Single-Image Piece-wise Planar 3D Reconstruction via Associative Embedding

Zehao Yu[1*]   Jia Zheng[1*]   Dongze Lian[1]   Zihan Zhou[2]   Shenghua Gao[1†]

[1]ShanghaiTech University   [2]The Pennsylvania State University

{yuzh,zhengjia,liandz,gaoshh}@shanghaitech.edu.cn   zzhou@ist.psu.edu

Abstract

Single-image piece-wise planar 3D reconstruction aims to simultaneously segment plane instances and recover 3D plane parameters from an image. Most recent approaches leverage convolutional neural networks (CNNs) and achieve promising results. However, these methods are limited to detecting a fixed number of planes with certain learned order. To tackle this problem, we propose a novel two-stage method based on associative embedding, inspired by its recent success in instance segmentation. In the first stage, we train a CNN to map each pixel to an embedding space where pixels from the same plane instance have similar embeddings. Then, the plane instances are obtained by grouping the embedding vectors in planar regions via an efficient mean shift clustering algorithm. In the second stage, we estimate the parameter for each plane instance by considering both pixel-level and instance-level consistencies. With the proposed method, we are able to detect an arbitrary number of planes. Extensive experiments on public datasets validate the effectiveness and efficiency of our method. Furthermore, our method runs at 30 fps at the testing time, thus could facilitate many real-time applications such as visual SLAM and human-robot interaction. Code is available at `https://github.com/svip-lab/PlanarReconstruction`.

1. Introduction

Single-image 3D reconstruction is a fundamental problem in computer vision, with many applications in emerging domains such as virtual and augmented reality, robotics, and social media. In this paper, we address this challenging problem by recovering a piece-wise planar 3D model of a scene, that is, to find all the plane instances in a single RGB image and estimate their 3D parameters, as shown in Figure 1. The piece-wise planar model provides a compact representation of the 3D scene, which could benefit many applications such as SLAM and human-robot interaction.

---
[*]Equal contribution
[†]Corresponding author

Input image   Plane instance segmentation
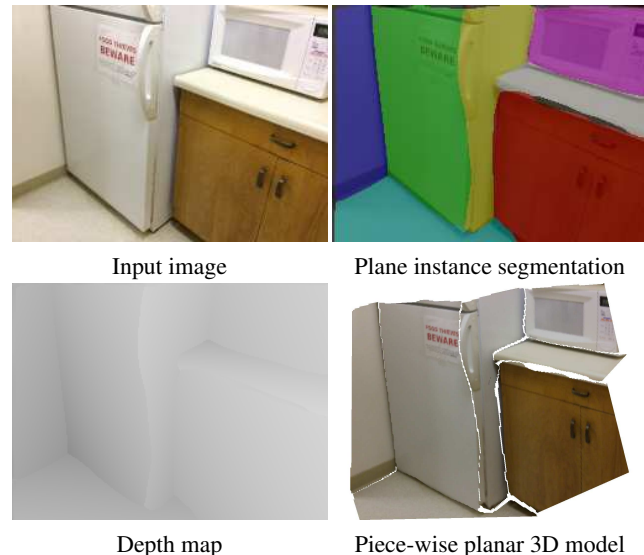
Depth map   Piece-wise planar 3D model

Figure 1: Piece-wise planar 3D reconstruction.

In the literature, most existing methods tackle this problem in a *bottom-up* manner [6, 2, 24, 31, 15, 10, 12]. They first extract geometric primitives such as straight line segments, vanishing points, corners, junctions, and image patches from the image. These primitives are then grouped into planar regions based on their geometric relationships. However, in practice, detecting the constituent geometric primitives itself is highly challenging, often resulting in a large number of missed detections (*e.g.*, due to poorly textured surfaces, lighting conditions) and outliers (*e.g.*, due to the presence of non-planar objects). As a result, statistical techniques such as RANSAC or Markov Random Field (MRF) are commonly employed to produce the final 3D models. But such techniques often break down when the percentage of missed and irrelevant detections is high, and are only applicable to restrictive scenarios (*e.g.*, Manhattan world scenes). Further, the optimization of the statistical model is time-consuming, which greatly limits their application in real-time tasks.

Different from bottom-up methods, a *top-down* approach [13] overcomes the aforementioned difficulties by analyzing the image in a holistic fashion, without resorting

to local geometric primitives. Recently, [23, 30] train CNNs to directly predict plane segmentation and plane parameters from a single image. These methods are shown to achieve the state-of-the-art performance on multiple indoor and outdoor datasets. Despite their advantages, current learning-based methods come with their own limitations. In particular, due to the lack of prior knowledge about the number and specific order of planes in an image, they are limited to detecting a fixed number of planes with certain learned order, thus may be not flexible enough to handle variations in real-world scene structure.

