Recovering 3D Planes from a Single Image via Convolutional Neural Networks

Fengting Yang and Zihan Zhou

The Pennsylvania State University
{fuy34, zzhou}@ist.psu.edu

Abstract. In this paper, we study the problem of recovering 3D planar surfaces from a single image of man-made environment. We show that it is possible to directly train a deep neural network to achieve this goal. A novel plane structure-induced loss is proposed to train the network to simultaneously predict a plane segmentation map and the parameters of the 3D planes. Further, to avoid the tedious manual labeling process, we show how to leverage existing large-scale RGB-D dataset to train our network without explicit 3D plane annotations, and how to take advantage of the semantic labels come with the dataset for accurate planar and non-planar classification. Experiment results demonstrate that our method significantly outperforms existing methods, both qualitatively and quantitatively. The recovered planes could potentially benefit many important visual tasks such as vision-based navigation and human-robot interaction.

Keywords: 3D reconstruction; plane segmentation; deep learning

1 Introduction

Automatic 3D reconstruction from a single image has long been a challenging problem in computer vision. Previous work have demonstrated that an effective approach to this problem is exploring structural regularities in man-made environments, such as planar surfaces, repetitive patterns, symmetries, rectangles and cuboids [12], [21], [14], [15], [33], [28], [5]. Further, the 3D models obtained by harnessing such structural regularities are often attractive in practice, because they provide a high-level, compact representation of the scene geometry, which is desirable for many applications such as large-scale map compression, semantic scene understanding, and human-robot interaction.

In this paper, we study how to recover 3D planes – arguably the most common structure in man-made environments – from a single image. In the literature, several methods have been proposed to fit a scene with a piecewise planar model. These methods typically take a *bottom-up* approach: First, geometric primitives such as straight line segments, corners, and junctions are detected in the image. Then, planar regions are discovered by grouping the detected primitives based on their spatial relationships. For example, [6], [27], [3], [34] first detect line segments in the image, and then cluster them into several classes, each associated with

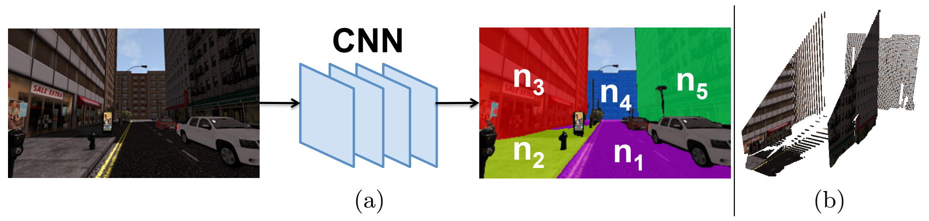

Fig. 1. We propose a new, end-to-end trainable deep neural network to recover 3D planes from a single image. **(a)** Given an input image, the network simultaneously predicts (i) a plane segmentation map that partitions the image into planar surfaces plus non-planar objects, and (ii) the plane parameters $\{\mathbf{n}_j\}_{j=1}^m$ in 3D space. **(b)** With the output of our network, a piecewise planar 3D model of the scene can be easily created.

a prominent vanishing point. [21] further detects junctions formed by multiple intersecting planes to generate model hypotheses. Meanwhile, [16], [9], [11] take a learning-based approach to predict the orientations of local image patches, and then group the patches with similar orientations to form planar regions.

However, despite its popularity, there are several inherent difficulties with the bottom-up approach. *First*, geometric primitives may not be reliably detected in man-made environments (e.g., due to the presence of poorly textured or specular surfaces). Therefore, it is very difficult to infer the geometric properties of such surfaces. *Second*, there are often a large number of irrelevant features or outliers in the detected primitives (e.g., due to presence of non-planar objects), making the grouping task highly challenging. This is the main reason why most existing methods resort to rather restrictive assumptions, e.g., requiring "Manhattan world" scenes with three mutually-orthogonal dominant directions or a "box" room model, to filter outliers and produce reasonable results. But such assumptions greatly limit the applicability of those methods in practice.

In view of these fundamental difficulties, we take a very different route to 3D plane recovery in this paper. Our method does not rely on grouping low-level primitives such as line segments and image patches. Instead, inspired by the recent success of convolutional neural networks (CNNs) in object detection and semantic segmentation, we design a novel, end-to-end trainable network to directly identify all planar surfaces in the scene, and further estimate their parameters in the 3D space. As illustrated in Fig. 1, the network takes a single image as input, and outputs (i) a segmentation map that identifies the planar surfaces in the image and (ii) the parameters of each plane in the 3D space, thus effectively creating a piecewise planar model for the scene.

One immediate difficulty with our learning-based approach is the lack of training data with annotated 3D planes. To avoid the tedious manual labeling process, we propose a novel plane structure-induced loss which essentially casts our problem as one of single-image depth prediction. Our key insight here is that, if we can correctly identify the planar regions in the image and predict the plane parameters, then we can also accurately infer the depth in these regions.

In this way, we are able to leverage existing large-scale RGB-D datasets to train our network. Moreover, as pixel-level semantic labels are often available in these datasets, we show how to seamlessly incorporate the labels into our network to better distinguish planar and non-planar objects.

In summary, **the contributions of this work** are: (i) We design an effective, end-to-end trainable deep neural network to directly recover 3D planes from a single image. (ii) We develop a novel learning scheme that takes advantage of existing RGB-D datasets and the semantic labels therein to train our network without extra manual labeling effort. Experiment results demonstrate that our method significantly outperforms, both qualitatively and quantitatively, existing plane detection methods. Further, our method achieves real-time performance at the testing time, thus is suitable for a wide range of applications such as visual localization and mapping, and human-robot interaction.

