Deep Recognition of Vanishing-Point-Constrained Building Planes in Urban Street Views

Zhiliang Zeng, Mengyang Wu, Wei Zeng, *Member, IEEE* and Chi-Wing Fu, *Member, IEEE*

***Abstract*—This paper presents a new approach to recognizing vanishing-point-constrained building planes from a single image of street view. We first design a novel convolutional neural network (CNN) architecture that generates geometric segmentation of per-pixel orientations from a single street-view image. The network combines two-stream features of general visual cues and surface normals in gated convolution layers, and employs a deeply supervised loss that encapsulates multi-scale convolutional features. Our experiments on a new benchmark with fine-grained plane segmentations of real-world street views show that our network outperforms state-of-the-arts methods of both semantic and geometric segmentation. The pixel-wise segmentation exhibits coarse boundaries and discontinuities. We then propose to rectify the pixel-wise segmentation into perspectively-projected quads based on spatial proximity between the segmentation masks and exterior line segments detected through an image processing. We demonstrate how the results can be utilized to perspectively overlay images and icons on building planes in input photos, and provide visual cues for various applications.**

***Index Terms*—Image segmentation, plane reconstruction, augmented reality, geometric reasoning, vanishing point, street view**

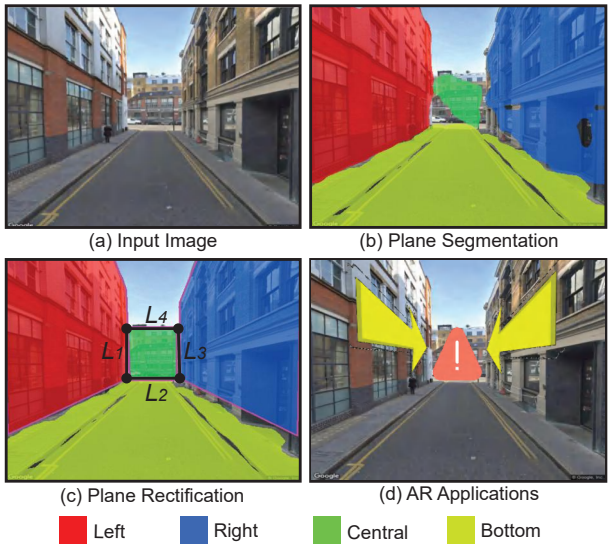


Fig. 1. From a single image of street view (a), we predict pixel-wise plane segmentations (b) and rectify them into perspectively-projected quads (c) that are more suitable for overlaying images/icons (d).

I. Introduction

MANY applications and games on smartphones and see-through glasses have been developed to enrich our views of the physical world. Typically, they provide context-aware information through a virtual overlay on the real scene. Yet, it remains challenging to naturally incorporate virtual objects in views of the physical world, since object placements in reality are governed by physical rules [35]. For instance, without considering the rules, a virtual cup would appear to float in mid-air instead of resting on a table. To improve the virtual object incorporation, one common approach is to align virtual objects with planar surfaces [25], [35], which however, are not readily available in reality.

To recognize planes, one can register the scene with prior knowledge, e.g., physical markers. However, the approach is clearly incompetent for supporting city-scale image overlay applications. In comparison, vision-based tracking techniques are more ubiquitous and practically feasible for general scenarios [54]. Thus, recent tools are mostly devoted to recognizing planes in camera views without explicit markers.

Conventional image processing and understanding methods [21], [22], [41], [18] make use of handcrafted visual cues to recognize planes, *e.g.*, color, texture, and gradient. With the success of deep learning, great advancements have been achieved that can directly extract features from imagery data to infer per-pixel geometric properties. However, due to the lack of plane annotations, recent network-based methods [45], [30], [29], [52] are supervised by depth map, then further infer and refine plane geometry based on the depth predictions. Clearly, acquiring a noise-free depth map is a difficult task, especially in outdoor environments like street views.

To overcome limitations of the depth-map methods, we propose a new approach as depicted in Fig. 1. Given an input street-view image (*e.g.*, Fig. 1(a)), we first roughly locate planar regions for buildings and ground in the image view; see Fig. 1(b). Here, we color-code each region according to its orientation in respective to camera position, e.g., blue for the right- and yellow for bottom-oriented, etc. Just like the outputs from many existing neural networks for image segmentation, the extracted regions often exhibit coarse boundaries and discontinuities, thereby insufficient for supporting image overlay applications, which expect rectified planar surfaces [15], [14]. As illustrated in Fig. 1(c), the geometry of each (perspectively-projected) building and ground region can be approximated by a quad region in the image space. For the case of building regions, the quad would possess a pair of vertical lines and a pair of nearly-horizontal lines that meet at a vanishing point. By rectifying a quad and obtaining the associated projection, we can then arrange virtual objects and texture images as an overlay on the corresponding buildings; see Fig. 1(d).

