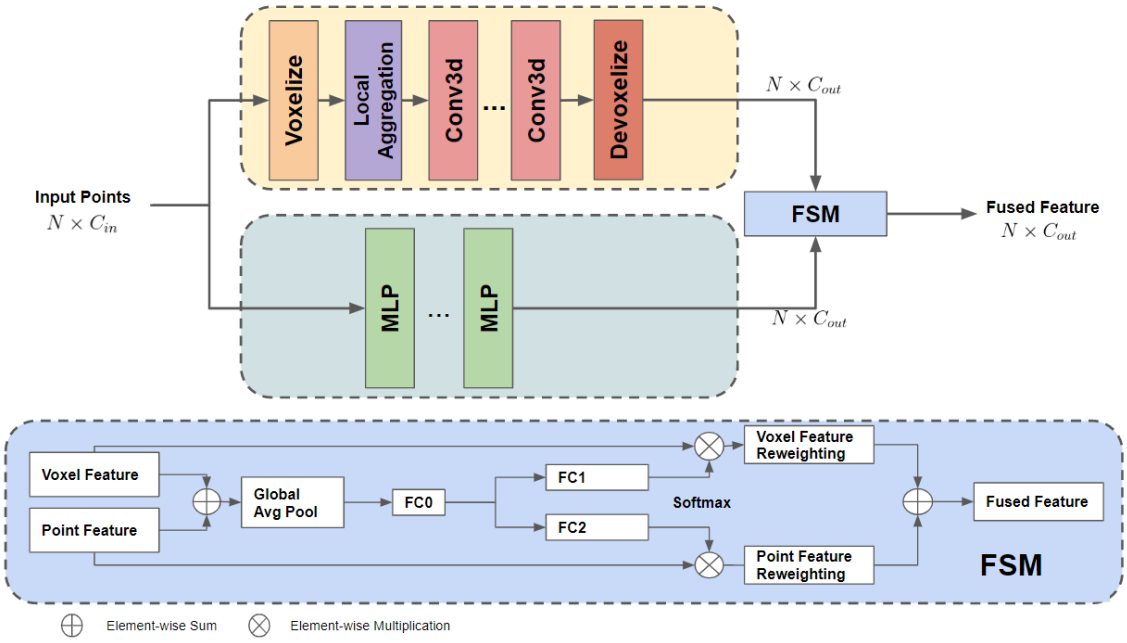


Figure 5.2: Illustration of our PVC layer (top) and Feature Selection Module (bottom).

outputs from all PVC layers and use MLPs to produce a compact feature. Serving as a global guidance, this feature map is concatenated with the output of each PVC layer, then pass through a channel attention module to enhance feature representation of specific semantics. This attention module, inspired by [138], aggregates weighted features of all the channels into the original features, and models discriminability between channels. The output from channel attention module produce an auxiliary loss. We add up all the losses and average the prediction probabilities for the final prediction. We show that incorporating channel attention module in our network boosts performances and not necessarily slow down inference speed(see Section 4.4).

