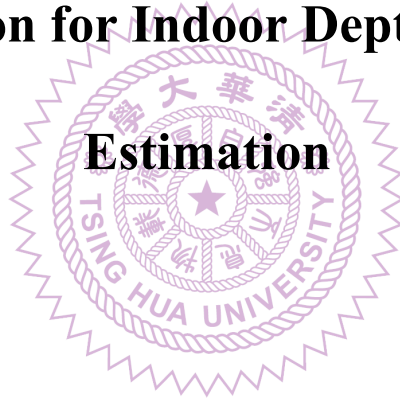


國立清華大學

博士論文

全景室內深度與格局偵測

360 Perception for Indoor Depth and Layout



系所別：電機工程學系 博士班 組別：系統組

學號姓名：108061805 王福恩 (Fu-En Wang)

指導教授：孫民 博士 (Prof. Min Sun)

中華民國 112 年 2 月



Acknowledgements

在我待在 VSLab 的這將近七年半的時間裡，我最感謝我的指導教授 - 孫民，從我大三開始加入實驗室並著手大學專題時老師就非常耐心的幫助我完成各個研究議題並且總是能給予我實用的意見讓我的研究進度能更加順利，在我第一次投稿 ACCV 時，因為我當時還沒有任何投稿論文的經驗，我還記得老師非常仔細的幫我修改論文並指出我寫作上的各種問題，也非常感謝朱宏國與陳煥宗教授共同指導我的論文，同時我也非常感謝學長胡厚寧在實驗室一起熬夜趕這篇論文中各種實驗與 debug。除此之外，我非常感謝蔡易軒與邱維辰教授共同指導我並發表了 CVPR 與 TPAMI 的論文，同時也感謝共同作者葉鈺萱的幫助並幫我分擔了許多實驗的工作與負擔，而因為我去微軟實習的關係，我也非常感謝賴尚宏教授指導我的論文並將實習的成果發表到 AAAI。另外，VSLab 的所有成員也都在我讀博士的期間給予我很多鼓勵也給了我很多研究方向上的建議，真的非常謝謝大家。



摘要

近年來，隨著消費者級別的全景相機越來越普及，深度學習運用在全景相機上的相關演算法在計算機視覺領域開始得到許多重視。此外，因為全景像機能夠同時拍到周圍 360 度資訊的關係，室內自主系統也開始在使用全景像機進行室內定位與導航，然而，全景影像的場景理解技術至今都沒有成熟的演算法能夠讓大家有效率的去運用，因此，本論文將針對室內自主系統中至關重要的兩個項目 (1) 室內環境深度預測，與 (2) 室內格局預測來進行討論並提出新穎高效率的演算法來提高未來室內自主系統相關應用的可行性。首先，針對深度預測的部分，我們透過結合不同投影資訊的方式來提高既有方法在預測的深度圖上容易產生模糊的問題，同時，我們提出了兩個全新的網路架構 BiFuse 和 BiFuse++ 來大幅改善全景影像深度預測的精確度；針對格局預測的部分，我們則結合了 BiFuse++ 與 LED²-Net 來同時運用不同投影的資訊與一維表示法並精確預測出室內格局的資訊。

關鍵字：全景影像、深度學習、深度預測、格局預測



Abstract

In recent years, as consumer-level 360° cameras become popular and affordable by most people, algorithms that utilize deep learning and panoramas become important topics in computer vision. Moreover, since 360° cameras are capable of capturing all surrounding information around the camera, indoor autonomous systems start to adopt these useful sensors for indoor localization and navigation tasks. However, efficient approaches for dealing with these tasks haven't been studied well in computer vision field. Hence, in this paper, we focus on the two important tasks in indoor autonomous systems: 1) Indoor Depth Estimation, and 2) Indoor Layout Estimation. For Indoor Depth Estimation, we utilize the information from different projections of panoramas and propose two novel framework, BiFuse and BiFuse++, to significantly improve the problems existing in previous works that the predicted depth maps from networks are usually blurred. For Indoor Layout Estimation, we utilize BiFuse++ and LED²-Net to simultaneously use the information from different projections and 1D representation to precisely estimate layouts from panoramas.

Keywords: 360, deep-learning, depth-estimation, layout-estimation



Declaration

This dissertation includes the following contents that have been already published in TPAMI (2022) and CVPR (2020 and 2021).

1. **Fu-En Wang**, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, Yi-Hsuan Tsai,
“**BiFuse: Monocular 360 Depth Estimation via Bi-Projection Fusion**”
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. [1]
2. **Fu-En Wang**, Yu-Hsuan Yeh, Yi-Hsuan Tsai, Wei-Chen Chiu, Min Sun,
“**BiFuse++: Self-Supervised and Efficient Bi-Projection Fusion for 360 Depth Estimation**”
IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2022. [2]
3. **Fu-En Wang**, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, Yi-Hsuan Tsai,
“**LED²-Net: Monocular 360° Layout Estimation via Differentiable Depth Rendering**”
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Oral Presentation, 2021. [3]

The contents of [3] was included in Yu-Hsuan Yeh’s graduate thesis, and this dissertation will focus on the extension of BiFuse++ and LED²-Net, and the improvement introduced

by BiFuse++ architecture in 360° layout estimation task.



Contents

	Page
Acknowledgements	3
摘要	5
Abstract	7
Declaration	9
Contents	11
List of Figures	13
List of Tables	19
Chapter 1 BiFuse: Monocular 360° Depth Estimation via Bi-Projection Fusion	1
1.1 Introduction	1
1.2 Related Work	4
1.3 Our Approach	6
1.3.1 Preliminary	6
1.3.2 Proposed Spherical Padding	8
1.3.3 Proposed BiFuse Network	11
1.3.4 Implementation Details	14
1.4 Experimental Results	15



1.4.1	Evaluation Metric and Datasets	15
1.4.2	Overall Performance	17
1.4.3	More Results and Ablation Study	19
1.5	Conclusions	22
 Chapter 2 BiFuse++: Self-supervised and Efficient Bi-projection Fusion for		
360° Depth Estimation		27
2.1	Introduction	27
2.2	Related Works	32
2.3	Approach	36
2.3.1	Spherical Projection	37
2.3.2	Our BiFuse++ Framework	39
2.4	Experimental Results	44
2.4.1	Evaluation Metrics and Datasets	45
2.4.2	Implementation Details	47
2.4.3	Results of Supervised Scenario	48
2.4.4	Computational Comparison	51
2.4.5	Results of Self-Supervised Scenario	53
2.5	Conclusion	57
 Chapter 3 BiFuse++ and LED²-Net		59
3.1	Introduction	59
3.2	Experiments	61
3.2.1	Experimental Results	62

Chapter 4 Conclusions

63

References

65





List of Figures

Figure 1.1 Our BiFuse network estimates the 360° depth from a monocular image using both equirectangular and cubemap projections. A bi-projection fusion component is proposed to leverage both projections inspired by both peripheral and foveal vision of the human eye. Given the estimated 360° depth, a complete 3D point cloud surrounding the camera can be generated to serve downstream applications. 2

Figure 1.2 Field-of-view (FoV) comparison. Equirectangular projection has the largest FoV compared to each face on the cubemap projection with (solid-line) or without (dash-line) the proposed spherical padding. 5

Figure 1.3 Spherical padding v.s. cube padding. Cube padding directly pads the feature of the connected faces. In addition to obvious inconsistency at the boundary, the values of four corners are undefined. In [6], the values are only chosen by the closest side. In our proposed spherical padding, the padding area is calculated with spherical projection. As a result, both the missing corner and inconsistency at the boundary can be addressed. 8

Figure 1.4 The proposed BiFuse Network. Our network consists of two branches B_e and B_c . The input of B_e is an RGB equirectangular image, while B_c takes the corresponding cubemap as input. We replace the first convolution layer in B_e with a Pre-Block [9, 34]. For the decoder, we adopt up-projection [4] modules. For each convolution and up-projection layer in B_c , we apply our spherical padding to connect feature maps of six faces. Most importantly, between feature maps from B_e and B_c , we use the proposed bi-projection fusion module to share information between two feature representations. Finally, we add a Conv module [36] to unify two depth predictions from B_e and B_c 9

Figure 1.5 Illustration for the transformation presented in Equation (1) of our main manuscript. As shown on the right-hand side, given a 3-dimensional coordinate (x, y, z) on the face of cubemap, it can be transformed into the corresponding coordinate (θ, ϕ) in terms of equirectangular representation. 9

Figure 1.6 The cubemap with length w and padding size γ . We keep the focal length the same ($0.5w$) and calculate a new FoV σ' 10

Figure 1.7 Illustration of our fusion block. The symbol \times denotes element-wise multiplication while the symbol $+$ denotes element-wise summation. 13

Figure 1.8 Qualitative results of Matterport3D. The black area in the ground truth depth map indicates invalid pixels. 17

Figure 1.9 Qualitative results of Stanford2D3D. The black area in the ground truth depth map indicates invalid pixels. 17

Figure 1.10 Qualitative results of PanoSUNCG. The black area in the ground truth depth map indicates invalid pixels. 19

Figure 1.11 Qualitative results of 360D. The black area in the ground truth depth map indicates invalid pixels. 19

Figure 1.12 Qualitative result of different padding methods. For clear visualization, we plot the inverse depth to compare different padding methods. . 20

Figure 1.13 Matterport3D dataset 22

Figure 1.14 Matterport3D dataset 22

Figure 1.15 PanoSUNCG dataset	23
Figure 1.16 PanoSUNCG dataset	23
Figure 1.17 Stanford2D3D dataset	24
Figure 1.18 Stanford2D3D dataset	24
Figure 1.19 360D dataset	24
Figure 1.20 360D dataset	25
Figure 2.1 Our BiFuse++ is a self-supervised framework of monocular 360° depth estimation. The depth estimation network (DepthNet) is a bi-projection architecture consisting of two encoders and a shared decoder. The in- puts of DepthNet are equirectangular and cubemap projections of refer- ence panorama I_t . Between each adjacent layer of encoders, the fea- ture maps of two projections are fused by our proposed fusion module (green arrows). To achieve self-supervised learning, an additional net- work (PoseNet) takes three adjacent panoramas (I_{t-1} , I_t , and I_{t+1}) in a video sequence as inputs and infers the corresponding camera motions (P_{t-1} and P_{t+1}). We then compute the photo consistency error based on the predicted depth map and camera motions to jointly train the two net- works.	28
Figure 2.2 360-SelfNet [5] trained on real-world images. The spherical pho- tometric loss cannot deal with the low-texture area and thus produces un- stable depth maps (red indicates a large depth value). Note that we mask out the photographer at the bottom left and right region.	31

Figure 2.3 The overview of our DepthNet. Our DepthNet consists of two encoders (B_e and B_c) based on ResNet-34 and a single shared decoder that unifies the feature maps from the two encoders. The inputs are the equirectangular and cubemap projections converted from a single panorama, and the output is the corresponding equirectangular depth map. During the encoding procedure, the feature maps of B_e and B_c are fused by our proposed fusion module (green). Unlike [1] and [43], our fusion module refines the original feature maps and the refined ones are then passed into next layers of B_e and B_c . To preserve complete details in the final predicted depth maps, we add three skip-connections by concatenating the fused feature maps ($f_{fuse}^1, f_{fuse}^2, f_{fuse}^3$) from fusion modules with decoded feature maps. Then, we extract multi-scale depth maps (d_1, d_2, d_3 , and d_4) from these concatenated feature maps by 1x1 convolutional layers. 36

Figure 2.4 The architecture of our fusion module. The feature maps from equirectangular and cubemap branches are first concatenated and passed into three convolutional blocks. Then, we add a skip connection to the original feature maps and obtain the fused feature maps f'_{equi} and f'_{cube} , which are the inputs of the next convolutional layers. In addition, the other fused feature map f_{fuse} are concatenated in the decoding process later. 37

Figure 2.5 The overview of our PoseNet. Our PoseNet is based on ResNet-18 and the inputs are three sequential panoramas (I_{t-1}, I_t, I_{t+1}) in a video and PoseNet infers the corresponding camera motion P_{t-1} and P_{t+1} . To suppress the ambiguity of photo consistency error in occluded areas and stabilize the training, PoseNet estimates four occlusion masks X_s to find the occluded areas. 42

Figure 2.6 The qualitative results on Matterport3D, Stanford2D3D, and PanoSUNCG under the supervised scenario (every two rows show qualitative results of each dataset). Note that the dark blue and red colors indicate close and far distance, and we use red circles to highlight the inconsistent predictions of all approaches. 48

Figure 2.7	The distortion introduced by equirectangular projection. When the pitch of the camera is 0° , the structure of the room is clear. As the pitch becomes larger, the effect of equirectangular distortion is more obvious. The distortion affects the training stability when applying existing approaches designed for perspective cameras to panoramas.	49
Figure 2.8	The 3D reconstruction comparison of BiFuse++ with other baselines. We note that the red circles indicate the incorrect depth prediction. Our BiFuse++ is able to preserve the corner details, while the other approaches predict inconsistent results.	50
Figure 2.9	The qualitative results on PanoSUNCG under self-supervised scenario.	53
Figure 2.10	The effect of Contrast-Aware Photometric Loss (CAPL). The Spherical Photometric Loss (SPL) cannot deal with the low-texture area and thus produces unstable depth maps (red indicates a large depth value). Note that we mask out the photographer at the bottom left and right region. . .	55
Figure 2.11	The Spherical Photometric Loss (SPL) is a degenerated loss. The BiFuse++ w/ SPL indeed reaches a lower SPL (0.44) compared to BiFuse++ w/ CAPL (0.49). However, the quality of BiFuse++ w/ SPL is worse than BiFuse++ w/ CAPL.	56
Figure 3.1	The task for layout estimation [3]. Given a single panorama, we train a neural network that takes the panorama as input and infers the corresponding 3D structure.	60
Figure 3.2	The motivations of LED ² -Net [3]. The corners of a room layout are shown in the left figure. When we introduce same errors to these corners, the IoU difference introduced from these corner errors can be hugely different (right figure).	60
Figure 3.3	The horizon depth of LED ² -Net [3]. LED ² -Net proposes a layout-to-depth module to convert predicted and ground truth layout to their corresponding horizon depth maps.	61



List of Tables

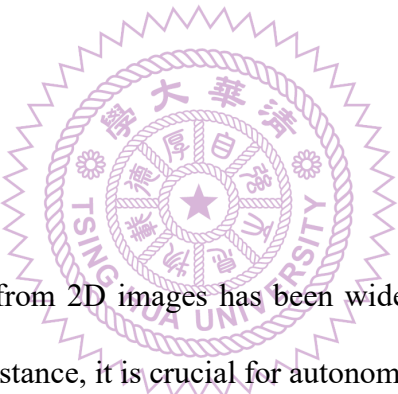
Table 1.1	Quantitative results on real-world datasets: Matterport3D and Stanford2D3D.	15
Table 1.2	Quantitative results on virtual-world datasets: PanoSUNCG and 360D.	16
Table 1.3	Comparison of padding methods on the cubemap branch.	18
Table 1.4	Qualitative results of fusion methods on Matterport3D.	20
Table 1.5	Quantitative results on the Matterport3D dataset with comparisons to different fusion strategies. Training and evaluation is based on the cubemap coordinate system.	21
Table 2.1	The quantitative results on Matterport3D [8].	44
Table 2.2	The quantitative results on Stanford2D3D [10]. Note that SliceNet* is a re-implemented version.	45
Table 2.3	The quantitative results on PanoSUNCG [5].	45
Table 2.4	The quantitative results after applying rotation noise on Matterport3D [8].	50
Table 2.5	The quantitative results after applying rotation noise on Stanford2D3D [10].	50
Table 2.6	The number of parameters of different fusion modules (we set the channels to 512).	52
Table 2.7	The computational comparison of fusion approaches.	52
Table 2.8	The quantitative results on PanoSUNCG [5] under the self-supervised scenario. Our BiFuse++ with Spherical Photometric Loss (SPL) [5] or Contrast-Aware Photometric Loss (CAPL) outperforms other baselines. SSIM stands for structural similarity.	52

Table 3.1	The quantitative results of Realtor360 [7].	62
Table 3.2	The quantitative results of Matterport3D [8].	62



Chapter 1 BiFuse: Monocular 360° Depth Estimation via Bi-Projection Fusion

1.1 Introduction



Inferring 3D structure from 2D images has been widely studied due to numerous practical applications. For instance, it is crucial for autonomous systems like self-driving cars and indoor robots to sense the 3D environment since they need to navigate safely in 3D. Among several techniques for 3D reconstruction, significant improvement has been achieved in monocular depth estimation due to the advance of deep learning and availability of large-scale 3D training data. For example, FCRN [4] achieves monocular depth estimation by their proposed up-projection module. However, most of the existing methods are designed for a camera with normal field-of-view (FoV). As 360° camera becomes more and more popular in recent years, the ability to infer the 3D structure of a camera's complete surrounding has motivated the study of monocular 360° depth estimation.

In this paper, we propose an end-to-end trainable neural network leveraging two common projections – equirectangular and cubemap projection – as inputs to predict the depth

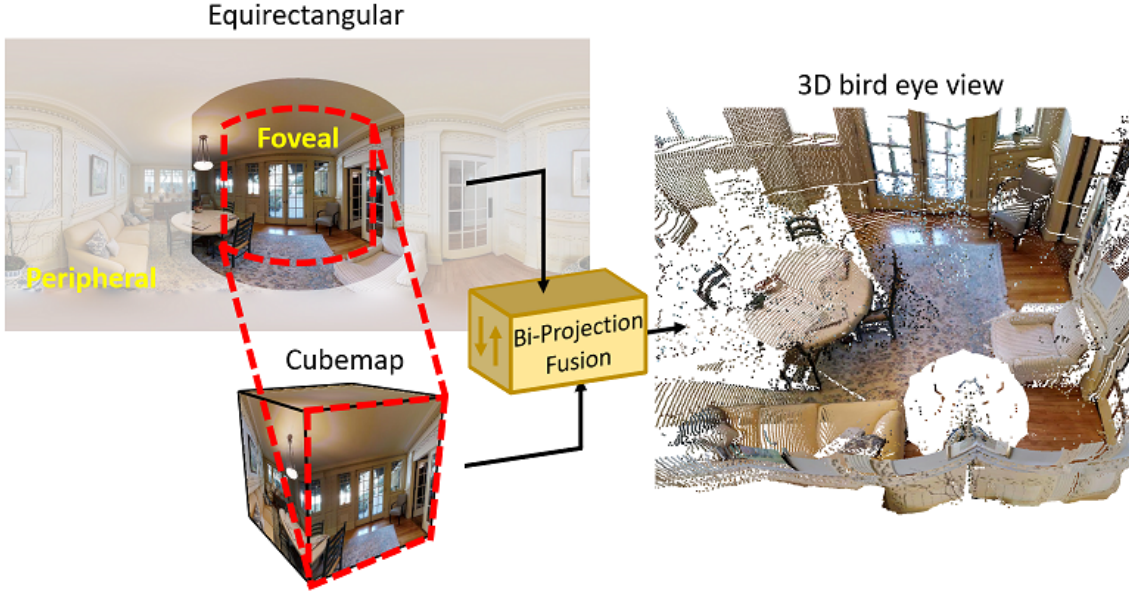


Figure 1.1: Our BiFuse network estimates the 360° depth from a monocular image using both equirectangular and cubemap projections. A bi-projection fusion component is proposed to leverage both projections inspired by both peripheral and foveal vision of the human eye. Given the estimated 360° depth, a complete 3D point cloud surrounding the camera can be generated to serve downstream applications.

map of a monocular 360° image. Our main motivation is to combine the capability from both peripheral and foveal vision like the human eye (see Fig. 1.1 for the illustration). Note that, equirectangular projection provides a wide field-of-view mimicking a peripheral vision, whereas cubemap projection provides a smaller but non-distorted field-of-view mimicking the foveal vision. On the one hand, equirectangular projection allows all surrounding information to be observed from a single 2D image but introduces distortion. On the other hand, cubemap projection avoids distortion but introduces discontinuity at the boundary of the cube. Considering both projections would have the complementary property to each other, where we refer to our method as *BiFuse*.

