

Indoor Panorama Planar 3D Reconstruction via Divide and Conquer

Cheng Sun^{1,2}

chengsun@gapp.nthu.edu.tw

Ning-Hsu Wang¹

albert100121@gapp.nthu.edu.tw

Chi-Wei Hsiao¹

chiweihsiao@gapp.nthu.edu.tw

Min Sun^{1,3}

sunmin@ee.nthu.edu.tw

Hwann-Tzong Chen^{1,4}

htchen@cs.nthu.edu.tw

Abstract

Indoor panorama typically consists of human-made structures parallel or perpendicular to gravity. We leverage this phenomenon to approximate the scene in a 360-degree image with (H)orizontal-planes and (V)ertical-planes. To this end, we propose an effective divide-and-conquer strategy that divides pixels based on their plane orientation estimation; then, the succeeding instance segmentation module conquers the task of planes clustering more easily in each plane orientation group. Besides, parameters of V-planes depend on camera yaw rotation, but translation-invariant CNNs are less aware of the yaw change. We thus propose a yaw-invariant V-planar reparameterization for CNNs to learn. We create a benchmark for indoor panorama planar reconstruction by extending existing 360 depth datasets with ground truth H&V-planes (referred to as “PanoH&V” dataset) and adopt state-of-the-art planar reconstruction methods to predict H&V-planes as our baselines. Our method outperforms the baselines by a large margin on the proposed dataset. Code is available at <https://github.com/sunset1995/PanoPlane360>.

1. Introduction

Reconstructing planar surfaces from single-view images have many applications such as interior modeling, AR/VR, robot navigation, and scene understanding. Generally, an indoor scene consisting of human-made structures can be approximated by a small set of dominant planes, making planar reconstruction suitable for 3D indoor modeling.

Recent works on planar reconstruction [17, 18, 20] employ state-of-the-art instance segmentation methods and achieve promising results. However, these works are mostly trained on the planar datasets derived from ScanNet [6] or NYUv2 [23] with a small field-of-view (FoV). Data of this kind require multiple images to reconstruct entire

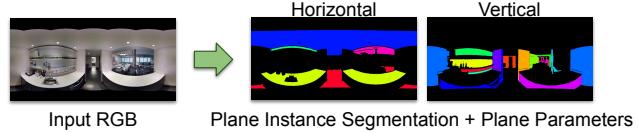


Figure 1: Planar surfaces of human-made structures are mostly horizontal or vertical with respect to the gravity direction. Given an RGB panorama, we propose to model a 3D scene as horizontal or vertical planes (the *H&V-planes*).

scenes, which would cost more computational time and resources. As 360° devices get popularized, the amount of 360° data has significantly increased, with many panorama datasets [4, 2, 26] being released to facilitate learning-based methods. By taking input in 360° format, 3D reconstruction of an entire scene can be done with only one snapshot. Considering the benefit of real-world 360° data in planar reconstruction and the gap of existing literature in this research field, we believe the planar reconstruction task from panorama imagery is worthy of investigation.

In this work, we construct the first real-world indoor 360° H&V-plane dataset (PanoH&V dataset), where we simplify the planar reconstruction task by focusing on horizontal and vertical planes (illustrated in Fig. 1). To this end, we extend existing large-scale 360° datasets by fitting the provided depth modality with horizontal and vertical planes. The H&V-planes are similar to the concept of the Manhattan world [5] and the Atlanta world [22], but we only constrain the extracted planes to be vertical or horizontal without assuming other inter-plane relationship. With the extended modality from large-scale H&V-planes, we train two current state-of-the-art planar reconstruction methods, PlaneRCNN [17] and PlaneAE [32], and report their performance to serve as the strong baselines in our benchmark.

We find existing planar reconstruction methods suboptimal when applied to the presented PanoH&V dataset. First, existing methods employ instance segmentation to detect planes from visual cue with less consideration about the estimated geometry. In practice, some planes are easier to differentiate through geometry instead of visual appearance

¹National Tsing Hua University

²ASUS AICS Department

³Joint Research Center for AI Technology and All Vista Healthcare

⁴Aeolus Robotics

(e.g., walls with similar appearance). Second, plane parameters depend on camera poses, but 360° camera yaw rotations are left-right circular shifts on equirectangular images and hard for the translation-invariant CNNs to observe. In contrast to the existing works, our method addresses the above issues appropriately: *i*) We use a divide-and-conquer strategy. The proposed *surface orientation grouping* distinguishes pixels of different plane orientations, so the succeeding instance segmentation module applied to each group can focus on a simpler subproblem. *ii*) We then propose a residual form *yaw-invariant parameterization* for V-planar geometry such that it is independent of the camera yaw rotation. We show that the yaw-invariant parameterization brings significant improvements in V-plane orientations estimation, which also benefits other methods.

We summarize the contribution of this work in two aspects. In terms of **technical contribution**, the proposed method consists of *i*) a divide-and-conquer strategy for the task of plane instance segmentation, which exploits the estimated plane orientations to divide the task into multiple simpler subproblems; *ii*) a yaw-invariant vertical plane parameterization addressing the 360° yaw ambiguity, which can also boost other existing methods. In terms of **system contribution**, we construct a new real-world 360° piece-wise planar benchmark, which focuses on evaluating horizontal and vertical planes. Finally, our approach outperforms the two strong baselines adapted from existing state-of-the-art planar reconstruction methods on the new benchmark.

2. Related work

Reconstructing 3D planes from an image involves two subtasks: segmenting plane instances and estimating plane geometric parameters. To solve the problems, PlaneNet [18] trains CNN and DCNN that reconstruct a fixed number of planes by estimating plane parameters and plane segmentation masks both in an instance-wise manner. PlaneRecover [29] also predicts a fixed number of planes but learns directly from depth modality with plane structure-induced loss. Recent state-of-the-art approaches relax the constraint on the number of planes by exploiting popular frameworks in instance segmentation. PlaneRCNN [17] modifies the two-stage architecture Mask R-CNN [11] with object category classification replaced by plane geometry prediction, followed by a network to refine the segmentation masks. PlaneAE [32] predicts per-pixel plane parameters and adopts associative embedding [3, 9, 16, 20], which trains a network to map each pixel to embedding space and then clusters the embedded pixels to generate instances. DualRPN [13] groups planes into object and layout categories, each with its network branch, and infers plane representations for both the visible and occlusion parts. A plane post-refinement method [21] is recently proposed, which improves the results of the existing methods (e.g., PlaneRCNN and PlaneAE) by

refining the inter-planar relationship.

Previous works rarely use the plane geometry prior when segmenting plane instances. Plane parameter estimation may correlate with instance segmentation either via loss [18, 29, 32] or by an additional module for refinement [17, 21]. In contrast, our method directly groups pixels based on plane orientations so that the plane segmentation module can detect unique planes in each group separately.

Estimating per-pixel surface normals for vertical planes from an equirectangular image is challenging for CNNs. Specifically, vertical-plane parameters depend on 360° camera's yaw rotation, but the counterpart left-right circular shifting on the equirectangular image is less discerned by the translation-invariant CNNs. Although the surface normal is a fundamental property to many 360° applications aside from the planar reconstruction, existing methods [30, 24, 28] which estimate surface normal from an equirectangular image are less aware of the 360° camera yaw ambiguous problem. A workaround by [8] is to use CoordConv [19] to make the model condition on image u -coordinates so that the yaw ambiguous problem in 360° is alleviated. However, this relies on the deep net capability to learn the relationship between the u -coordinates and the plane orientations. On the contrary, we propose a yaw-invariant parameterization for vertical planes, which solves the yaw-rotation ambiguous problem adequately.

3. PanoH&V dataset

In this section, we first introduce the large-scale panoramic public datasets used to construct our dataset (Sec. 3.1). We then show the statistical analysis on these datasets to support the validity of scene approximation with H&V-planes (Sec. 3.2). Finally, we outline the automatic H&V-plane annotation algorithm from depth modality (Sec. 3.3).