2 Related Work

3D plane recovery from a single image. Existing approaches to this problem can be roughly grouped into two categories: geometry-based methods and appearance-based methods. *Geometry-based methods* explicitly analyze the geometric cues in the 2D image to recover 3D information. For example, under the pinhole camera model, parallel lines in 3D space are projected to converging lines in the image plane. The common point of intersection, perhaps at infinity, is called the vanishing point [13]. By detecting the vanishing points associated with two sets of parallel lines on a plane, the plane's 3D orientation can be uniquely determined [6], [27], [3]. Another important geometric primitive is the junction formed by two or more lines of different orientations. Several work make use of junctions to generate plausible 3D plane hypotheses or remove impossible ones [21], [34]. And a different approach is to detect rectangular structures in the image, which are typically formed by two sets of orthogonal lines on the same plane [26]. However, all these methods rely on the presence of strong regular structures, such as parallel or orthogonal lines in a Manhattan world scene, hence have limited applicability in practice.

To overcome this limitation, *appearance-based methods* focus on inferring geometric properties of an image from its appearance. For example, [16] proposes a diverse set of features (e.g., color, texture, location and shape) and uses them to train a model to classify each superpixel in an image into discrete classes such as "support" and "vertical (left/center/right)". [11] uses a learning-based method to predict continuous 3D orientations at a given image pixel. Further, [9] automatically learns meaningful 3D primitives for single image understanding. Our method also falls into this category. But unlike existing methods which take a bottom-up approach by grouping local geometric primitives, our method trains a network to directly predict global 3D plane structures. Recently, [22] also proposes a deep neural network for piecewise planar reconstruction from a single image. But its training requires ground truth 3D planes and does not take advantage of the semantic labels in the dataset.

Machine learning and geometry. There is a large body of work on developing machine learning techniques to infer pixel-level geometric properties of the scene, mostly in the context of depth prediction [30], [7] and surface normal prediction [8], [18]. But few work has been done on detecting mid/hight-level 3D structures with supervised data. A notable exception which is also related to our problem is the line of research on indoor room layout estimation [14], [15], [28], [5], [20]. In these work, however, the scene geometry is assumed to follow a simple "box" model which consists of several mutually orthogonal planes (e.g., ground, ceiling, and walls). In contrast, our work aims to detect 3D planes under arbitrary configurations.

3 Method

3.1 Difficulty in Obtaining Ground Truth Plane Annotations

As most computer vision problems, a large-scale dataset with ground truth annotations is needed to effectively train the neural network for our task. Unfortunately, since the planar regions often have complex boundaries in an image, manual labeling of such regions could be very time-consuming. Further, it is unclear how to extract precise 3D plane parameters from an image.

To avoid the tedious manual labeling process, one strategy is to automatically convert the per-pixel depth maps in existing RGB-D datasets into planar surfaces. To this end, existing multi-model fitting algorithms can be employed to cluster 3D points derived from the depth maps. However, this is not an easy task either. Here, the fundamental difficulty lies in the choice of a proper threshold in practice to distinguish the inliers of a model instance (e.g., 3D points on a particular plane) from the outliers, *regardless of which algorithm one chooses.*

To illustrate this difficulty, we use the SYNTHIA dataset [29] which provides a large number of photo-realistic synthetic images of urban scenes and the corresponding depth maps (see Section 4.1 for more details). The dataset is generated by rendering a virtual city created using the Unity game development platform. Thus, the depth maps are noise-free. To detect planes from the 3D point cloud, we apply a popular multi-model fitting method called J-Linkage [31]. Similar to the RANSAC technique, this method is based on sampling consensus. We refer interested readers to [31] for a detailed description of the method.

A key parameter of J-Linkage is a threshold ϵ which controls the maximum distance between a model hypothesis (i.e., a plane) and the data points belonging to the hypothesis. In Fig. 2, we show example results produced by J-Linkage with different choices of ϵ. As one can see in Fig. 2(c), when a small threshold ($\epsilon = 0.5$) is used, the method breaks the building facade on the right into two planes. This is because the facade is not completely planar due to small indentations (e.g., the windows). When a large threshold ($\epsilon = 2$) is used (Fig. 2(d)), the stairs on the building on the left are incorrectly grouped with another building. Also, some objects (e.g., cars, pedestrians) are merged with the ground. If we use these results as ground truth to train a deep neural network, the network will

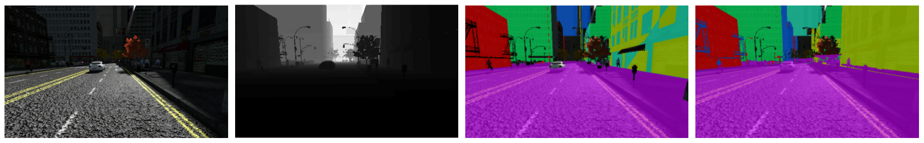

Fig. 2. Difficulty in obtaining ground truth plane annotations. **(a-b):** Original image and depth map. **(c-d):** Plane fitting results generated by J-Linkage with $\epsilon = 0.5$ and $\epsilon = 2$, respectively.

also likely learn the systematic errors in the estimated planes. And the problem becomes even worse if we want to train our network on real datasets. Due to the limitation of existing 3D acquisition systems (e.g., RGB-D cameras and LIDAR devices) and computational tools, the depth maps in these datasets are often noisy and of limited resolution and limited reliable range. Clustering based on such depth maps is prone to errors.