However, the FoV of the foveal vision could be too small, which degrades the effectiveness of our fusion scheme (Fig. 1.2). To tackle this issue, cube padding (CP) methods [5,6] have been proposed to expand field-of-view from neighboring faces on the cube. Nevertheless, using cube padding may result in geometric inconsistency at the boundary

that introduces non-negligible distortion effect. Therefore, we propose spherical padding (SP) which pads the boundary by considering the spherical geometry and reduces the boundary inconsistency. Finally, instead of naively combining features of both branches (e.g., [7]), we propose a bi-projection fusion procedure with learnable masks to balance the information shared between two projections. The source code and pretrained models are available to the public¹.

We apply our method to four panorama datasets: Matterport3D [8], PanoSUNCG [5], 360D [9] and Stanford2D3D [10]. Our experimental results show that the proposed method performs favorably against the current state-of-the-art (SOTA) methods. In addition, we present extensive ablation study for each of the proposed modules, including the spherical padding and fusion schemes. Our contributions are summarized as follows:

1. We propose an end-to-end two-branch network, which incorporates both equirectangular and cubemap projections, to mimic the combination of peripheral and foveal vision of the human eye, respectively.
2. To share the information of different projections, we propose a bi-projection fusion procedure with learnable masks to balance the information from two projections.
3. We propose spherical padding to extend the field-of-view of cubemap projection and reduce the boundary inconsistency of each face.

¹<https://fuenwang.ml/project/bifuse>

1.2 Related Work

We describe the related work regarding monocular depth estimation and 360° perception in the following.

Monocular Depth Estimation. Saxena *et al.* [11] is one of the pioneer work on learning to estimate monocular depth. After several years of development using classical machine learning approaches, deep learning contributes to the latest significant improvement in performance. Eigen *et al.* [12] first use a deep neural network to estimate the depth map from a single image. Later on, Laina *et al.* [4] utilize ResNet [13] as the encoder and propose an up-projection module for the upsampling procedure along with the reverse Huber loss to improve depth estimation. In addition, Lee *et al.* [14] try to predict depth using several cropped images and combine them in the Fourier domain. To further refine depth predictions, [15–19] integrate conditional random fields (CRF) into deep neural network to achieve better performance. For instance, Cao *et al.* [15] formulate depth estimation as a classification problem and use CRF to refine the final prediction.

Moreover, other attempts have been made to advance depth estimation. Fu *et al.* [20] use dilated convolution to increase the receptive field and apply the ordinal regression loss to preserve the spatial relation between each neighboring class. With photometric loss, unsupervised training for depth estimation [21–27] can be achieved. Godard *et al.* [21] use stereo pairs to predict disparity based on the left-right consistency, while Zhou *et al.* [22] propose two networks to estimate both depth and ego-motion from video sequences. In addition, Yang *et al.* [25] use depth-normal consistency to improve depth prediction. However, for the above-mentioned methods, they are designed for a camera with normal FoV

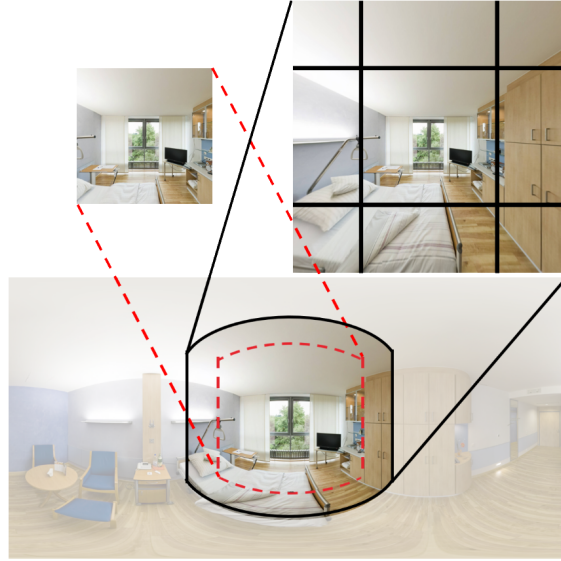


Figure 1.2: Field-of-view (FoV) comparison. Equirectangular projection has the largest FoV compared to each face on the cubemap projection with (solid-line) or without (dash-line) the proposed spherical padding.

without considering the property of 360° images.

360° Perception. Recently, omnidirectional cameras has become a popular media, which encourages people to work on panorama related tasks [28, 29]. For instance, due to the large field-of-view, room layout can be inferred from panorama [7, 29, 30]. However, the performance usually suffers from the distortion of equirectangular projection. To overcome this issue, several approaches are proposed. Cheng *et al.* [6] convert panorama into cubemap. For each face, they replace the original zero padding with their proposed cube padding method to remove the boundary inconsistency. Built upon [6], Wang *et al.* [5] use cubemap and cube padding for unsupervised panorama depth estimation.

To make the network aware of the distortion, spherical convolution methods are proposed recently [31–34]. Considering this property, Zioulis *et al.* [9] propose OmniDepth and adopt spherical layers in [34] as the pre-processing module. However, it still remains a challenge when applying spherical CNNs using deeper networks on the depth

task. Ederet *et al.* [35] tackle the 360° depth estimation as multi-task learning of depth, surface normal, and plane boundary. However, the surface normal from depth map is usually noisy especially in real-world scenarios which limits the scalability outside synthetic scenes. Different from existing works above, we improve the learning mechanism via utilizing a two-branch network from the perspective of the human eye system and propose a spherical padding scheme to maintain the geometric consistency in the cubemap representation. Our experiments shows that our method achieves state-of-the-art performance in both real-world and synthetic scenes.

1.3 Our Approach

In this paper, we aim to take advantage of two different representations for 360° images, equirectangular and cubemap projections, for improving the monocular 360° depth estimation. In the following, we sequentially detail the cubemap projection with our proposed spherical padding procedure in Sec. 1.3.1 and 1.3.2, bi-projection fusion scheme in Sec. 1.3.3, and the overall network architecture in Sec. 1.3.3.

1.3.1 Preliminary

For a cubemap representation with sides of equal length w , we denote its six faces as $f_i, i \in \{B, D, F, L, R, U\}$, corresponding to the ones on the back, down, front, left, right and up, respectively. Each face can be treated as the image plane of an independent camera with focal length $\frac{w}{2}$, in which all these cameras share the same center of projection (i.e., the center of the cube) but with different poses. When we set the origin of the world coordinate system to the center of the cube, the extrinsic matrix of each camera coordinate system can

be simply defined by a rotation matrix R_{f_i} and zero translation. Given a pixel p_i on the image plane f_i with its coordinate (x, y, z) on the corresponding camera system, where $0 \leq x, y \leq w - 1$ and $z = \frac{w}{2}$, we can transform it into the equirectangular representation by a simple mapping:

$$\begin{aligned} q_i &= R_{f_i} \cdot p_i, \\ \theta_{f_i} &= \arctan\left(\frac{q_i^x}{q_i^z}\right), \\ \phi_{f_i} &= \arcsin\left(\frac{q_i^y}{|q_i|}\right), \end{aligned} \quad (1.1)$$

where θ_{f_i} and ϕ_{f_i} are longitude and latitude in equirectangular projection; and q_i^x, q_i^y, q_i^z are the x, y, z components of q_i respectively. As this mapping is reversible, we are able to easily perform both equirectangular-to-cube and cube-to-equirectangular transformations, which are denoted as **E2C** and **C2E**, respectively.

In Figure 1.5 we provide the detailed illustration of the equirectangular-to-cube and cube-to-equirectangular transformations (**E2C** and **C2E**). To be specific, when doing **E2C**, we need to sample pixels on equirectangular coordinate in order to acquire the corresponding texture for each face of the cube representation. However, if we directly project from equirectangular coordinates onto the cube, we could have some faces with incomplete pixels as these pixels cannot be mapped from the integer equirectangular coordinates. As a result, the technique of **inverse mapping** is usually adopted to solve this problem: for each coordinate on the cube, we compute its corresponding coordinate in the equirectangular system and copy the pixel value. Please note here that, if the corresponding coordinate in the equirectangular system is with floating numbers, the interpolation is applied on its neighboring integer coordinates to retrieve the interpolated pixel value.

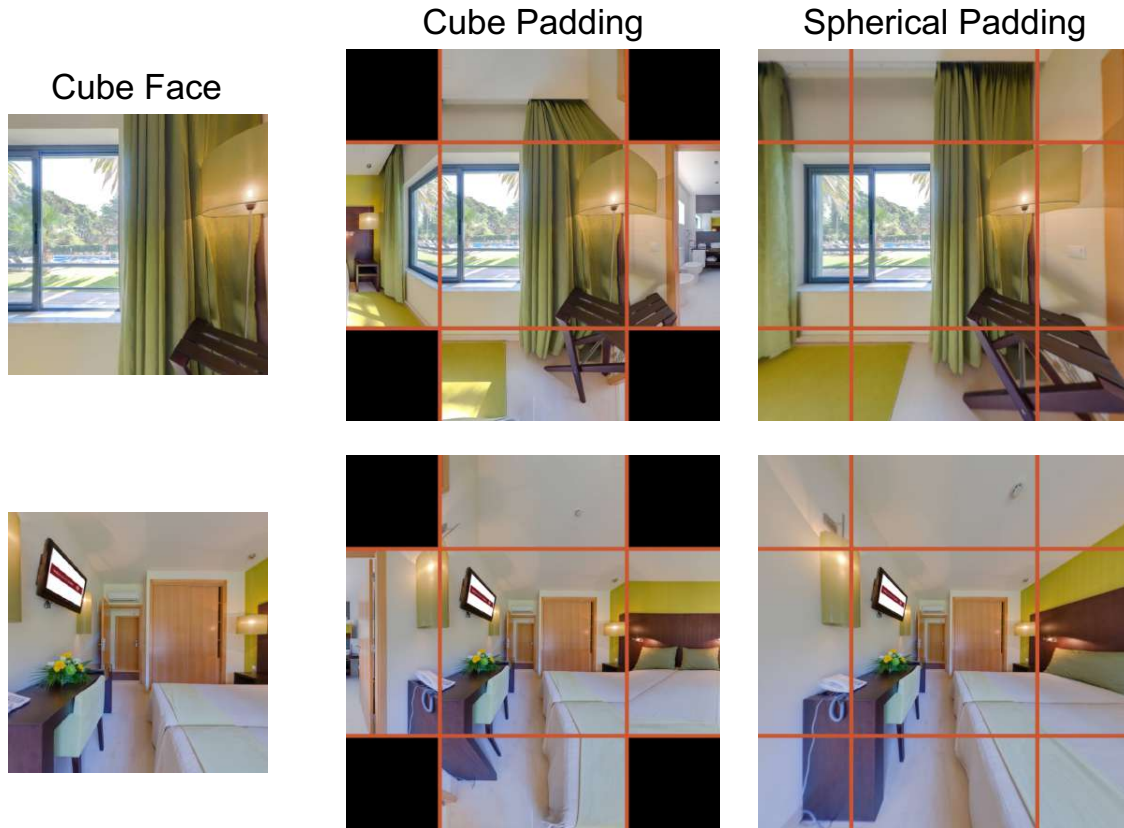


Figure 1.3: Spherical padding v.s. cube padding. Cube padding directly pads the feature of the connected faces. In addition to obvious inconsistency at the boundary, the values of four corners are undefined. In [6], the values are only chosen by the closest side. In our proposed spherical padding, the padding area is calculated with spherical projection. As a result, both the missing corner and inconsistency at the boundary can be addressed.

1.3.2 Proposed Spherical Padding

Due to the distortion in the equirectangular projection, directly learning a typical convolutional neural network to perform monocular depth estimation on equirectangular images would lead to unstable training process and unsatisfying prediction [6]. In contrast, the cubemap representation suffers less from distortion but instead produces large errors since the discontinuity across the boundaries of each face [5,6]. In order to resolve this issue for cubemap projection, Cheng *et al.* [6] propose the cube padding (CP) approach to utilize the connectivity between faces on the cube for image padding. However, solely padding the feature map of a face by using the features from its neighboring faces does

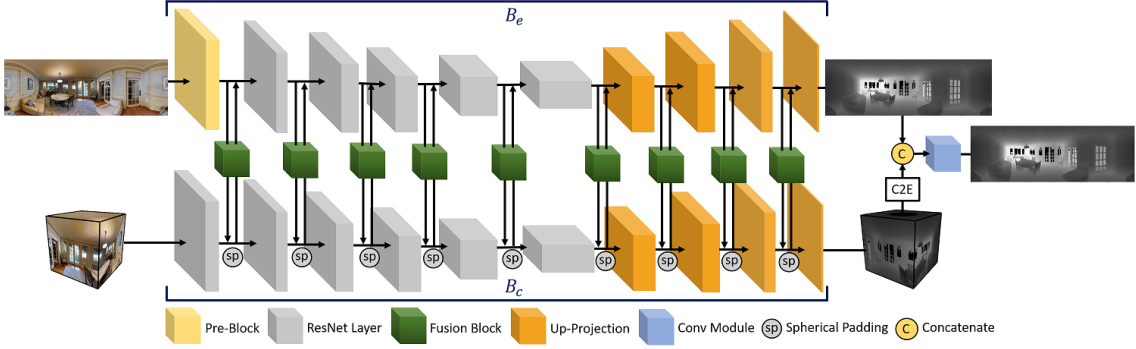


Figure 1.4: The proposed BiFuse Network. Our network consists of two branches B_e and B_c . The input of B_e is an RGB equirectangular image, while B_c takes the corresponding cubemap as input. We replace the first convolution layer in B_e with a Pre-Block [9, 34]. For the decoder, we adopt up-projection [4] modules. For each convolution and up-projection layer in B_c , we apply our spherical padding to connect feature maps of six faces. Most importantly, between feature maps from B_e and B_c , we use the proposed bi-projection fusion module to share information between two feature representations. Finally, we add a Conv module [36] to unify two depth predictions from B_e and B_c .

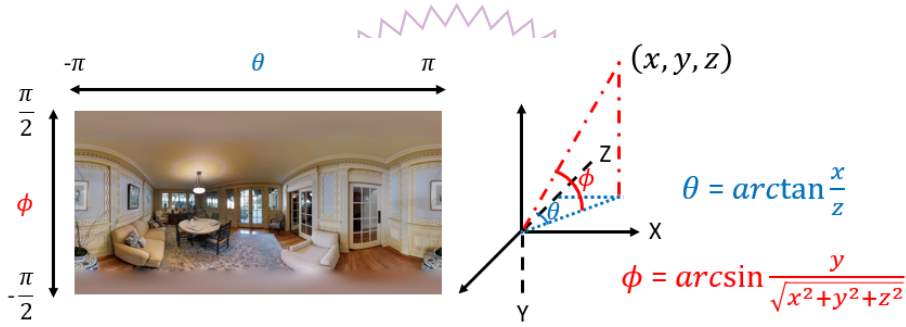


Figure 1.5: Illustration for the transformation presented in Equation (1) of our main manuscript. As shown on the right-hand side, given a 3-dimensional coordinate (x, y, z) on the face of cubemap, it can be transformed into the corresponding coordinate (θ, ϕ) in terms of equirectangular representation.

not follow the characteristic of perspective projection. Therefore, here we propose the spherical padding (SP) method, which pads the feature according to spherical projection. As such, we can connect each face with the geometric relationship. A comparison between the cube padding [6] and our proposed spherical padding is illustrated in Fig. 1.3.

The most straightforward way to apply spherical padding for cubemap is to first transform all the faces into a unified equirectangular image by C2E. Then, we extend the original FoV $\sigma = 90^\circ$ to σ' , and map it back to the cubemap by E2C. As a result, we can pad them on each face completely without missing parts (i.e., undefined areas in cube

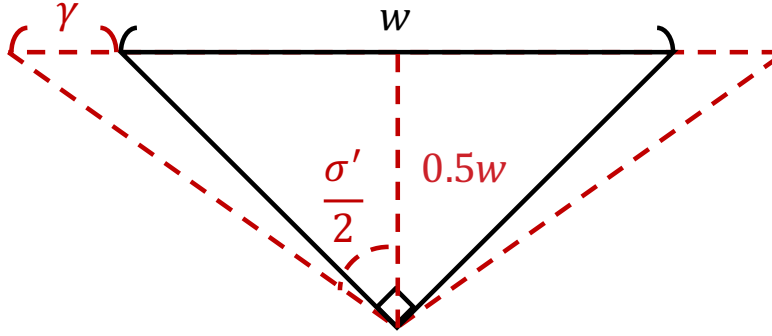


Figure 1.6: The cubemap with length w and padding size γ . We keep the focal length the same ($0.5w$) and calculate a new FoV σ' .

padding of Fig. 1.3) and with consistent geometry. Specifically, given a cubemap with side length w and Fov $\sigma = 90^\circ$, the C2E transformation is identical to the inverse calculation of (1.1). When we apply spherical padding with padding size γ , which is determined by the padding size in the convolution layer (e.g., $\gamma=1$ for a 3×3 convolution layer), we update the side length of a cube face to $w' = w + 2\gamma$, and the corresponding FoV becomes $\sigma' = 2 \arctan \frac{w/2+\gamma}{w/2}$ after padding, as illustrated in Fig. 1.6. Hence, for mapping from equirectangular image back to the padded cubemap, we should use both w' and σ' to derive the correct E2C transformation for spherical padding.

Efficient Transformation. We have described the overall concept of our spherical padding.

However, the above procedure consists of both C2E and E2C transformations, which could require heavy computational cost. Therefore, we simplify this procedure by deriving a direct mapping function between two cube faces. Given two cube faces f_i and f_j , we first denote the geometric transformation between their camera coordinate systems as a rotation matrix $R_{f_i \rightarrow f_j}$. Then the mapping from a pixel p_i in f_i to f_j can be established upon

the typical projection model of pinhole cameras:

$$K = \begin{bmatrix} w/2 & 0 & w/2 \\ 0 & w/2 & w/2 \\ 0 & 0 & 1 \end{bmatrix}, \quad (1.2)$$

$$p_j = K \cdot R_{f_i \rightarrow f_j} \cdot p_i,$$

$$x = \frac{p_j^x}{p_j^z}, y = \frac{p_j^y}{p_j^z},$$

where (x, y) represents the 2D location of p_i after being mapped onto the image plane of f_j . Since this mapping only needs to be computed once for all the pixels on the padding region, the computational cost of applying spherical padding is comparable with cube padding, without any E2C or C2E transformation included.

1.3.3 Proposed BiFuse Network

We have introduced our spherical padding method that enlarges the field-of-view while maintaining the geometric consistency at the boundary, to improve the cubemap representation as one branch of the proposed BiFuse network. In Fig. 1.4, we show our complete two-branch network motivated by the human eye system with peripheral and foveal vision.

Overall, our model consists of two encoder-decoder branches which take the equirectangular image and cubemap as input, respectively, where we denote the equirectangular branch as B_e and the cubemap one as B_c . As mentioned in Sec. 1.1, each branch has its benefit but also suffers from some limitations. To jointly learn a better model while sharing both advantages, we utilize a bi-projection fusion block that bridges the information

across two branches, which will be described in the following. To generate the final prediction, we first convert the prediction of cubemap to the equirectangular view and adopt a convolution module to combine both predictions.

Bi-Projection Fusion. To encourage the information shared across two branches, we empirically find that directly combining feature maps [7] from B_e and B_c would result in unstable gradients and training procedure, and thus it is keen to develop a fusion scheme to balance two branches. Inspired by the recent works in multi-tasking [37, 38], we focus on balancing the feature map from two different representations. To achieve this goal, we propose a bi-projection fusion module H : given feature maps h_e and h_c from B_e and B_c in each layer respectively, we estimate the corresponding feature maps $h'_e = H_e(h_e)$ and $h'_c = H_c(\text{C2E}(h_c))$, where H_e and H_c indicate a convolution layer.