3.1. Panorama dataset sources

We construct our dataset from three public 360° RGB-D datasets, including Matterport3D [4] and Stanford2D3D [2] in real-world scenes, and Structure3D [34] in synthetic environments. These three datasets consist of large-scale aggregation of panoramic RGB images and depth maps. All panorama images in this work are represented in the equirectangular format with resolution of 512×1024 . Similar to [17, 18] deriving plane modality to train the learning-based method, our annotations of H&V-planar masks and parameters are derived from the ground-truth 360° depths.

We assume all panorama images are aligned with the gravity direction. In case that g-sensor and tripod are not equipped with the 360° camera, and the image is not aligned, we can use the voting-based algorithm mentioned in [10, 31, 33, 35] for vanishing point (VP) detection and panoramic image alignment. Panoramas generally provide

enough context to extract the gravity direction, and we will not lose any pixel or introduce any padding after image alignment.

3.2. H&V-plane scene approximation

We aim to approximate the indoor scene by horizontal planes (*H-planes*) and vertical planes (*V-planes*) whose surface normals are parallel and perpendicular to the gravity direction respectively. H&V-plane is a restrictive version of the general piecewise plane and can fit the Manhattan world [5] (MW) and the Atlanta world [22] (AW), but, unlike MW and AW, the H&V-planes do not assume any relationship between V-planes. Although the use of H&V-planes for scene approximation complies with our intuition, we further perform two quantitative analyses on three large-scale indoor panorama datasets, which include various indoor scenes (*e.g.*, classroom, office, living room, kitchen) to justify the validity and applicability of H&V-plane assumption.

- 1) For each pair of vertically adjacent pixels in aligned 360° images, we calculate the angle between the floor plane and the line passing through the two projected 3D points. The histogram of angles computed from all images of the three indoor panoramic datasets is shown in Fig. 2. Imagine that we take a vertical slice of the 3D point cloud (corresponding to an image column) and move from the bottom point to its upper adjacent point and repeat the process until we reach the top point—Most of the moving directions will be either horizontal (0° to 10°) or vertical (80° to 90°). Such distribution suggests the gravity aligned nature of indoor scene structures.
- 2) We approximate the scenes projected from the depth modality with our H-planes and V-planes extraction algorithms (described in Sec. 3.3). The statistics of the derived H&V-plane annotation on the three indoor panoramic datasets are shown in Fig. 3. We can see that, in general, more than 80% of the pixels in an image can be covered (less than 5cm discrepancy) by roughly 20 H&V-planes. This result suggests that our H&V-plane approximation are suitable for modeling the gist of an indoor scene.

3.3. Ground-truth H&V-plane annotation

Following the gravity aligned and H&V-plane assumptions, we present the first 360° planar dataset with training, validation, and test sets. The annotations include H&V-plane masks and plane parameters, and all images and annotations are in the same resolution of 512×1024 . We use the same analysis as in Fig. 2 to classify each pixel into H-pixel, V-pixel, or other. RANSAC is then performed on H-pixels and V-pixels to extract instance mask and plane geometry for H-planes and V-planes. The statistical information of

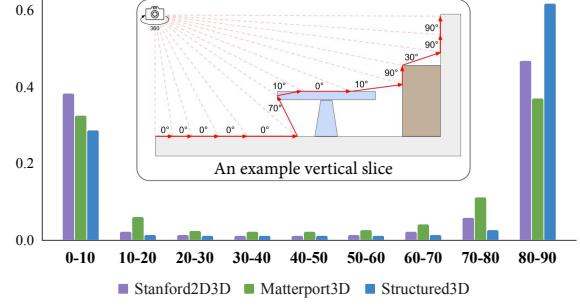


Figure 2: The histogram of angles between the floor plane and the line connecting two vertically adjacent pixels (both transformed into 3D points with ground truth depth). The center subfigure illustrates the angles of points in a vertical slice. The analysis shows that gravity-aligned structures dominate the indoor datasets, supporting us to approximate a scene with horizontal and vertical planes (H&V-planes).

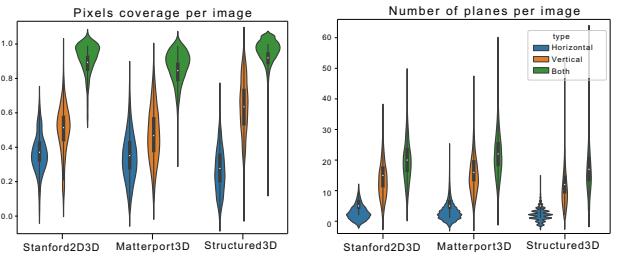


Figure 3: The violin plot illustrates the statistics of the constructed H&V-plane benchmark. We show all the combinations between the three panoramic datasets (*x*-axis) and the type of planes (blue bars for H-planes, orange bars for V-planes, and green bars for both). The left canvas shows the per-image pixel coverage of H&V-planes. The *y*-axis is the ratio of covered pixels. The right canvas shows the number of H&V-planes per image.

the constructed dataset is depicted in Fig. 3. Please refer to supplementary material for the full description of the H&V-plane extraction algorithm.

4. Approach

Our task is to reconstruct H&V-planes from a single 360° RGB image. An overview of our approach is depicted in Fig. 4. Our first step is to partition an image into H-planar, V-planar, and non-planar regions (Sec. 4.1). Our model predicts planar depth and V-plane orientation both in pixel-level, which will be used to derive the plane parameters (Sec. 4.2). In Sec. 4.3, a divide-and-conquer strategy for plane instance segmentation is proposed to make use of the learned pixels' H&V-planar and geometric information. Finally, implementation details are provided in Sec. 4.4.

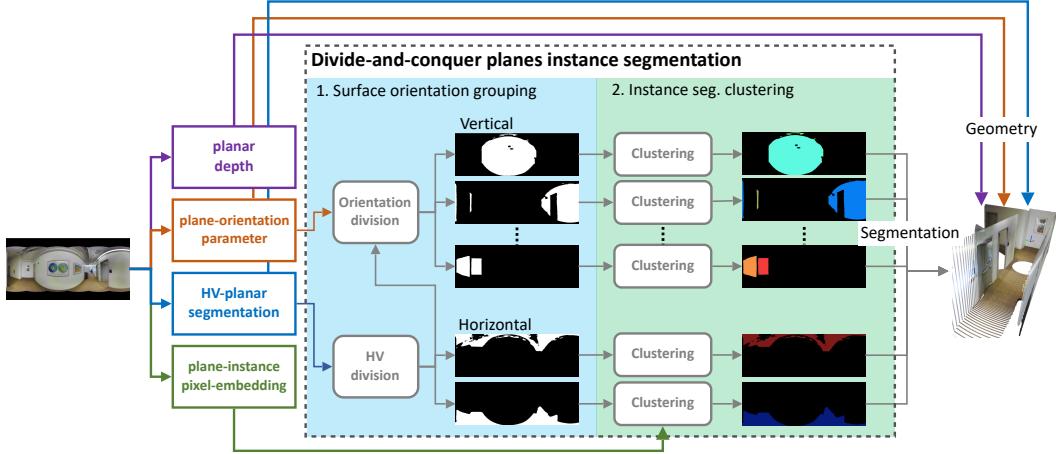


Figure 4: An overview of the proposed method. We employ an encoder-decoder network to extract pixel-level latent features from the input RGB panorama, followed by different Conv1x1 layers to produce each modality. We detail the HV-planar segmentation and geometric characteristics estimation in Sec. 4.1 and Sec. 4.2 respectively. In Sec. 4.3, a divide-and-conquer strategy is proposed exploiting all the pixel-level predictions for plane instance segmentation. We first *divide* pixels according to the estimated surface orientation. In the *conquer* stage, we employ associative embedding [32] to distinguish individual plane instances in an orientation group, which only involves pixels of a similar plane orientation and thus easier to solve.