In this paper, we propose a novel *CNN-based, bottom-up* approach which takes the best of both worlds, while avoiding the limitations of existing methods. To make this possible, our key insight is that we can detect plane instances in an image by computing the likelihood that two pixels belong to the same plane instance and then use these likelihoods to group similar pixels together. Unlike traditional bottom-up methods which perform grouping on geometric primitives, our similarity metric is based on a deep embedding model, following its recent success in pose estimation [25], object detection [19], and instance segmentation [9, 5, 17]. Next, we mask the non-planar pixels with a planar/non-planar segmentation map generated by another CNN branch. Finally, an efficient mean shift clustering algorithm is employed to cluster the masked pixel embeddings into plane instances.

Following the plane instance segmentation, we design a plane parameter network by considering both pixel-level accuracy and instance-level consistencies. We first predict the plane parameter at each pixel, then combine those predictions with the plane instances to generate the parameter of each plane. Note that, unlike existing CNN methods, we restrict our networks to make local predictions (*i.e.*, pixel-wise embedding vectors, and plane parameters) and group these predictions in a bottom-up fashion. This enables our method to generate an arbitrary number of planes and avoid being restricted to any specific order or spatial layout.

In summary, **our contributions** are as follows: i) We present a novel two-stage deep learning framework for piece-wise planar 3D reconstruction. Based on the deep associate embedding model, we design a multi-branch, end-to-end trainable network which can detect an arbitrary number of planes and estimate their parameters simultaneously. ii) We propose a fast variant of mean shift clustering algorithm to group pixel embeddings into plane instances, which achieves real-time performance at the testing time. iii) Extensive experiments on two challenging datasets, ScanNet [4] and NYUv2 [27], validate the effectiveness and efficiency of our method.

2. Related Work

2.1. Single-View Planar Reconstruction

Geometry-based methods. Geometry-based methods [6, 2, 24, 20] recover 3D information based on geometric cues in 2D image. For example, Delage *et al.* [6] first extract line segments, vanishing points, and superpixels from the image. Then an MRF model is used to label the superpixels with a predefined set of plane classes (*i.e.*, three dominant plane orientations under the Manhattan world assumption). Similarly, Barinova *et al.* [2] assume that the environment is composed of a flat ground and vertical walls, and use a Conditional Random Field (CRF) model to label the detected primitives. Lee *et al.* [20] detect a collection of line segments and vanishing points in an image, and search for the building model in a hypothesis set that best matches the collection of geometric primitives. However, all these approaches rely on strong assumptions about the scene, which limit their applicability in practice.

Appearance-based methods. Appearance-based methods infer geometric properties of an image based on its appearance. Early works [15, 10, 12] take a *bottom-up* approach. They first predict the orientations of local image patches, and then group the patches with similar orientations to form planar regions. Hoiem *et al.* [15] define a set of discrete surface layout labels, such as "support", "vertical", and "sky", and use a number of hand-crafted local image features (*e.g.*, color, texture, location, and perspective) to train a model to label each superpixel in an image. Haines and Calway [12] learn to predict continuous 3D orientations for pre-segmented regions and cast plane detection as an optimization problem with an MRF model. Fouhey *et al.* [10] first detect convex/concave edges, occlusion boundaries, superpixels, and their orientations, then formulate the grouping problem as a binary quadratic program under the Manhattan world assumption. Our method also falls into this category. Different from existing methods, we cast plane detection as an instance segmentation problem, in which we learn a similarity metric to directly segment plane instances in an image, and then estimate plane parameter for each plane instance.