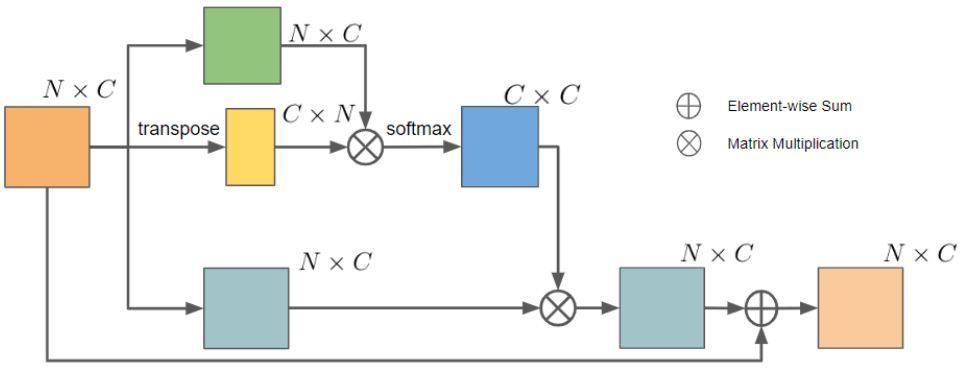


Figure 5.3: Channel attention module

5.4 Experiment

In this section, we evaluate the performance of our method in point cloud learning tasks including object classification, object part segmentation, and indoor scene segmentation. For parameter settings, our network has a total of four PVC layers. From the first to the last layer, grid size is set to $G_0 = 32, G_1 = 16, G_2 = 8, G_3 = 4$ respectively. Number of neighbors for KNN search is 32. Dilation step is $n_0 = 1, n_1 = 2, n_2 = 4, n_3 = 8$. Our method is implemented with PyTorch [125] and run on a Nvidia TitanXP GPU. Batch size for training is set to be 16. We use the Adam optimizer [126] with default settings. The learning rate is initialized as 0.001 and decays by a rate of 0.5 every 20 epochs. Object classification converges around 100 epochs, part segmentation converges at 80 epochs, and indoor scene segmentation converges at 120 epochs. The full version of our network has $C = 64$ feature channels for the first PVC layer. Feature channel for l-th layer is $C \times 2^{l-1}$.

5.4.1 Object Classification

Datasets and Implementation Details We evaluate our network on ModelNet40 [11] for 3D object classification task. ModelNet40 contains 12311 meshed CAD models from 40 categories. The dataset has 9843 objects for training and 2468 objects for testing. We prepare the dataset following PointNet [105] conventions. We random sample N points and only use normal as input feature. As illustrated in Figure 5.1, the output from all PVC layers are concatenated to form a global feature. Max pooling followed by a fully connected layer are then used to produce a classification score.

Evaluation We train three versions of classification network with variance on number of points and number of feature channels. Shown in Table 5.1, our method has a good balance between performance and efficiency.

Table 5.1: Results of ModelNet40 [11] classification. $1 \times C$ represents our full size model.

Method	Input Data	OA	Latency(ms)
PointNet [105]	8×1024	89.2	**15**
Ours($0.5 \times C$)	8×1024	**91.7**	20
PointNet++ [106]	8×1024	91.9	27
DGCNN [139]	8×1024	91.9	27
Ours($0.75 \times C$)	8×1024	**92.3**	**26**
Grid-GCN(full) [135]	16×1024	**93.1**	42
Ours($1 \times C$)	8×2048	92.5	**35**

5.4.2 Shape Segmentation

Data and Implementation Details We conduct experiment on ShapeNetPart[12] for shape segmentation. ShapeNetPart is a collection of 16681 point clouds (14006 for training, 2874 for testing) from 16 categories, each annotated with 2-6 labels. Input features are normals only, while point coordinates (x, y, z) are incorporated in the network for voxelization and local aggregation. We random sample 2048 points for training and use the original points

for testing.

Evaluation A comparison of our method and previous approaches is listed in Table 5.2. We report our result with mean instance IoU. We train three versions of our method, a compact network with $0.5 \times C$ feature channels, a medium size network with $0.75 \times C$ feature channels, and a full size network. Our compact network is able to achieve the same results as PointNet++[106], with $2\times$ speedup, and 0.7G less memory consumption. Comparing with DGCNN our compact method is $2\times$ faster and only half of its memory usage. Comparing with PV CNN[131], our method achieves comparable results.

Table 5.2: Results of object part segmentation on ShapeNetPart [12]. Our method achieves comparable performance with fast inference speed, and low GPU consumption.

Method	Input Data	InstanceIoU	Latency (ms)	GPU usage (G)
PointNet [105]	8×2048	83.7	**22**	1.5
3D-Unet [140]	Volume(8×96^3)	84.6	682	8.8
PointNet++ [106]	8×2048	85.1	78	2.0
DGCNN [139]	8×2048	85.1	88	2.4
PV CNN($0.5 \times C$) [131]	8×2048	**85.5**	22	**1.0**
Ours($0.5 \times C$)	8×2048	**85.5**	32	1.3
PointCNN [124]	8×2048	86.1	136	2.5
PV CNN($1 \times C$) [131]	8×2048	86.2	**51**	**1.6**
Ours($1 \times C$)	8×2048	**86.3**	68	2.3