4.1. H&V-planar segmentation

We train two binary classifiers to be activated on H-planar pixels and V-planar pixels respectively. A pixel is recognized as “non-planar” when the corresponding probabilities from both classifiers are below a certain threshold (we simply set to 0.5); otherwise, it will be classified as H-pixels or V-pixels according to which classifier gives a higher probability. An alternative is to use 3-way softmax classification, but we do not observe performance differences and thus stick to two binary classifiers with fewer output channels. We use binary cross entropy loss (BCE) as H&V-planar training objective:

$$L_{\text{HVseg}} = \frac{1}{N} \sum_i (\text{BCE}(m_i^h, m_i^{h*}) + \text{BCE}(m_i^v, m_i^{v*})), \quad (1)$$

where the subscript i is the pixel index, N is the total number of pixels, $\{m^h, m^v\}$ are the predicted H&V-planar probability maps, and $\{m^{h*}, m^{v*}\}$ are the ground truth masks. Based on the segmented H&V-pixels, we exploit their prior to give different treatments in the succeeding modules.

4.2. Planar geometry estimation

Our model estimates two geometric characteristics at pixel-level—H&V-planar depth d and V-planar surface normal θ . The H&V-planar depth d is used to recover both H-planes and V-planes, while the V-plane orientation θ only relates to V-planes. In below, the plane parameter is denoted as $\vec{n} = [x \ y \ z]$, which is the unit surface normal scaled by the plane offset; image coordinate is denoted as $u \in [-\pi, \pi]$ and $v \in [-\frac{\pi}{2}, \frac{\pi}{2}]$; we use subscript i to denote the index of a pixel. In testing phase, the plane parameter of a detected

plane is simply determined by the dimension-wise median of \vec{n}_i in the segmented instance mask.

4.2.1 H-planar geometry

The unit surface normal of an H-plane is either $[0 \ 0 \ 1]$ or $[0 \ 0 \ -1]$ (corresponding to horizontal planes above or below the camera respectively) and can be determined accordingly as the pixel is located at the upper half or the bottom half of an equirectangular image. Thus, the only one degree-of-freedom of an H-plane is the plane offset. Given the estimated planar depth d , we derive the H-plane offset at the pixel-level by

$$z_i = d_i \cdot \sin v_i. \quad (2)$$

Thus, $\vec{n}_i = z_i \cdot [0 \ 0 \ 1]$ is the plane parameter of the H-planar pixel, where the sign is determined by $\sin(v_i)$. The training objective for H-planar geometry is

$$L_{\text{Hgeo}} = \frac{1}{|I_H|} \sum_{i \in I_H} |z_i - z_i^*|, \quad (3)$$

where I_H is the set of all H-pixels indices and z_i^* is the ground truth H-plane offset of the pixel.

4.2.2 V-planar geometry

For a pixel belonging to a V-plane, we derive the V-plane surface normal from the estimated θ_i as $[\cos \theta_i \ \sin \theta_i \ 0]$. The V-plane offset depends on the V-plane surface normal and the estimated planar depth d_i , which is

$$o_i = d_i \cdot \cos v_i \cdot [\cos \theta_i \ \sin \theta_i] \cdot [\cos u_i \ \sin u_i]^T. \quad (4)$$

Thus, $\vec{n}_i = o_i \cdot [\cos \theta_i \sin \theta_i 0]$ is the plane parameter of the V-planar pixel. Note that only θ and d are model predictions, while o and \vec{n} are derived from θ and d . The V-planar loss is

$$L_{V\text{geo}} = \frac{1}{|I_V|} \sum_{i \in I_V} \|\vec{n}_i - \vec{n}_i^*\|_1 + \text{CosineLoss}(\theta_i, \theta_i^*), \quad (5)$$

where \vec{n}_i^* and θ_i^* are ground-truth V-planar geometries and

$$\text{CosineLoss}(\theta_i, \theta_i^*) = 1 - (\cos \theta_i \cos \theta_i^* + \sin \theta_i \sin \theta_i^*). \quad (6)$$

Yaw-invariant parameterization for V-plane orientations. The accuracy of the estimated V-plane orientation θ is critical in our method. It affects the planar geometry quality and also involves in the proposed divide-and-conquer process of plane instance segmentation (Sec. 4.3). However, V-plane orientation estimation in a 360° image is a challenging task for CNNs. The reason is that the V-planar surface orientation co-varies with the 360° camera yaw rotation, but the translation-invariant CNNs are less aware of the corresponding left-right circular shifting on the equirectangular image. Employing CoordConv [19] to condition the CNNs on image coordinate is a workaround for this issue.

To address the yaw ambiguity more effectively, we propose re-parameterizing θ_i into residual form with respect to the pixel yaw viewing angle u_i such that it is invariant to the 360° camera yaw rotation. More specifically, instead of appending u_i to input like CoordConv [19], we subtract it from the target orientation:

$$\theta'_i = \theta_i^* - u_i, \quad (7)$$

which is the re-parameterized yaw-invariant V-plane orientation. The proposed representation enables the model to infer the V-plane orientation without the knowledge about the 360° camera yaw rotation.

4.3. Divide-and-conquer for plane instance segmentation

Motivated by the strong prior in indoor scenes where most plane instances share a small number of distinct orientations (*e.g.*, Manhattan world [5] and Atlanta world [22]), we propose to integrate plane orientation information into the main process of plane instance segmentation. We use a divide-and-conquer strategy. In the *divide* stage, we divide pixels by surface orientation grouping, which forms multiple simpler subproblems. In the *conquer* stage, we apply pixel embedding clustering [32] to identify unique plane instances in each orientation group.

We find a similar idea in the recent DualRPN [13], where the planes are separated into two predefined groups—object and layout—each processed by its branch. In contrast to DualRPN, we do not need semantic annotation for our *divide*

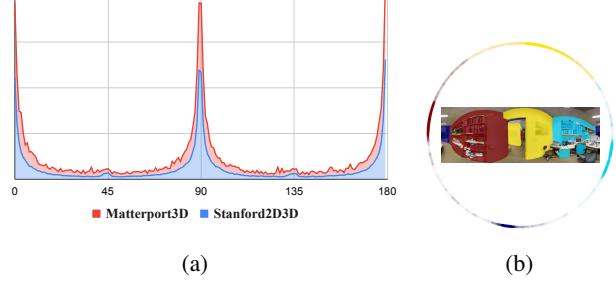


Figure 5: (a) Statistics of angles between all pairs of planes in an image. The results are averaged across the entire Stanford2D3d and subsampled Matterport3D. (b) An example of surface orientation grouping result. Different colors denote different groups. The outer circle represents the distribution of the estimated V-plane orientations.

stage, and our orientation groups are automatically determined, where there are more than six groups in most of the cases (*e.g.*, the Manhattan world captured by panorama).

4.3.1 Divide: surface orientation grouping

The analysis on our dataset (see Fig. 5) shows that, in an indoor panorama, most plane instances share similar, opposite, or perpendicular plane orientations with other planes. Hence, the per-pixel V-planar surface orientations are distributed primarily around a small number of angles. Utilizing such regularity, we divide pixels of an image into groups of similar plane normals to early separate plane instances of dissimilar orientations. We first use the predicted H&V-planar mask (Sec. 4.1) for dividing H&V-pixels (and ignoring “non-planar” pixels). For H-pixels, the two surface orientation groups (*i.e.* $\vec{n} = [0 \ 0 \ \pm 1]$) can be easily determined by pixels’ v -coordinates (Sec. 4.2.1). For V-pixels, we use a voting process to detect the prominent V-plane orientation peaks. Specifically, we quantize the estimated θ of V-pixels into circular bins followed by a peak-finding algorithm (see supplementary material for more detail). V-pixels are assigned to their nearest peak to form surface orientation groups (see Fig. 5b for an example in practice).

4.3.2 Conquer: pixel embedding clustering

To further cluster pixels in each surface orientation group into plane instance masks, we employ pixel embedding, which is originally dealing with instance segmentation [3, 9, 16] and also found to be effective for plane instance segmentation [32]. We follow the setting in [32] for the pixel embedding—embedding dimension set to 2; trained with losses L_{pull} to pull the pixels of a plane instance toward their centroid and L_{push} to push the centroids of different planes away from each other. In this work, we use different pixel embedding spaces for H-planes and V-planes, so

$L_{\text{pull}} = L_{\text{Hpull}} + L_{\text{Vpull}}$ and $L_{\text{push}} = L_{\text{Hpush}} + L_{\text{Vpush}}$. In the testing phase, the mean shift clustering is applied to each orientation group separately.