Recently, several CNN-based methods have been proposed to directly predict global 3D plane structures. Liu *et al.* [23] propose a deep neural network that learns to infer plane parameters and assign plane IDs (segmentation masks) to each pixel in a single image. Yang and Zhou [30] cast the problem as a depth prediction problem and propose a training scheme which does not require ground truth 3D planes. However, these approaches are limited to predicting a fixed number of planes, which could lead to a degraded performance in complex scenes. Concurrently, Liu *et al.* [22] address this problem using a proposal-based instance segmentation framework, *i.e.*, Mask R-CNN [14].
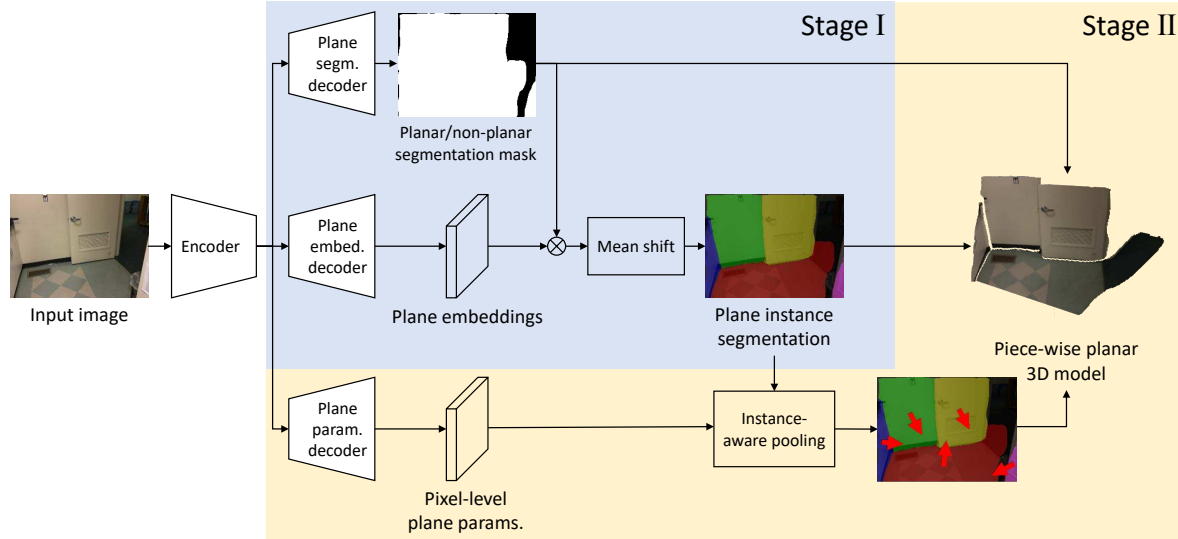
Figure 2: Network architecture. In the **first stage**, the network takes a single RGB image as input, and predicts a planar/non-planar segmentation mask and pixel-level embeddings. Then, an efficient mean shift clustering algorithm is applied to generate plane instances. In the **second stage**, we estimate parameter of each plane by considering both pixel-level and instance-level geometric consistencies.

Instead, we leverage a proposal-free instance segmentation approach [5] to solve this problem.

2.2. Instance Segmentation

Popular approaches to instance segmentation first generate region proposals, then classify the objects in the bounding box and segment the foreground objects within each proposal [14]. Recent work on associative embedding [25] and their extensions in object detection [19] and instance segmentation [9, 5, 17] provide a different solution. These methods learn an embedding function that maps pixels into an embedding space where pixels belonging to the same instance have similar embeddings. Then, they use a simple cluster technique to generate instance segmentation results. Newell *et al.* [25] introduce associative embedding in the context of multi-person pose estimation and extend it to proposal-free instance segmentation. De Brabandere *et al.* [5] propose a discriminative loss to learn the instance embedding, then group embeddings to form instances using a mean shift clustering algorithm. Kong and Fowlkes [17] introduce a recurrent model to solve the pixel-level clustering problem. Our method is particularly inspired by these work where we treat each plane in an image as an instance, and utilize the idea of associative embedding to detect plane instances. But we further propose i) an efficient mean shift algorithm to cluster plane instances, and ii) an end-to-end trainable network to jointly predict plane instance segmentation and plane parameters, which is not obvious in the context of original instance segmentation problem.

3. Method

Our goal is to infer plane instances and plane parameters from a single RGB image. We propose a novel two-stage method with a multi-branch network to tackle this problem. In the first stage, we train a CNN to obtain planar/non-planar segmentation map and pixel embeddings. We then mask the pixel embeddings with the segmentation map and group the masked pixel embeddings by an efficient mean shift clustering algorithm to form plane instances. In the second stage, we train a network branch to predict pixel-level plane parameters. We then use an instance-aware pooling layer with the instance segmentation map from the first stage to produce the final plane parameters. Figure 2 shows the overall pipeline of our method.

3.1. Planar/Non-Planar Segmentation

We first design an encoder-decoder architecture to distinguish the planar and non-planar regions. We use an extended version of ResNet-101-FPN [21] as an encoder.[1] The ResNet-101 implemented by [32, 33] is pretrained on ImageNet [7] for image classification. The decoder predicts planar/non-planar segmentation map for each pixel. Since the two classes are imbalanced in man-made environments, we use the balanced cross entropy loss as adopted in [29, 3]:

$$L_S = -(1-w)\sum_{i\in\mathcal{F}}\log p_i - w\sum_{i\in\mathcal{B}}\log(1-p_i), \quad (1)$$

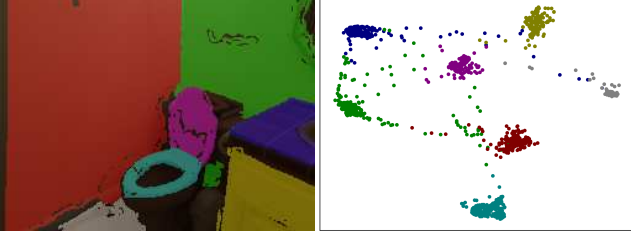[1]See supplementary materials for more details of network architecture.