5.4.3 Indoor Scene Segmentation

Data and Implementation Details We conduct experiments on S3DIS [1] for large-scale indoor scene segmentation. S3DIS [1] is a challenging dataset which consists of point clouds collected from six areas. Following the convention [105, 124], we leave out area 5 for testing purpose. For data preparation, we split rooms into $2m \times 2m$ blocks, with $0.5m$ padding along each side (x, y). These context points do not involve in neither loss computation nor prediction during testing. We use color as input feature, point coordinates

Table 5.3: Results of indoor scene segmentation on S3DIS[1], evaluated on Area 5. We report the result using mean Intersection-Over-Union(mIoU) metric. Compared with previous methods, our method is able to achieve top-ranking results while being lightweight and fast.

Method	Input Data	mIoU	Latency (ms)	GPU usage (G)
PointNet [105]	8×4096	43.0	**21**	1.0
PointNet++ [106]	8×4096	52.3	-	-
3D-Unet [140]	Volume(8×96^3)	55.0	575	6.8
DGCNN [139]	8×4096	48.0	178	2.4
PointCNN [124]	16×2048	57.3	282	4.6
PV CNN++($0.5 \times C$) [131]	4×8192	57.6	41	**0.7**
PV CNN++($1 \times C$) [131]	4×8192	59.0	70	0.8
Grid-GCN(full) [135]	8×4096	57.8	26	-
Ours($0.5 \times C$)	4×8192	60.2	34	1.4
Ours($0.75 \times C$)	4×8192	60.8	51	1.8
Ours($1 \times C$)	4×8192	**61.7**	71	2.0
Ours($1 \times C$, sparse)	4×8192	61.4	42	0.9

are incorporated in the network for voxelization and local aggregation. At training time, we random sample 8192 points from block data, and use original points at testing time.

To demonstrate the great potential of our proposed network, we also design experiments which replace regular 3D convolution layers with sparse 3D convolutions. We adopt Minkowski [119] sparse convolution in our experiment. Sparse convolution enables our network to process high-dimensional data with further speedup and reduce computation load on GPU. **Evaluation** A list of comparison of our method and previous approaches is shown in Table 5.3. We also train three versions for indoor scene segmentation. Compared with PV-CNN++ [131], our method is faster while able to achieve 2.9% higher mIoU score.

5.4.4 Ablation Study

To show the effectiveness of our proposed method, we gradually add a component while keeping the rest unchanged. To see the gain of each component, we train a baseline net-

work consists of four PVC layers. The baseline does not use local feature aggregation, instead an averaged feature of all points fall into the same voxel is taken. Fusion method is summation only. And no channel attention module(CAM) is used for prediction. Final prediction is directly produced from global feature without deep supervision. Experiments are conducted on ShapeNetPart[12]. The baseline model is the compact version ($0.5 \times C$). From Table 5.4, we can see that each component is able to boost baseline method without necessarily increase latency too much.

Table 5.4: Ablation studies on ShapeNetPart[12].

	mIoU	gain	Latency(ms)
Baseline	84.6	-	26
w/ Local Aggregation	84.8	+0.2	26
w/ FSM	85.2	+0.6	28
w/ CAM	84.9	+0.3	30
Full(Local Aggregation + FSM + CAM)	85.5	+0.9	32

5.5 Conclusion

In this work, we propose a novel approach for fast and effective 3D point cloud learning. We designed a lightweight network that can incorporate both fine-grained point features and multi-scale local neighborhood information. We introduce feature selection module and deep supervision into our network for performance improvement. Experimental results on several point cloud datsets demonstrate that our method achieves comparable results while being fast and memory efficient.

Chapter 6

Summary and concluding remarks

Both monocular depth estimation and point cloud segmentation are essential tasks for 3D scene understanding in computer vision.

The difficulty of monocular depth estimation for 360 images is exacerbated by the distortion issue and the lack of the availability of large-scale labeled dataset. In **Chapter 2**, we propose a novel model agnostic two-stage framework *PanoDepth* for generating dense high-quality depth from a monocular 360 input. Extensive experiments show that *PanoDepth* outperforms state-of-the-art approaches by a large margin. Our stereo matching sub-network in the later stage adapts to the 360 geometry and achieves top-ranking performance in 360 stereo matching.