To produce feature maps that benefit both branches, we first concatenate h'_e and h'_c , and then pass it to a convolution layer with the *sigmoid* activation to estimate a mask M to balance the fusion procedure. Finally, we generate feature maps \bar{h}_e and \bar{h}_c as the input to the next layer as:

$$\begin{aligned}\bar{h}_e &= h_e + M \cdot h'_c, \\ \bar{h}_c &= h_c + \text{E2C}((1 - M)) \cdot \text{E2C}(h'_e).\end{aligned}\tag{1.3}$$

Note that we use C2E and E2C operations in the fusion procedure to ensure that features and the mask M are in the same projection space.

The overview of our fusion block is provided in Figure 1.7. First, the inputs are the feature maps from equirectangular and cubemap branches, which are fed into their

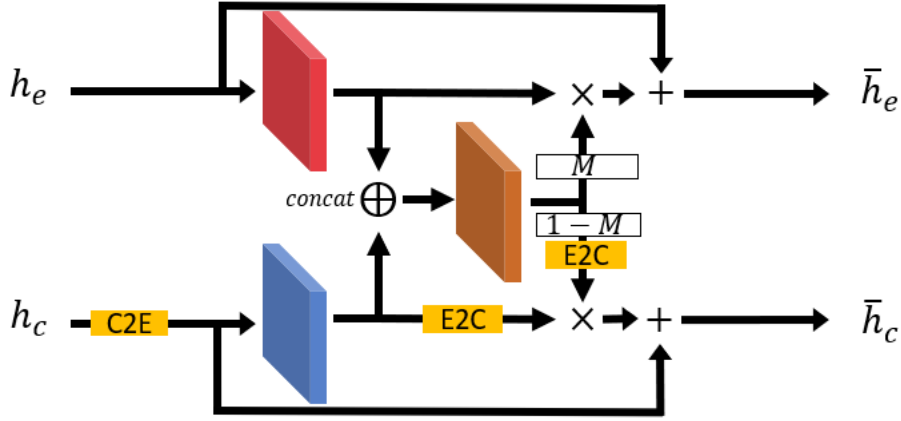


Figure 1.7: Illustration of our fusion block. The symbol \times denotes element-wise multiplication while the symbol $+$ denotes element-wise summation.

corresponding convolution layers respectively. After concatenating their outputs after convolutions (denoted as $h'_e = H_e(h_e)$ and $h'_c = H_c(C2E(h_c))$ respectively), another convolution layer is used to infer a weighting mask M in order to balance the fusion between two branches. Finally, we obtain the final outputs by having $\bar{h}_e = h_e + M \cdot h'_e$ and $\bar{h}_c = h_c + E2C((1 - M)) \cdot E2C(h'_e)$. In this way, the information between two branches can be well shared.

Loss Function. We adopt the reverse Huber loss [4] as the objective function for optimizing predictions from both B_e and B_c :

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2 + c^2}{2c} & |x| > c. \end{cases} \quad (1.4)$$

The overall objective function is then written as:

$$\mathcal{L} = \sum_{i \in P} \mathcal{B}(D_e^i - D_{GT}^i) + \mathcal{B}(C2E(D_c)^i - D_{GT}^i), \quad (1.5)$$

where D_e and D_c are the predictions produced by B_e and B_c respectively; D_{GT} is the ground truth depth in the equirectangular representation; and P indicates all pixels where

there is a valid depth value in the ground truth map. We note that the C2E operation is required on converting D_c into the equirectangular form before computing the loss.

Network Architecture. For each branch, we adopt the ResNet-50 [13] architecture as the encoder and use the up-projection module proposed by [4] as the decoder. Similar to [9] that considers the equirectangular property, we replace the first convolution layer of ResNet-50 in B_e with a spherical Pre-Block that has multi-scale kernels with size of (3,9), (5,11), (5,7), and (7,7), where their output feature maps are concatenated together as a 64-channel feature map and further fed into the next layer. In the cubemap branch B_c , we replace the original zero padding operation with our spherical padding among every adjacent layer (Fig. 1.4).

Furthermore, the proposed bi-projection fusion block as in (1.3) is inserted between every two layers between B_e and B_c in both encoder and decoder, in which each H_e and H_c in one fusion module contains a convolution layer which has the same channel number as the input feature map. Finally, to combine the predictions from B_e and B_c , we adopt a module with several convolution layers as in [36].

1.3.4 Implementation Details

We implement the network using the PyTorch [39] framework. We use Adam [40] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our batch size is 16 and the learning rate is set to 0.0003. For training our model, we first learn B_e and B_c branches independently without using the fusion scheme as the warm-up training stage for 40 epochs, and then update only bi-projection fusion modules for another 40 epochs. Finally, we train the entire network for 20 epochs.

Table 1.1: Quantitative results on real-world datasets: Matterport3D and Stanford2D3D.

Dataset	Method	MRE ↓	MAE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Matterport3D	FCRN [4]	0.2409	0.4008	0.6704	0.1244	0.7703	0.9174	0.9617
	OmniDepth (bn) [9]	0.2901	0.4838	0.7643	0.1450	0.6830	0.8794	0.9429
	Equi	0.2074	0.3701	0.6536	0.1176	0.8302	0.9245	0.9577
	Cube	0.2505	0.3929	0.6628	0.1281	0.7556	0.9135	0.9612
	Ours w/ fusion	0.2048	0.3470	0.6259	0.1134	0.8452	0.9319	0.9632
Stanford2D3D	FCRN [4]	0.1837	0.3428	0.5774	0.1100	0.7230	0.9207	0.9731
	OmniDepth (bn) [9]	0.1996	0.3743	0.6152	0.1212	0.6877	0.8891	0.9578
	Equi	0.1428	0.2711	0.4637	0.0911	0.8261	0.9458	0.9800
	Cube	0.1332	0.2588	0.4407	0.0844	0.8347	0.9523	0.9838
	Ours w/ fusion	0.1209	0.2343	0.4142	0.0787	0.8660	0.9580	0.9860

1.4 Experimental Results

In this section, we conduct experiments on four panorama benchmark datasets: Matterport3D [8], PanoSUNCG [5], 360D [9] and Stanford2D3D [10], both quantitatively and qualitatively. We mainly compare our method with the baseline FCRN [4] and the OmniDepth [9] approach, which is the current state-of-the-art for single panorama depth estimation. In addition, we compare different variants of the proposed framework to validate the effectiveness of our designed modules. Source code and models will be made available to the public.

1.4.1 Evaluation Metric and Datasets

We evaluate the performance by standard metrics in depth estimation, including MAE, MRE, RMSE, RMSE (log), and δ . Details of each dataset are introduced below and we use the same setting to compare all the methods.

Matterport3D. Matterport3D contains 10,800 panorama and the corresponding depth ground truth captured by Matterport’s Pro 3D Camera. This dataset is the largest real-

Table 1.2: Quantitative results on virtual-world datasets: PanoSUNCG and 360D.

Dataset	Method	MRE ↓	MAE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
PanoSUNCG	FCRN [4]	0.0979	0.1346	0.3973	0.0692	0.9223	0.9659	0.9819
	OmniDepth [9]	0.1143	0.1624	0.3710	0.0882	0.8705	0.9365	0.9650
	Equi	0.0687	0.0836	0.2902	0.0496	0.9529	0.9787	0.9886
	Cube	0.0628	0.0891	0.2946	0.0508	0.9453	0.9780	0.9890
	Ours w/ fusion	0.0592	0.0789	0.2596	0.0443	0.9590	0.9823	0.9907
360D	FCRN [4]	0.0699	0.1381	0.2833	0.0473	0.9532	0.9905	0.9966
	OmniDepth [9]	0.0931	0.1706	0.3171	0.0725	0.9092	0.9702	0.9851
	Equi	0.0606	0.1172	0.2667	0.0437	0.9667	0.9920	0.9966
	Cube	0.0613	0.1167	0.2739	0.0447	0.9688	0.9908	0.9956
	Ours w/ fusion	0.0615	0.1143	0.2440	0.0428	0.9699	0.9927	0.9969

world dataset for indoor panorama scenes, which makes it challenging as the depth map from ToF sensors usually has noise or missing value in certain areas. In practice, we filter areas with missing values during training. To train and test our network, we follow the official split which takes 61 rooms for training and the others are for testing. We resize the resolution of image and depth map into 512×1024 .

Stanford2D3D. Stanford2D3D is collected from three kinds of buildings in the real world, containing six large-scale indoor areas. The dataset contains 1413 panoramas and we use one of official splits that takes fifth area (area_5) for testing, and the others are for training. During training and testing, we resize the resolution of image and depth map into 512×1024 .

PanoSUNCG. PanoSUNCG contains 103 scenes of SUNCG [41] and has 25,000 panoramas. In experiments, we use the official training and testing splits, where 80 scenes are for training and 23 for testing. For all panoramas, we resize them to 256×512 and filter out pixels with depth values larger than 10 meters.

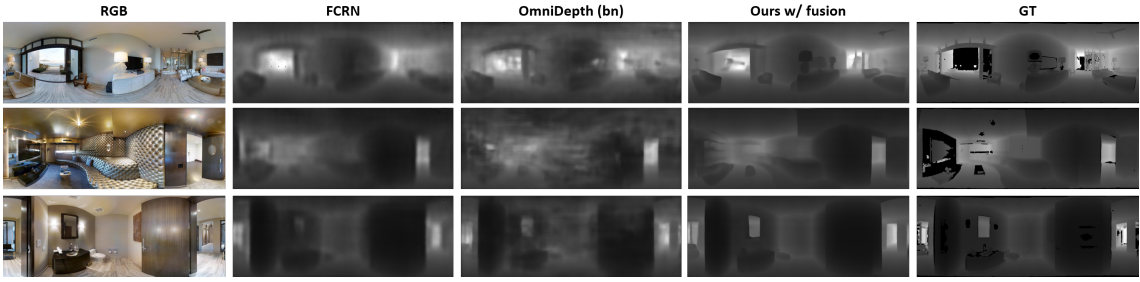


Figure 1.8: Qualitative results of Matterport3D. The black area in the ground truth depth map indicates invalid pixels.

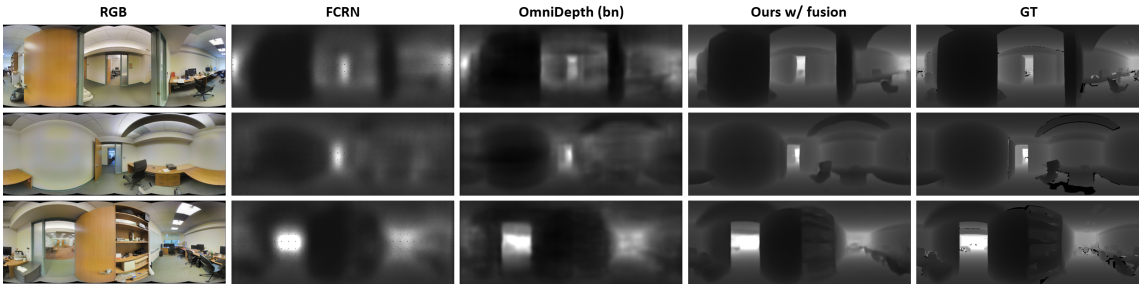


Figure 1.9: Qualitative results of Stanford2D3D. The black area in the ground truth depth map indicates invalid pixels.

360D. 360D dataset is collected by OmniDepth [9], including two synthetic datasets, SunCG and SceneNet and two realistic datasets, Stanford2D3D and Matterport3D. They use path tracing renderer to render four datasets and place spherical cameras in the virtual environment to acquire photo-realistic panoramas with the resolutions 256×512 . For each panorama, they apply augmentation by 90° , 180° and 270° . In total, 360D contains 35,977 panoramas, where 34,679 of them are used for training and the rests are for testing.

1.4.2 Overall Performance

We first present results of using two baselines, each with a single branch, and compare them with our proposed two-branch framework: 1) **Equi**: the equirectangular branch B_e without bi-projection fusion; 2) **Cube**: the cubemap branch B_c with cube padding [6] without our fusion scheme; 3) **Ours w/ fusion**: our final model of applying the proposed spherical padding to the cubemap branch B_c and integrating our bi-projection fusion to

Table 1.3: Comparison of padding methods on the cubemap branch.

Dataset	Method	MRE	MAE	RMSE
Matterport3D	Cube w/ zp	0.2577	0.4136	0.6934
	Cube w/ cp	0.2505	0.3929	0.6628
	Cube w/ sp	0.2254	0.3660	0.6327
Stanford2D3D	Cube w/ zp	0.1457	0.2667	0.4511
	Cube w/ cp	0.1332	0.2588	0.4407
	Cube w/ sp	0.1259	0.2388	0.4269
PanoSUNCG	Cube w/ zp	0.1195	0.1367	0.3441
	Cube w/ cp	0.0628	0.0891	0.2946
	Cube w/ sp	0.0600	0.0840	0.2874
360D	Cube w/ zp	0.0761	0.1382	0.2819
	Cube w/ cp	0.0610	0.1163	0.2722
	Cube w/ sp	0.0588	0.1145	0.2614

both branches.

In Table 1.1 and 1.2, we show quantitative comparisons on four datasets as mentioned above. Overall, our fusion model performs favorably against FCRN [4] and OmniDepth [9], as well as our baselines using the single branch (i.e., Equi or Cube). This validates the effectiveness of the proposed two-branch network, in which the equirectangular view provides a larger field-of-view and the cubemap one focuses on non-distorted regions.

Moreover, on Matterport3D and Stanford2D3D, we find that the official implementation of OmniDepth (originally designed for the 360D [9] dataset) has difficulty to converge on these two datasets, and thus we add batch normalization [42] to successfully train the model, which is denoted as **OmniDepth (bn)** in Table 1.1.

Qualitative Comparisons. From Fig. 1.8 to 1.11, we present qualitative results of depth maps on four datasets. Compared to the FCRN and OmniDepth methods, our model is able to produce sharper results around boundaries. This can be attributed by the foveal view

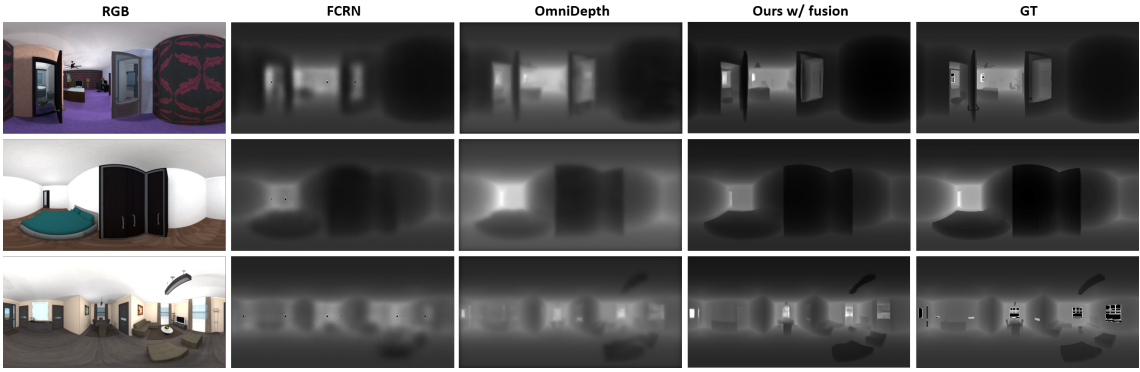


Figure 1.10: Qualitative results of PanoSUNCG. The black area in the ground truth depth map indicates invalid pixels.

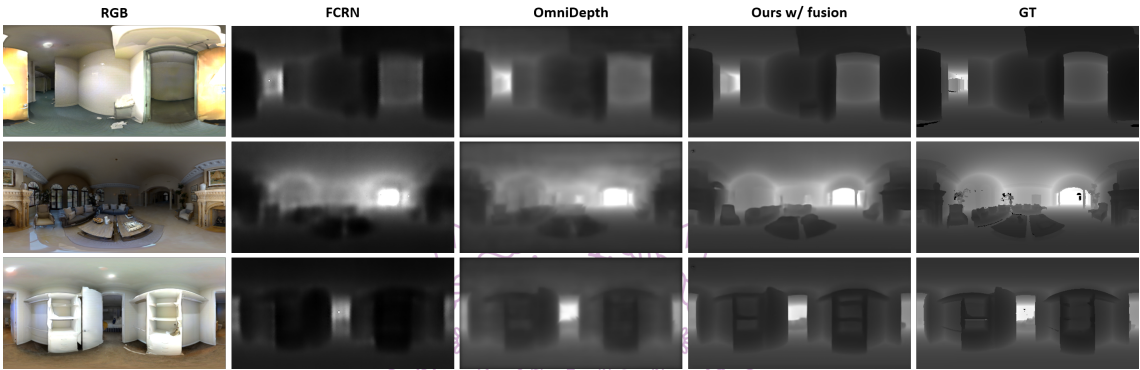


Figure 1.11: Qualitative results of 360D. The black area in the ground truth depth map indicates invalid pixels.

capturing detailed information, while the peripheral view with larger FoV provides global context.

1.4.3 More Results and Ablation Study

Effect of Spherical Padding. To further study the effects of spherical padding in the cubemap, we compare the proposed spherical padding (SP) with the other two padding methods, i.e., zero padding (ZP) and cube padding (CP).

Quantitative results on only the cubemap branch are shown in Table 1.3. By applying our spherical padding, the cubemap branch B_c outperforms other padding methods significantly. In addition, Fig. 1.12 shows qualitative comparisons of applying different

Table 1.4: Qualitative results of fusion methods on Matterport3D.

Method	MRE	MAE	RMSE
Yang <i>et al.</i> [7]	0.2662	0.4842	0.7364
Average	0.2658	0.4405	0.7202
Ours	0.2048	0.3470	0.6259

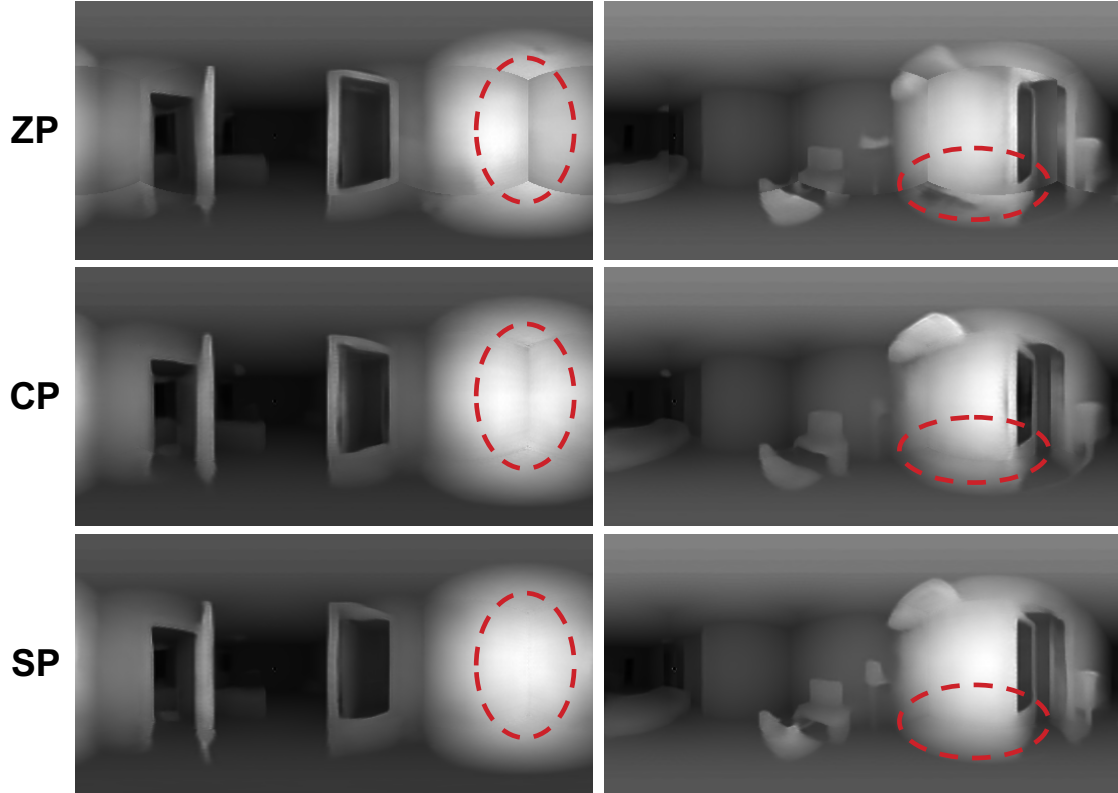


Figure 1.12: Qualitative result of different padding methods. For clear visualization, we plot the inverse depth to compare different padding methods.

padding methods. When using zero padding, the depth maps of six faces have obvious boundary artifacts. After using cube padding, the boundary effect becomes more smooth, but it is still observable because the cube padding does not follow the geometric relationship. By applying the proposed spherical padding, we are able to maintain the boundary as spherical padding is calculated using the spherical projection.