Figure 3: The distribution of plane embeddings. The points with different colors denote learnt embeddings from different plane instances.

where $\mathcal{F}$ and $\mathcal{B}$ are the set of foreground and background pixels, respectively. p_i is the probability that i-th pixel belongs to foreground (*i.e.*, planar regions), and w is the foreground/background pixel-number ratio.

3.2. Embedding Model

Our plane instance segmentation is inspired by recent work on associative embedding [25, 9, 5, 17]. The main idea of associative embedding is to predict an embedding vector for each visual unit such that if some visual units belong to the same instance label, the distance between their embedding vectors should be small so that they can be easily grouped together.

For our task, we use a plane embedding branch to map pixels to some embedding space, as shown in Figure 3. This branch shares the same high-level feature maps with the plane segmentation branch. To enforce pixels in the same plane instance are closer than those in different planes, we use the discriminative loss in [5]. The loss consists of two terms, namely a "pull" loss and a "push" loss. The "pull" loss pulls each embedding to the mean embedding of the corresponding instance (*i.e.* the instance center), whereas the "push" loss pushes the instance centers away from each other.

$$L_E = L_{pull} + L_{push}, \qquad (2)$$

where

$$L_{pull} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} \max\left(\|\mu_c - x_i\| - \delta_{\mathrm{v}}, 0\right), \quad (3)$$

$$L_{push} = \frac{1}{C(C-1)} \sum_{\substack{c_A=1 \\ c_A \neq c_B}}^{C} \sum_{c_B=1}^{C} \max\left(\delta_{\mathrm{d}} - \|\mu_{c_A} - \mu_{c_B}\|, 0\right).$$
$$(4)$$

Here, C is the number of clusters $\mathcal{C}$ (planes) in the ground truth, N_c is the number of elements in cluster c, x_i is the pixel embedding, μ_c is the mean embedding of the cluster c, and δ_{v} and δ_{d} are the margin for "pull" and "push" losses, respectively.

Intuitively, if the pixel embeddings are easily separable (*i.e.*, the inter-instance distance is larger then δ_{d}, or the distance between an embedding vector and its center is smaller than δ_{v}), the penalty is zero. Otherwise, the penalty will increase sharply. Thus, the loss acts like hard example mining since it only penalizes difficult cases in the embedding.

3.3. Efficient Mean Shift Clustering

Once we have the embedding vector for each pixel, we group them to form plane instances. Mean shift clustering is suitable for this task since the number of plane instances is not known *a priori*. However, the standard mean shift clustering algorithm computes pairwise distance on all pairs of pixel embedding vectors at each iteration. The complexity of each iteration is $O(N^2)$ where N is the number of pixels in the image. In practice, N is very large even for a small size image. For example, in our experiments, $N = 192 \times 256$, making the standard algorithm inapplicable.

To tackle this problem, we propose a fast variant of the mean shift clustering algorithm. Instead of shifting all pixels in embedding space, we only shift a small number of anchors in embedding space and assign each pixel to the nearest anchor. Specifically, let k, d denote the number of anchors per dimension and the embedding dimension, respectively, we generate k^d anchors uniformly in the embedding space. We then compute pairwise potential between anchor a_j and embedding vector x_i as follows:

$$p_{ij} = \frac{1}{\sqrt{2\pi} b} \exp\left(-\frac{m_{ij}^2}{2b^2}\right), \qquad (5)$$

where b is the bandwidth in mean shift clustering algorithm and $m_{ij} = \|a_j - x_i\|_2$ is the distance between a_j and x_i. The shift step of each anchor in each iteration t can be expressed as:

$$a_j^t = \frac{1}{Z_j^t} \sum_{i=1}^{N} p_{ij}^t \cdot x_i, \qquad (6)$$

where $Z_j^t = \sum_{i=1}^{N} p_{ij}^t$ is a normalization constant. To further speed up the process, we filter out those anchors with low local density at the beginning of clustering.