3.2 A New Plane Structure-Induced Loss

The challenge in obtaining reliable labels motivates us to develop alternative training schemes for 3D plane recovery. Specifically, we ask the following question: Can we leverage the wide availability of large-scale RGB-D and/or 3D datasets to train a network to recognize geometric structures such as planes *without* obtaining ground truth annotations about the structures?

To address this question, our key insight is that, if we can recover 3D planes from the image, then we can use these planes to (partially) explain the scene geometry, which is generally represented by a 3D point cloud. Specifically, let $\{I_i, D_i\}_{i=1}^n$ denote a set of n training RGB image and depth map pairs with known camera intrinsic matrix K.[1] Then, for any pixel $\mathbf{q} \doteq [x, y, 1]^T$ (in homogeneous coordinates) on image I_i, it is easy to compute the corresponding 3D point as $Q = D_i(\mathbf{q}) \cdot K^{-1}\mathbf{q}$. Further, let $\mathbf{n} \in \mathbb{R}^3$ represents a 3D plane in the scene. If Q lies on the plane, then we have $\mathbf{n}^T Q = 1$.[2]

With the above observation, assuming there are m planes in the image I_i, we can now train a network to simultaneously output (i) a per-pixel probability map S_i, where $S_i(\mathbf{q})$ is an $(m+1)$-dimensional vector with its j-th element $S_i^j(\mathbf{q})$ indicating the probability of pixel $\mathbf{q}$ belonging to the j-th plane,[3] and (ii) the

[1] Without loss of generality, we assume a constant K for all images in the dataset.

[2] A common way to represent a 3D plane is $(\tilde{\mathbf{n}}, d)$ where $\tilde{\mathbf{n}}$ is a normal vector and d is the distance to the camera center. In this paper, we choose a more succinct parametrization: $\mathbf{n} \doteq \tilde{\mathbf{n}}/d$. Note that $\mathbf{n}$ can uniquely identify a 3D plane, assuming the plane is not through the camera center (which is valid for real world images).

[3] In this paper, we use $j = 0$ to denote the "non-planar" class.

plane parameters $\Pi_i = \{\mathbf{n}_i^j\}_{j=1}^m$, by minimizing the following objective function:

$$\mathcal{L} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\sum_{\mathbf{q}} S_i^j(\mathbf{q}) \cdot |(\mathbf{n}_i^j)^T Q - 1|\right) + \alpha \sum_{i=1}^{n} \mathcal{L}_{reg}(S_i), \tag{1}$$

where $\mathcal{L}_{reg}(S_i)$ is a regularization term preventing the network from generating a trivial solution $S_i^0(\cdot) \equiv 1$, i.e., classifying all pixels as non-planar, and α is a weight balancing the two terms.

Before proceeding, we make two important observations about our formulation Eq. (1). *First*, the term $|(\mathbf{n}_i^j)^T Q - 1|$ measures the deviation of a 3D scene point Q from the j-th plane in I_i, parameterized by $\mathbf{n}_i^j$. In general, for a pixel $\mathbf{q}$ in the image, we know from perspective geometry that the corresponding 3D point must lie on a ray characterized by $\lambda K^{-1}\mathbf{q}$, where λ is the depth at $\mathbf{q}$. If this 3D point is also on the j-th plane, we must have

$$(\mathbf{n}_i^j)^T \cdot \lambda K^{-1}\mathbf{q} = 1 \Longrightarrow \lambda = \frac{1}{(\mathbf{n}_i^j)^T \cdot K^{-1}\mathbf{q}}. \tag{2}$$

Hence, in this case, λ can be regarded as the depth at $\mathbf{q}$ constrained by $\mathbf{n}_i^j$. Now, we can rewrite the term as:

$$|(\mathbf{n}_i^j)^T Q - 1| = |(\mathbf{n}_i^j)^T D_i(\mathbf{q}) \cdot K^{-1}\mathbf{q} - 1| = |D_i(\mathbf{q})/\lambda - 1|. \tag{3}$$

Thus, the term $|(\mathbf{n}_i^j)^T Q - 1|$ essentially compares the depth λ induced by the j-th predicted plane with the ground truth $D_i(\mathbf{q})$, and penalizes the difference between them. In other words, our formulation casts the 3D plane recovery problem as a depth prediction problem.

Second, Eq. (1) couples plane segmentation and plane parameter estimation in a loss that encourages consistent explanations of the visual world through the recovered plane structure. It mimics the behavior of biological agents (e.g., humans) which also employ structural priors for 3D visual perception of the world [32]. This is in contrast to alternative methods that rely on ground truth plane segmentation maps and plane parameters as direct supervision signals to tackle the two problems separately.

3.3 Incorporating Semantics for Planar/Non-Planar Classification

Now we turn our attention to the regularization term $\mathcal{L}_{reg}(S_i)$ in Eq. (1). Intuitively, we wish to use the predicted planes to explain as much scene geometry as possible. Therefore, a natural choice of $\mathcal{L}_{reg}(S_i)$ is to encourage plane predictions by minimizing the cross-entropy loss with constant label 1 at each pixel. Specifically, let $p_{plane}(\mathbf{q}) = \sum_{j=1}^m S_i^j(\mathbf{q})$ be the sum of probabilities of pixel $\mathbf{q}$ being assigned to each plane, we write

$$\mathcal{L}_{reg}(S_i) = \sum_{\mathbf{q}} -1 \cdot \log(p_{plane}(\mathbf{q})) - 0 \cdot \log(1 - p_{plane}(\mathbf{q})). \tag{4}$$

Note that, while the above term effectively encourages the network to explain every pixel in the image using the predicted plane models, it treats all pixels equally. However, in practice, some objects are more likely to form meaningful planes than others. For example, a building facade is often regarded as a planar surface, whereas a pedestrian or a car is typically viewed as non-planar. In other words, if we can incorporate such high-level semantic information into our training scheme, the network is expected to achieve better performance in differentiating planar vs. non-planar surfaces.