4.4. Implementation details

We extract pixel-level latent features by employing the standard encoder-decoder architecture. In below, we use ConvBlock to denote a sequence of Conv3x3, BN, and ReLU. We use ResNet-101 [12] as our backbone. The channels of each ResNet’s output features are first reduced to 128 by two ConvBlocks. In the decoder, the features from a lower resolution are upsampled and concatenated with higher resolution features, followed by a ConvBlock to fuse the two sources of features. In the upsampling process, the channels remain 128 except the finest scale (spatial stride 2), whose channels are set to 64. Finally, each modality is produced by a Conv1x1 layer from the finest scale features. All the predictions are upsampled by 2 to the input resolution. See supplementary material for the architecture diagram. The overall training objective is

$$L = 0.1 L_{\text{HVseg}} + L_{\text{Hgeo}} + L_{\text{Vgeo}} + L_{\text{pull}} + L_{\text{push}} . \quad (8)$$

We give the H&V-planar segmentation loss L_{HVseg} a smaller weight as we find it converges quickly after just a few epochs.

5. Experiments

5.1. Baselines construction

We manage to adapt two current state-of-the-art planar reconstruction approaches—PlaneAE [32]¹ and PlaneRCNN [17]²—with their official implementation to our newly constructed benchmark. For simplicity and consistency to ours, the input panorama format is equirectangular with resolution 512×1024 , and we make necessary changes to the baseline methods accordingly. Using cubemap as the input format is an alternative, but more detailed designs should be considered for the separated faces (*e.g.*, feature delivering strategy, crossing-multi-faces bounding boxes merging or masks merging strategy), which is out of scope for this work. Observing the success of processing equirectangular with planar CNNs in estimating depth [14] and layout [35, 25], we believe fine-tuning the ImageNet [7] pre-trained planar CNNs on equirectangular is suitable for our application.

Common adaptation. We employ the left-right circular padding [27] for all convolution layers. The u -coordinate is concatenated as one of the input channels to alleviate the ambiguity of yaw rotation. All methods use ResNet-101 [12] as the backbone. The unit surface normals have two degree-of-freedom θ_u, θ_v , where θ_v in our case is produced by a binary classifier as they are either 0 or ± 1 (the sign can be determined by pixel vertical location).

¹<https://github.com/svip-lab/PlanarReconstruction>

²<https://github.com/NVlabs/planercnn>

PlaneRCNN [17]. We modify RoIAlign and NMS to handle the left-right circular coordinate system. We have normal clusters for vertical and horizontal planes.

PlaneAE [32]. The Efficient Mean Shift is scaled according to the image resolution—number of iterations, anchors and sampled points are set to 18, 36, and 11k respectively.

5.2. Data split

We follow the official to split the scenes into training, validation, and test sets, but remove data that have too many missing depth values while extracting the ground-truth H&V-planes (Sec. 3). Finally, Stanford2D3D [2] contains 1,040 images for training and 372 images for validation; Matterport3D [4] contains 7,275, 1,189, and 1,005 for training/validation/test; Structured3D [34] contains 18,332 for training, 1,771 for validation, and 1,691 for test.

5.3. Training protocol

We use Adam optimizer [15] with learning rate 1e-4 and 2 panoramas in a mini-batch. The number of training epochs for Stanford2D3D, Matterport3D, and Structured3D are 100, 20, and 10 respectively. All methods, including ours, use the same training protocol.

5.4. Evaluation metrics

Following previous works [17, 29, 32], we evaluate the performance of plane instance segmentation with some standard clustering metrics [1]: Adjusted Rand Index (ARI↑), Variation of Information (VI↓), and Segmentation Covering (SC↑) which only consider the unique plane instance segmentation results on the 2D image. To evaluate 3D reconstruction quality, plane and pixel recalls are used under different planar depth discrepancy thresholds and mask IoU over 0.5. We report the results by averaging the plane recall and pixel recall under depth threshold of 5cm, 10cm, 20cm, 30cm, and 60cm. Pixel-level depth accuracy on ground truth planar region are also reported.

5.5. Results

Quantitative evaluation. In Table 1, we report comparisons between our approach and two competitive baselines on three presented 360° datasets. Our method achieves state-of-the-art performance on all planar metrics. For the metrics that only relate to segmentation quality, our method shows more improvement on Matterport3D, which contains more complex scenes captured in luxury houses comparing to the performance gain on Stanford2D3D and the synthetic Structured3D dataset. For metrics considering 3D reconstruction quality, our approach outperforms both baselines by a large margin. PlaneRCNN is trained against with ground truth depth while PlaneAE and our method only learn from plane parameters. On depth-based evaluation, our method still shows competitive results comparing to PlaneRCNN.

Method	Segmentation Quality			Per-pixel Recall ↑	Per-plane Recall ↑	Depth accuracy		
	ARI↑	VI↓	SC↑			Rel.↓	\log_{10} ↓	RMSE↓
Matterport3D [4] test set								
PlaneRCNN [17]	0.574	2.022	0.632	0.473	0.336	0.125	0.052	0.329
PlaneAE [32]	0.673	1.944	0.640	0.498	0.381	0.187	0.080	0.528
Ours	0.686	1.894	0.660	0.544	0.410	0.132	0.053	0.326
Stanford2D3D [2] validation set								
PlaneRCNN [17]	0.682	1.677	0.703	0.452	0.297	0.119	0.055	0.386
PlaneAE [32]	0.765	1.536	0.733	0.520	0.341	0.140	0.071	0.671
Ours	0.768	1.514	0.742	0.627	0.430	0.093	0.041	0.327
Structured3D [34] test set								
PlaneRCNN [17]	0.726	1.393	0.743	0.654	0.522	0.071	0.029	0.137
PlaneAE [32]	0.821	1.175	0.785	0.728	0.591	0.122	0.054	0.464
Ours	0.824	1.150	0.794	0.794	0.657	0.057	0.025	0.126

Table 1: The new benchmark. Our method outperforms the two adapted strong baselines.

Method	yaw-invariant	Per-pixel recall ↑	Per-plane recall ↑
Matterport3D [4] validation set			
PlaneRCNN [17]	✓	0.471	0.313
		0.495	0.326
PlaneAE [32]	✓	0.484	0.362
		0.495	0.369
Stanford2D3D [2] validation set			
PlaneRCNN [17]	✓	0.452	0.297
		0.481	0.313
PlaneAE [32]	✓	0.520	0.341
		0.555	0.368

Table 2: Applying the yaw-invariant parameterization to the two baselines shows better planar reconstruction results.

Qualitative results. In Fig. 6, we show the reconstructed visualization by PlaneRCNN [17], PlaneAE [32], and ours. Owing to our yaw-invariant representation, our geometric quality is generally better than the others’. In addition, our reconstructed planes are aligned better with each other.

Processing time. The average processing time over 50 scenes (with GeForce RTX 2080 Ti) for PlaneRCNN, PlaneAE and ours are 137ms, 144ms and 364ms respectively. The proposed D&C takes 224ms, which can be improved by applying the mean shift to each division in parallel instead of current sequential implementation.

5.6. Ablation study

In this section, several ablation studies are shown to further exemplify the effectiveness of the proposed method.

Can the proposed 360° yaw-invariant plane parameterization benefit other baselines? One challenge of 360° plane orientation estimation is that the non-horizontal normals are related to camera yaw rotation, but the CNNs are less aware of the counterpart 2D left-right circular shifting

CoordConv [19]	yaw-invariant	V-plane orientation error (deg°) ↓
Stanford2D3D [2] validation set		
✓		7.29
	✓	5.97
✓	✓	5.18
	✓	5.21

Table 3: We train four networks with different settings to compare CoordConv [19] and the proposed yaw-invariant parameterization on V-plane orientation estimation.

on the equirectangular image. In addition to CoordConv [19] strategy, we propose a 360° camera yaw-invariant representation for V-plane normals, which ensures the ground truth normal is independent of the ambiguous camera yaw rotation (see Sec. 4.2 for detail). The proposed representation is applied to other baselines, and the results are reported in Table 2. We observe consistent improvements on all metrics by applying the proposal to PlaneRCNN [17] and PlaneAE [32].