In **Chapter 3**, to address the spherical distortion presented in 360 images, we use gnomonic projection-based tangent image presentation. To alleviate the discrepancy between patches, we introduce a geometry-aware fusion mechanism which fuse 3D geometric features with the image features. A self-attention transformer is leveraged into our pipeline to globally aggregate information from patches, which leads to more consistent patch-wise

predictions. We further extend the geometry-aware fusion with an iterative refining scheme which further improves the depth estimation with more structural details. We show that using tangent images effectively mitigates the distortion issue, and the incorporation of the geometric features as well as transformer significantly improves the depth estimation performance.

For the point cloud analysis, we use different point convolution operators to effectively encode point shapes. In **Chapter 4**, we propose an architecture named SPNet for 3D point cloud semantic segmentation. We introduce a SPConv operator to effectively learn point cloud geometry. We demonstrate that with Poisson disk sampling as well as feature propagation, our network can go deep without losing much inherent shape patterns. Our framework outperforms many competing approaches proved by experimental results on public large-scale datasets.

In **Chapter 5**, we designed a lightweight network that can incorporate both fine-grained point features and multi-scale local neighborhood information. The network consists of Point-voxel convolution layer, that takes the advantages of both point and voxel grid represenations. We also introduce feature selection module and deep supervision into our network for performance improvement. Experimental results on several point cloud datsets demonstrate that our method achieves comparable results while being fast and memory efficient.

Bibliography

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 448–465, 2018.

[3] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. Bifuse: Monocular 360 depth estimation via bi-projection fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2020.

[4] Xinjing Cheng, Peng Wang, Yanqi Zhou, Chenye Guan, and Ruigang Yang. Omnidirectional depth extension networks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 589–595. IEEE, 2020.

[5] Wei Zeng, Sezer Karaoglu, and Theo Gevers. Joint 3d layout and depth prediction from a single indoor panorama image. In *European Conference on Computer Vision*, pages 666–682. Springer, 2020.

[6] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, Federico Alvarez, and Petros Daras. Spherical view synthesis for self-supervised 360 depth estimation. In *2019 International Conference on 3D Vision (3DV)*, pages 690–699. IEEE, 2019.

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[8] Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360sd-net: 360 stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 582–588. IEEE, 2020.

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[11] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

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

[13] Marc Eder, Mykhailo Shvets, John Lim, and Jan-Michael Frahm. Tangent images for mitigating spherical distortion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12426–12434, 2020.

[14] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Hohonet: 360 indoor holistic understanding with latent horizontal features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2573–2582, 2021.

[15] Hualie Jiang, Zhe Sheng, Siyu Zhu, Zilong Dong, and Rui Huang. Unifuse: Unidirectional fusion for 360° panorama depth estimation. *IEEE Robotics and Automation Letters*, 2021.

[16] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[17] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 155–163, 2018.

[18] Lemonia Argyriou, Daphne Economou, and Vassiliki Bouki. Design methodology for 360 immersive video applications: the case study of a cultural heritage virtual tour. *Personal and Ubiquitous Computing*, pages 1–17, 2020.

[19] Matthias Berning, Takuro Yonezawa, Till Riedel, Jin Nakazawa, Michael Beigl, and Hide Tokuda. parnorama: 360 degree interactive video for augmented reality prototyping. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1471–1474, 2013.

[20] Naveen Appiah and Nitin Bandaru. Obstacle detection using stereo vision for self-driving cars. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 926–932, 2011.

[21] Gyula Pudics, Miklós Zsolt Szabó-Resch, and Zoltán Vámossy. Safe robot navigation using an omnidirectional camera. In *2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 227–231. IEEE, 2015.

[22] Michael S. Feurstein. Towards an integration of 360-degree video in higher education. In *DeLFI Workshops*, 2018.

[23] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *arXiv preprint arXiv:2003.06620*, 2020.

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

[25] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[26] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051. IEEE, 2019.

[27] Lam Huynh, Phong Nguyen-Ha, Jiri Matas, Esa Rahtu, and Janne Heikkila. Guiding monocular depth estimation using depth-attention volume. *arXiv preprint arXiv:2004.02760*, 2020.