After the algorithm converges, we merge nearby anchors to form clusters $\tilde{\mathcal{C}}$, where each cluster $\tilde{c}$ corresponds to a plane instance. Specifically, we consider two anchors belongs to the same cluster if their distance less than bandwidth b. The center of this cluster is the mean of anchors belonging to this cluster.

Finally, we associate pixel embeddings to clusters using soft assignment:

$$S_{ij} = \frac{\exp\left(-m_{ij}\right)}{\sum_{j=1}^{\tilde{C}} \exp\left(-m_{ij}\right)}. \qquad (7)$$

Algorithm 1 Efficient Mean Shift Clustering.

1: **Input:** pixel embeddings $\{x_i\}_{i=1}^N$, hyper-parameters k, d, b, and T
2: initialize k^d anchors uniformly in the embedding space
3: **for** $t = 1$ to T **do**
4:     compute pairwise potential term p_{ij}^t with Eq. (5)
5:     conduct mean shift for each anchor with Eq. (6)
6: **end for**
7: merge nearby anchors to form clusters $\tilde{C}$
8: **Output:** instance segmentation map S with Eq. (7)

The details of the proposed algorithm are shown in Algorithm 1. Note that the bandwidth b can be determined by the desired margin in the training stage of vector embedding. The complexity of each iteration of our algorithm is $O(k^d N)$. As long as $k^d \ll N$, our algorithm can be performed much more efficiently.

3.4. Plane Parameter Estimation

Given an image, the previous stage provides us a plane instance segmentation map. Then we need to infer the 3D parameter for each plane instance. To this end, we further design a plane parameter branch to predict the plane parameter for each pixel. Then, using the instance segmentation map, we aggregate the output of this branch to form an instance-level parameter for each plane instance.

Specifically, the branch output a $H \times W \times 3$ plane parameter map. Following [30], we define the plane parameter as $n \in \mathbb{R}^3$. For 3D points Q lies on this plane, we have $n^T Q = 1$.[2] We use L1 loss to supervise the learning of per-pixel plane parameters:

$$L_{PP} = \frac{1}{N} \sum_{i=1}^N \|n_i - n_i^*\|, \qquad (8)$$

where n_i is the predicted plane parameter and n_i^* is the ground truth plane parameter for i-th pixel.

Instance-aware pooling. In practice, we find that pixel-level parameter supervision is not sufficient, as it may not produce consistent outputs across the entire plane instance. Therefore we propose to further aggregate the pixel-level parameters into an instance-level parameter:

$$n_j = \frac{1}{Z_j} \sum_{i=1}^N S_{ij} \cdot n_i, \qquad (9)$$

where $Z_j = \sum_{i=1}^N S_{ij}$ is a normalization constant. It acts like a global average pooling but with different attention for different plane instances.

---
[2]We represent a 3D plane by $n \doteq \tilde{n}/d$, where $\tilde{n} \in \mathcal{S}^2$ and d denote the surface normal and plane distance to the origin.

Following [30], we enforce the instance-level parameter to be consistent with the scene geometry. To be specific, we compare the depth map inferred from the plane parameter with the ground truth depth map using the following loss:

$$L_{IP} = \frac{1}{N\tilde{C}} \sum_{j=1}^{\tilde{C}} \sum_{i=1}^N S_{ij} \cdot \|n_j^T Q_i - 1\|, \qquad (10)$$

where Q_i is the 3D point at pixel i inferred from ground truth depth map.

Note that our approach to plane parameter estimation is different from previous methods [23, 30]. Those methods first predict plane parameter and then associate each pixel with a particular plane parameter. In contrast, we first group pixels into plane instances and then estimate the parameter for each plane instance. We argue that our approach is more adequate because segmentation can uniquely determine an instance.

Finally, to simultaneously infer plane instance segmentation and plane parameters, the overall training loss of our method is:

$$L = L_S + L_E + L_{PP} + L_{IP}. \qquad (11)$$

4. Experiments

In this section, we conduct experiments to evaluate the performance of the proposed method on two public datasets: ScanNet [4] and NYUv2 [27]. Due to space limitations, we refer readers to supplementary materials for additional experiment results, including ablation studies about the mean shift clustering algorithm and plane parameter estimation.

4.1. Implementation Details

We implement our model with PyTorch [26]. We use Adam optimizer [16] with a learning rate of 10^{-4} and a weight decay of 10^{-5}. The batch size is set to 16. The network is trained for 50 epochs on one NVIDIA TITAN XP GPU device. We train the network with margins $\delta_v = 0.5$, $\delta_d = 1.5$. We set the embedding dimension $d = 2$, number of anchors per dimension $k = 10$, and the bandwidth $b = \delta_v$ in the mean shift clustering algorithm. The number of iterations T is set to 5 in training and set to 10 in testing. Our model is trained in an end-to-end manner.