Fusion Schemes. To validate our fusion module, we conduct two baselines on Matterport3D using the fusion method proposed in [7] via directly adding up two feature maps and the feature averaging scheme. We show this ablation study in Table 1.4. From the

results, our method is consistently better than the baselines. For instance, the MAE is improved by 28% and 23% comparing with Yang *et al.* [7] and Average. In addition, we find the training of this baseline is unstable and has the convergence problem as the gradients from different branches cannot be well balanced. This shows the benefit of integrating our bi-projection fusion scheme which applies several masks to balance features of two branches.

More results and analysis. To demonstrate that our fusion scheme is beneficial to the depth estimation, we build up two model variants with different fusion strategies as the baselines to make comparison: 1) a fusion method proposed in [7] via directly adding up two feature maps, and 2) simply averaging predictions from two branches. In addition to the training/evaluation in the equirectangular coordinate (Table 1.4), we further provide comparisons in the perspective coordinate. Table 1.5 shows the quantitative results evaluated on the Matterport3D dataset, and our full model with the bi-directional fusion module is able to outperform other baselines with other fusion strategies, which validates the benefit of our fusion scheme to improve depth estimation in 360° cameras.

Table 1.5: Quantitative results on the Matterport3D dataset with comparisons to different fusion strategies. Training and evaluation is based on the cubemap coordinate system.

Different fusion	MRE ↓	MAE ↓	RMSE ↓	RMSE (log) ↓	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
Yang <i>et al.</i> [7]	0.4652	0.2616	0.5490	0.1541	0.8661	0.9453	0.9665
Average	0.4798	0.2642	0.5488	0.1550	0.8630	0.9460	0.9672
Ours	0.4512	0.2473	0.5343	0.1518	0.8792	0.9485	0.9676

Qualitative Result of Point Clouds For better visualization on the comparison between our proposed method and the baselines, here we show several qualitative results on four datasets (two cases each) in terms of point clouds, which are based on the depth estimation produced by different approaches: **Matterport3D** in Fig. 1.13 and Fig. 1.14; **PanoSUNCG**

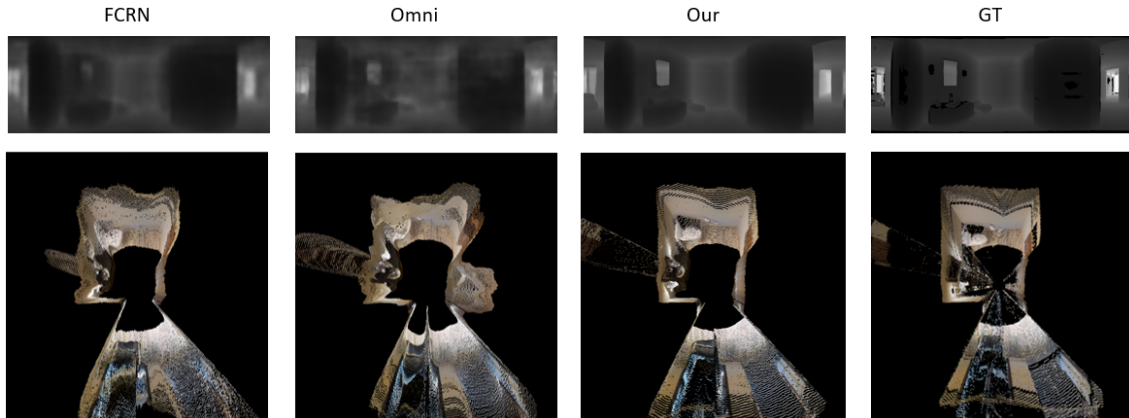


Figure 1.13: Matterport3D dataset

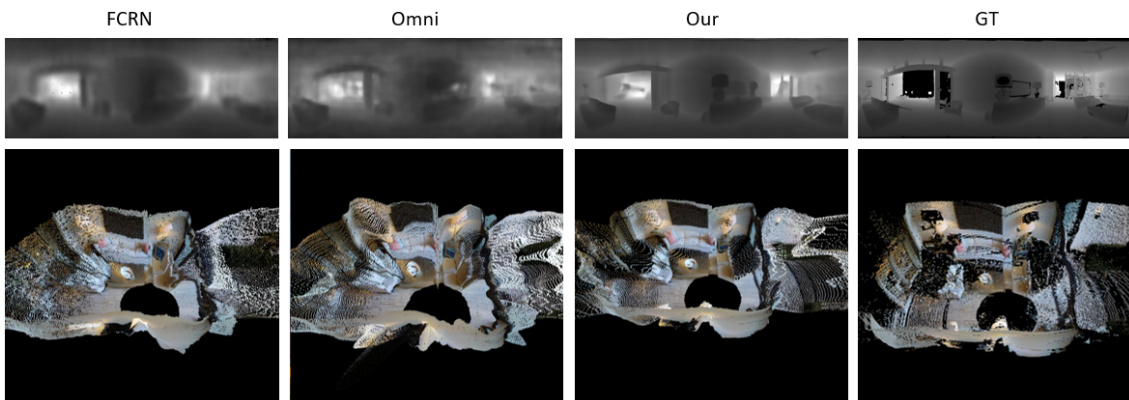


Figure 1.14: Matterport3D dataset

in Fig. 1.15 and Fig. 1.16; **Stanford2D3D** in Fig. 1.17 and Fig. 1.18; **360D** in Fig. 1.19 and Fig. 1.20, respectively. Each figure has two rows, the first row shows depth predictions, while the second row provides corresponding point clouds in bird eye view. The point cloud results clearly demonstrate that our prediction has sharper boundary than other methods (i.e., FCRN and Omni) and are closer to the ground truth (GT).

1.5 Conclusions

In this paper, we propose an end-to-end 360° depth estimation network which incorporates both equirectangular and cubemap projections to mimic peripheral and foveal vision as the human eye. Since the two projections have the complementary property,

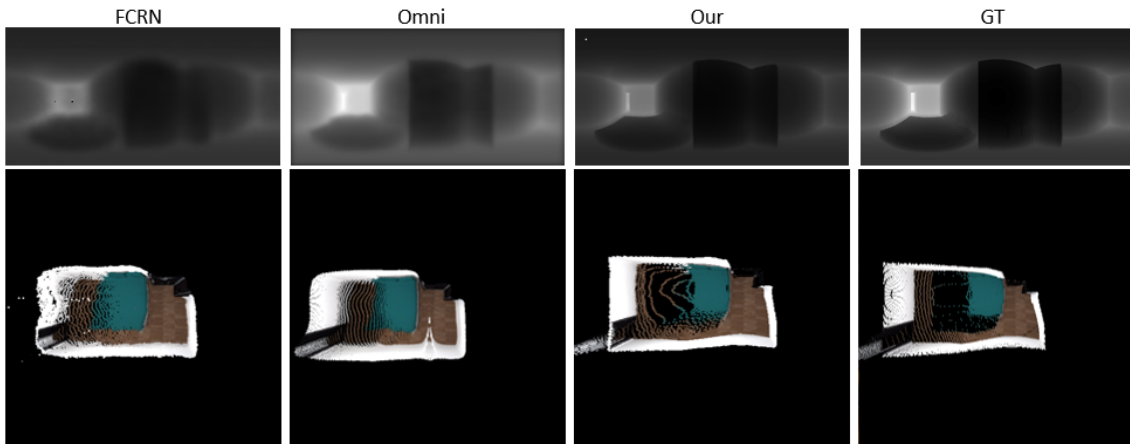


Figure 1.15: PanoSUNCG dataset

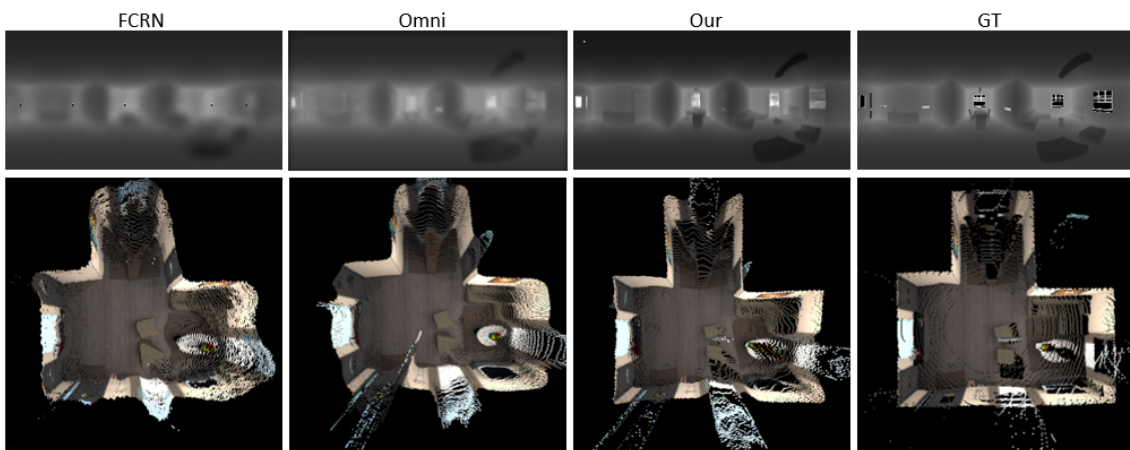


Figure 1.16: PanoSUNCG dataset

we fuse their features by our bi-projection fusion module. Furthermore, to extend the field-of-view of the cubemap projection and eliminate the boundary inconsistency of each cube face, we propose spherical padding which connects features from neighboring faces. Experimental results demonstrate that our method achieves state-of-the-art performance.

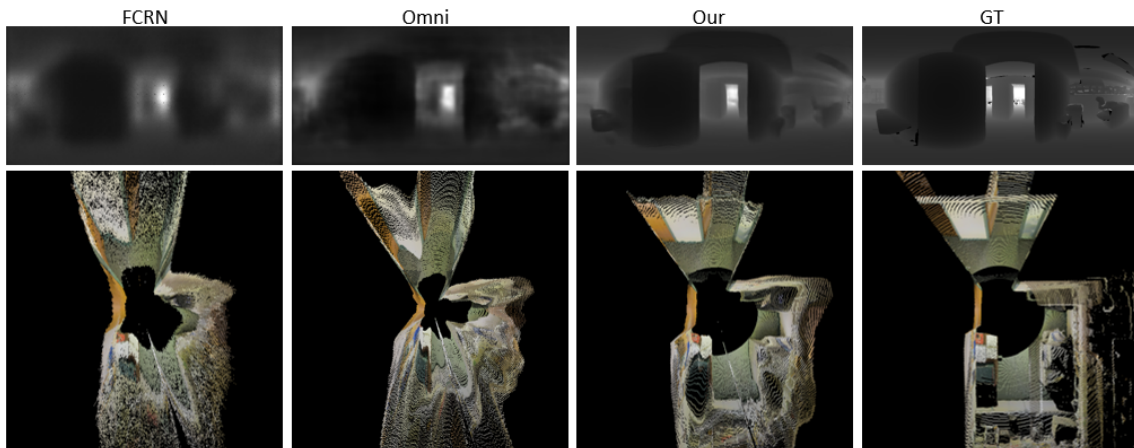


Figure 1.17: Stanford2D3D dataset

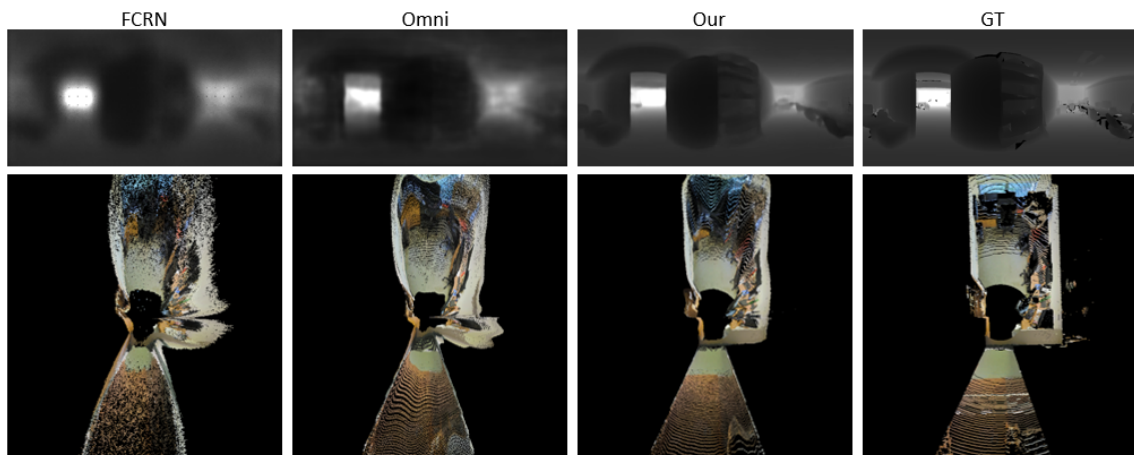


Figure 1.18: Stanford2D3D dataset

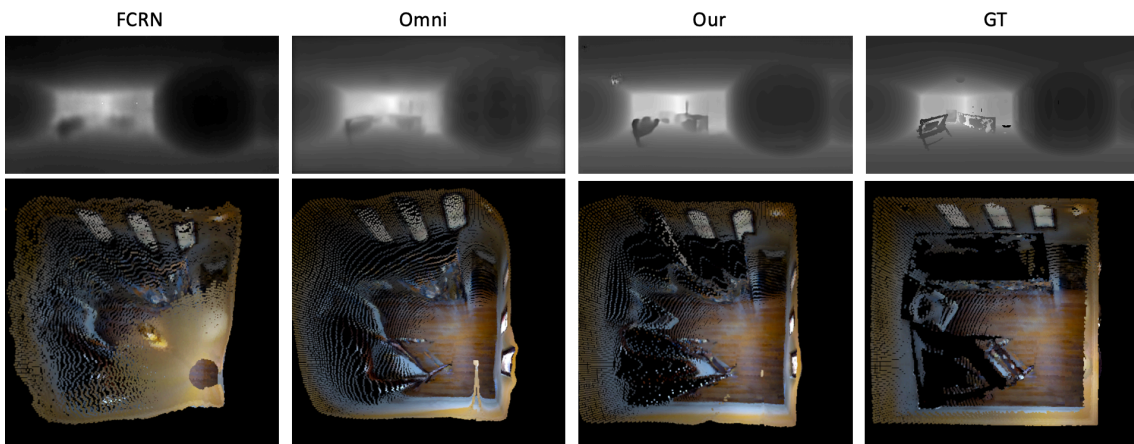


Figure 1.19: 360D dataset

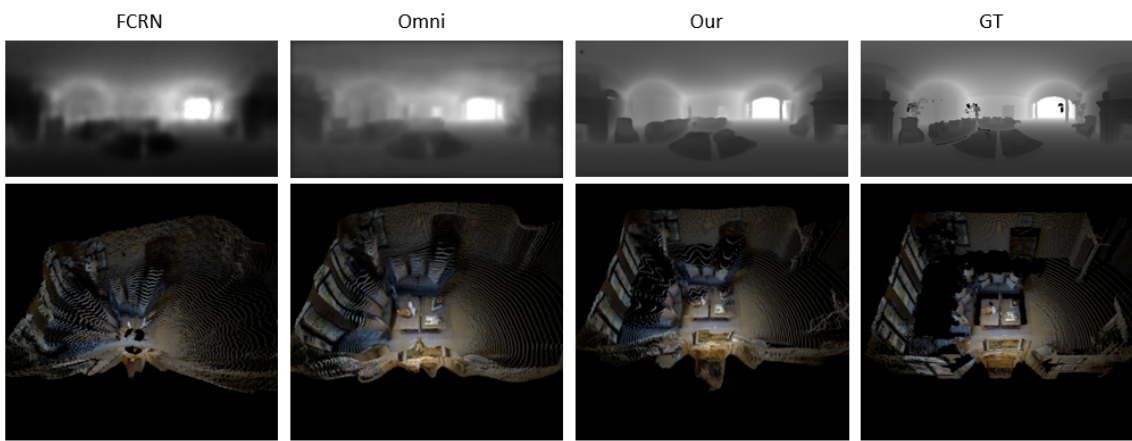


Figure 1.20: 360D dataset



Chapter 2 **BiFuse++: Self-supervised and Efficient Bi-projection Fusion for 360° Depth Estimation**

2.1 Introduction



Depth estimation from a single image is a crucial technique for many applications. For indoor autonomous systems, the geometric information of depth maps is necessary to improve navigation and exploration efficiency. Moreover, a thoughtful understanding of the environment provided by depth maps is also required to ensure the safety of the surrounding people. Traditionally, depth maps are captured by scanners such as LiDARs, structured light, or other time-of-flight (ToF) sensors. Such sensors are typically costly and, thus, have limited usages depending on the scenarios. With the advance of deep neural networks, depth estimation from the perspective cameras becomes a possible solution for these tasks. For instance, FCRN [4] is one popular framework for monocular depth estimation. However, most existing frameworks are designed only for perspective cameras with limited field-of-view (FoV), whereas some depth sensors (e.g., LiDARs) offer

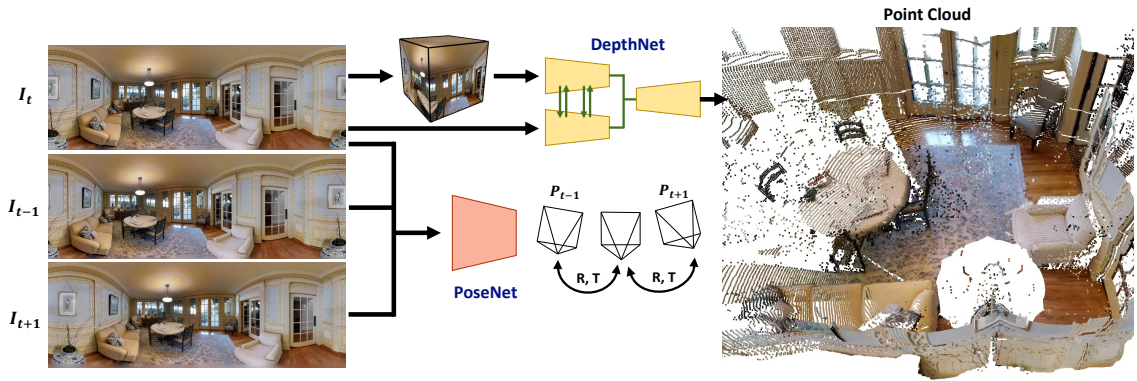


Figure 2.1: Our BiFuse++ is a self-supervised framework of monocular 360° depth estimation. The depth estimation network (DepthNet) is a bi-projection architecture consisting of two encoders and a shared decoder. The inputs of DepthNet are equirectangular and cubemap projections of reference panorama I_t . Between each adjacent layer of encoders, the feature maps of two projections are fused by our proposed fusion module (green arrows). To achieve self-supervised learning, an additional network (PoseNet) takes three adjacent panoramas (I_{t-1} , I_t , and I_{t+1}) in a video sequence as inputs and infers the corresponding camera motions (P_{t-1} and P_{t+1}). We then compute the photo consistency error based on the predicted depth map and camera motions to jointly train the two networks.