Motivated by this observation, we propose to further utilize the semantic labels in the existing datasets. Take the SYNTHIA dataset as an example. The dataset provides precise pixel-level semantic annotations for 13 classes in urban scenes. For our purpose, we group these classes into "planar" = {building, fence, road, sidewalk, lane-marking} and "non-planar" = {sky, vegetation, pole, car, traffic signs, pedestrians, cyclists, miscellaneous}. Then, let $z(\mathbf{q}) = 1$ if pixel $\mathbf{q}$ belongs to one of the "planar" classes, and $z(\mathbf{q}) = 0$ otherwise, we can revise our regularization term as:

$$\mathcal{L}_{reg}(S_i) = \sum_{\mathbf{q}} -z(\mathbf{q}) \cdot \log(p_{plane}(\mathbf{q})) - (1 - z(\mathbf{q})) \cdot \log(1 - p_{plane}(\mathbf{q})). \quad (5)$$

Note that the choices of planar/non-planar classes are dataset- and problem-dependent. For example, one may argue that "sky" can be viewed as plane at infinity, thus should be included in the "planar" classes. Regardless the particular choices, we emphasize that here we provide a flexible way to incorporate high-level semantic information (generated by human annotators) to the plane detection problem. This is in contrast to traditional geometric methods that solely rely on a single threshold to distinguish planar vs. non-planar surfaces.

3.4 Network Architecture

In this paper, we choose a *fully convolutional network* (FCN), following its recent success in various pixel-level prediction tasks such as semantic segmentation [23], [2] and scene flow estimation [25]. Fig. 3 shows the overall architecture of our proposed network. To simultaneously estimate the plane segmentation map and plane parameters, our network consists of two prediction branches, as we elaborate below.

Plane segmentation map. To predict the plane segmentation map, we use an encoder-decoder design with skip connections and multi-scale side predictions, similar to the DispNet architecture proposed in [25]. Specifically, the encoder takes the whole image as input and produces high-level feature maps via a convolutional network. The decoder then gradually upsamples the feature maps via deconvolutional layers to make final predictions, taking into account also the features from different encoder layers. The multi-scale side predictions further allow the network to be trained with deep supervision. We use ReLU for all layers except for the prediction layers, where the softmax function is applied.

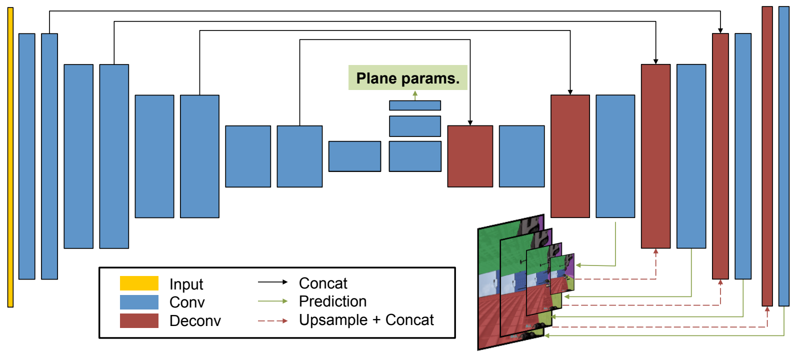

Fig. 3. Network architecture. The width and height of each block indicates the channel and the spatial dimension of the feature map, respectively. Each reduction (or increase) in size indicates a change by a factor of 2. The first convolutional layer has 32 channels. The filter size is 3 except for the first four convolutional layers (7, 7, 5, 5).

Plane parameters. The plane parameter prediction branch shares the same high-level feature maps with the segmentation branch. The branch consists of two stride-2 convolutional layers ($3 \times 3 \times 512$) followed by a $1 \times 1 \times 3m$ convolutional layer to output the parameters of the m planes. Global average pooling is then used to aggregate predictions across all spatial locations. We use ReLU for all layers except for the last layer, where no activation is applied.

Implementation details. Our network is trained from scratch using the publicly available Tensorflow framework. By default, we set the weight in Eq. (1) as $\alpha = 0.1$, and the number of planes as $m = 5$. During training, we adopt the Adam [17] method with $\beta_1 = 0.99$ and $\beta_2 = 0.9999$. The batch size is set to 4, and the learning rate is set to 0.0001. We also augment the data by scaling the images with a random factor in [1, 1.15] followed by a random cropping. Convergence is reached at about 500K iterations.

4 Experiments

In this section, we conduct experiments to study the performance of our method, and compare it to existing ones. All experiments are conducted on one Nvidia GTX 1080 Ti GPU device. At testing time, our method runs at about 60 frames per second, thus are suitable for potential real-time applications.[4]

4.1 Datasets and Ground Truth Annotations

SYNTHIA: The recent SYNTHIA dataset [29] comprises more than 200,000 photo-realistic images rendered from virtual city environments with precise pixelwise depth maps and semantic annotations. Since the dataset is designed to

[4] Please refer to supplementary materials for additional experiment results about (i) the choice of plane number and (ii) the effect of semantic labels.

facilitate autonomous driving research, all frames are acquired from a virtual car as it navigates in the virtual city. The original dataset contains seven different scenarios. For our experiment, we select three scenarios (SEQS-02, 04, and 05) that represents city street views. For each scenario, we use the sequences for all four seasons (spring, summer, fall, and winter). Note that, to simulate real traffic conditions, the virtual car makes frequent stops during navigation. As a result, the dataset has many near-identical frames. We filter these redundant frames using a simple heuristic based on the vehicle speed. Finally, from the remaining frames, we randomly sample 8,000 frames as the training set and another 100 frames as the testing set.