Z. Zeng, M. Wu, and C.-W. Fu are with the Chinese University of Hong Kong. e-mail: {zlzeng,mywu,cwfu}@cse.cuhk.edu.hk.

W. Zeng is with SIAT, Chinese Academy of Sciences. W. Zeng is the corresponding author. e-mail: wei.zeng@siat.ac.cn.

Manuscript received October 17, 2019; revised February 21, 2020.

Yet, to achieve the goal is non-trivial. First, urban scenes often have variant lighting conditions and clutter objects. Geometry-based approaches relying on geometric primitives (*e.g.*, line segments [6], [16] and vanishing points [40], [46]) are prone to fail for occlusions such as pedestrians and trees. Second, outdoor scenes typically exhibit much more complex geometry than indoor ones. Geometric reasoning methods for inferring layouts of indoor scenes, which assume simple geometric properties including corner connections [28] and boxy configurations [32], [13], [27], are not applicable hereof.

Instead, we propose to rectify coarse plane regions by exploiting their associations with line segments. Our main insight is that plane regions in an urban scene are mostly artifacts such as grounds and facades, which embrace a wide variety of straight lines like edges of windows and roads. At the core of our method, we first design a novel plane segmentation network that predicts pixel-wise plane orientations in respective to camera position. To improve segmentation accuracy, we leverage gated convolutional layers that integrate two-stream features of general visual cues and surface normals, and a deeply supervised loss that encapsulates multi-scale convolutional features. Next, we infer correlation between line segments and plane segmentation masks based on their spatial proximity. This filters out line segments irrelevant to plane regions, and limits the feasible sets of line segments to these which share the same orientation label. Lastly, we infer one horizontal and one vertical vanishing point for each set of line segments, by which we can deduce four intersecting lines forming a perspectively-projected quad.

The main contributions of our work are:

- We build a new benchmark of street-view images with fine-grained orientation labels from three different metropolises in the world (Sec. IV-A).
- We design a novel CNN architecture with gating mechanism and deep supervision to improve plane segmentation (Sec. III-B). Experiment results reveal comparable advancements over state-of-the-art networks for semantic and geometric segmentations (Sec. IV-B).
- We propose to rectify coarse plane segmentations into quads based on their spatial proximity with line segments (Sec. III-C), and demonstrate the use of these rectified quads for overlaying virtual objects on building planes in urban street views (Sec. V).

II. Related Work

Augmented reality (AR) aims to seamlessly interweave virtual contents with the physical world [25], [26]. One feasible solution is to spatially align virtual objects with planes in real environments [35], [15]. This requires not only precise plane regions, but also their geometric properties such as orientation and perspectives. For instance, to overlay arrow icons on ground for navigation, we need a quad plane beneath the camera, and its heading to guide the arrow direction. To obtain the information, a conventional approach is to register the scene with a database of the reality (*e.g.*, markers, 3D reference models) using key features, such as the prominent lines [38], building silhouettes [24], symmetry and repetition patterns [14]. Such a database, however, is often not readily available. Therefore, plane recovery by image understanding without prior is regarded as a more feasible approach [54].

This work is related to augmented reality, but focuses mainly on detecting quad planes in single-view images that are typically urban street-view photos. The detected planes can be utilized to facilitate real and virtual objects combinations, serving as a component in general AR systems [3]. Particularly, to handle street-view photos, the challenges include the occlusions and complex geometry in outdoor urban scenes. We hereby develop a new framework that first utilizes advanced CNN techniques of gating mechanism and deep supervision to predict plane regions, and further rectifies the regions based on their spatial proximity with line segments.

Image segmentation infers scene context by dividing an image into regions, each corresponding to a semantic object (*e.g.*, vehicle, pedestrian) or a geometric context (*e.g.*, orientation to camera). Overall, the problem can be modeled as a multi-class classification task, which can be addressed by a flat classifier (*e.g.*, Boosting [44], Random Forests [43]) using hand-crafted features. Performance relies on the expressiveness of the features, which however, are unlikely robust for complex urban scenes. Benefiting from the advancements of deep CNN-based models (*e.g.*, FCN [42], DeconvNet [34], SegNet [4], and DeepLab [8]), and fine-labeled datasets (e.g., [5], [2], [10]), recent researches on image segmentation of urban scenes have reached a new level in terms of accuracy and generality.

For plane segmentation, conventional methods [21], [22], [41], [18] relied on general visual cues, *e.g.*, textures, colors, and gradients. Recently, end-to-end CNNs [45], [29], [30], [52] improved the performance by formulating plane segmentation as a supervised depth prediction problem. The prior studies inspired us that both general visual cues and surface normals (closely related to depth information [37]) can be utilized to infer plane regions. We thus encapsulate the two-stream features using a gating mechanism, which has shown to be effective for various image segmentation tasks [36], [48]. Furthermore, we enrich convolutional features [31] by integrating hierarchical feature maps into the loss function. Experimental results reveal comparable advancements over the latest networks.