Can the proposed plane orientation parameterization alone address the yaw ambiguity of 360° camera? To show the yaw ambiguity problem of 360° images and the proposed yaw-invariant representation can effectively address it, we train all combinations of the use of CoordConv and our yaw-invariant parameterization. Table 3 shows that the per-pixel normal error in degree over the V-planar pixels. As mentioned in Sec. 4.2, V-planar normals depend on 360° camera yaw rotation, but CNN layers cannot handle the yaw change very well. Consequently, the model with neither CoordConv nor the proposed yaw-invariant parameterization yields inferior V-planar geometry quality. By simply conditioning the model on image u -coordinates, the orientation error is decreased by a margin. Finally, we show that re-parameterizing the plane parameter to be yaw-invariant can achieve superior orientation quality. Employing Co-

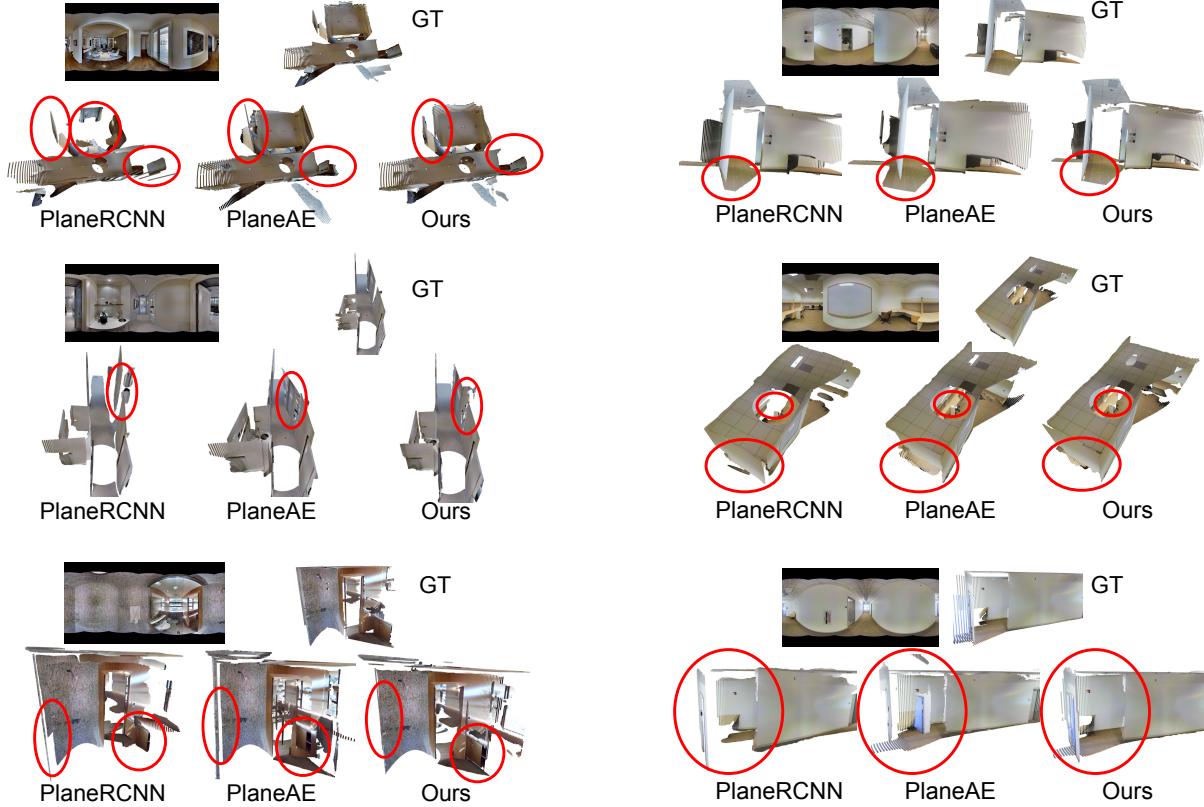


Figure 6: We show some qualitative comparisons of the 3D reconstruction results among the three methods. We highlight the differences in red circles. See supplement for more comparisons on 2D and 3D mesh visualization in bird-eye view.

ordConv upon our parameterization does not improve the result, suggesting that the proposed parameterization alone has adequately solved the yaw ambiguity.

The advantage of the *divide* stage in our divide-and-conquer strategy. The *divide* stage can early separate planes that are otherwise challenging to differentiate by the visual cues using the baseline method. Three representative examples are highlighted in Fig. 7, where we find some planes are merged by the baseline method PlaneAE [32] as it considers only pixel embedding. In contrast, these falsely merged planes can be separated early and easily by our *divide* stage using orientation estimation.

6. Conclusion

We propose a method that benefits from the divide-and-conquer strategy of approximating an indoor panoramic scene by horizontal and vertical planes. We point out a critical issue about the camera yaw ambiguity in panorama, which is neglected by previous methods. A simple and effective yaw-invariant parameterization is proposed to solve the ambiguity adequately. Last but not least, we construct the first real-world H&V-planar reconstruction benchmark by extending the existing large-scale panorama datasets with

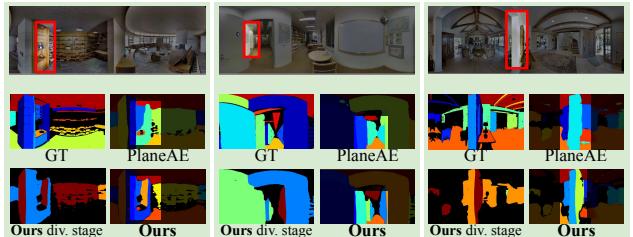


Figure 7: Three examples to demonstrate the effectiveness of our divide-and-conquer plane instance segmentation strategy. In comparison with PlaneAE [32], which fails in the highlighted regions, our method can separate the undetected plane from its neighboring planes in the *divide* stage.

H&V-plane modality, and we apply two state-of-the-art plane instance segmentation methods to serve as the strong baselines for our dataset. We show that our method achieves superior performance on this newly constructed benchmark.

Acknowledgements: This work was supported in part by the MOST, Taiwan under Grants 110-2634-F-001-009 and 110-2634-F-007-016, MOST Joint Research Center for AI Technology and All Vista Healthcare. We thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

Supplementary material

A. Ground-truth H&V-plane annotation

This section provides more details about the extraction of *HV-planes* from ground-truth depth.

A.1. Local analysis

The goal is to classify each pixel into ‘H-pixel’, ‘V-pixel’, or ‘other’ for later HV-plane extraction. The same rule is generally applied to all pixels, so we describe how we classify a pixel p for simplicity as follows. We relocate the world origin to the 3D coordinate of point p (same process applied to other points); the z-axis is aligned with gravity, and all the 3D points from the same image column are on the plane formed by y-axis and z-axis. Let $(0, y_t, z_t)$ and $(0, y_b, z_b)$ be the 3D coordinates from the top and bottom adjacent pixels respectively. Let d be the depth value of pixel p . We calculate the following information:

```

 $T_H \leftarrow \min(d \cdot 5, 40)$  {threshold for ‘H’}
 $T_V \leftarrow 90 - T_H$  {threshold for ‘V’}
 $\theta_t \leftarrow \arctan2(|z_t|, |y_t|)$  {degree in range  $[0^\circ, 90^\circ]$ }
 $\theta_b \leftarrow \arctan2(|z_b|, |y_b|)$  {degree in range  $[0^\circ, 90^\circ]$ }
 $type_t \leftarrow \text{thresholdHVO}(\theta_t, T_H, T_V)$ 
 $type_b \leftarrow \text{thresholdHVO}(\theta_b, T_H, T_V)$ 

```

T_H and T_V are depth-dependent thresholds (the farther, the more tolerant). The procedure ‘thresholdHVO’ is defined as

```

if  $\theta < T_H$  and  $\max(|z_t|, |z_b|) < 0.1$  then
    type  $\leftarrow$  ‘H’
else if  $\theta > T_V$  and  $\max(|y_t|, |y_b|) < 0.1$  then
    type  $\leftarrow$  ‘V’
else
    type  $\leftarrow$  ‘O’
end if

```

We then classify the pixel based on the states of $type_t$ and $type_b$ using the rule depicted in Table 4. The ‘TBD’ in Table 4 needs to be further determined by θ_t and θ_b :

```

if  $type_t = ‘V’$  and  $type_b = ‘H’$  then
    if  $90 - \theta_t < \theta_b$  then
        return ‘V-pixel’
    else
        return ‘H-pixel’
    end if
else if  $type_t = ‘H’$  and  $type_b = ‘V’$  then
    if  $\theta_t < 90 - \theta_b$  then
        return ‘H-pixel’
    else
        return ‘V-pixel’
    end if
end if

```

$type_t / type_b$	‘H’	‘V’	‘O’
‘H’	‘H-pixel’	TBD	‘H-pixel’
‘V’	TBD	‘V-pixel’	‘V-pixel’
‘O’	‘H-pixel’	‘V-pixel’	‘other’

Table 4: Type of a pixel according to the relationship with its top and bottom adjacent pixels

Finally, we use connected component (CC) analysis on ‘H-pixel’ and ‘V-pixel’, and re-assign small CC into ‘other’.