For quantitative evaluation, we need to label all the planar regions in the test images. As we discussed in Section 3.1, automatic generation of ground truth plane annotations is difficult and error-prone. Thus, we adopt a semi-automatic method to interactively determine the ground truth labels with user input. To label one planar surface in the image, we ask the user to draw a quadrilateral region within that surface. Then, we fit a plane to the 3D points (derived from the ground truth depth map) that fall into that region to obtain the plane parameters *and* an instance-specific estimate of the variance of the distance distribution between the 3D points and the fitted plane. Note that, with the instance-specific variance estimate, we are able to handle surfaces with varying degrees of deviation from a perfect plane, but are commonly regarded as "planes" by humans. Finally, we use the plane parameters and the variance estimate to find all pixels that belong to the plane. We repeat this process until all planes in the image are labeled.

Cityscapes: Cityscapes [4] contains a large set of real street-view video sequences recorded in different cities. From the 3,475 images with publicly available fine semantic annotations, we randomly select 100 images for testing, and use the rest for training. To generate the planar/non-planar masks for training, we label pixels in the following classes as "planar" = {ground, road, sidewalk, parking, rail track, building, wall, fence, guard rail, bridge, and terrain}.

In contrast to SYNTHIA, the depth maps in Cityscapes are highly noisy because they are computed from stereo correspondences. Fitting planes on such data is extremely difficult even with user input. Therefore, to identify planar surfaces in the image, we manually label the boundary of each plane using polygons, and further leverage the semantic annotations to refine it by ensuring that the plane boundary aligns with the object boundary, if they overlap.

4.2 Methods for Comparison

As discussed in Section 2, a common approach to plane detection is to use geometric cues such as vanishing points and junction features. However, such methods all make strong assumptions on the scene geometry, e.g., a "box"-like model for indoor scenes or a "vertical-ground" configuration for outdoor scenes. They would fail when these assumptions are violated, as in the case of SYNTHIA and Cityscapes datasets. Thus, we do not compare to these methods. Instead, we compare our method to the following *appearance-based methods*:

Depth + Multi-model Fitting: For this approach, we first train a deep neural network to predict pixel-level depth from a single image. We directly adopt the DispNet architecture [25] and train it from scratch with ground truth depth data. Following recent work on depth prediction [19], we minimize the berHu loss during training.

To find 3D planes, we have then applied two different multi-model fitting algorithms, namely J-Linkage [31] and RansaCov [24], on the 3D points derived from the predicted depth map. We call the corresponding methods **Depth + J-Linkage** and **Depth + RansaCov**, respectively. For fair comparison, we only keep the top-5 planes detected by each method. As mentioned earlier, a key parameter in these methods is the distance threshold ϵ. We favor them by running J-Linkage or RansaCov multiple times with various values of ϵ and retaining the best results.

Geometric Context (GC) [16]: This method uses a number of hand-crafted local image features to predict discrete surface layout labels. Specifically, it trains decision tree classifiers to label the image into three main geometric classes {support, vertical, sky}, and further divide the "vertical" class into five subclasses {left, center, right, porous, solid}. Among these labels, we consider the "support" class and "left", "center", "right" subclasses as four different planes, and the rest as non-planar.

To retrain their classifiers using our training data, we translate the labels in SYNTHIA dataset into theirs[5] and use the source code provided by the authors[6]. We found that this yields better performance on our testing set than the pre-trained classifiers provided by the authors. We do not include this method in the experiment on Cityscapes dataset because it is difficult to determine the orientation of the vertical structures from the noisy depth maps.

Finally, we note that there is another closely related work [11], which also detects 3D planes from a single image. Unfortunately, the source code needed to train this method on our datasets is currently unavailable. And it is reported in [11] that its performance on plane detection is on par with that of GC. Thus, we decided to compare our method to GC instead.

4.3 Experiment Results

Plane segmentation. Fig. 4 shows example plane segmentation results on SYNTHIA dataset. We make several important observations below.

First, Neither Depth + J-Linkage nor Depth + RansaCov performs well on the test images. In many cases, they fail to recover the individual planar surfaces (except the ground). To understand the reason, we show the 3D point cloud derived from the predicted depth map in Fig. 5. As one can see, the point cloud tends to be very noisy, making the task of choosing a proper threshold ϵ in the

[5] sky$\rightarrow$sky, {road, sidewalk, lane-marking}$\rightarrow$support, and the rest$\rightarrow$vertical. For the "building" and "fence" classes in the SYNTHIA dataset, we fit 3D planes at different orientations to determine the appropriate subclass label (i.e., left/center/right).