Geometric reasoning methods make use of geometric properties of objects and relationships between objects to recover the surface layout of a scene [17]. Unlike pixel-wise image segmentation, the methods can generate well-structured quads that are preferable for image overlay applications. A well-known example is the 'Manhattan world,' which assumes that planes of an artificial scene lie in one of the three mutually orthogonal orientations [11]. This assumption provides a basis for researches on indoor surface layout estimation by delineating a box model [20], [28], [32], [13], [27]. Nevertheless, planes in outdoor scenes have more complex layouts, thus limiting the applications of this simple boxy configuration. Some others are devoted to analyzing the geometry of outdoor scenes, *e.g.*, [33], [17], [51]. A typical example is [33]. The method first localizes line segments through Canny edge detector, then applies Markov Random Field to construct a graph structure to group the line segments, and finally parses

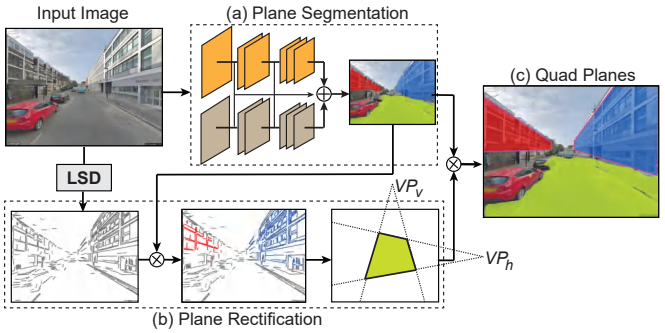


Fig. 2. An overview of our approach. (a) The plane segmentation network boosts the segmentation performance using the gated feature maps of both the general visual cues and surface normals. LSD denotes line segment detector. (b) The plane rectification module refines the coarse plane segmentations and produces the vanishing-point-constrained quads (VP denotes vanishing point). (c) The left-, right- and central-oriented quads (color-coded in red, green, and blue, respectively) of building planes are the final results.

rectangles using a max-sum solver. The method is effective for inferring quad planes in uncluttered environments. However, the max-sum solver is prone to converge to a local optimum when excessive (irrelevant) line segments from non-planar objects are localized. These outliers can largely slow down the graph construction, as the method needs to searches over the entire space of all line segment pairs. To work around the issue, we exploit the associations between line segments and plane segmentations, by which we formulate a robust and efficient line segment grouping strategy.

III. Method

Given a monocular RGB image $I \in \mathbb{R}^{3\times W\times H}$ with dimensions $W \times H$ as input, we predict a set of quad planes $\mathcal{S} = \{S_1, ..., S_n\}$. Each S_i can be specified as a 5-tuple: $S_i = \{L_1^i, L_2^i, L_3^i, L_4^i, o^i\}$, where L_j^i denotes the j^{th} line segment in the image space that forms S_i, and $o^i \in O$ denotes the orientation of S_i in the camera's perspective (see Sec. IV-A for details). We employ a two-point perspective with a horizontal vanishing point VP_h^i and a vertical vanishing point VP_v^i to jointly determine the four lines in the quad (Fig. 2 (bottom)). Note that VP_h^i or VP_v^i may locate at infinity.

A. Overview

Our method has the following twofold modules (see Fig. 2):

Plane Segmentation (Sec. III-B). We assume that a street-view image consists of finite planes, i.e., piecewise constant regions. Thus, we can employ an end-to-end CNN to generate a piecewise plane segmentation of the image. The network takes I as input, and predicts a segmentation mask $M \in \mathbb{R}^{W\times H}$, where $M(i)$ denotes the predicted orientation of the i-th pixel of I. Specifically, we employ features of both general visual cues and surface normals, and encapsulate multi-scale adaptive features to boost the network training (Fig. 3).

Plane Rectification (Sec. III-C). We design this module to rectify the coarse pixel-wise plane segmentation and produce quad planes, for better supporting image overlay applications. We first employ a conventional line segment detector (LSD) [16] to identify the line segments in I. Next, we utilize the spatial proximity between regions in the segmentation mask M and the detected line segments to group the line segments. Finally, we deduce VP_h and VP_v, and reconstruct four intersecting line segments for each quad plane.

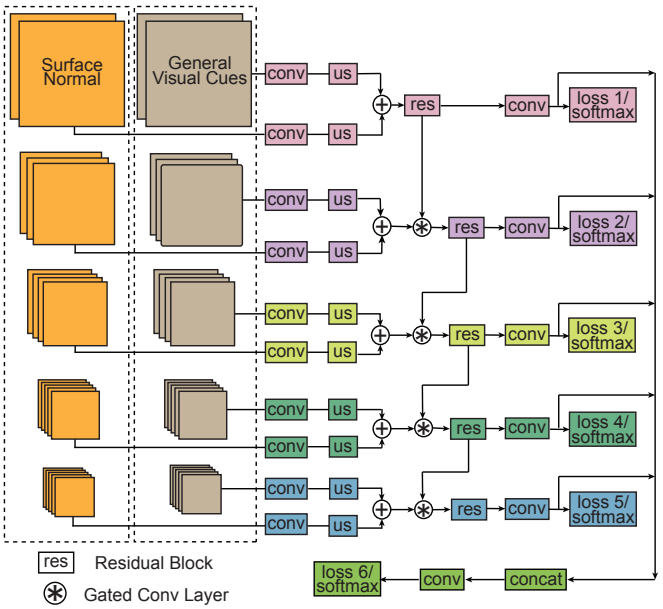


Fig. 3. Our network architecture encodes features of the surface normals and general visual cues. The two-stream features are processed through a set of residual blocks and gated convolutional layers. In the end, the multi-scale feature maps are encapsulated into a deeply-supervised loss.