360° field-of-view.

With the rising availability of consumer-level spherical cameras, the omnidirectional camera turns into a good choice for indoor autonomous systems. By capturing the 360° information into a single panorama, 360° cameras can significantly increase the navigation and exploration efficiency. Coming along with this, 360° perception has become an important topic in computer vision. For instance, OmniDepth [9] and BiFuse [1] are frameworks for monocular 360° depth estimation. These frameworks use the ground truth depth maps captured by depth sensors as the supervisory signals to train the network. In general, there are two major issues for such 360° depth estimation frameworks: 1) Unlike common perspective cameras, the distortion introduced by equirectangular projection is extremely large, especially near the north and south poles on equirectangular coordinates. 2) the large number of depth maps captured by sensors is necessary to train the networks and thus highly increases the cost of data collection. Since the distortion can be removed by converting a single equirectangular image into several perspective ones where each of

them only covers limited FoVs (e.g., cubemap projection), BiFuse [1] utilizes the combination of equirectangular and cubemap projection as input to the framework. In this way, the 360° context can be preserved with the equirectangular projection, while the areas with large distortions can be resolved by the cubemap one. However, it is still necessary for the training of BiFuse to adopt a large-scale dataset consisting of precise depth ground truths from depth sensors, and the cost of data collection is usually large since the depth sensors with 360° FoV are expensive and not affordable by most consumers. Therefore, the requirement for a large amount of ground truth depth maps still prevents BiFuse from being extended to various scenes, thus reducing its scalability.

To reduce the cost of data collection, self-supervised learning approaches of monocular depth estimation like SfMLearner [22] are designed for normal FoV cameras. By utilizing SfMLearner and the cubemap projection, 360-SelfNet [5] is the first framework for self-supervised 360° depth estimation. However, only using cubemap and cubepadding [6] cannot provide complete information and also harms the stability of depth consistency around the cubemap boundary as addressed in [1]. To this end, combining both the equirectangular and cubemap projections appear to be a potential solution for self-supervised 360° depth estimation, in which such a combination has not been studied in this field.

In this paper, we extend the previous BiFuse [1] work and propose an advanced framework “BiFuse++” for self-supervised monocular 360° depth estimation. As illustrated in Figure 2.1, there are two networks, DepthNet and PoseNet, in our framework. DepthNet first estimates the depth map of the reference panorama I_t and PoseNet estimates the corresponding camera motions between adjacent panoramas (I_{t-1} and I_{t+1}). We then compute their photo consistency error to achieve self-supervised training. Our DepthNet is a bi-projection architecture which takes equirectangular and cubemap projections as in-

puts (I_t) to estimate the corresponding depth map. Motivated by UniFuse [43], we adopt a single decoder to unify feature maps from equirectangular and cubemap branches. To improve the performance and efficiency, we propose a new fusion module to exchange the information between different projections. Unlike UniFuse that simply infers equirectangular and cubemap feature maps from two independent encoders, our fusion module first fuses feature maps from two projections, and then the fused ones are further passed into the next layers of encoders. In this way, our encoders can directly retrieve the information from another branch and preserve more complete details on the predicted depth maps.

To infer the camera motions between adjacent panoramas, 360-SelfNet [5] adopts an additional network that takes the concatenation of adjacent panoramas as input and uses an encoder to extract the camera motions. In addition, a decoder consisting of transposed convolutional layers is adopted to infer the occlusion masks that are further leveraged in photometric loss. Instead of following 360-SelfNet to adopt an encoder-decoder architecture, our PoseNet is a single encoder that takes three adjacent panoramas (I_{t-1} , I_t , and I_{t+1}) as input, and infers the backward and forward camera motions (P_{t-1} and P_{t+1}), i.e., the camera motions from I_t to I_{t-1} and from I_t to I_{t+1} . We then directly adopt additional convolutional layers in the encoder to extract occlusion masks at different scales. With the predicted depth map and camera motions, we can achieve self-supervised training based on the photo consistency assumption. However, we find that the spherical photometric loss proposed by 360-SelfNet has a degeneration problem in low-texture areas and severely harms the training performance in real-world videos (see Figure 2.2). To this end, we propose “Contrast-Aware Photometric Loss (CAPL)” to deal with the degeneration. In addition to achieving self-supervised training on 360° videos, our BiFuse++ is also efficient and effective to be adopted in supervised training.

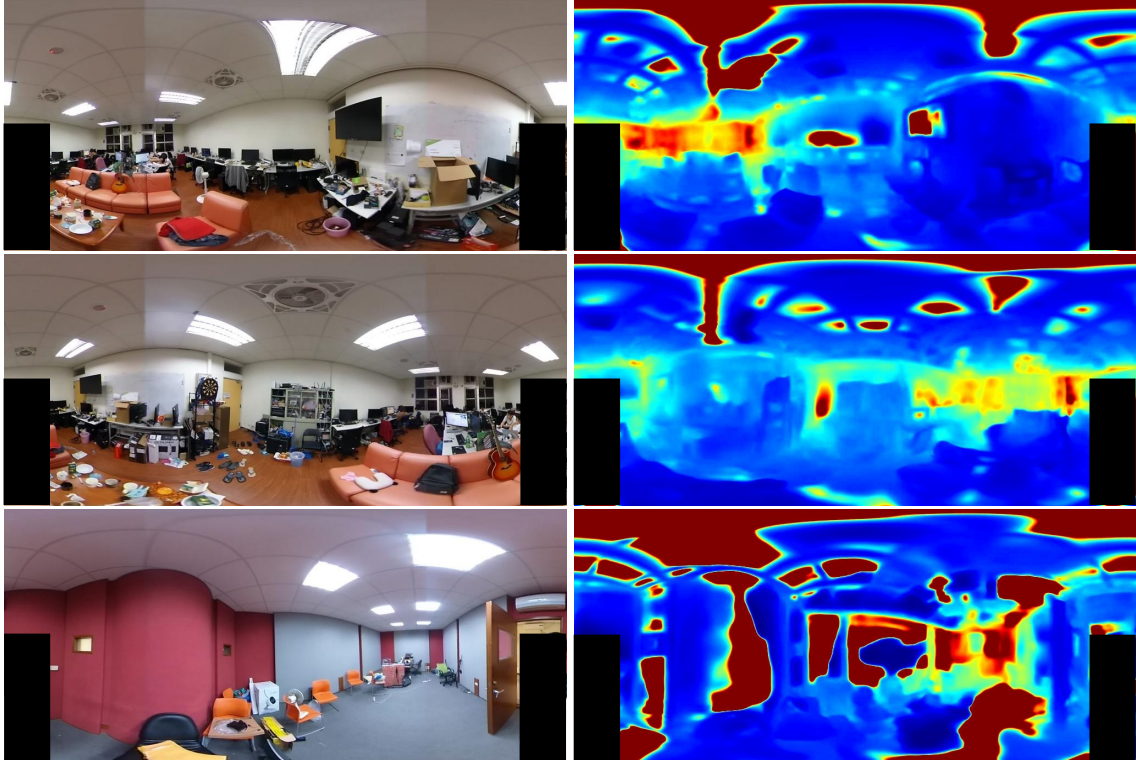


Figure 2.2: 360-SelfNet [5] trained on real-world images. The spherical photometric loss cannot deal with the low-texture area and thus produces unstable depth maps (red indicates a large depth value). Note that we mask out the photographer at the bottom left and right region.

To validate the applicability of BiFuse++, we conduct extensive experiments under both supervised and self-supervised scenarios. For the self-supervised scenario, we perform experiments on the PanoSUNCG [5] dataset. For the supervised scenario, we evaluate our method on Matterport3D [8], Stanford2D3D [10], and PanoSUNCG [5]. In general, our BiFuse++ achieves state-of-the-art performance under self-supervised scenarios and is comparable with HoHoNet [44] for supervised training. To investigate the benefit of incorporating cubemap projection, we add rotation noise into both training and testing datasets. BiFuse++ is shown to be robust for panoramic distortion. Furthermore, we estimate the computational cost of different fusion architectures, and our BiFuse++ achieves the lowest inference memory usage among the fusion approaches. In general, BiFuse++ reduces about 80% of parameters compared to BiFuse, while achieving a sig-

nificantly better performance of depth estimation. Hence, our proposed method is efficient and effective for 360° depth estimation.

To summarize, our contributions are the following:

1. BiFuse++ is the first work that integrates the bi-projection fusion architecture into self-supervised monocular 360° depth estimation.
2. We propose a new fusion module to improve the efficiency while preserving complete details in the depth maps.
3. Our framework achieves state-of-the-art performance under self-supervised training scenario and comparable with recent approaches under supervised training.

2.2 Related Works

Supervised Monocular Depth Estimation. Saxena *et al.* [11] is the pioneering work lifting a 2D image into a 3D model. With the advance in deep learning, approaches based on convolutional neural networks are studied. Eigen *et al.* [12] first propose using a deep neural network to estimate the depth map from a single image. Laina *et al.* [4] adopt ResNet [13] as the encoder and propose an up-projection module to upsample the feature maps. In addition, reverse Huber loss is proposed to balance the difference between small and large error areas of the estimated depth maps. To further improve the estimated depth maps, several approaches utilizing conditional random field (CRF) are proposed [15–19]. Cao *et al.* [15] treat depth estimation as a classification task and apply CRF to refine it. By leveraging ordinal regression into the deep neural network, Fu *et al.* [20] propose a deep ordinal regression network (DORN) that formulates depth estimation as a classification

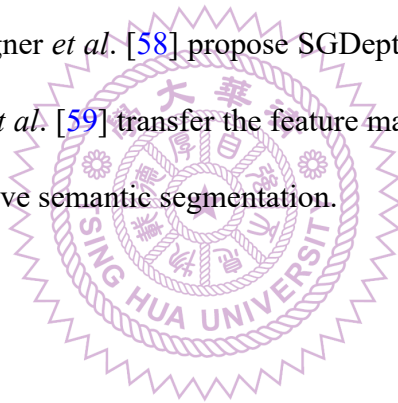
task. Ranftl *et al.* [45] propose to combine data from different sources and mix several datasets to greatly improve monocular depth estimation. Bhat *et al.* [46] propose to predict the depth maps as a linear combination of bins.

However, accurate depth maps measured by laser scanners like LiDARs or Kinect are required for the methods above, which significantly increases the cost of collecting a large amount of training data, and thus self-supervised approaches are studied.

Self-Supervised Monocular Depth Estimation. Xie *et al.* [47] collect the training data from 3D movies, i.e., the left and right frames, and propose Deep3D that infers the left or right frame of the input image to convert a 2D image into a 3D one. Garg *et al.* [48] propose a stereopsis-based framework that takes a single image of a rectified stereo pairs as input and infers the corresponding depth maps by image reconstruction error. Godard *et al.* [21] use the stereo image pairs and left-right photometric consistency to achieve self-training of monocular depth estimation. Since stereo cameras are still less popular than monocular ones for consumer-level devices, self-training approaches using sequential videos are studied. Zhou *et al.* [22] propose using two sub-networks to estimate both the monocular depth map and the camera pose of the sequential pairs in the training stage. In this way, a depth estimation network can be directly trained with a large number of monocular videos without any annotation or calibration, which highly improves the scalability of depth estimation. Vijayanarasimhan *et al.* [49] propose forward-backward constraints and leverage [50] to deal with the rigid motions of dynamic objects in the scene. Yang *et al.* [25] use depth-normal consistency to improve the depth estimation. Godard *et al.* [51] propose an automatic masking technique to efficiently mask out the moving objects. Johnston *et*

al. [52] use self-attention and discrete disparity to improve depth estimation. Guizilini *et al.* [53] propose PackNet to improve the generalization ability of the depth estimation network on the out-of-domain data. Bian *et al.* [54] propose a self-discovered masking scheme to detect moving objects in the videos.

To encode the structural information into the network, object-level information is used to improve the performance of self-supervised depth estimation. Guizilini *et al.* [55] utilize an additional segmentation network to guide the depth estimation network. Chen *et al.* [56] propose SceneNet to jointly constrain semantic and geometric understanding with content consistency. Zhu *et al.* [57] propose explicit border consistency between segmentation and depth map. Klingner *et al.* [58] propose SGDepth to solve moving objects by semantic guidance. Hoyer *et al.* [59] transfer the feature maps of a self-supervised depth estimation network to improve semantic segmentation.



360° Perception. Recently, since omnidirectional cameras, e.g., fisheye and 360° cameras, are widely used, people have started to focus on topics of panoramas. To extend the existing deep neural network techniques to panoramas, the distortion introduced by the equirectangular projection increases the instability of performance. Cheng *et al.* [6] first use cubemap projection to solve the distortion and propose cubepadding to extend the receptive field of each face. Wang *et al.* [60] incorporate circular padding and rotation invariance into the deep neural network. In addition to avoiding the distortion with projections, several distortion-aware convolutional approaches are proposed. Esteves *et al.* [32] and Cohen *et al.* [31] propose spherical CNNs by using Fourier transformation to implement the spherical correlation. Su *et al.* [33,34] use different convolutional kernels

according to the longitude and latitude on the equirectangular projection, and also adapt a pretrained model of perspective camera for inference procedure.

360° Depth Estimation. Based on the cube padding strategy (Cheng *et al.* [6]), Wang *et al.* [5] propose 360-SelfNet that is the first framework of self-supervised 360° depth estimation. Zioulis *et al.* [9] incorporate SphConv (Su *et al.* [34]) into the encoder to overcome the panoramic distortion and propose a framework for 360° depth estimation. Zioulis *et al.* [61] use CoordConv (Liu *et al.* [62]) and trinocular view synthesis to improve the performance. Inspired by Yang *et al.* [7] adopting a combination of different projections, Wang *et al.* [1] propose BiFuse that is a two-branch architecture and utilizes cubemap and equirectangular projections. Since the cube padding does not follow the projection geometry, “spherical padding” based on the spherical projection is introduced. Moreover, to better leverage the information of two projections, i.e., equirectangular and cubemap, “bi-projection fusion” is applied to fuse the feature maps. To improve efficiency, Jiang *et al.* [43] propose UniFuse to simplify the framework of BiFuse. To improve depth prediction, Jin *et al.* [63] and Zeng *et al.* [64] utilize layout information to provide more context to neural networks. By leveraging 1-D representation proposed in HorizonNet (Sun *et al.* [65]), Sun *et al.* [44] and Pintore *et al.* [66] propose HoHoNet and SliceNet to train a 360° depth estimation network.

However, only a few of the abovementioned works try to discuss monocular 360° depth estimation under the self-supervised training scenario. The appropriate and efficient design of networks under such a scenario has not been studied well in the literature. In this paper, we propose “BiFuse++”, a self-supervised 360° depth estimation framework,

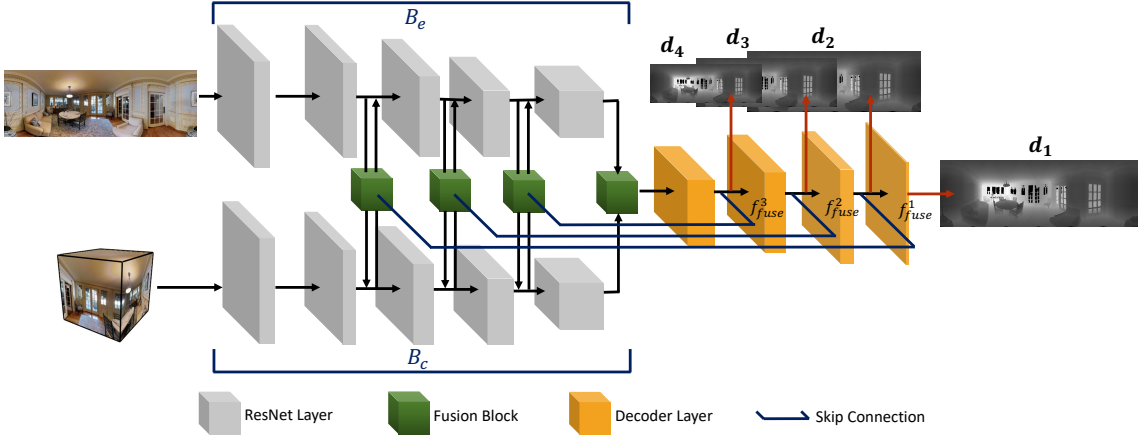


Figure 2.3: The overview of our DepthNet. Our DepthNet consists of two encoders (B_e and B_c) based on ResNet-34 and a single shared decoder that unifies the feature maps from the two encoders. The inputs are the equirectangular and cubemap projections converted from a single panorama, and the output is the corresponding equirectangular depth map. During the encoding procedure, the feature maps of B_e and B_c are fused by our proposed fusion module (green). Unlike [1] and [43], our fusion module refines the original feature maps and the refined ones are then passed into next layers of B_e and B_c . To preserve complete details in the final predicted depth maps, we add three skip-connections by concatenating the fused feature maps ($f_{fuse}^1, f_{fuse}^2, f_{fuse}^3$) from fusion modules with decoded feature maps. Then, we extract multi-scale depth maps (d_1, d_2, d_3 , and d_4) from these concatenated feature maps by 1×1 convolutional layers.

to improve the depth estimation efficiency and accuracy. Our framework can be adopted in both supervised and self-supervised training scenarios, and we conduct extensive experiments to verify BiFuse++ under the two scenarios. We will detail our approach in the following sections.

2.3 Approach

Our BiFuse++ is the first framework that utilizes bi-projection architecture in self-supervised training for monocular 360° depth estimation. In addition, we propose several components to improve the performance and efficiency of BiFuse [1] and 360-SelfNet [5]:

1. We propose an advanced bi-projection architecture that significantly reduces the model size while improving depth estimation performance compared with BiFuse.

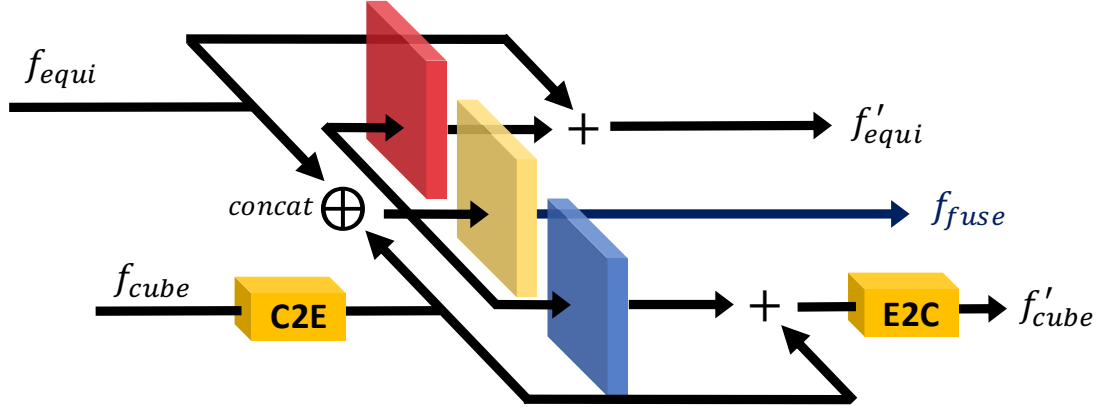


Figure 2.4: The architecture of our fusion module. The feature maps from equirectangular and cubemap branches are first concatenated and passed into three convolutional blocks. Then, we add a skip connection to the original feature maps and obtain the fused feature maps f'_{equi} and f'_{cube} , which are the inputs of the next convolutional layers. In addition, the other fused feature map f_{fuse} are concatenated in the decoding process later.

2. We propose a new fusion module that is able to effectively share the information between different projections while taking the lowest number of parameters.
3. To improve the training stability of 360-SelfNet on real-world videos (see Figure 2.2), we propose Contrast-Aware Photometric Loss to balance photo consistency error difference between high-texture and low-texture areas.