[6] http://dhoiem.cs.illinois.edu/

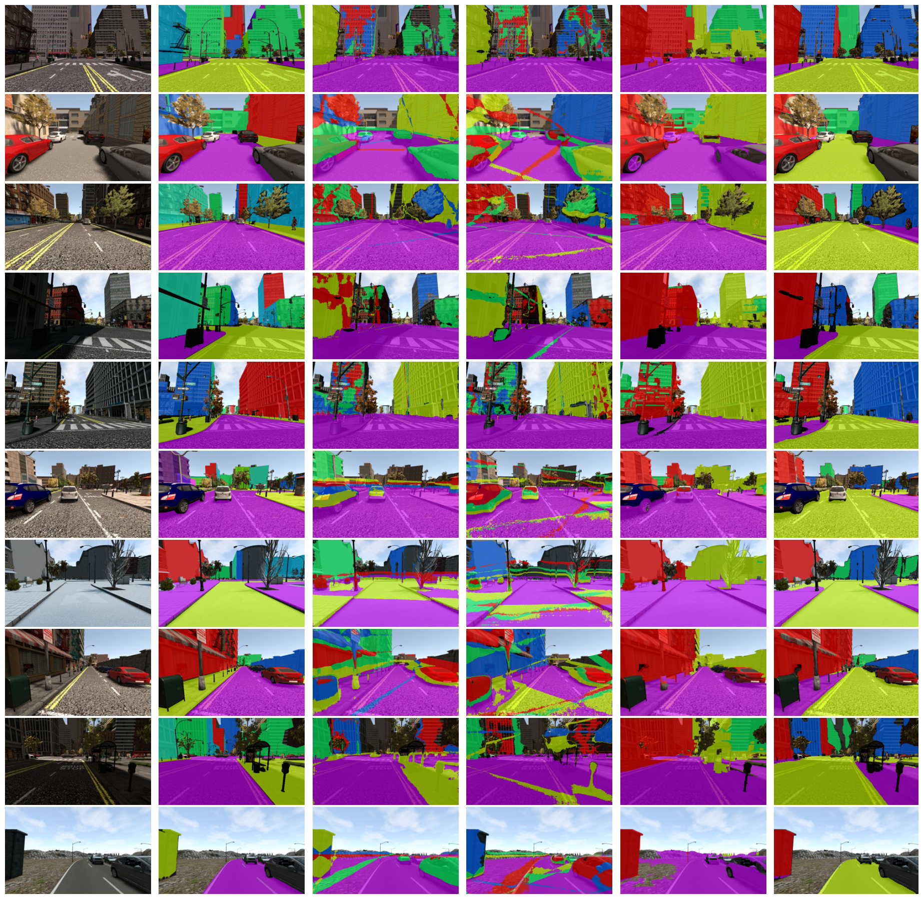

Fig. 4. Plane segmentation results on SYNTHIA. **From left to right:** Input image; Ground truth; Depth + J-Linkage; Depth + RansaCov; Geometric Context; Ours.

multi-model fitting algorithm extremely hard, if possible at all – if ϵ is small, it would not be able to tolerate the large noises in the point cloud; if ϵ is large, it would incorrectly merge multiple planes/objects into one cluster. Also, these methods are unable to distinguish planar and non-planar objects due to lack of ability to reason about the scene semantics.

Second, GC does a relatively good job in identifying major scene categories (e.g., separating the ground, sky from buildings). However, it has difficulty in determining the orientation of vertical structures (e.g., Fig. 4, first and fifth rows). This is mainly due to the coarse categorization (left/center/right) used by this method. In complex scenes, such a discrete categorization is often ineffective and ambiguous. Also, recall that GC is unable to distinguish planes that have the same orientation but are at different distances (e.g., Fig. 4, fourth row), not to mention finding the precise 3D plane parameters.

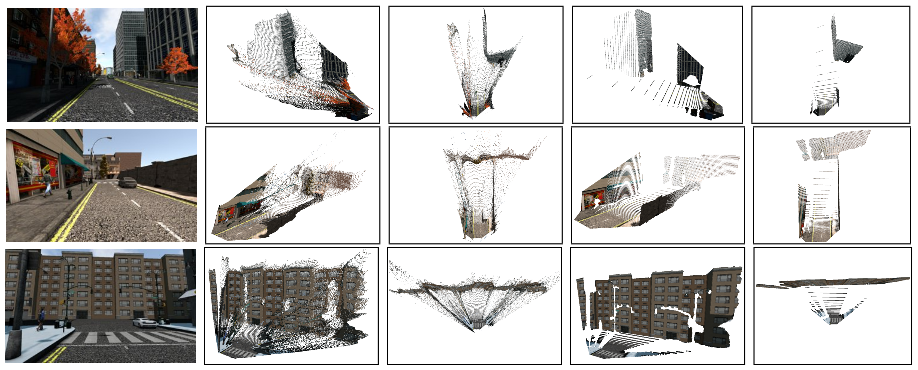

Fig. 5. Comparison of 3D models. **First column:** Input image. **Second and third columns:** Model generated by depth prediction. **Fourth and fifth columns:** Model generated by our method.

Table 1. Plane segmentation results. **Left:** SYNTHIA. **Right:** Cityscapes.

Method	RI	VOI	SC
Depth+J-Linkage	0.825	1.948	0.589
Depth+RansaCov	0.810	2.274	0.550
Geo. Context [16]	0.846	1.626	0.636
Ours	**0.925**	**1.129**	**0.797**

Method	RI	VOI	SC
Depth+J-Linkage	0.713	2.668	0.450
Depth+RansaCov	0.705	2.912	0.431
Ours (w/o fine-tuning)	0.759	1.834	0.597
Ours (w/ fine-tuning)	**0.884**	**1.239**	**0.769**

Third, our method successfully detects most prominent planes in the scene, while excluding non-planar objects (e.g., trees, cars, light poles). This is no surprise because our supervised framework implicitly encodes high-level semantic information as it learns from the labeled data provided by humans. Interestingly, one may observe that, in the last row of Fig. 4, our method classifiers the unpaved ground next to the road as non-planar. This is because such surfaces are not considered part of the road in the original SYNTHIA labels. Fig. 5 further shows some piecewise planar 3D models obtained by our method.