B. Plane Segmentation

We develop a new network architecture for segmenting a street-view image into quad planes. Fig. 3 depicts the whole network architecture. Overall, the network encodes two-stream features of both the general visual cues and surface normals through a series of gated convolutional layers. In the end, we encapsulate the features from all the stages into a deeply-supervised loss, so as to efficiently capture both the coarse features at high-stage and the fine details at low-stage.

1) Two-Stream Feature Encoder: Prior studies have formulated plane segmentation as a depth map prediction problem [45], [29], [30], [52]. However, depth map is usually noisy and difficult to acquire for real street-view images. On the other hand, other researches [21], [20], [18] have shown that general visual cues (*e.g.*, colors and textures) can be utilized to recognize planes in a single image. Thus, we propose to aggregate the features of surface normals and general visual cues to improve the plane segmentation performance.

Here, we adopt the GeoNet [37], a state-of-the-art network for surface normal prediction, to generate a surface normal feature map. Also, we use the VGG16 network [47] to encode general visual cues. Both feature extraction streams have five stages of convolutional layers; see Fig. 3(a). At each stage, the feature map size is halved, while the depth is doubled.

2) Gated Convolutional Layer: At the core of the network, we design a series of gated convolutional (GC) layers to facilitate the plane segmentation. Here, we denote the feature map of the surface normals as $F_t^n \in \mathbb{R}^{D_t\times W_t\times H_t}$ and the feature

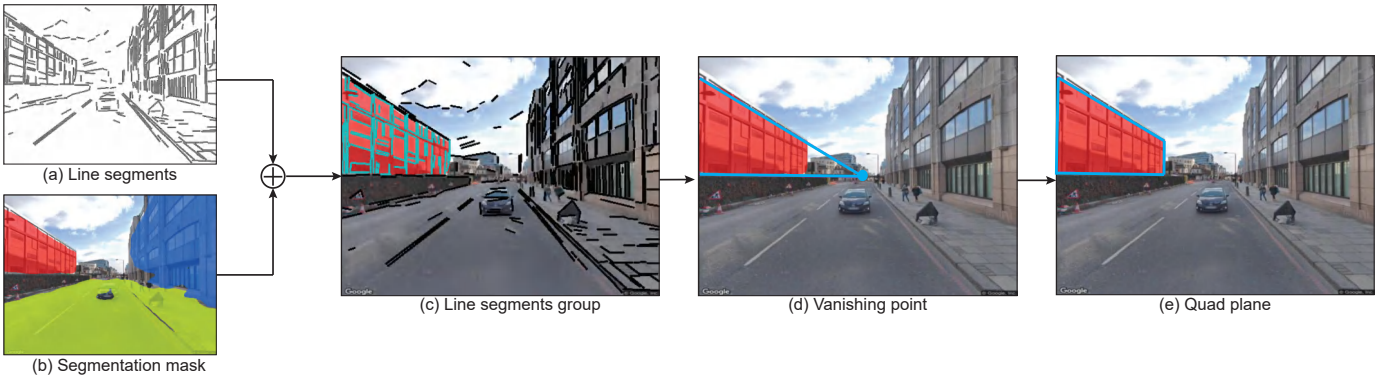


Fig. 4. Procedure of plane rectification. The module takes (a) line segments and (b) plane segmentation mask as inputs. (c) The line segments are partitioned into groups depending on their spatial proximities with the segmentation masks. (d) For each group of line segments, a vertical vanishing point and a horizontal vanishing point can be identified. Notice here the vertical vanishing point locates at infinity. (e) Four intersecting line segments, which form a quad plane as the final result, can be identified using two-point perspective of the vanishing points.

map of the general visual cues as $F_t^v \in \mathbb{R}^{D_t \times W_t \times H_t}$ at stage $t \in \{1,2,3,4,5\}$, where D_t, W_t, and H_t indicate the depth, width, and height of the feature maps at stage t, respectively.

Overall, the gating mechanism is designed for the following two purposes: First, since the surface normals and the general visual cues are closely related, we need to aggregate their features and generate a fused feature map F_t^f, combining the strengths of F_t^n and F_t^v. Second, since not all features in F_t^f are helpful for plane segmentation, we compute an attention map from stage t to supervise the learning at stage $t+1$.

The fusion function can be summarized as follows.

$$F_t^f = (F_t^n \otimes w_{1*1}) \oplus (F_t^v \otimes w_{1*1}), \tag{1}$$

where $\otimes$ and $\oplus$ stands for the convolution operator and element-wise sum operator, respectively. w_{1*1} represents a $1*1$ convolutional kernel. We employ a bilinear interpolation to upsample F_t^n and F_t^v to $\mathbb{R}^{H \times W}$, then we can combine the two features by using an element-wise sum operator.