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

[29] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[30] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1007–1015, 2018.

[31] Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9799–9809, 2019.

[32] Jamie Watson, Oisin Mac Aodha, Daniyar Turmukhambetov, Gabriel J Brostow, and Michael Firman. Learning stereo from single images. *arXiv preprint arXiv:2008.01484*, 2020.

[33] Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360 imagery. In *Advances in Neural Information Processing Systems*, pages 529–539, 2017.

[34] Marc Eder, True Price, Thanh Vu, Akash Bapat, and Jan-Michael Frahm. Mapped convolutions. *arXiv preprint arXiv:1906.11096*, 2019.

[35] Marc Eder, Pierre Moulon, and Li Guan. Pano popups: Indoor 3d reconstruction with a plane-aware network. In *2019 International Conference on 3D Vision (3DV)*, pages 76–84. IEEE, 2019.

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

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

[38] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. 2018.

[39] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. *arXiv preprint arXiv:1912.06378*, 2019.

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

[41] Marc Eder and Jan-Michael Frahm. Convolutions on spherical images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–5, 2019.

[42] Yu-Chuan Su and Kristen Grauman. Kernel transformer networks for compact spherical convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9442–9451, 2019.

[43] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 518–533, 2018.

[44] Qiang Zhao, Chen Zhu, Feng Dai, Yike Ma, Guoqing Jin, and Yongdong Zhang. Distortion-aware cnns for spherical images. In *IJCAI*, pages 1198–1204, 2018.

[45] Hong-Xiang Chen, Kunhong Li, Zhiheng Fu, Mengyi Liu, Zonghao Chen, and Yulan Guo. Distortion-aware monocular depth estimation for omnidirectional images. *IEEE Signal Processing Letters*, 28:334–338, 2021.

[46] David Gallup, Jan-Michael Frahm, Philippos Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

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

[48] Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291, 2018.

[49] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[50] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018.

[51] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, 2017.

[52] Faisal Mahmood and Nicholas J Durr. Deep learning and conditional random fields-based depth estimation and topographical reconstruction from conventional endoscopy. *Medical image analysis*, 48:230–243, 2018.

[53] Ruibo Li, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang. Deep attention-based classification network for robust depth prediction. In *Asian Conference on Computer Vision*, pages 663–678. Springer, 2018.

[54] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[55] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[56] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems*, pages 35–45, 2019.

[57] Greire Payen de La Garanderie, Amir Atapour Abarghouei, and Toby P Breckon. Eliminating the blind spot: Adapting 3d object detection and monocular depth estimation to 360 panoramic imagery. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 789–807, 2018.

[58] Keisuke Tateno, Nassir Navab, and Federico Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[59] Lei Jin, Yanyu Xu, Jia Zheng, Junfei Zhang, Rui Tang, Shugong Xu, Jingyi Yu, and Shenghua Gao. Geometric structure based and regularized depth estimation from 360 indoor imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[60] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[61] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

[62] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.

[63] Changhee Won, Jongbin Ryu, and Jongwoo Lim. Sweepnet: Wide-baseline omnidirectional depth estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6073–6079, 2019.

[64] Changhee Won, Jongbin Ryu, and Jongwoo Lim. Omnimvs: End-to-end learning for omnidirectional stereo matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8987–8996, 2019.

[65] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[66] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018.

[67] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[68] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 302–317, 2018.

[69] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. A simple and efficient rectification method for general motion. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 496–501. IEEE, 1999.

[70] Julien Valentin, Adarsh Kowdle, Jonathan T Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski,

et al. Depth from motion for smartphone ar. *ACM Transactions on Graphics (ToG)*, 37(6):1–19, 2018.

[71] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019.

[72] Rafael Monroy, Sebastian Lutz, Tejo Chalasani, and Aljosa Smolic. Salnet360: Saliency maps for omni-directional images with cnn. *Signal Processing: Image Communication*, 69:26–34, 2018.

[73] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019.

[74] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020.

[75] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[76] Jiachen Yang, Tianlin Liu, Bin Jiang, Wen Lu, and Qinggang Meng. Panoramic video quality assessment based on non-local spherical cnn. *IEEE Transactions on Multimedia*, 23:797–809, 2020.