For quantitative evaluation, we use three popular metrics [1] to compare the plane segmentation maps obtained by an algorithm with the ground truth: Rand index (RI), variation of information (VOI), and segmentation covering (SC). Table 1(left) compares the performance of all methods on SYNTHIA dataset. As one can see, our method outperforms existing methods by a significant margin w.r.t all evaluation metrics.

Table 1(right) further reports the segmentation accuracies on Cityscapes dataset. We test our method under two settings: (i) directly applying our model trained on SYNTHIA dataset, and (ii) fine-tuning our network on Cityscapes dataset. Again, our method achieves the best performance among all methods. Moreover, fine-tuning on the Cityscapes dataset significantly boost the performance of our network, despite that the provided depth maps are very noisy. Finally, we show example segmentation results on Cityscapes in Fig. 6.

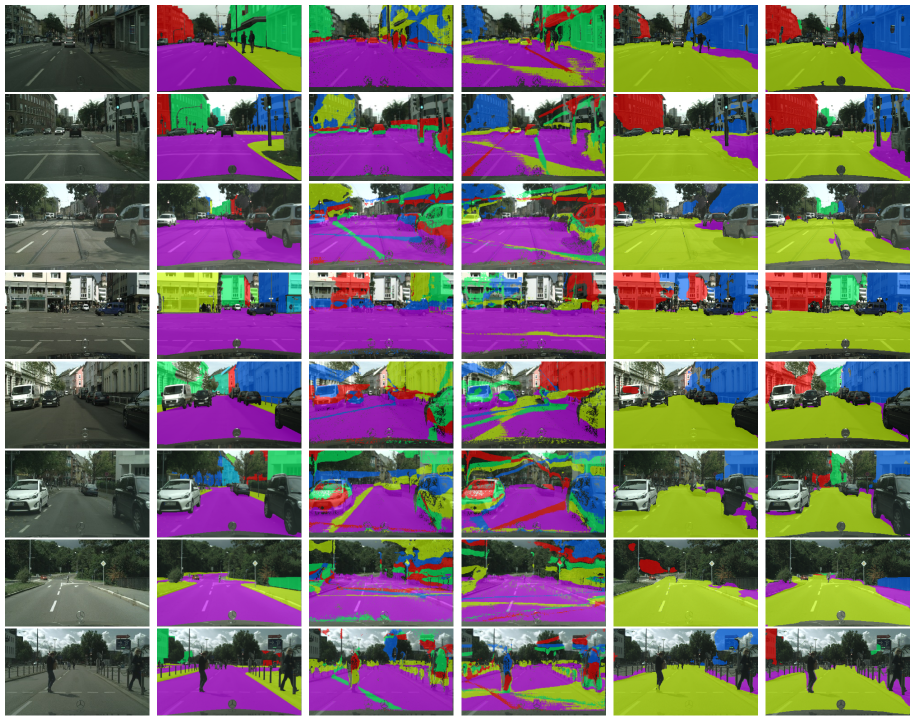
Fig. 6. Plane segmentation results on Cityscapes. **From left to right:** Input image; Ground truth; Depth + J-Linkage; Depth + RansaCov; Ours (w/o fine-tuning); Ours (w/ fine-tuning).

Depth prediction. To further evaluate the quality of the 3D planes estimated by our method, we compare the depth maps derived from the 3D planes with those obtained via standard depth prediction pipeline (see Section 4.2 for details). Recall that our method outputs a per-pixel probability map $S(\mathbf{q})$. For each pixel $\mathbf{q}$ in the test image, we pick the 3D plane with the maximum probability to compute our depth map. We exclude pixels which are considered as "non-planar" by our method, since our network is not designed to make depth predictions in that case.

As shown in Table 2, our method achieves competitive results on both datasets, but the accuracies are slightly lower than those of standard depth prediction pipeline. The decrease in accuracy may be partly attributed to that our method is designed to recover large planar structures in the scene, therefore ignores small variations and details in the scene geometry.

Failure cases. Fig. 7 shows typical failure cases of our method, which include occasionally separating one plane into two (first column) or merging multiple planes into one (second column). Interestingly, for the formal case, one can still obtain a decent 3D model (Fig. 5, last row), suggesting opportunities to further

Table 2. Depth prediction results.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
SYNTHIA							
Train set mean	0.3959	3.7348	10.6487	0.5138	0.3420	0.6699	0.8221
DispNet+berHu loss	0.0451	**0.2226**	**1.6491**	**0.0755**	**0.9912**	**0.9960**	**0.9976**
Ours	**0.0431**	0.3643	2.2405	0.0954	0.9860	0.9948	0.9966
Cityscapes							
Train set mean	0.2325	4.6558	15.4371	0.5093	0.6127	0.7352	0.8346
DispNet+berHu loss	**0.0855**	**0.7488**	**5.1307**	**0.1429**	**0.9222**	**0.9776**	**0.9907**
Ours	0.1042	1.4938	6.8755	0.1869	0.8909	0.9672	0.9862

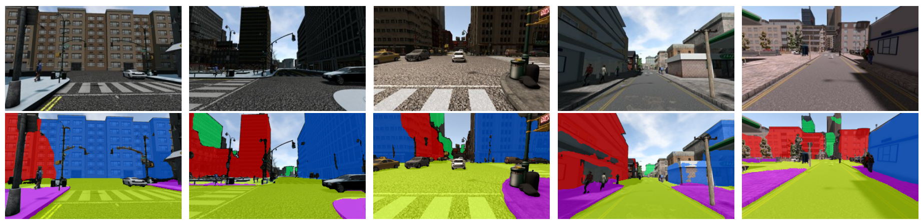
Fig. 7. Failure examples.

refine our results via post-processing. Our method also has problem with curved surfaces (third column).