A.2. Horizontal planes extraction

A horizontal plane has only one degree of freedom that can be parameterized by the signed distance (for up/down in 3D) to the camera height. We identify horizontal planes in a manner similar to non-maximum suppression:

1. Using linear search to find an H-plane covering the largest number of remaining H-pixels within 5cm;
2. Removing any connected component (CC) on the image that contains fewer than 1,000 pixels (*i.e.* $\approx 0.2\%$ of the image size);
3. Masking out inlier H-pixels.

The process iterates until no H-plane can be found. The extracted H-planes are refined by reassigning all H-pixels to their nearest H-planes and recalculating the parameters of H-planes based on new inlier H-pixels.

A.3. Vertical planes extraction

A vertical plane has two degrees of freedom, which can be characterized by $\vec{n} = [x \ y \ 0] = o \cdot [\cos \theta \ \sin \theta \ 0]$ —the unit surface normal $[\cos \theta \ \sin \theta \ 0]$ scaled by the plane offset o . Similar to H-plane extraction, we apply the following process:

1. RANSAC finding a plane \vec{n} that covers the largest number of V-pixels within a threshold $T = \min(o \cdot 0.05, 0.2)$; A 3D point is said to be an inlier if the distance to the plane is less than the threshold;
2. Keeping only the largest CC as a V-plane instance since the detected plane \vec{n} by RANSAC extends infinitely in 3D and could cover multiple non-connected planes.
3. Masking out inlier V-pixels.

The process iterates until the largest CC contains fewer than 1,000 pixels. The extracted V-planes are refined by reassigning border pixels of instances to their nearest V-planes. This step could cut an instance into different CCs; only the largest CC of the instance remains, and the other smaller-sized CCs are reassigned to the majority of their neighbors. Finally, V-plane parameters are refined by solving the least-squares based on the new inliers.

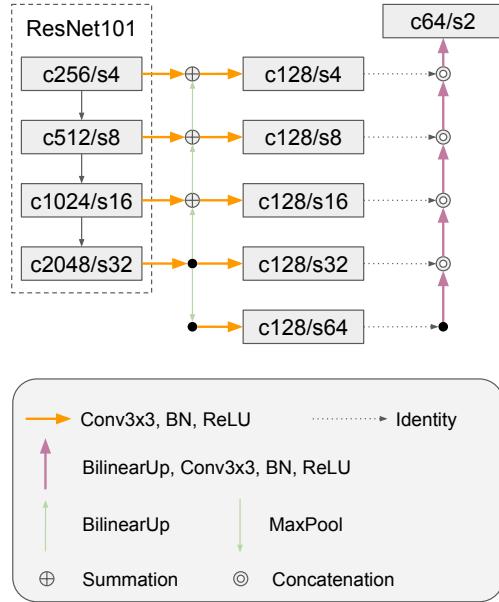


Figure 8: Detailed network architecture employed to extract per-pixel features. The “ cX/sY ” indicates that the feature tensor has X latent channels with spatial stride Y .

B. Relation with panorama room layout

The room layout reconstruct the highest-level structure of a room, which consists of only a few facades and the result is up to a scale. Besides, most of the room layout reconstruction methods currently only consider one-floor-one-ceiling model. In plane evaluation metrics, many factors can even make the detected layout facades to be counted as false positive, *e.g.*, planes of furniture or objects taking a large amount of pixels of a wall, beam, column, gates or doors to another area.

C. Detailed network architecture

The architecture for pixel-level feature extraction is depicted in Fig. 8. We employ ResNet101 as our backbone network, which provides features in four different output stride. We use ConvBlock to denote a sequence of Conv3x3, BN, ReLU. Each backbone feature tensor is reduced to 128 latent channels by two ConvBlocks, where the intermediate features are summed with the upsampled coarser-scale features to capture a longer range of context. We also add one extra level by MaxPool. Finally, from coarse to fine levels, the features are upsampled, refined by ConvBlock and concatenated with the finer level features. The overall encoder-decoder network produces a pixel-level feature tensor at output stride 2 with 64 latent channels, which is then followed by different Conv1x1 layer to estimate each modality for HV-planes reconstruction.

D. Visualization of yaw-rotation ambiguity in 360° images

We visualize the effect of camera yaw rotation on V-plane parameters in a 360° image in Fig 9, where planes are colored according to the angles of the plane orientations. We can see that values of the standard plane representation vary with different yaw rotations of 360° camera. This property, however, is inconsistent with the fact that the convolutional neural network is translation-invariant and therefore less aware of the yaw rotation.

To address the yaw ambiguity effectively, we propose to re-parameterize the ground-truth plane orientation θ_i^* of each pixel i into *residual form* with respect to the pixel yaw viewing angle u_i such that it is invariant to the 360° camera yaw rotation:

$$\theta'_i = \theta_i^* - u_i. \quad (9)$$

We show the same scene as Fig 9 but with our camera yaw invariant plane representation in Fig 10, where the ground-truth orientation is now camera yaw-invariant.

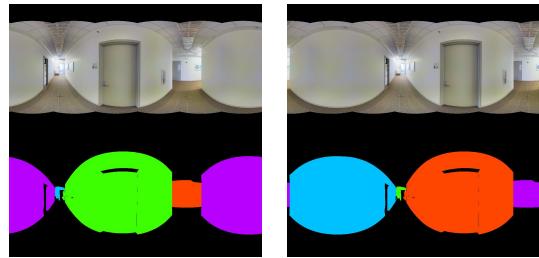


Figure 9: RGB and ground-truth V-planes colored by the angles of plane orientations. The left and right figures are two views of the exact same scene with different camera yaw rotations.

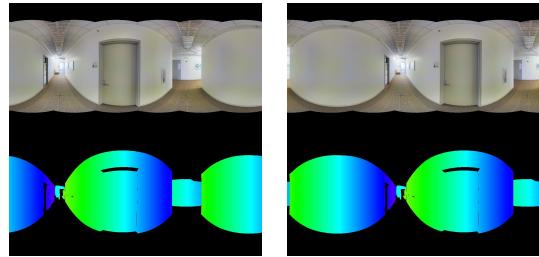


Figure 10: The proposed camera yaw-invariant plane representation.

E. V-planar orientation grouping

Our proposed plane instance segmentation module is geometry-aware. More specifically, we group all pixels with

Method	Segmentation Quality			Recall↑	
	ARI↑	VI↓	SC↑	Pixel	Plane
PlaneAE [32]	0.765	1.536	0.733	0.520	0.341
w/o yaw-inv	0.762	1.555	0.734	0.542	0.360
w/o ori-gp	0.764	1.542	0.738	0.545	0.369
w/o yaw-inv	0.762	1.554	0.732	0.619	0.417
full	0.768	1.514	0.742	0.627	0.430

Table 5: More quantitative results on Stanford2d3d of our method’s intermediate version. A strong baseline—PlaneAE—is also shown for comparison. *yaw-inv*: the yaw-invariant representation. *ori-gp*: the orientation grouping as a pre-filtering before mean shift.

similar plane orientations in the first stage of the plane instance segmentation process. We describe below how we instantiate the grouping process for pixels classified as ‘V-planar’. Let $\theta_i \in [-\pi, \pi]$ denote the reconstructed V-plane orientation in radian of a V-planar pixel with index i . We discretize all $\{\theta_i \mid \text{pixel } i \text{ is V-planar}\}$ into 360 bins. Note that the histogram is circular, where the first bin is adjacent to the last bin. We then find all prominent peaks on the histogram with two criteria: *i*) the vote is larger than any bins within the nearest 50 bins, and *ii*) it has at least 100 votes. V-pixels are assigned to their nearest peak to form surface orientation groups.