After fusion, we compute an attention map $\alpha_t \in \mathbb{R}^{H \times W}$ using

$$\alpha_t = \sigma(Res(F_t^f)), \tag{2}$$

where $Res(\cdot)$ denotes a residual convolutional block and $\sigma(\cdot)$ denotes the sigmoid function. Intuitively, α_t can be seen as an attention map that marks the important features of higher weights. Next, we apply the attention map α_t at stage t to the gated feature map at stage $t+1$ as

$$\hat{F}_{t+1}^f = ((\alpha_t \odot F_{t+1}^f) \oplus F_{t+1}^f) \otimes w_{t+1}, \tag{3}$$

where $\odot$ stands for an element-wise product operator, and w_{t+1} represents the channel-wise weighting kernel at stage $t+1$. $\hat{F}_{t+1}^f$ is passed to Eqn. (2) to compute the attention map α_{t+1} at stage $t+1$, then α_{t+1} is passed on to the next stage $t+2$. This procedure is repeated until the final stage.

3) Deeply-Supervised Loss: We encapsulate the multi-scale convolutional features into discriminative representations, so as to enhance the performance with fewer convolutional layers. Here, we apply a $1*1$ convolutional kernel to the gated feature map $\hat{F}_t^f$, yielding an activation map X_t for each stage t. In the end, we further concatenate X_t for all stages $t \in \{1,2,3,4,5\}$, and apply a convolution layer to create a fused activation map X_{fuse}. By this, all X_t and X_{fuse} share the same dimensions as the ground-truth plane instance Y. Thus, we formulate a deeply-supervised loss function as

$$\mathcal{L} = \mathcal{C}e(X_{fuse}, Y) + \sum_{t=1}^{5} \mathcal{C}e(X_t, Y), \tag{4}$$

where $\mathcal{C}e(\cdot)$ stands for a general softmax cross entropy function. We employ $\mathcal{L}$ to guide the network training process.

C. Plane Rectification

The gated network produces the coarse segmentation mask $M \in \mathbb{R}^{W \times H}$. To better support image overlay generation, we further rectify the pixel-wise segmentations into quad planes. As depicted in Fig. 4, the module takes line segments (denoted as L) detected by a line segment detector (LSD) [16], together with the segmentation mask M as inputs. For each line $l \in L$, we first dilate line l with a $k \times k$ kernel (here, we use 5), yielding a bag of pixels denoted as P^l. Then, we can measure the correlation between l and orientation class $o_i \in O$ by

$$corr(l, o_i) = \frac{\sum_{j=1}^{|P^l|} \mathbb{1}(M(P_j^l) = o_i)}{|P^l|}, \tag{5}$$

where j runs over all pixels in P^l, and $\mathbb{1}$ is an indicator function. Based on the correlation $corr(l, o_i)$, we can then identify the following three scenarios of associating l and o_i.

1) If $corr(l, o_i) \geq thre_{up}$, we consider that l is inside a plane of orientation o_i. For instance, line segments of windows, doors, and billboards are contained within the quad plane of a building facade. These line segments are useful for inferring the plane's perspectives, but are not helpful for identifying the plane's boundaries.
2) If $thre_{up} > corr(l, o_i) \geq thre_{low}$, we consider that l lies in the boundary of a coarse plane of orientation o_i. Such scenario typically happens to line segments of a building roof or footprint. These line segments are informative for both the plane's perspectives and boundaries.
3) If $corr(l, o_i) < thre_{low}$, we consider that l is not associated with any plane of orientation o_i. Such line segments are ignored when rectifying planes of orientation o_i.

where $thre_{up}$ and $thre_{low}$ denote the upper and lower thresholds, respectively, and are set to 0.7 and 0.3, respectively, in our implementation. In this way, we can find a set of line

segments for each orientation o_i. Further, we employ spatial proximity to divide them into groups $\mathcal{L}^1$,...,$\mathcal{L}^n$, corresponding to plane instance S_1,...,S_n. For each group of line segments $\mathcal{L}^i$, we use a voting-based approach proposed by [46] to estimate the vanishing points (Fig. 4(d)): (i) summarize the orientation histogram of $\mathcal{L}^i$, (ii) identify the vertical (zenith) and horizon lines, and (iii) find the vertical VP_v^i and horizontal VP_h^i accordingly. Next, we connect VP_h^i with the top- and bottom-most pixels of S_i, and VP_v^i with the left- and right-most pixels of S_i. The operations yield four intersection line segments $L_1^i, L_2^i, L_3^i, L_4^i$ (Fig. 4(e)). In case VP_h^i locates at infinity, the two horizontal lines are in parallel; whilst if VP_v^i locates at infinity, the two vertical lines are in parallel. By then, we can rectify the coarse plane instance S_i into a quad plane with orientation o_i, *i.e.*, $\{L_1^i, L_2^i, L_3^i, L_4^i, o^i\}$.