We first explain the spherical projection and introduce basic transformations in Sec. 2.3.1. In section 2.3.2, we first detail the proposed fusion module of BiFuse++. Then, we introduce the design of our entire framework, including network architectures and the adaptation to supervised and self-supervised training scenarios. Lastly, we explain the proposed Contrast-Aware Photometric Loss and other loss functions we adopted in BiFuse++.

2.3.1 Spherical Projection

For a cubemap representation of which side length is equal to w , we denote i as the six faces $i \in \{B, D, F, L, R, U\}$, to represent the faces of back, down, front, left, right,

and up, respectively. Since the field-of-view (FoV) of each face is equal to 90° ; each face can be considered as a perspective camera whose focal length is $\frac{w}{2}$ and all faces share the same center point in world coordinate. Since the viewing direction of each face is fixed in cubemap projection, the corresponding extrinsic matrix of each camera can be defined by a rotation matrix R_i . For a pixel p on a certain face i , we can transform it into the coordinate on the equirectangular projection by the following mapping:

$$K = \begin{bmatrix} w/2 & 0 & w/2 \\ 0 & w/2 & w/2 \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.1)$$

$$q = R_i \cdot K^{-1} \cdot \hat{p},$$

$$\theta = \arctan\left(\frac{q^x}{q^z}\right),$$

$$\phi = \arcsin\left(\frac{q^y}{|q|}\right).$$

where w is the dimension of i and \hat{p} is the homogeneous representation of p ; and θ and ϕ are longitude and latitude in equirectangular projection; and q^x, q^y, q^z are x-y-z components of q , respectively. We call such mapping as cube-to-equirectangular (C2E) transformation. Since C2E mapping is reversible, we call the reverse one as equirectangular-to-cube (E2C) transformation. We detail E2C transformation in the following:

$$q^x = \sin(\theta) \cdot \cos(\phi),$$

$$q^y = \sin(\phi),$$

$$q^z = \cos(\theta) \cdot \cos(\phi),$$

$$\hat{p} = K \cdot R_i^T \cdot q.$$

(2.2)

Both E2C and C2E transformation are extensively used in the architecture of BiFuse. For the convenient purpose, we use $\pi(q) = (\theta, \phi)$ and $\pi^{-1}(\theta, \phi) = q$ to represent the forward and inverse spherical projection. To convert an equirectangular depth map to the corresponding point cloud, and project them onto another equirectangular image, we define the following mapping function:

$$\begin{aligned} \hat{q} &= R \cdot d \cdot \pi^{-1}(\theta, \phi) + t, \\ (\hat{\theta}, \hat{\phi}) &= \pi\left(\frac{\hat{q}}{|\hat{q}|}\right). \end{aligned} \tag{2.3}$$

where d is the depth value, R and t are the camera pose (rotation and translation) of target equirectangular image, and $(\hat{\theta}, \hat{\phi})$ are the projected longitude and latitude on target equirectangular image, respectively.

2.3.2 Our BiFuse++ Framework

The overview of our BiFuse++ is illustrated in Figure 2.1. To achieve self-supervised learning for monocular 360° depth estimation, our training process takes three adjacent panoramas extracted from video sequences, and we adopt two networks, i.e., DepthNet and PoseNet, to estimate the depth map and camera motions. In our DepthNet, we use our proposed fusion module to exchange the information of different projections. With the predicted depth map and camera motions, we propose “Contrast-Aware Photometric Loss” to self-supervise the two networks. The details of each component are explained in the following.

Fusion Module. The overview of our fusion module is illustrated in Figure 2.4. Our fusion module consists of three convolutional layers and their inputs are the concatenation

of f_{equi} and f_{cube} that are the feature maps of different projections, i.e., equirectangular and cubemap. The red and blue layers are adopted as the residual block to refine the feature maps of different projections, while the last convolution layer (yellow) learns a fused feature map of both projections. Thus, there are three output feature maps, f'_{equi} , f'_{cube} , and f_{fuse} , from our fusion module. Specifically, we generate the three feature maps as:

$$\begin{aligned}
 f'_{equi} &= f_{equi} + H_e(f_{equi} \oplus C2E(f_{cube})) , \\
 f'_{cube} &= E2C(C2E(f_{cube}) + H_c(f_{equi} \oplus C2E(f_{cube}))) , \\
 f_{fuse} &= H_f(f_{equi} \oplus C2E(f_{cube})) ,
 \end{aligned} \tag{2.4}$$

where H_e , H_c , and H_f are convolutional layers, and \oplus is the concatenation operation. f_{fuse} is then leveraged in our decoder. f'_{equi} and f'_{cube} are then passed into the next convolutional layer of the encoder in our network. In this way, the image details can be well preserved in our final predicted depth maps.

Depth Estimation Network (DepthNet). The overview of our DepthNet is illustrated in Figure 2.3. We take the equirectangular and cubemap projections converted from a single panoramic image as inputs, and our DepthNet predicts the corresponding depth map in equirectangular projection. DepthNet consists of two encoders based on ResNet-34 [13] to extract the feature maps from equirectangular and cubemap panoramas. We apply our fusion module to fuse the feature maps between each ResNet layer of the two encoders. Different from [43], we have the refined feature maps from our fusion module (f'_{equi} and f'_{cube} in Figure 2.4) forwarded into the next layers of our encoders. In this way, the benefits of different projections can be early received in our encoders, and we find that such a mechanism can well preserve the details of panoramas. Similar to [43], we

adopt a single UNet-like decoder to simplify the decoder of BiFuse. We add three skip-connections by concatenating the fused feature maps from our fusion modules (f_{fuse} in Figure 2.4 and $f_{fuse}^{1,2,3}$ in Figure 2.3) with our decoder layers. In addition, we adopt sub-pixel convolution [67] as our decoder layers to improve both final accuracy and reduce memory consumption compared to [43] and [1]. After each decoder layer, we extract the corresponding depth maps of four scales $\{d_s\}_{s=1}^4$ by 1x1 convolutional layers, and we follow [22] to add a sigmoid layer after them, with α and β to control the range of the depth value.

$$d'_s = \alpha \cdot \rho(f_s) + \beta, \tag{2.5}$$

$$d_s = \frac{1}{d'_s}.$$

where s denotes the scale, f_s is the output of the four convolutional layers, α and β are hyper-parameters, ρ is the sigmoid function, and d_s is the depth map of scale s . In this paper, we follow [22] and set $\alpha = 10$ and $\beta = 0.01$.

Pose Estimation Network (PoseNet). To achieve self-supervised depth estimation on videos, both depth and camera motion are required to estimate photo consistency errors. Hence, we adopt an addition network (PoseNet) to estimate the corresponding camera motions between adjacent frames. As illustrated in Figure 2.5, we adopt a single ResNet-18 [13] encoder of which inputs are the concatenation of three sequential panoramas (I_{t-1} , I_t , and I_{t+1}). We estimate the backward and forward camera motions P_{t-1} and P_{t+1} to jointly calculate their photo consistency errors. Since photo consistency has ambiguity in occluded areas; we have four additional 3x3 convolutional layers that take the four feature maps from ResNet-18 as inputs and estimate the occlusion masks at four scales, which are denoted as $\{X_s\}_{s=1}^4$. We then use the occlusion masks to suppress ambiguous areas and

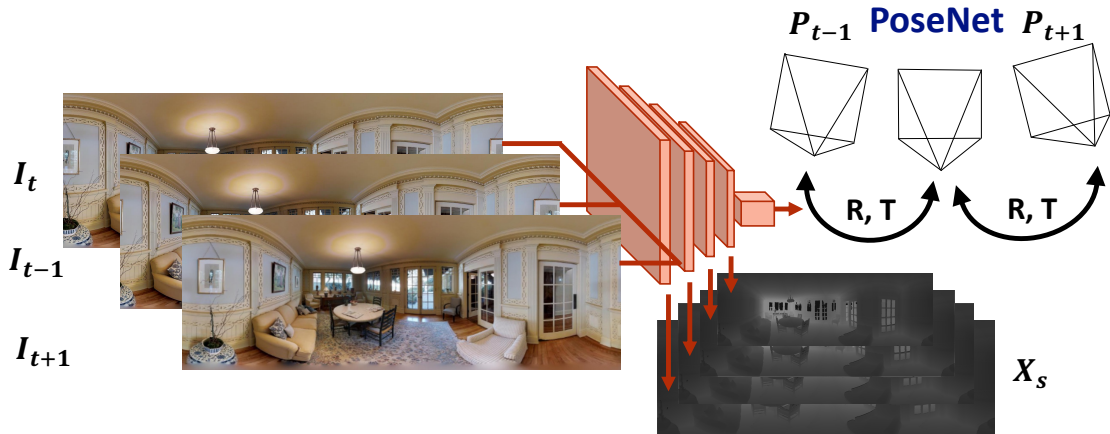


Figure 2.5: The overview of our PoseNet. Our PoseNet is based on ResNet-18 and the inputs are three sequential panoramas (I_{t-1}, I_t, I_{t+1}) in a video and PoseNet infers the corresponding camera motion P_{t-1} and P_{t+1} . To suppress the ambiguity of photo consistency error in occluded areas and stabilize the training, PoseNet estimates four occlusion masks X_s to find the occluded areas.

stabilize the training.



Self-supervised Loss Function. We use the photo consistency assumption, i.e., the image intensity is consistent across reprojected frames given the depth and camera motion, to train our DepthNet and PoseNet in a self-supervised fashion. Regarding the loss function, 360-SelfNet [5] first proposes “Spherical Photometric Loss (SPL)” to calculate the photo consistency error with spherical projection. However, 360-SelfNet cannot estimate stable depth maps on low-texture areas in real-world videos (Figure 2.2). We find that there is a degeneration problem in SPL, i.e., the lower SPL is not always equal to more accurate depth maps, and we show supporting evidence in Section 2.4. The degeneration comes from the ambiguity of photo consistency assumption that the consistency errors of pixels in low-texture areas are meaningless; such ambiguity can seriously harm the training results in real-world videos. To this end, we propose “Contrast-Aware Photometric Loss

(CAPL)” to prevent networks from being affected by these low-texture areas:

$$\mathcal{L}_{CAPL}^s(I_{t,s}) = \sum_{p=1}^N X_s(p) \cdot \sigma(I_{t,s}(p)) \cdot \delta(I_{t,s}(p)) , \quad (2.6)$$

$$\delta(I_{t,s}(p)) = |I_{t,s}(p) - \hat{I}_{t-1,s}(p)| + |I_{t,s}(p) - \hat{I}_{t+1,s}(p)| ,$$

where p denotes pixels, N is the total number of pixels, X_s is the predicted occlusion mask, and σ is the standard deviation of p in a 5x5 window. δ is the photo consistency error of $I_{t,s}$; $\hat{I}_{t-1,s}$ and $\hat{I}_{t+1,s}$ are the warped panoramas of $I_{t,s}$ after being reprojected onto $I_{t-1,s}$ and $I_{t+1,s}$ by the predicted depth map d_s and camera motions (P_{t-1} and P_{t+1}), respectively. δ is first multiplied by the occlusion mask to remove unreasonable depth values, and then we use the standard deviation of p to solve the degeneration problem since the standard deviation in low-texture areas are usually small. In this way, CAPL can significantly improve the quality of predicted depth maps in real-world videos.

In addition to CAPL, we adopt two regularization terms to provide constraints for occlusion masks and predicted depth maps. To prevent predicted occlusion masks from decaying to zero, we apply a binary cross-entropy loss to the masks:

$$\mathcal{L}_m(X_s) = - \sum_{p=1}^N \log(X_s(p)) , \quad (2.7)$$

This regularization provides a large penalty when the values in occlusion masks are small.

To reduce the noise on the predicted depth maps, we apply smooth regularization to the predicted depth map:

$$\mathcal{L}_{sm}(d_s) = \sum_{p=1}^N |\nabla(d_s(p))| . \quad (2.8)$$

Supervised and Self-Supervised Training. In this paper, our proposed framework is evaluated under both supervised and self-supervised training scenarios. In supervised

training, our DepthNet is directly trained with ground truth depth maps, and we adopt reverse Huber loss [4] as our loss function \mathcal{L}_{berHu} :

$$\mathcal{L}_{berHu} = \sum_{s=1}^4 \sum_{p=1}^N \mathcal{B}(d_s(p), \hat{d}_s(p)), \quad (2.9)$$

$$\mathcal{B}(d_s(p), \hat{d}_s(p)) = \begin{cases} |d_s(p) - \hat{d}_s(p)| & |d_s(p) - \hat{d}_s(p)| \leq c, \\ \frac{(d_s(p) - \hat{d}_s(p))^2 + c^2}{2c} & |d_s(p) - \hat{d}_s(p)| > c, \end{cases} \quad (2.10)$$

where p is pixels, while d_s and \hat{d}_s are the predicted and ground truth depth maps, respectively. c is typically set to $0.2 \cdot \max(|d_s(p) - \hat{d}_s(p)|)$.

In self-supervised training, DepthNet and PoseNet are trained with three abovementioned loss terms: 1) Contrast-Aware Photometric Loss, 2) occlusion mask regularization, and 3) smooth regularization. The final loss function is then established as:

$$\mathcal{L}_{ss} = \sum_{s=1}^4 \mathcal{L}_{CAPL}^s(I_{t,s}) + w_1 \cdot \mathcal{L}_m(X_s) + w_2 \cdot \mathcal{L}_{sm}(d_s), \quad (2.11)$$

where w_1 and w_2 are hyper-parameters.

2.4 Experimental Results

Table 2.1: The quantitative results on Matterport3D [8].

Method	MAE ↓	MRE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
FCRN	0.4008	0.2409	0.6704	0.1244	0.7703	0.9174	0.9617
OmniDepth	0.4838	0.2901	0.7643	0.1450	0.6830	0.8794	0.9429
BiFuse	0.3470	0.2048	0.6259	0.1134	0.8452	0.9319	0.9632
UniFuse	0.3160	0.1592	0.5485	0.0926	0.8490	0.9463	0.9747
SliceNet	0.3296	0.1764	0.6133	0.1045	0.8716	0.9483	0.9716
HoHoNet	0.2862	0.1488	0.5138	0.0871	0.8786	0.9519	0.9771
BiFuse++	0.2842	0.1424	0.5190	0.0862	0.8790	0.9517	0.9772

We first introduce the common evaluation metrics, the benchmark datasets (Sec. 2.4.1),

Table 2.2: The quantitative results on Stanford2D3D [10]. Note that SliceNet* is a re-implemented version.

Method	MAE ↓	MRE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
FCRN	0.3428	0.1837	0.5774	0.1100	0.7230	0.9207	0.9731
OmniDepth	0.3743	0.1996	0.6152	0.1212	0.6877	0.8891	0.9578
BiFuse	0.2343	0.1209	0.4142	0.0787	0.8660	0.9580	0.9860
UniFuse	0.2198	0.1195	0.3875	0.0747	0.8686	0.9621	0.9870
SliceNet*	0.2484	0.1249	0.4370	0.0873	0.8377	0.9414	0.9777
HoHoNet	0.2027	0.1014	0.3834	0.0668	0.9054	0.9693	0.9886
BiFuse++	0.2173	0.1117	0.3720	0.0727	0.8783	0.9649	0.9884

Table 2.3: The quantitative results on PanoSUNCG [5].

Method	MAE ↓	MRE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
FCRN	0.1346	0.0979	0.3973	0.0692	0.9223	0.9659	0.9819
OmniDepth	0.1624	0.1143	0.3710	0.0882	0.8705	0.9365	0.9650
BiFuse	0.0789	0.0592	0.2596	0.0443	0.9590	0.9823	0.9907
UniFuse	0.0776	0.0528	0.2704	0.0441	0.9591	0.9825	0.9906
BiFuse++	0.0688	0.0524	0.2477	0.0414	0.9630	0.9835	0.9911

and implementation details (Sec. 2.4.2). For performance evaluation, we validate the improved accuracy of BiFuse++ network architecture with supervised training scenarios (Sec. 2.4.3) on three datasets: Matterport3D [8], Stanford2D3D [10], and PanoSUNCG [5]. We further test the robustness of our method by evaluating the performance under rotation noise. Moreover, we validate the computational efficiency of BiFuse++ with respect to existing approaches (Sec. 2.4.4). Then, we use the BiFuse++ network with low inference memory to conduct self-training efficiently on PanoSUNCG [5] (Sec. 2.4.5). Moreover, we also capture several videos in the real-world environment and conduct qualitative comparisons to show the applicability of BiFuse++.

2.4.1 Evaluation Metrics and Datasets

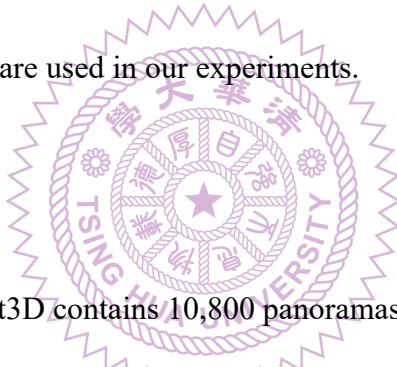
We use standard evaluation protocols in depth estimation, i.e., MAE (mean absolute error), MRE (mean relative error), RMSE (root mean square error), RMSE (log) (scale-

invariant root mean square error), and δ (threshold). For RMSE (log), we use log-10 for the computation. During evaluations, we follow [1] to ignore the area in which ground truth depth values are larger than 10 meters. Since the scale of self-training results is ambiguous, we follow [22] and apply median alignment before evaluation for self-supervised scheme:

$$d' = d \cdot \frac{\text{median}(\hat{d})}{\text{median}(d)}. \quad (2.12)$$

where d is the predicted depth map, \hat{d} is the ground truth one, and d' is the median-aligned depth map used for evaluation.

The following datasets are used in our experiments.



Matterport3D. Matterport3D contains 10,800 panoramas and the corresponding depth ground truth captured by Matterport’s Pro 3D Camera, a structured-light scanner. This dataset is the largest real-world dataset for indoor panorama scenes. However, the depth maps from sensors usually have noise or missing value in certain areas. In practice, we filter areas with missing values during training. We follow the official split to train and test our network, which takes 61 rooms for training and the others for testing. We resize the resolution of images and depth maps into 512×1024 .

Stanford2D3D. Stanford2D3D is collected from three kinds of buildings in the real world, containing six large-scale indoor areas. The dataset contains 1413 panoramas, and we use one of the official splits that takes the fifth area (area 5) for testing, and the others

are for training. During training and testing, we resize the resolution of images and depth maps into 512×1024 .

PanoSUNCG. PanoSUNCG contains 103 scenes of SunCG [41] and has 25,000 panoramas. In our experiments, we use the official training and testing splits, where 80 scenes are for training and 23 for testing. For the supervised scheme, we resize them to 256×512 and filter out pixels with depth values larger than 10 meters. For the self-supervised one, we keep the original resolution 512×1024 .

2.4.2 Implementation Details

We implement our network using PyTorch [39]. We use Adam [40] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our batch size is 8, and the learning rate is 0.0003. Unlike BiFuse [1], training the network and fusion module separately, we train the entire framework jointly. The ResNet encoders of DepthNet and PoseNet are first pretrained on ImageNet [68], and we apply uniform initialization to all other layers. We train the networks for 100 epochs for supervised scenarios, while the networks are trained for 60 epochs for self-supervised scenarios. Following [5], we set w_1 and w_2 of Equation (2.11) to 0.1 and 0.01, respectively.