Other failures are typically associated with our assumption that there are at most $m = 5$ planes in the scene. For example, in Fig. 7, fourth column, the building on the right has a large number of facades. And it becomes even more difficult when multiple planes are at great distance (fifth column). We leave adaptively choosing the plane number in our framework for future work.

5 Conclusion

This paper has presented a novel approach to recovering 3D planes from a single image using convolutional neural networks. We have demonstrated how to train the network, without 3D plane annotations, via a novel plane structure-induced loss. In fact, the idea of exploring structure-induced loss to train neural networks is by no means restricted to planes. We plan to generalize the idea to detect other geometric structures, such as rectangles and cuboids.

Another promising direction for future work would be to improve the generalizability of the networks via unsupervised learning, as suggested by [10]. For example, it is interesting to probe the possibility of training our network without depth information, which is hard to obtain in many real world applications.

Acknowledgement. This work is supported in part by a startup fund from Penn State and a hardware donation from Nvidia.

References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011)
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
3. Barinova, O., Konushin, V., Yakubenko, A., Lee, K., Lim, H., Konushin, A.: Fast automatic single-view 3-d reconstruction of urban scenes. In: ECCV (2). pp. 100–113 (2008)
4. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. pp. 3213–3223 (2016)
5. Dasgupta, S., Fang, K., Chen, K., Savarese, S.: Delay: Robust spatial layout estimation for cluttered indoor scenes. In: CVPR. pp. 616–624 (2016)
6. Delage, E., Lee, H., Ng, A.Y.: Automatic single-image 3d reconstructions of indoor manhattan world scenes. In: ISRR. pp. 305–321 (2005)
7. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS. pp. 2366–2374 (2014)
8. Fouhey, D.F., Gupta, A., Hebert, M.: Data-driven 3d primitives for single image understanding. In: ICCV. pp. 3392–3399 (2013)
9. Fouhey, D.F., Gupta, A., Hebert, M.: Unfolding an indoor origami world. In: ECCV. pp. 687–702 (2014)
10. Garg, R., Kumar, B.G.V., Carneiro, G., Reid, I.D.: Unsupervised CNN for single view depth estimation: Geometry to the rescue. In: ECCV. pp. 740–756 (2016)
11. Haines, O., Calway, A.: Recognising planes in a single image. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1849–1861 (2015)
12. Han, F., Zhu, S.C.: Bottom-up/top-down image parsing by attribute graph grammar. In: ICCV. pp. 1778–1785 (2005)
13. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000)
14. Hedau, V., Hoiem, D., Forsyth, D.A.: Recovering the spatial layout of cluttered rooms. In: ICCV. pp. 1849–1856 (2009)
15. Hedau, V., Hoiem, D., Forsyth, D.A.: Thinking inside the box: Using appearance models and context based on room geometry. In: ECCV. pp. 224–237 (2010)
16. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision **75**(1), 151–172 (2007)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
18. Ladicky, L., Zeisl, B., Pollefeys, M.: Discriminatively trained dense surface normal estimation. In: ECCV. pp. 468–484 (2014)
19. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3DV. pp. 239–248 (2016)
20. Lee, C., Badrinarayanan, V., Malisiewicz, T., Rabinovich, A.: Roomnet: End-to-end room layout estimation. In: ICCV. pp. 4875–4884 (2017)
21. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: CVPR. pp. 2136–2143 (2009)
22. Liu, C., Yang, J., Ceylan, D., Yumer, E., Furukawa, Y.: Planenet: Piece-wise planar reconstruction from a single rgb image. In: CVPR (2018)

23. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. pp. 3431–3440 (2015)
24. Magri, L., Fusiello, A.: Multiple models fitting as a set coverage problem. In: CVPR. pp. 3318–3326 (2016)
25. Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR. pp. 4040–4048 (2016)
26. Micusík, B., Wildenauer, H., Kosecka, J.: Detection and matching of rectilinear structures. In: CVPR (2008)
27. Micusík, B., Wildenauer, H., Vincze, M.: Towards detection of orthogonal planes in monocular images of indoor environments. In: ICRA. pp. 999–1004 (2008)
28. Ramalingam, S., Pillai, J.K., Jain, A., Taguchi, Y.: Manhattan junction catalogue for spatial reasoning of indoor scenes. In: CVPR. pp. 3065–3072 (2013)
29. Ros, G., Sellart, L., Materzynska, J., Vázquez, D., López, A.M.: The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: CVPR. pp. 3234–3243 (2016)
30. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 824–840 (2009)
31. Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. In: ECCV. pp. 537–547 (2008)
32. Witkin, A.P., Tenenbaum, J.M.: On the role of structure in vision. In: Beck, J., Hope, B., Rosenfeld, A. (eds.) Human and Machine Vision, pp. 481–543. Academic Press (1983)
33. Xiao, J., Russell, B.C., Torralba, A.: Localizing 3d cuboids in single-view images. In: NIPS. pp. 755–763 (2012)
34. Yang, H., Zhang, H.: Efficient 3D room shape recovery from a single panorama. In: CVPR (2016)