IV. Experiments and Results

A. Dataset and Implementation

1) Our Dataset: Our network focuses on plane segmentations of outdoor street views. To improve the robustness of the trained network model, we consider the following three requirements for compiling our dataset: **R1**) The street views should be taken from various cities, such that the trained network model can be more general; **R2**) The street views should consist of both the street and side views, such that in the future, we can simulate user behaviors of looking around when using AR applications on the streets and pavements; and **R3**) The detected quad-plane regions should be suitable for placing virtual objects, *i.e.*, quad-plane regions of small areas and faraway from the camera should be omitted.

To meet requirement **R1**, we opt to select three major cities in different parts of the world, *i.e.*, New York, Hong Kong, and City of London. Example street views are presented in the first row of Fig. 5. From the figure, we can observe the distinct landscapes of these cities, such as, the buildings in New York and Hong Kong are in general taller than those in London. For each city, we first randomly sample 500 positions in its central area. Second, for each sample position, we extract its geographic information (latitude *lat* & longitude *lon*) and identify the street heading (h). Last, we crawl a view at each sample position by passing *lat*, *lon*, h + θ, with image size 480 (W)×360 (H) into the Google Street View API [1]. Here, θ is a random value in the range of [0^o, 180^o], where 0^o and 180^o are street views along the vehicle roadway, whilst 90^o are street views perpendicular to the roadway and looking towards to the side (**R2**). Other parameters, including the field of view and pitch, are set to their default values according to the API. Specifically, the default value of pitch is set to 0, meaning that the camera's view direction is simply horizontal.

By this, we crawl a total of 1,500 (500×3) raw street-view images for the three cities. Next, we manually label the quad plane regions on each image using the LabelMe toolbox [49]. Each plane region is labeled as an orientation label of left (LT), right (RT), central (CT), and bottom (BT). Thus, the set of orientation classes O in this work is specified as $O = \{LT, RT, CT, BT\}$. We shall note that many pixels are not assigned with any label. The second row in Fig. 5 presents the labeling results. Notice that some ground floors of corridors or transparent glasses are not labeled, as we consider they are not suitable for placing virtual objects.

TABLE I
Quantitative comparisons with two semantic segmentation networks (LDN, HK, and NYC denote the City of London, Hong Kong, and New York City, respectively).

		DeepLabV3+	PSPNet	Ours
pixel accuracy	LDN	0.819	0.779	**0.822**
	HK	0.810	0.749	**0.811**
	NYC	0.796	0.746	**0.804**
	Overall	0.809	0.758	**0.813**
mean accuracy	LDN	0.844	0.799	**0.853**
	HK	0.757	0.736	**0.820**
	NYC	0.751	0.721	**0.795**
	Overall	0.787	0.757	**0.828**
mean IoU	LDN	0.695	0.630	**0.706**
	HK	0.677	0.589	**0.696**
	NYC	0.639	0.558	**0.663**
	Overall	0.673	0.596	**0.691**

2) Implementation: For each city, we randomly select 400 street views for network training, and take the remaining 100 for testing. Also, we employ a data augmentation method to enlarge the training set: we scale up the training images to resolution 150% & 200%, then randomly crop a 480×360 region on each scaled image. Hence, we generate in total 3,600 (400 × 3 cities × 3 scales) images as the training data. We trained our network on a workstation with an NVidia Titan Xp GPU card. The training process ran in total 100k iterations. The momentum optimizer with a polynomial decay learning rate starting at *1e-3* was employed to update the parameters. We employed a small batch size of two together with group normalization to improve the training accuracy [50].

B. Results of Plane Segmentation Network

Since the problem is modeled as a piecewise segmentation task, we first compare the results with two recent neural networks for semantic segmentation tasks, *i.e.*, PSPNet [53] and DeepLabV3+ [9]. Besides, we compare the results with two latest neural networks specifically designed for plane segmentations, *i.e.*, PlaneRecover [52] and PlaneRCNN [29].

1) Comparisons with Semantic Segmentation Networks: First, we trained the PSPNet [53] and DeepLabV3+ [9] models on our training dataset using hyper-parameters proposed in the original papers. For each network training, we checked the model performance every two training epochs, and chose the one with the best segmentation results as the final model. Last, we tested the final models on the testing datasets. We employed three widely-used metrics for pixel-wise segmentation evaluation, i.e., *pixel accuracy*, *mean accuracy*, and *mean intersection over union (IoU)*. Let n_{ij} be the number of pixels of orientation i predicted to have orientation j, and let $t_i = \sum_j n_{ij}$ be the total number of pixels of orientation i. The three metrics can be written as

- pixel accuracy: $\sum_i^{|O|} n_{ii} \;/\; \sum_i^{|O|} t_i$;
- mean accuracy: $(1/|O|)\sum_i^{|O|} n_{ii}/t_i$; and
- mean IoU: $(1/|O|)\sum_i^{|O|} n_{ii}/(t_i + \sum_j^{|O|} n_{ji} - n_{ii})$.