F. More quantitative comparison of intermediate version

We show more intermediate version of our method in Table 5. Our based method (w/o *yaw-inv*, w/o *ori-gp*) shows slightly better result on pixel-recall and plane-recall comparing to PlaneAE, which could be due to the difference in model architecture and the output modalities for representing plane parameter. By applying the *ori-gp*, we can see consistent improvement on segmentation quality, especially when it works with the *yaw-inv* (the quality of *ori-gp* depend on the quality of the planar yaw-angle estimation). Further, adding *yaw-inv* to our system leads to a significant improvement on pixel-recall and plane-recall which considering not only the segmentation quality but also geometry quality.

G. More qualitative results reconstructed by our method.

We show more visual results on the two real-world dataset—Stanford2D3D [2] dataset and Matterport3D [4] dataset—in Fig. 11, Fig. 12 and Fig. 14.

H. Qualitative comparisons with baselines

We compare more qualitative results of the two baselines—PlaneRCNN [17] and PlaneAE [32]—and our method in

Fig. 13. The first column shows input RGB and ground-truth plane instance segmentation. The second to the fourth columns respectively show results reconstructed by PlaneRCNN [17], PlanarReconstruct [32], and our method. For each scene, we show the planar depth error (clipped to 3 meters) in the top row and the predicted plane instance segmentation in the bottom row. We use red circles to highlight obvious errors.

Fig. 11 12 13 14 are in the next few pages.

References

- [1] Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.
- [2] Iro Armeni, Sasha Sax, Amir Roshan Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017.
- [3] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017.
- [4] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 667–676, 2017.
- [5] James M. Coughlan and Alan L. Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 845–851. MIT Press, 2000.
- [6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2432–2443, 2017.
- [7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.
- [8] Marc Eder, Pierre Moulon, and Li Guan. Pano popups: Indoor 3d reconstruction with a plane-aware network. In *2019 International Conference on 3D Vision, 3DV 2019, Québec City, QC, Canada, September 16-19, 2019*, pages 76–84, 2019.
- [9] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017.

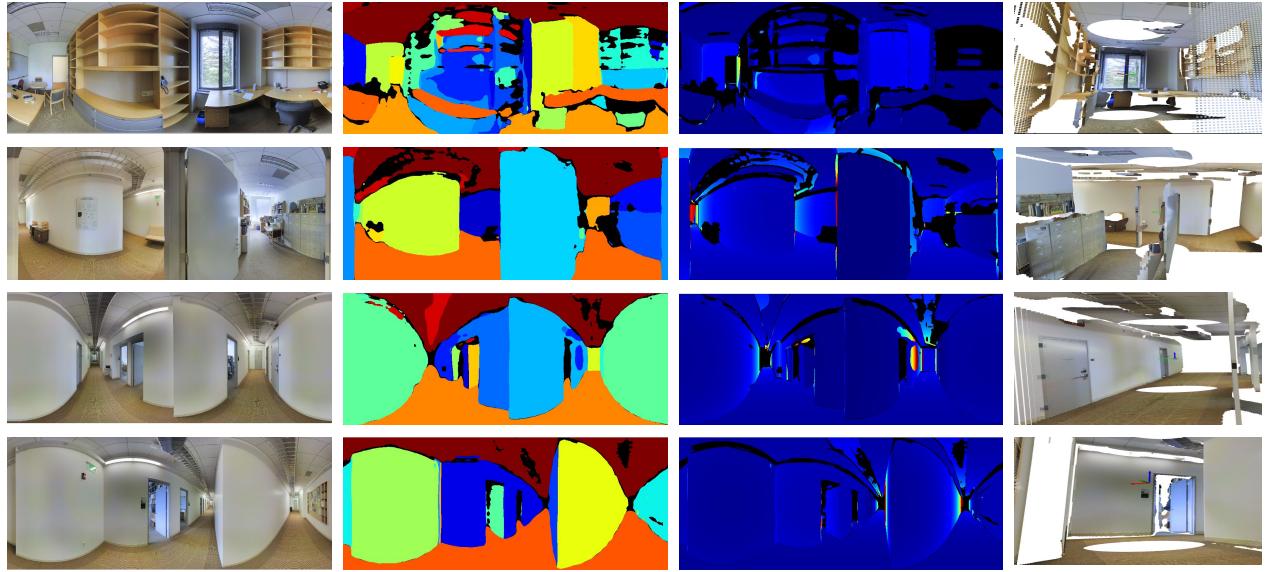


Figure 11: Visual results on Stanford2D3D [2] validation set. The first to the fourth columns show input RGB, plane instance segmentation detected by our method, planar depth error, and snapshot in the reconstructed 3D planar model.

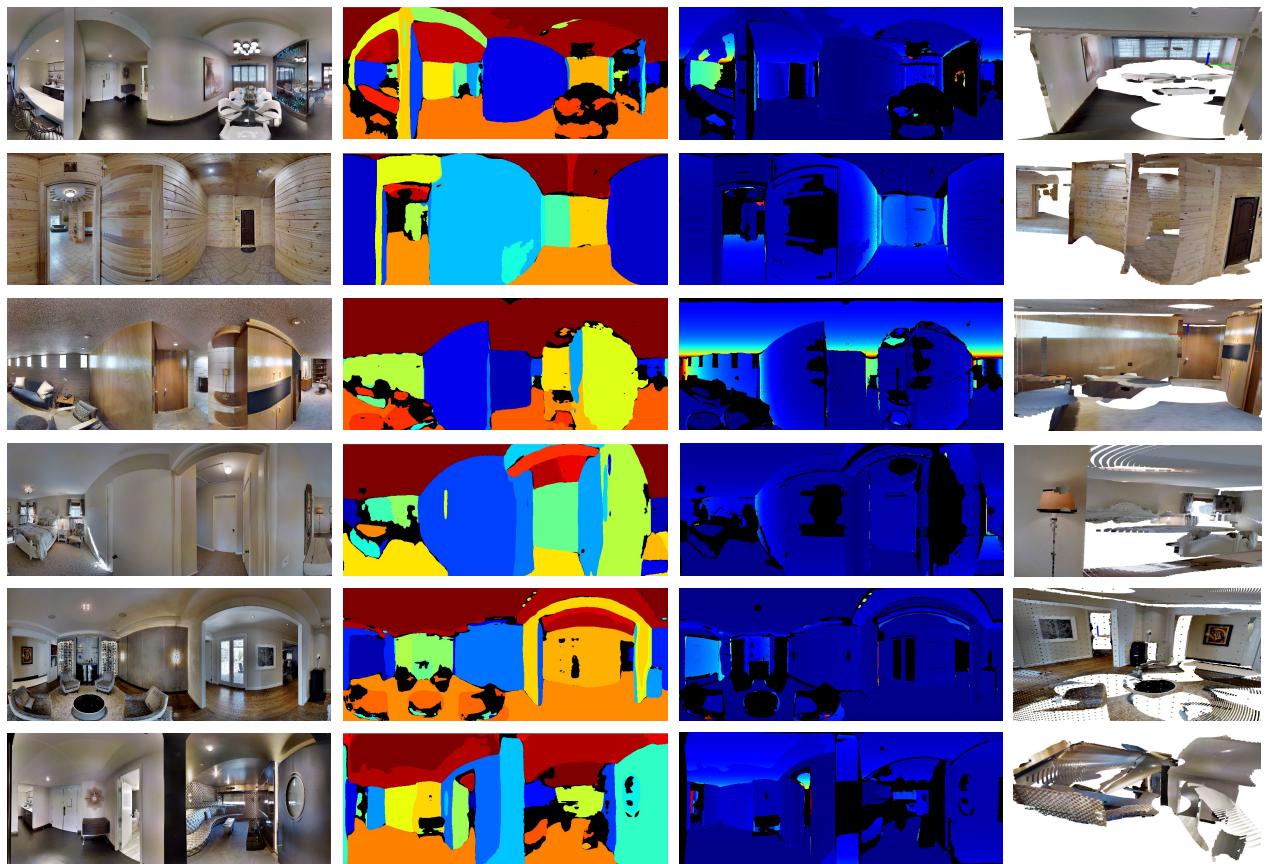


Figure 12: Visual results on Matterport3D [4] test set. The first to the fourth columns show input RGB, plane instance segmentation detected by our method, planar depth error, and snapshot in the reconstructed 3D planar model.