[77] Shih-Han Chou, Yi-Chun Chen, Kuo-Hao Zeng, Hou-Ning Hu, Jianlong Fu, and Min Sun. Self-view grounding given a narrated 360 video. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[78] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2vid: Automatic cinematography for watching 360 videos. In *Asian Conference on Computer Vision*, pages 154–171. Springer, 2016.

[79] Ravi Garg, B. G. Vijay Kumar, Gustavo Carneiro, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, volume 9912, pages 740–756, 2016.

[80] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908, pages 842–857, 2016.

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

[82] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3838, 2019.

[83] Juan-Ting Lin, Dengxin Dai, and Luc Van Gool. Depth estimation from monocular images and sparse radar data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 10233–10240. IEEE, 2020.

[84] Clara Fernandez-Labrador, José M Fácil, Alejandro Perez-Yus, Cédric Demonceaux, Javier Civera, and José J Guerrero. Corners for layout: End-to-end layout recovery from 360 images. *arXiv:1903.08094*, 2019.

[85] Fu-En Wang, Hou-Ning Hu, Hsien-Tzu Cheng, Juan-Ting Lin, Shang-Ta Yang, Meng-Li Shih, Hung-Kuo Chu, and Min Sun. Self-supervised learning of depth and camera motion from 360 videos. In *Asian Conference on Computer Vision*, pages 53–68. Springer, 2018.

[86] Hsien-Tzu Cheng, Chun-Hung Chao, Jin-Dong Dong, Hao-Kai Wen, Tyng-Luh Liu, and Min Sun. Cube padding for weakly-supervised saliency prediction in 360 videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2018.

[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[88] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[89] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, October 2021.

[90] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[91] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.

[92] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021.

[93] Harold Scott Macdonald Coxeter. Introduction to geometry. 1961.

[94] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[95] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[97] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[98] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[99] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3684–3692, 2018.

[100] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[101] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[102] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[103] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017.

[104] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.

[105] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[106] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[107] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.

[108] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.

[109] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[110] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7440–7449, 2019.

[111] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[112] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[113] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[114] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds. *arXiv preprint arXiv:1909.09287*, 2019.

[115] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

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

[117] Yangyan Li, Soeren Pirk, Hao Su, Charles R Qi, and Leonidas J Guibas. Fpnn: Field probing neural networks for 3d data. In *Advances in Neural Information Processing Systems*, pages 307–315, 2016.

[118] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[119] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[120] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[121] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018.

[122] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[123] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018.

[124] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018.

[125] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[126] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[127] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.

[128] 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. *arXiv preprint arXiv:1911.11236*, 2019.

[129] Wenkai Han, Chenglu Wen, Cheng Wang, Xin Li, and Qing Li. Point2node: Correlation learning of dynamic-node for point cloud feature modeling. *arXiv preprint arXiv:1912.10775*, 2019.

[130] Alexandre Boulch. Convpoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 2020.

[131] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 963–973, 2019.

[132] Daniel Maturana and Sebastian Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 3471–3478. IEEE, 2015.

[133] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[134] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[135] Qiangeng Xu. Grid-gcn for fast and scalable point cloud learning. *arXiv preprint arXiv:1912.02984*, 2019.

[136] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated point convolutions: On the receptive field of point convolutions. *arXiv preprint arXiv:1907.12046*, 2019.

[137] Ashish Sinha and Jose Dolz. Multi-scale guided attention for medical image segmentation. *arXiv preprint arXiv:1906.02849*, 2019.

[138] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[139] 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 (TOG)*, 38(5):1–12, 2019.

[140] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

VITA

Yuyan Li was born and raised in Huangshan, Anhui, China. She studied Electrical Engineering at Tongji University, Shanghai, where she earned bachelor degree in 2013. She started PhD program in Computer Science department at University of Missouri in 2014. She joined Dr. Ye Duan's lab and started the research under Dr. Duan's supervision. Her research interests mainly focus on 3D scene understanding and deep learning for RGBD images. She will start her career in industry after graduation. She will continue doing researches and contribute to the computer vision community.