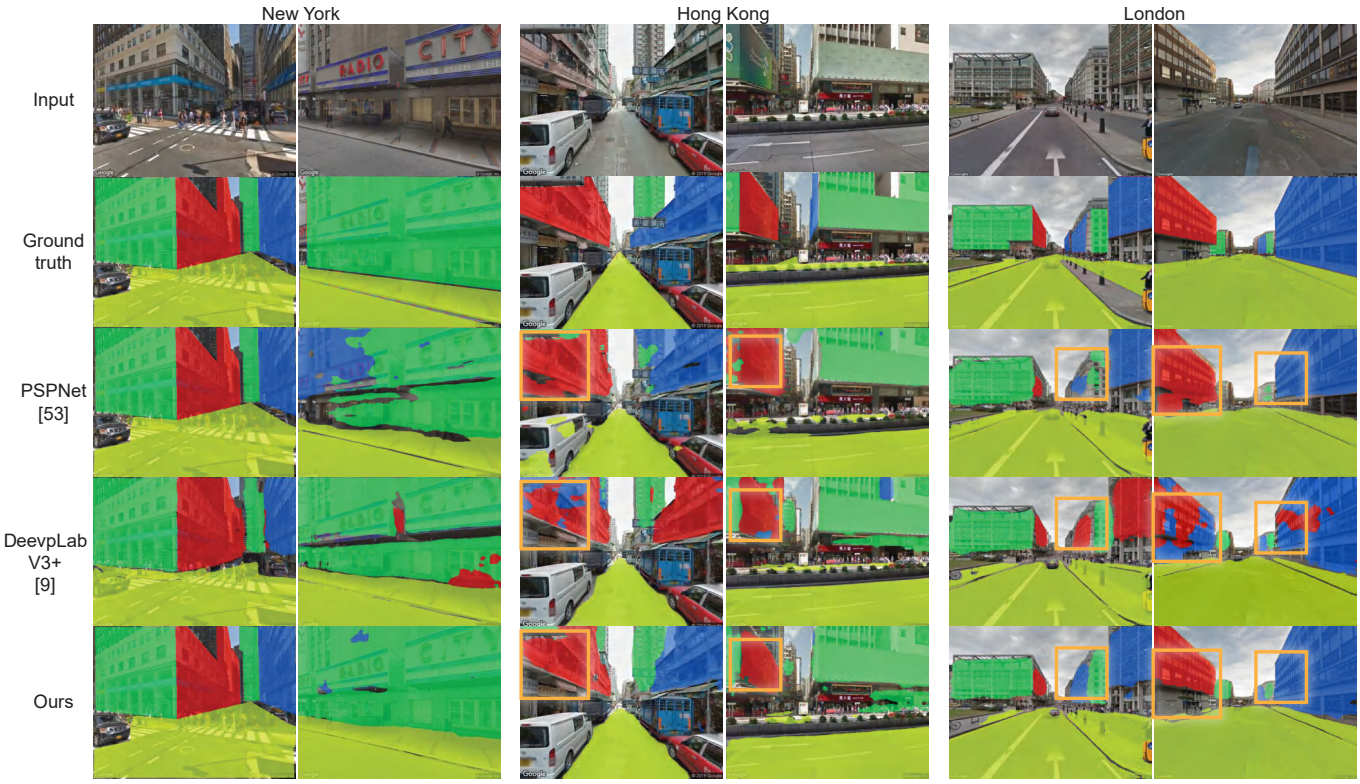


Fig. 5. Qualitative comparisons with general segmentation networks. From top to bottom: input images, ground truths, results of PSPNet [53], results of DeepLabV3+ [9], and our results. From left to right: New York×2, Hong Kong×2, and City of London×2.

Table I shows the quantitative comparison results with the two semantic segmentation networks. Our network outperforms both PSPNet [53] and DeepLabV3+ [9] for all the three metrics. This is probably because the semantic segmentation CNNs employ only the general visual features for segmentation, while our network utilizes a gated fusion module to consider both the general visual cues and surface normal features. Besides, we can notice that improvement of our model over PSPNet [53] is much more than that over DeepLabV3+ [9]. This is probably because DeepLabV3+ [9] utilizes deeper layers than PSPNet [53] and ours.

Fig. 5 shows qualitative comparisons of predictions for street views of New York, Hong Kong, and London. Overall, our network generates better segmentations than others. Specifically, we can notice that our results contain fewer holes, especially in comparison with DeepLabV3+ [9]. The small holes may not affect much the quantitative metrics presented in Table I. Yet, they can greatly affect the final results by the plane rectification module; see Sec. IV-C for more discussions.

2) Comparisons with Plane Segmentation Networks: Both PlaneRecover [52] and PlaneRCNN [29] rely on the depth information for pixel-wise plane segmentation. Specifically, PlaneRecover [52] is essentially a depth prediction network that relies heavily on high-quality depth maps to supervise the network training. Unfortunately, our dataset is not coupled with such information. Hence, we experimented to predict the depth directly from the input images. The results predicted by struct2depth [7], a state-of-the-art neural network for depth prediction are, however, incompetent. On the other hand, PlaneRCNN [29] requires the intrinsic camera parameters for estimating the plane offset, which is again absent in our dataset. So, we opt to reuse the pre-trained models of PlaneRecover [52] on SYNTHIA [39], PlaneRCNN [29] on ScanNet [12], and ours on the training dataset. For a fair comparison, we randomly selected 50 images from three unseen datasets of SYNTHIA [39] (except for PlaneRecover [52]), KITTI [2], and CityScape [10]. The input sizes are set to resolution 480×360 for consistency.