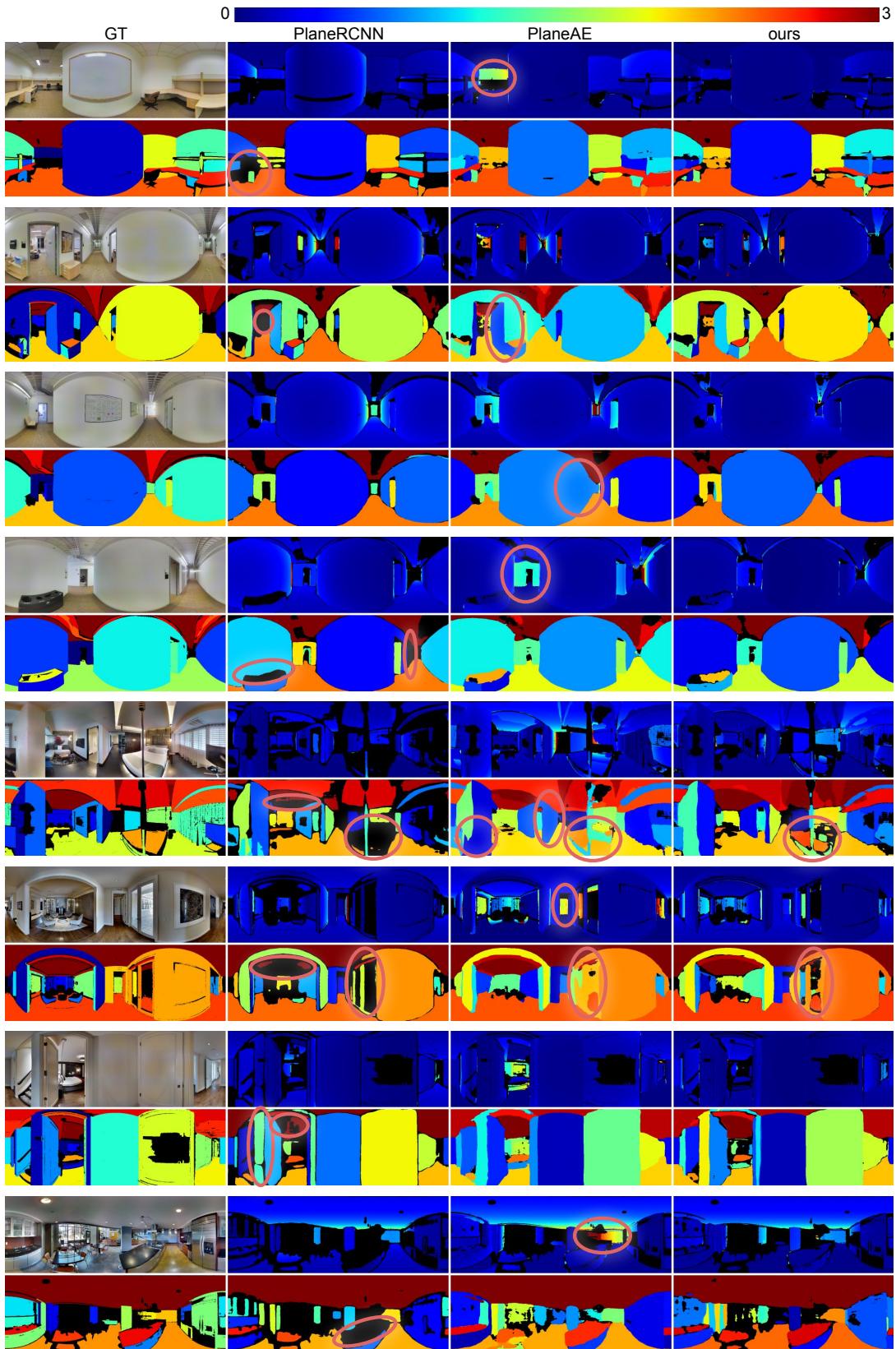


Figure 13: Qualitative comparisons with baselines. (See main text for details.)



Figure 14: 3D mesh visualization from our approach.

- [10] Clara Fernandez-Labrador, Alejandro Pérez-Yus, Gonzalo López-Nicolás, and Josechu J. Guerrero. Layouts from panoramic images with geometry and deep learning. *IEEE Robotics and Automation Letters*, 3(4):3153–3160, 2018.
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.
- [13] Ziyu Jiang, Buyu Liu, Samuel Schulter, Zhangyang Wang, and Manmohan Chandraker. Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 110–118. IEEE, 2020.
- [14] Lei Jin, Yanyu Xu, Jia Zheng, Junfei Zhang, Rui Tang, Shugong Xu, Jingyi Yu, and Shenghua Gao. Geometric structure based and regularized depth estimation from 360 indoor imagery. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 886–895. IEEE, 2020.
- [15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [16] Shu Kong and Charless C. Fowlkes. Recurrent pixel embedding for instance grouping. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9018–9028, 2018.
- [17] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3d plane detection and reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4450–4459, 2019.
- [18] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. PlaneNet: piece-wise planar reconstruction from a single RGB image. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2579–2588, 2018.
- [19] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 9628–9639, 2018.
- [20] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2277–2287, 2017.
- [21] Yiming Qian and Yasutaka Furukawa. Learning pairwise inter-plane relations for piecewise planar reconstruction. In *Computer Vision - ECCV 2020 - European Conference*, 2020.
- [22] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), with CD-ROM, 27 June - 2 July 2004, Washington, DC, USA*, pages 203–209. IEEE Computer Society, 2004.
- [23] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, pages 746–760, 2012.
- [24] Shuran Song, Andy Zeng, Angel X. Chang, Manolis Savva, Silvio Savarese, and Thomas A. Funkhouser. Im2pano3d: Extrapolating 360° structure and semantics beyond the field of view. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3847–3856, 2018.
- [25] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. HorizonNet: learning room layout with 1d representation and pano stretch data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1047–1056, 2019.
- [26] Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360sd-net: 360° stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 582–588. IEEE, 2020.
- [27] Tsun-Hsuan Wang, Hung-Jui Huang, Juan-Ting Lin, Chan-Wei Hu, Kuo-Hao Zeng, and Min Sun. Omnidirectional CNN for visual place recognition and navigation. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 2341–2348, 2018.
- [28] Jiu Xu, Björn Stenger, Tommi Kerola, and Tony Tung. Pano2cad: Room layout from a single panorama image. In *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*, pages 354–362, 2017.
- [29] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, volume 11214 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2018.
- [30] Hao Yang and Hui Zhang. Efficient 3d room shape recovery from a single panorama. In *2016 IEEE Conference on Com-*

- puter Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5422–5430, 2016.
- [31] Yang Yang, Shi Jin, Ruiyang Liu, Sing Bing Kang, and Jingyi Yu. Automatic 3d indoor scene modeling from single panorama. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3926–3934, 2018.
 - [32] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1029–1037, 2019.
 - [33] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. PanoContext: A whole-room 3d context model for panoramic scene understanding. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI*, pages 668–686, 2014.
 - [34] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A large photo-realistic dataset for structured 3d modeling. *CoRR*, abs/1908.00222, 2019.
 - [35] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. LayoutNet: reconstructing the 3d room layout from a single RGB image. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2051–2059, 2018.