4.2. Results on ScanNet Dataset

We first evaluate our method on ScanNet dataset [4] generated by [23]. The ground truth is obtained by fitting planes to a consolidated mesh of ScanNet and project them back to individual frames. The generating process also incorporates semantic annotations from ScanNet. The resulting dataset contains 50,000 training and 760 testing images with resolution 256×192.
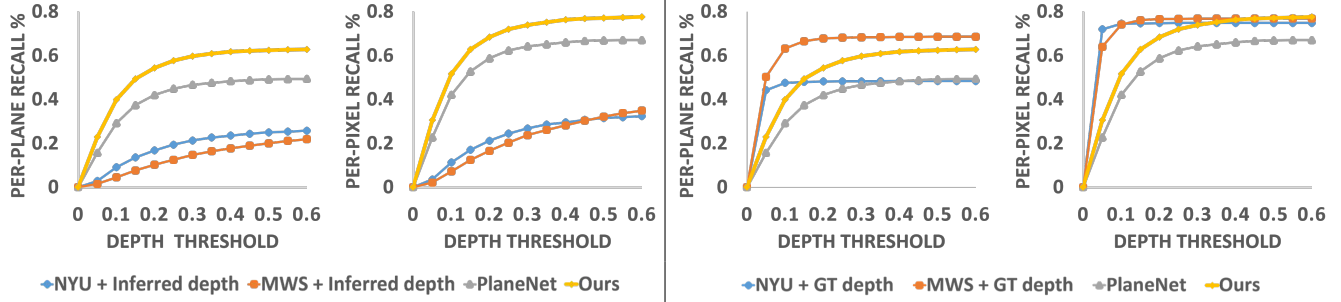
Figure 4: Plane and pixel recalls on the ScanNet dataset. Please see the supplementary materials for exact numbers.

Figure 5: Piece-wise planar 3D reconstruction results on the ScanNet dataset. In the plane instance segmentation results, black color indicates non-planar regions.

Methods for comparison. We compare our method with the recent CNN-based method PlaneNet [23], and two bottom-up methods NYU-Toolbox [27] and Manhattan World Stereo (MWS) [11].[3] NYU-Toolbox [27] is a popular plane detection algorithm that uses RANSAC to extracts plane hypotheses and Markov Random Field (MRF) to optimize plane segmentation. Manhattan World Stereo (MWS) [11] employs Manhattan world assumption for plane extraction and utilizes vanishing lines in the pairwise terms of MRF. For bottom-up methods, we use the same network architecture as ours to predict pixel-level depth map. Following [18], we minimize the berHu loss during training. Alternatively, we also use ground truth depth map as input for these methods.

Evaluation metric. Following [23], we use plane and pixel recalls as our evaluation metrics. The plane recall is the per-

centage of correctly predicted ground truth planes, and the pixel recall is the percentage of pixels within the correctly predicted planes. A ground-truth plane is considered correctly predicted if i) one of the predicted planes has more than 0.5 intersection-over-union (IOU) score, and ii) the mean depth difference over the overlapping region is less than a threshold, which varies from 0.05m to 0.6m with an increment of 0.05m. In addition, we also use surface normal difference as the threshold in our experiment.

Quantitative evaluation. Figure 4 shows the pixel and plane recalls of all methods. As shown in the first two plots, our method significantly outperforms all competing methods when inferred depth maps are used. Furthermore, we achieve competitive or better results even when the bottom-up methods are provided with the ground truth depth maps, as shown in the last two plots. This clearly demonstrates the effectiveness of our method. Furthermore, we obtain consistent results when the surface normal difference is adopted as the threshold (see supplementary materials).

---

[3]We obtain the implementation of these methods from PlaneNet [23] at https://github.com/art-programmer/PlaneNet.

Figure 6: Plane instance segmentation results on the ScanNet dataset.

Input image   NYU-Toolbox [27]   MWS [11]   PlaneNet [23]   Ours   Ground truth

Table 1: Runtime comparison (* denotes CPU time).

Method	NYU-Toolbox [27]	MWS [11]	PlaneNet [23]	Ours
FPS	0.14*	0.05*	1.35	**32.26**

Table 2: Plane instance segmentation results on the NYUv2 test set.