Supervised and Self-supervised Training. In the supervised training scenario, we directly use the monocular images and corresponding depth maps provided by the above-mentioned datasets to train our DepthNet. In other words, the training is similar to monoc-

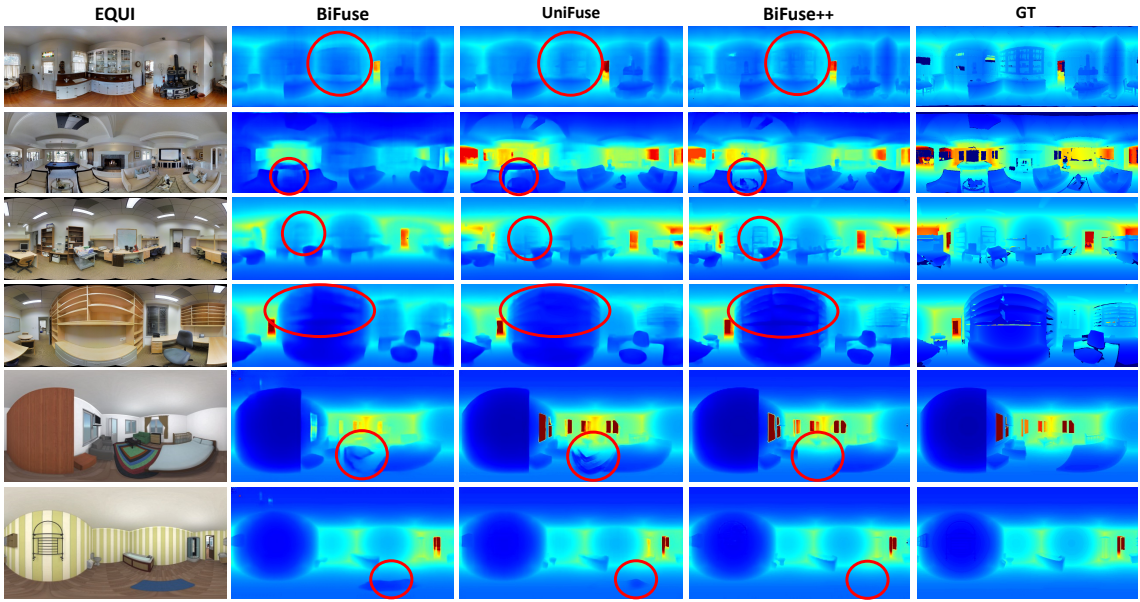


Figure 2.6: The qualitative results on Matterport3D, Stanford2D3D, and PanoSUNCG under the supervised scenario (every two rows show qualitative results of each dataset). Note that the dark blue and red colors indicate close and far distance, and we use red circles to highlight the inconsistent predictions of all approaches.

ular depth estimation like BiFuse [1]. Since we can directly acquire supervision from ground truth depth maps for training DepthNet with Equation (2.9), our PoseNet is not involved in this scenario. For the self-supervised scenario, since we cannot access ground truth depth maps for training, we use the RGB video sequences from PanoSUNCG [5] to self-supervise both DepthNet and PoseNet. Specifically, PoseNet takes three sequential panoramas (I_{t-1} , I_t , I_{t+1} in Figure 2.5) as input and infers the camera motions between them, while DepthNet takes only I_t as input and predicts the corresponding depth map. With the predicted monocular depth map and camera motions, we can self-supervise the two networks with Equation (2.11).

2.4.3 Results of Supervised Scenario

For supervised scenarios, we compare BiFuse++ with works of monocular depth estimation, including approaches designed for both perspective and spherical cameras.

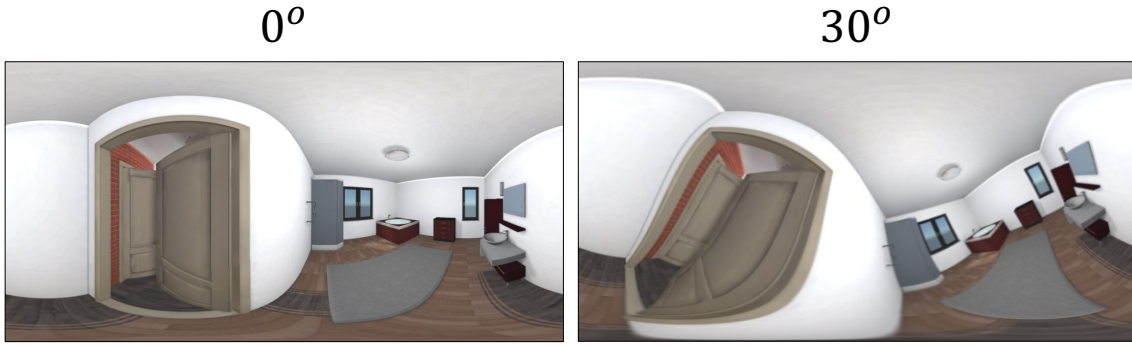


Figure 2.7: The distortion introduced by equirectangular projection. When the pitch of the camera is 0° , the structure of the room is clear. As the pitch becomes larger, the effect of equirectangular distortion is more obvious. The distortion affects the training stability when applying existing approaches designed for perspective cameras to panoramas.

1) FCRN [4], a strong approach designed for perspective cameras. 2) OmniDepth [9], a framework designed for spherical cameras that incorporates [34] into the architecture. For 1-D representation approaches, we compare our method with SliceNet [66] and HoHoNet [44].

The quantitative results are shown in Table 2.1-2.3. BiFuse++ achieves comparable results with the latest state-of-the-art 1D-representation HoHoNet and outperforms all other methods under the assumption that all input images are well aligned with the gravity direction.

Rotation Noise Evaluation. Since the spherical cameras adopted by the abovementioned datasets are well aligned with the gravity direction, the rotation of panoramas is usually small. This assumption benefits the most for 1-D representation like HoHoNet. To investigate the result without this assumption, we conduct the following experiment. For training and testing, we introduce rotation noise (see 0° and 30° rotation in Figure 2.7) on the Matterport3D and Stanford2D3D datasets. Specifically, we uniformly sample an angle between 30° and -30° to rotate panoramas and ground truth depth maps during each training iteration. For testing sets, we also apply the rotation noise, but the rotated angles

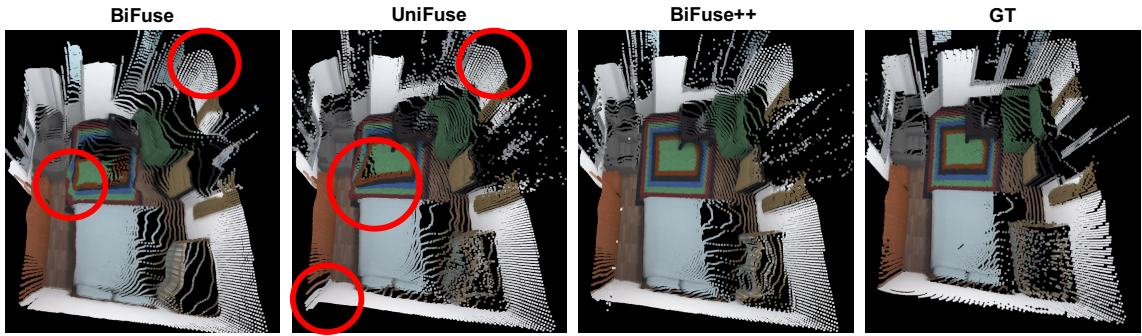


Figure 2.8: The 3D reconstruction comparison of BiFuse++ with other baselines. We note that the red circles indicate the incorrect depth prediction. Our BiFuse++ is able to preserve the corner details, while the other approaches predict inconsistent results.

of each image are consistent across all baselines. BiFuse++ is robust to rotation noise compared to other methods, as shown in Table 2.4-2.5.

Table 2.4: The quantitative results after applying rotation noise on Matterport3D [8].

Method	MAE	MRE	RMSE	RMSE (log)
UniFuse	0.3197	0.1627	0.5456	0.0941
SliceNet	0.3669	0.1863	0.6124	0.1058
HoHoNet	0.3085	0.1486	0.5385	0.0897
BiFuse++	0.3054	0.1521	0.5293	0.0897

Table 2.5: The quantitative results after applying rotation noise on Stanford2D3D [10].

Method	MAE	MRE	RMSE	RMSE (log)
UniFuse	0.2542	0.1289	0.4209	0.0819
SliceNet	0.2892	0.1459	0.5038	0.1001
HoHoNet	0.2210	0.1116	0.4001	0.0744
BiFuse++	0.2193	0.1134	0.3890	0.0742

Qualitative Discussion. The qualitative comparison of fusion approaches are shown in Figure 2.6. Compared with BiFuse [1], BiFuse++ achieves sharper results. This is because BiFuse adopts a simple architecture without any skip-connection layer inherited from FCRN [4] so that the image details encoded in the low-level feature maps cannot be well preserved. In contrast, BiFuse++ adopts a UNet-like architecture with three skip-connection layers (c.f., Figure 2.3), and the details are well recovered in the depth maps.

Compared with UniFuse [43], BiFuse++ recovered much clearer object boundaries. This is because UniFuse adopts two ResNet to encode features maps of equirectangular and cubemap projections independently so that the two encoders cannot effectively leverage the information from the other branch. In contrast, we pass the fused features (f'_{equi} and f'_{cube} in Figure 2.4) to the layers of the two encoders. In this way, the layers of encoders can directly retrieve the context from the other projection and preserve more details on the predicted depth maps eventually.

3D Comparison. To further show the difference in depth maps generated from fusion approaches, we show the corresponding point cloud visualizations in Figure 2.8. The point clouds of BiFuse and UniFuse are not capable of generating sharp corners, while BiFuse++ predicts accurate wall boundaries. Moreover, the depth of objects like carpets is noisier than the one of BiFuse++. For high variance areas like the edges between windows and walls, the results of BiFuse++ are closer to the ground truth. In contrast, the other baselines generate smooth results in these areas. Hence, our BiFuse++ is able to predict accurate point clouds and outperform the other baselines.

2.4.4 Computational Comparison

Before we apply BiFuse++ to self-training of depth estimation, we first examine the efficiency of different fusion approaches since the self-training procedure usually takes more resources and, thus, a memory-efficient framework is necessary. We estimate the efficiency with the following two aspects.

Number of Parameters. Since the number of parameters used by fusion modules ([1], [43], and BiFuse++) depends on the channels of input features, we fix the channel numbers of all approaches to 512 to fairly compare the module size of different fusion modules. As shown in Table 2.6, the fusion module of BiFuse takes the largest number of parameters, and BiFuse++ uses the smallest one. Under this setting, our new fusion module can reduce 55% of parameters than BiFuse does.

Table 2.6: The number of parameters of different fusion modules (we set the channels to 512).

	BiFuse++	UniFuse	BiFuse
Parameters	2.1 M	3.5 M	4.7 M

Table 2.7: The computational comparison of fusion approaches.

Approach	GFLOPs	Model size	Runtime Memory
BiFuse++	87.42	53.19 M	1762 Mb
UniFuse	62.58	30.26 M	2006 Mb
BiFuse	682.86	253.1 M	3346 Mb

Table 2.8: The quantitative results on PanoSUNCG [5] under the self-supervised scenario. Our BiFuse++ with Spherical Photometric Loss (SPL) [5] or Contrast-Aware Photometric Loss (CAPL) outperforms other baselines. SSIM stands for structural similarity.

Method	MAE ↓	MRE ↓	RMSE ↓	RMSE (log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
360-SelfNet (Equi)	0.2436	0.1499	0.5421	0.0959	0.8618	0.9463	0.9714
360-SelfNet	0.2344	0.1521	0.5121	0.0934	0.8479	0.9420	0.9726
UniFuse	0.2452	0.1458	0.4978	0.0920	0.8513	0.9398	0.9691
BiFuse++ w/ SSIM	0.3125	0.1852	0.5889	0.1068	0.7847	0.9313	0.9684
BiFuse++ w/ SPL	0.2083	0.1287	0.4695	0.0838	0.8838	0.9583	0.9778
BiFuse++ w/ CAPL	0.1815	0.1176	0.4321	0.0790	0.8974	0.9546	0.9773

Runtime Resources. To examine the computational resources of different fusion approaches, we adopt RTX2080Ti as the platform and use a single dummy image of resolution 512x1024x3 as the input of all tested frameworks. The results are shown in Table 2.7. Compared to BiFuse, we reduce 87% of GFLOPs and 79% of parameters. Moreover, BiFuse++ only needs half of the inference memory as BiFuse. Although our GFLOPs and

parameters are slightly larger than UniFuse, we use less inference memory since we adopt PixelShuffle [67] in the decoding process. In general, we significantly reduce the computational resources of BiFuse and make it more practical to be applied in self-training.

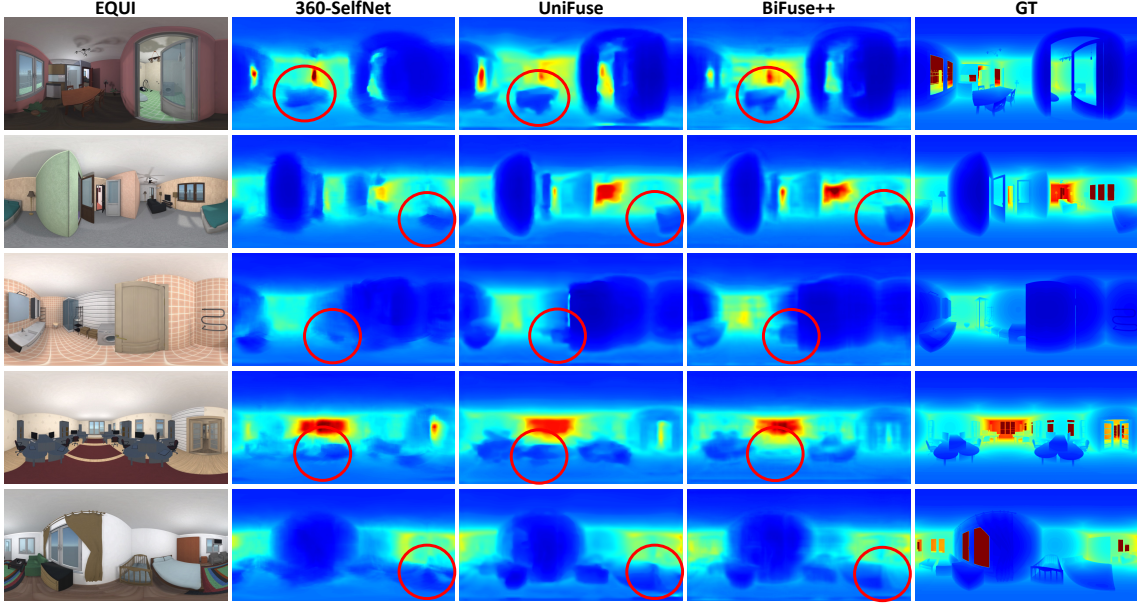


Figure 2.9: The qualitative results on PanoSUNCG under self-supervised scenario.

2.4.5 Results of Self-Supervised Scenario

We use the framework described in Section 2.3.2 to self-supervisedly train our DepthNet and PoseNet. For the qualitative and quantitative comparison, we conduct experiments on PanoSUNCG [5] to verify the applicability of BiFuse++. We compare BiFuse++ with three baselines: 1) 360-SelfNet (EQUI): the framework of [5], but the inputs are equirectangular panoramas. 2) 360-SelfNet: the framework proposed in [5]. 3) UniFuse: replace our DepthNet with the architecture of [43]. In addition, we compare variants of BiFuse++ with three different loss functions: 1) “BiFuse++ w/ SSIM”: our framework trained with structural similarity index (SSIM). 2) “BiFuse++ w/ SPL”: our framework trained with spherical photometric loss proposed by [5]. 3) “BiFuse++ w/ CAPL”: our framework trained with our proposed Contrast-Aware Photometric Loss. The quantitative

and qualitative results are shown in Table 2.8 and Figure 2.9, respectively.

Compared with our previous work 360-SelfNet [5], our framework “BiFuse++ w/ CAPL” quantitatively improves 360-SelfNet by 16% in RMSE and 23% in MAE, as shown in Table 2.8. Compared to UniFuse, we improved 13% in RMSE and 26% in MAE. Since the architecture of 360-SelfNet only adopts cubemap projection as input, the benefit of equirectangular projection is discarded, and thus 360-SelfNet introduces much noise to predicted depth maps, as shown in Figure 2.9. Although UniFuse can predict sharper depth maps than 360-SelfNet does, there are still obvious errors around object/wall boundaries. In contrast, BiFuse++ is capable of recovering more details of objects and has smaller errors around the object boundaries. Such an improvement comes from our fusion approach. During the encoding process, we pass the fused feature maps (f'_{equi} and f'_{cube} in Figure 2.4) to layers of encoders to preserve more details, and thus BiFuse++ achieves sharper depth predictions in the end.

Compared to the training results without CAPL (BiFuse++ w/ SPL), “BiFuse++ w/ CAPL” improves the RMSE and MAE by 8% and 13%, respectively (see Table 2.8). Such an improvement validates the reason why we design CAPL to prevent the network from focusing on low texture areas, i.e., to balance the difference between high-texture and low-texture areas. Comparing to training with SSIM loss (BiFuse++ w/ SSIM), the final performance is worse than training with spherical photometric loss. We have tried to apply other structure-based loss functions like weighted local contrast normalization (WLCN) proposed by [69], but the training fails to converge, and no reasonable related depth maps can be generated. Both SSIM and WLCN loss functions apply normalization to image patches, and we find such an operation eventually harms the training stability. In contrast, our Contrast-Aware Photometric Loss uses the standard deviation of image patches as the

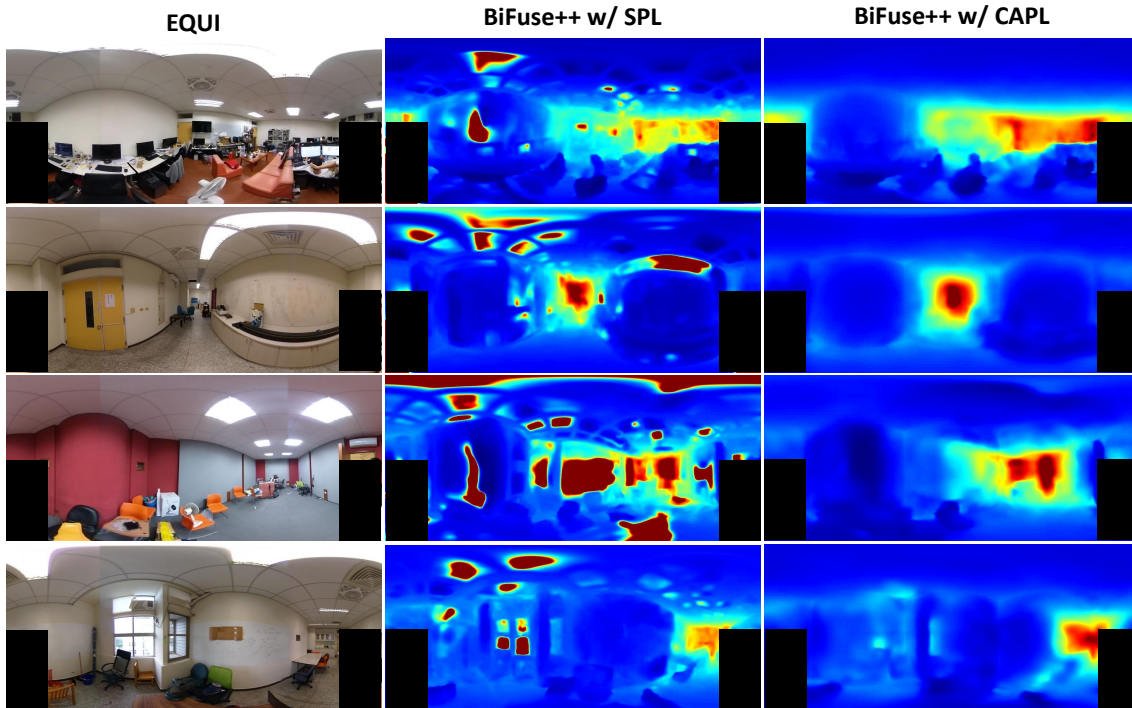


Figure 2.10: The effect of Contrast-Aware Photometric Loss (CAPL). The Spherical Photometric Loss (SPL) cannot deal with the low-texture area and thus produces unstable depth maps (red indicates a large depth value). Note that we mask out the photographer at the bottom left and right region.

weighting of photometric loss instead of directly applying normalization, and thus we can prevent numerical instability.