Method	RI ↑	VI ↓	SC ↑
GT Depth + NYU-Toolbox [27]	0.875	1.284	0.544
PlaneNet [23]	0.723	1.932	0.404
Ours	**0.888**	**1.380**	**0.519**

Qualitative evaluation. Figure 5 shows our reconstruction results for a variety of scenes. The qualitative comparisons against existing methods on plane instance segmentation are shown in Figure 6. We make the following observations: i) All methods perform well in simple scenarios (*e.g.*, the first row). ii) PlaneNet [23] and our method produce significantly better segmentation results in most cases (*e.g.*, the second and the third row). The poor performance by bottom-up methods is likely due to the noise in the predicted depth maps. In such cases, it is hard to select a proper threshold to distinguish inliers (*i.e.*, points on a plane) and outliers during the grouping stage. iii) PlaneNet sometimes misses small planes (*e.g.*, the chairs in the fourth row) or incorrectly merges multiple planes (*e.g.*, the cabinet and the door in the fifth row), while our approach is more robust in those cases. This is probably due to the assumption of a fixed number of planes in PlaneNet. Our approach is not restricted to such an assumption, thus performs better in detecting structures at different scales.

Speed. Table 1 shows the runtime comparison results

with other methods on the ScanNet dataset. All timings are measured on the same computing platform with Xeon E5-2630 @2.2GHz (20 cores) and a single NVIDIA TITAN XP GPU. Our method achieves the fastest speed of 32.26 fps on a single GPU, making it suitable for many real-time applications such as visual SLAM.

4.3. Results on NYUv2 Dataset

We further evaluate the performance of our method on the NYUv2 dataset [27], which contains 795 training images and 654 test images. Specifically, we conduct experiments to examine i) the generalizability of our learnt embedding on plane instance segmentation, and ii) the depth prediction accuracy of our method.

Plane instance segmentation. In this experiment, we directly use PlaneNet and our model trained on the ScanNet dataset to predict plane instances on the NYUv2 dataset. Following [23], we generate ground truth plane instances

Table 3: Comparison of depth prediction accuracy on the NYUv2 test set.

Method	Lower the better					Higher the better		
	Rel	Rel(sqr)	$\log_{10}$	RMSE_{iin}	$\mathrm{RMSE}_{\log}$	1.25	1.25^2	1.25^3
Eigen-VGG [8]	0.158	0.121	0.067	0.639	0.215	77.1	95.0	98.8
SURGE [28]	0.156	0.118	0.067	0.643	0.214	76.8	95.1	98.9
FCRN [18]	0.152	0.119	0.072	0.581	0.207	75.6	93.9	98.4
PlaneNet [23]	0.142	0.107	0.060	0.514	0.179	81.2	95.7	98.9
Ours (depth-direct)	**0.134**	**0.099**	**0.057**	**0.503**	**0.172**	**82.7**	**96.3**	**99.0**
Ours	0.141	0.107	0.061	0.529	0.184	81.0	95.7	**99.0**

in the test images by first fitting plane in each semantic instance using RANSAC and further merging two planes if the mean distance is below 10cm. For quantitative evaluation, we employ three popular metrics in segmentation [1, 30]: Rand index (RI), variation of information (VI), and segmentation covering (SC). As shown in Table 2, our method significantly outperforms PlaneNet in terms of all metrics. This suggests that our embedding-based approach is more generalizable than existing CNN-based method. And our method remains competitive against traditional bottom-up method NYU-Toolbox, even when the latter is provided with the ground truth depth maps. We refer readers to supplementary materials for qualitative results on the NYUv2 dataset.

Depth prediction. While our method demonstrates superior performance in piece-wise planar 3D reconstruction, it's also interesting to evaluate the network's capacity for per-pixel depth prediction. For this experiment, we fine-tune our network using the ground truth plane instances we generated on NYUv2 dataset. Table 3 compares the accuracy of the depth maps derived from our network output (i.e., the piece-wise planar 3D models) against those generated by standard depth prediction methods. As one can see, our method outperforms or is comparable to all other methods, which further verifies the quality of the 3D planes recovered by our method.

We have also trained a variant of our network, denoted as "Ours (depth-direct)", by fine-tuning the plane parameter prediction branch only using pixel-level supervision. Then, we use this branch to directly predict depth maps. As shown in Table 3, compared to this variant, using our piece-wise planar representation results in slight decrease in depth prediction accuracy, as such a representation sometimes ignores details and small variations in the scene structure.

4.4. Failure Cases

We show some failure cases in Figure 7. In the first example, our method merges the whiteboard and the wall into one plane. This may be because the appearance of these two planes are similar. A possible solution is to separate them by incorporating semantic information. In the second example, our method separates one plane into two planes (wall and headboard) because of the distinct appearance. One can

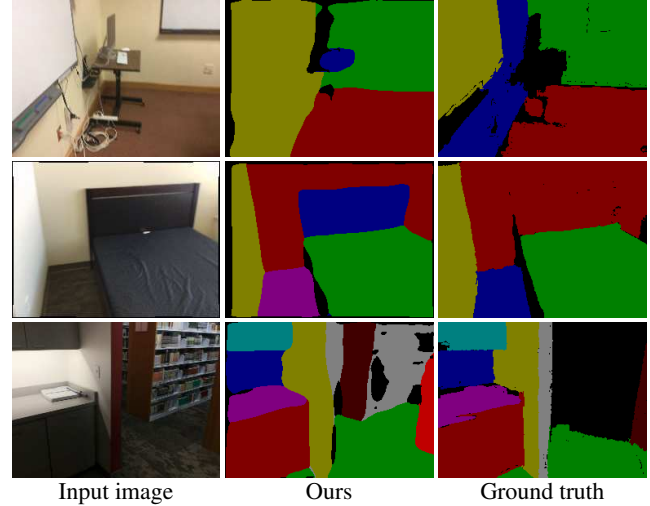| Input image	Ours	Ground truth

Figure 7: Failure cases.

easily merge these two planes using plane parameters in a post-processing step. In the third example, our method fails to segment the whole bookshelf. A possible reason is that the plane instance annotations obtained by fitting are not consistent, i.e., the bookshelf, in this case, is not labeled in the ground truth.

5. Conclusion

This paper proposes a novel two-stage method to single-image piece-wise planar 3D reconstruction. Specifically, we learn a deep embedding model to directly segment plane instances in an image, then estimate 3D parameter for each plane by considering pixel-level and instance-level geometric consistencies. The proposed method significantly outperforms the state-of-the-art methods while achieving real-time performance. While the proposed method has demonstrated promising results, it is still far from perfect. Possible future directions include i) incorporating semantic information to improve the reconstruction results, and ii) learning to reconstruct piece-wise planar 3D models from videos by leveraging the theory of multi-view geometry.

Acknowledgements

References

[1] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 8

[2] Olga Barinova, Vadim Konushin, Anton Yakubenko, KeeChang Lee, Hwasup Lim, and Anton Konushin. Fast automatic single-view 3-d reconstruction of urban scenes. In *ECCV*, pages 100–113, 2008. 1, 2

[3] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR*, pages 221–230, 2017. 3

[4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 2, 5

[5] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017. 2, 3, 4

[6] Erick Delage, Honglak Lee, and Andrew Y Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *ISRR*, pages 305–321, 2005. 1, 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 3

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015. 8

[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. 2, 3, 4

[10] David F Fouhey, Abhinav Gupta, and Martial Hebert. Unfolding an indoor origami world. In *ECCV*, pages 687–702, 2014. 1, 2

[11] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *CVPR*, pages 1422–1429, 2009. 6, 7

[12] Osian Haines and Andrew Calway. Recognising planes in a single image. *TPAMI*, 37(9):1849–1861, 2015. 1, 2

[13] Feng Han and Song-Chun Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*, pages 1778–1785, 2005. 1

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 2, 3

[15] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering Surface Layout from an Image. *IJCV*, 75(1):151–172, 2007. 1, 2

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[17] Shu Kong and Charless Fowlkes. Recurrent pixel embedding for instance grouping. In *CVPR*, pages 9018–9028, 2018. 2, 3, 4

[18] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, pages 239–248, 2016. 6, 8

[19] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 734–750, 2018. 2, 3

[20] David C Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, pages 2136–2143, 2009. 2

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3

[22] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. *CoRR*, abs/1812.04072, 2018. 2

[23] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *CVPR*, pages 2579–2588, 2018. 2, 5, 6, 7, 8

[24] Branislav Mičušík, Horst Wildenauer, and Markus Vincze. Towards detection of orthogonal planes in monocular images of indoor environments. In *ICRA*, pages 999–1004, 2008. 1, 2

[25] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NeurIPS*, pages 2277–2287, 2017. 2, 3, 4

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Workshop*, 2017. 5

[27] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 2, 5, 6, 7

[28] Peng Wang, Xiaohui Shen, Bryan Russell, Scott Cohen, Brian Price, and Alan Yuille. Surge: Surface regularized geometry estimation from a single image. In *NeurIPS*, pages 172–180, 2016. 8

[29] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, 2015. 3

[30] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *ECCV*, pages 85–100, 2018. 2, 5, 8

[31] Hao Yang and Hui Zhang. Efficient 3d room shape recovery from a single panorama. In *CVPR*, pages 5422–5430, 2016. 1

[32] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 3

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 127(3):302–321, 2018. 3