Training on real-world videos. To train BiFuse++ in real-world scenarios, we use the dataset we collected in 360-SelfNet [5], in which there are 25 video sequences of 5 rooms recorded with RICOH THETA V, to apply the self-training strategy adopted in Section 2.4.5 for experiments. To ensure the training stability and the small baseline between consecutive frames, we extract the raw videos with 5 frames per second. Since we do not have laser scanners like LiDARs to collect the depth ground truths, we show the training results qualitatively. As addressed in Section 2.3.2, directly applying spherical photometric loss [5] to the real-world videos results in unstable depth prediction under

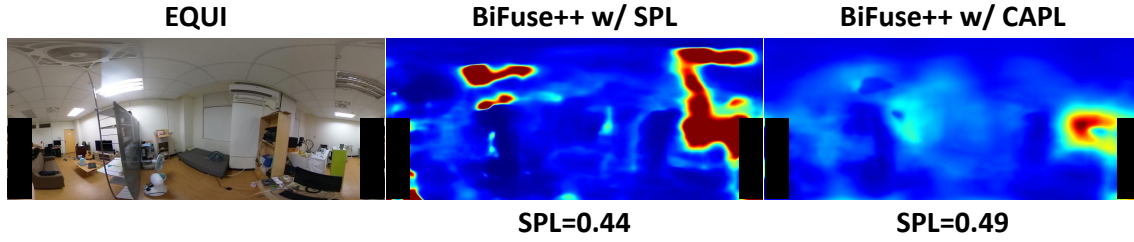


Figure 2.11: The Spherical Photometric Loss (SPL) is a degenerated loss. The BiFuse++ w/ SPL indeed reaches a lower SPL (0.44) compared to BiFuse++ w/ CAPL (0.49). However, the quality of BiFuse++ w/ SPL is worse than BiFuse++ w/ CAPL.

low-texture areas such as walls or floor, since the corresponding photometric loss is ambiguous, i.e., there is a degeneration problem in the spherical photometric loss. Thus, we propose CAPL to prevent the network from focusing on these areas overly and we compare the training results before/after applying CAPL in Figure 2.10. Without CAPL, the depth maps are noisy, especially on the wall and floor, while the results are smooth and stable after applying CAPL. Hence, our proposed BiFuse++ along with CAPL is a general self-supervised 360° depth estimation framework capable of estimating high-quality depth maps in both virtual and real-world environments.

Spherical Photometric Loss Degeneration. To investigate the reason why our CAPL can improve the results on real-world videos, we monitor the spherical photometric loss (SPL) value of “BiFuse++ w/ SPL” and “BiFuse++ w/ CAPL”. For BiFuse++ w/ SPL, the average spherical photometric loss value over the validation set is 0.45, while the average loss value becomes 0.48 after applying CAPL. This indicates that the network with a lower average spherical photometric loss cannot always produce better depth maps. As the example shown in Figure 2.11, the depth map with higher spherical photometric loss is better than the lower one. When there is no constraint on low-texture areas (w/ SPL),

we find that the networks still tend to keep minimizing the spherical photometric error on these areas even if the corresponding value is already small. Since there is small intensity noise introduced by camera sensors between videos frames, the spherical photometric loss is impossible to be zero when the perfect depth prediction and camera motion are given. The minimizing behavior of networks in low-texture areas severely harms the training stability. Thus, our CAPL uses the standard deviation of neighboring pixels to directly enforce our networks not to overly focus on these areas because the standard deviation of low-texture areas is always small.

2.5 Conclusion

We propose “BiFuse++”, the first bi-projection architecture for both self-supervised and supervised 360° depth estimation, extending our previous works 360-SelfNet [5] and BiFuse [1]. To improve the efficiency and scalability, we propose a new fusion module that adopts a residual connection and removes the mask module from the original fusion module. In addition, we follow [43] that removes a redundant decoder and adopts a pixelshffule upsampling strategy to improve efficiency. We conduct experiments on three benchmark datasets of 360° depth estimation and achieve state-of-the-art performance in self-supervised scenarios and comparable performance with state-of-the-art approaches in supervised scenarios.



Chapter 3 BiFuse++ and LED²-Net

Based on previous works [1,2], the bi-projection fusion scheme is proven to be helpful in 360° depth estimation. In BiFuse [1], we apply such a fusion to supervised training scenario. While in BiFuse++ [2], the fusion approach are used in both supervised and self-supervised training scenarios. However, applying bi-projection fusion in 360° layout estimation has not been widely studied yet. In this chapter, we utilize BiFuse++ architecture into my previous paper LED²-Net to show that 360° layout estimation can also benefit from our bi-projection fusion approaches.

3.1 Introduction

Monocular 360° layout estimation is an important technique in VR/AR applications. Given a room layout, the provided geometric cues can also help indoor autonomous systems to efficiently make decisions. An illustration of 360° layout estimation is shown in Figure 3.1. Given a single panorama which camera rotation is aligned with gravity, a deep neural network is trained to estimate the corner positions from it, and then the corner positions are projected to 3D coordinate and the 3D room layout is recovered.

To train the networks, previous works such as HorizonNet [65] directly regress the boundary positions along the horizontal axis in the panoramas. As shown in Figure 3.2,



Figure 3.1: The task for layout estimation [3]. Given a single panorama, we train a neural network that takes the panorama as input and infers the corresponding 3D structure.



Figure 3.2: The motivations of LED²-Net [3]. The corners of a room layout are shown in the left figure. When we introduce same errors to these corners, the IoU difference introduced from these corner errors can be hugely different (right figure).

such a direct regression does not consider the fact that each corner has different contributions for the final layout, and thus leads to a sub-optimal training performance. To solve this issue, my previous paper LED²-Net takes the geometric relation of corners into account. Specifically, LED²-Net propose a layout-to-depth transformation which is able to convert the predicted layout boundary points into a single horizon depth (Figure 3.3) map in a differentiable scheme. Then, the loss is calculated according to the ground truth depth maps from layout annotations. Based on the proposed differentiable depth rendering process in LED²-Net, the importance of each corner can be well considered in the final loss function.

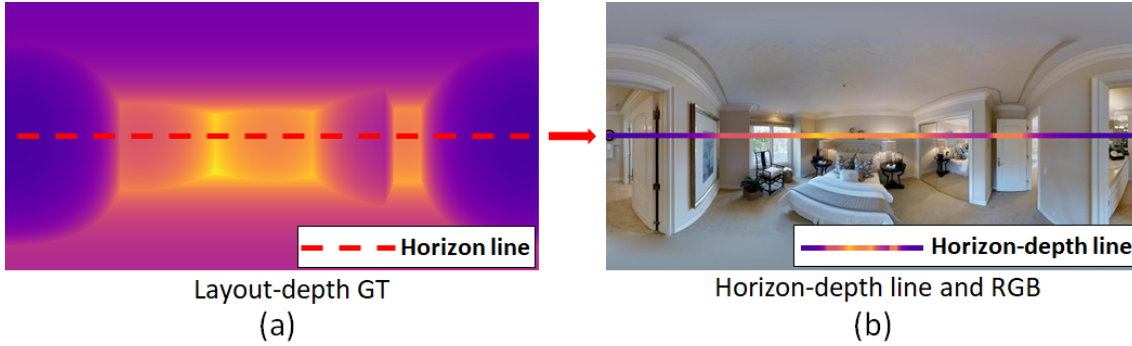


Figure 3.3: The horizon depth of LED²-Net [3]. LED²-Net proposes a layout-to-depth module to convert predicted and ground truth layout to their corresponding horizon depth maps.

To explore the applicability of bi-projection fusion in 360° layout estimation, we adopt the bi-projection architecture of BiFuse++ [2] and use the loss function of LED²-Net [3] to train the network. We conduct experiments on Realtor360 [7] and Matterport3D [8] datasets. We found that our new framework can perform better than LED²-Net, while reducing about 20% of network parameters (from 82M to 65M).

3.2 Experiments

We verify this architecture on both Realtor360 [7] and Matterport3D [8] datasets. We first introduce the two datasets and then show our experimental results in Section 3.2.1

Realtor360. This dataset is annotated by Yang *et al.* [7], which consists of 593 panoramas from Sun360 [70] dataset and 1980 panoramas collected from a private database. For our experiments, we follow [3] to use the official train and val splits.

Matterport3D. Zou *et al.* [71] and Wang *et al.* [72] manually annotate the ground truth layout from the original Matterport3D [8] dataset. In total, the dataset provides 2295 panoramas along with their layout annotations. We follow the official train/val/test splits

Table 3.1: The quantitative results of Realtor360 [7].

Method	Overall		4 corners		6 corners		8 corners		10+ corners	
	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)
LayoutNet [29]	65.84	62.77	80.41	76.60	60.50	57.87	41.16	41.16	22.35	22.35
DuLa-Net [7]	80.53	77.20	82.63	78.91	80.72	77.79	78.12	74.86	63.10	59.72
HorizonNet [65]	86.69	83.66	87.83	84.73	87.63	84.78	81.27	78.44	78.49	73.64
AtlantaNet [73]	80.36	74.59	83.42	77.05	80.67	75.01	73.72	69.31	59.43	55.51
LED ² -Net [3]	88.19	85.21	89.25	86.33	88.8	85.97	83.7	80.81	81.67	76.2
Ours	88.81	85.93	90.00	86.96	89.43	86.84	84.01	81.24	81.78	77.11

Table 3.2: The quantitative results of Matterport3D [8].

Method	Overall		4 corners		6 corners		8 corners		10+ corners	
	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)	2D IoU (%)	3D IoU (%)
LayoutNet [29]	78.73	75.82	84.61	81.35	75.02	72.33	69.79	67.45	65.14	63.00
DuLa-Net [7]	78.82	75.05	81.12	77.02	82.69	78.79	74.00	71.03	66.12	63.27
HorizonNet [65]	81.24	78.73	83.54	80.81	82.91	80.61	76.26	74.10	72.47	70.30
AtlantaNet [73]	82.09	80.02	84.42	82.09	83.85	82.08	76.97	75.19	73.19	71.62
LED ² -Net [3]	83.91	81.52	86.91	84.22	85.53	83.22	78.72	76.89	71.79	70.09
Ours	83.43	81.12	86.35	83.69	84.31	82.17	77.86	76.31	71.20	69.67

for conducting experiments.

3.2.1 Experimental Results

The quantitative results on Realtor360 [7] and Matterport3D [8] are shown in Table 3.1 and Table 3.2, respectively. For Matterport3D, our results are slightly worse than with our previous work LED²-Net [3]. For Realtor360, our BiFuse++ architecture can improve the 2D IoU from 88.19 to 88.81, while reducing the number of parameters from 82M to 65M. Such a improvement is primarily from our bi-projection module since the original LED²-Net and HorizonNet [65] only use a single ResNet-50 [13] as the encoders which take an equirectangular image as input without any information sharing between different projections. In comparison, our BiFuse++ adopt two ResNet-34 encoders to extract the features from different projections and use several bi-projection fusion modules to share their information, which improves the layout performance and reduce the total number of parameters.

Chapter 4 Conclusions

In this dissertation, I propose a bi-projection architecture BiFuse [1] which adopts a dual-branch framework utilizing both equirectangular and cubemap projections for monocular 360° depth estimation. BiFuse achieved the state-of-the-art depth accuracy in 2019. However, to reduce the model complexity and further improve the performance, I propose an more advanced framework BiFuse++ [2] which reduced about 55% of the model size compared to BiFuse. In addition, I verified this framework in both supervised and self-supervised training scenarios. For supervised scenario, the performance is comparable to HoHoNet [44], while the performance achieved state-of-the-art performance in self-supervised scenario. To further demonstrate the potentials of the bi-projection architecture, I extend BiFuse++ backbone to my previous paper LED²-Net which focuses on 360° layout estimation, and achieved a slightly better performance on Realtor360 [7] and a comparable result on Matterport3D [8].



References

- [1] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, “Bifuse: Monocular 360 depth estimation via bi-projection fusion,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [2] F.-E. Wang, Y.-H. Yeh, Y.-H. Tsai, W.-C. Chiu, and M. Sun, “Bifuse++: Self-supervised and efficient bi-projection fusion for 360° depth estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.
- [3] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, “Led2-net: Monocular 360deg layout estimation via differentiable depth rendering,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [4] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in *International Conference on 3D Vision (3DV)*, 2016.
- [5] F.-E. Wang, H.-N. Hu, H.-T. Cheng, J.-T. Lin, S.-T. Yang, M.-L. Shih, H.-K. Chu, and M. Sun, “Self-supervised learning of depth and camera motion from 360° videos,” in *Asian Conference on Computer Vision (ACCV)*, 2018.

- [6] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, T.-L. Liu, and M. Sun, “Cube padding for weakly-supervised saliency prediction in 360° videos,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [7] S.-T. Yang, F.-E. Wang, C.-H. Peng, P. Wonka, M. Sun, and H.-K. Chu, “Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [8] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, “Matterport3D: Learning from RGB-D data in indoor environments,” *International Conference on 3D Vision (3DV)*, 2017.
- [9] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, “OmniDepth: Dense depth estimation for indoors spherical panoramas,” in *European Conference on Computer Vision (ECCV)*, 2018.
- [10] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, “Joint 2d-3d-semantic data for indoor scene understanding,” *arXiv preprint arXiv:1702.01105*, 2017.
- [11] A. Saxena, M. Sun, and A. Y. Ng, “Make3d: Learning 3d scene structure from a single still image,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2008.
- [12] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [14] J.-H. Lee, M. Heo, K.-R. Kim, and C.-S. Kim, "Single-image depth estimation based on fourier domain analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [15] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," in *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2018.
- [16] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [17] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [18] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous crfs as sequential deep networks for monocular depth estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [19] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, "Structured attention guided convolutional neural fields for monocular depth estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [20] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

- [21] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [22] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [23] Z. Yin and J. Shi, “Geonet: Unsupervised learning of dense depth, optical flow and camera pose,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [24] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid, “Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [25] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, “Unsupervised learning of geometry from videos with edge-aware depth-normal consistency,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [26] C. Wang, J. Miguel Buenaposada, R. Zhu, and S. Lucey, “Learning depth from monocular videos using direct methods,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [27] H.-Y. Lai, Y.-H. Tsai, and W.-C. Chiu, “Bridging stereo matching and optical flow via spatiotemporal correspondence,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

- [28] N.-H. Wang, B. Solarte, Y.-H. Tsai, W.-C. Chiu, and M. Sun, “360sd-net: 360° stereo depth estimation with learnable cost volume,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [29] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, “Layoutnet: Reconstructing the 3d room layout from a single rgb image,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [30] Y. Zhang, S. Song, P. Tan, and J. Xiao, “Panocontext: A whole-room 3d context model for panoramic scene understanding,” in *European Conference on Computer Vision (ECCV)*, 2014.
- [31] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, “Spherical CNNs,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [32] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, “Learning so(3) equivariant representations with spherical cnns,” in *European Conference on Computer Vision (ECCV)*, 2018.
- [33] Y. Su and K. Grauman, “Kernel transformer networks for compact spherical convolution,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [34] Y.-C. Su and K. Grauman, “Learning spherical convolution for fast features from 360° imagery,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [35] M. Eder, P. Moulon, and L. Guan, “Pano popups: Indoor 3d reconstruction with a plane-aware network,” in *International Conference on 3D Vision (3DV)*, 2019.

- [36] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, “Demon: Depth and motion network for learning monocular stereo,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [37] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, “Segflow: Joint learning for video object segmentation and optical flow,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [38] Z. Zhang, Z. Cui, C. Xu, Z. Jie, X. Li, and J. Yang, “Joint task-recursive learning for semantic segmentation and depth estimation,” in *European Conference on Computer Vision (ECCV)*, 2018.
- [39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2014.
- [41] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, “Semantic scene completion from a single depth image,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [42] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning (ICML)*, 2015.

- [43] H. Jiang, Z. Sheng, S. Zhu, Z. Dong, and R. Huang, “Unifuse: Unidirectional fusion for 360° panorama depth estimation,” *IEEE Robotics and Automation Letters (RA-L)*, 2021.
- [44] C. Sun, M. Sun, and H.-T. Chen, “Hohonet: 360 indoor holistic understanding with latent horizontal features,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [45] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.
- [46] S. F. Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [47] J. Xie, R. Girshick, and A. Farhadi, “Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [48] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [49] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, “Sfm-net: Learning of structure and motion from video,” *arXiv preprint arXiv:1704.07804*, 2017.

- [50] A. Byravan and D. Fox, “Se3-nets: Learning rigid body motion using deep neural networks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [51] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, “Digging into self-supervised monocular depth estimation,” in *IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [52] A. Johnston and G. Carneiro, “Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [53] V. Guizilini, R. Amrus, S. Pillai, A. Raventos, and A. Gaidon, “3d packing for self-supervised monocular depth estimation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [54] J.-W. Bian, H. Zhan, N. Wang, Z. Li, L. Zhang, C. Shen, M.-M. Cheng, and I. Reid, “Unsupervised scale-consistent depth learning from video,” *International Journal of Computer Vision (IJCV)*, 2021.
- [55] V. Guizilini, R. Hou, J. Li, R. Amrus, and A. Gaidon, “Semantically-guided representation learning for self-supervised monocular depth,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [56] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang, “Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

- [57] S. Zhu, G. Brazil, and X. Liu, “The edge of depth: Explicit constraints between segmentation and depth,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [58] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, “Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance,” in *European Conference on Computer Vision (ECCV)*, 2020.
- [59] L. Hoyer, D. Dai, Y. Chen, A. Koring, S. Saha, and L. Van Gool, “Three ways to improve semantic segmentation with self-supervised depth estimation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [60] T.-H. Wang, H.-J. Huang, J.-T. Lin, C.-W. Hu, K.-H. Zeng, and M. Sun, “Omni-directional cnn for visual place recognition and navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [61] N. Zioulis, A. Karakottas, D. Zarpalas, F. Alvarez, and P. Daras, “Spherical view synthesis for self-supervised 360 depth estimation,” in *International Conference on 3D Vision (3DV)*, 2019.
- [62] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, “An intriguing failing of convolutional neural networks and the coordconv solution,” *arXiv*, 2018.
- [63] L. Jin, Y. Xu, J. Zheng, J. Zhang, R. Tang, S. Xu, J. Yu, and S. Gao, “Geometric structure based and regularized depth estimation from 360 indoor imagery,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

- [64] W. Zeng, S. Karaoglu, and T. Gevers, “Joint 3d layout and depth prediction from a single indoor panorama image,” in *European Conference on Computer Vision (ECCV)*, 2020.
- [65] C. Sun, C.-W. Hsiao, M. Sun, and H.-T. Chen, “Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [66] G. Pintore, M. Agus, E. Almansa, J. Schneider, and E. Gobbetti, “Slicenet: deep dense depth estimation from a single indoor panorama using a slice-based representation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [67] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [68] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [69] Y. Zhang, S. Khamis, C. Rhemann, J. Valentin, A. Kowdle, V. Tankovich, M. Schoenberg, S. Izadi, T. Funkhouser, and S. Fanello, “Activestereonet: End-to-end self-supervised learning for active stereo systems,” in *European Conference on Computer Vision (ECCV)*, 2018.

- [70] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, “Recognizing scene viewpoint using panoramic place representation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [71] C. Zou, J.-W. Su, C.-H. Peng, A. Colburn, Q. Shan, P. Wonka, H.-K. Chu, and D. Hoiem, “Manhattan room layout reconstruction from a single 360 image: A comparative study of state-of-the-art methods,” *International Journal of Computer Vision (IJCV)*, 2021.
- [72] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, “Layoutmp3d: Layout annotation of matterport3d,” *arXiv:2003.13516*, 2020.
- [73] G. Pintore, M. Agus, and E. Gobbetti, “Atlantnet: Inferring the 3d indoor layout from a single 360 image beyond the manhattan world assumption,” in *European Conference on Computer Vision (ECCV)*, 2020.