Plane segmentations by our model are coupled with orientation information, while PlaneRecover [52] and PlaneRCNN [29] predict only distinct plane regions. The metrics used for comparing semantic segmentation networks would be incompatible hereof. Thus, we simplify plane segmentations into only two classes of *plane* and *non-plane*. All pixels in predicted plane regions are *plane*, whilst the others are *non-plane*. We can then compare the segmentation accuracy using the F-measure [23] that is commonly employed for evaluating binary classification. The metric is expressed as

$$F_\beta = \frac{(1+\beta^2)Precision \times Recall}{\beta^2 Precision + Recall}, \tag{6}$$

where β^2 is set to 0.3, which follows the setting in [23]. *Precision* is the ratio of true-positive plane pixels over all plane pixels, while *recall* is the ratio of true-positive plane pixels over ground-truth plane pixels.

Table II shows the comparison results. Our network achieves the best performance on both KITTI [2] and CityScape [10] datasets, which are unseen to all three networks; PlaneRecover [52] produces better results on SYNTHIA [39], as the model was trained and tested on the same dataset.

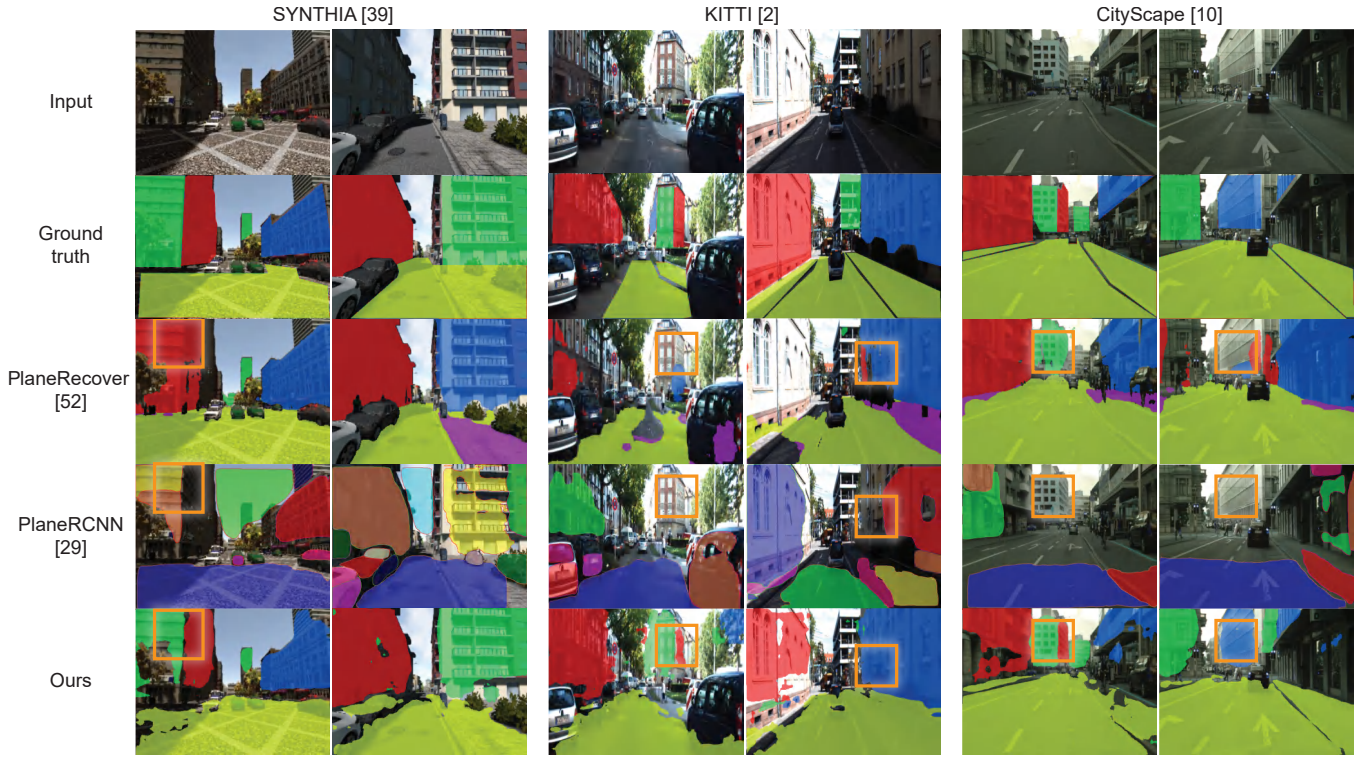


Fig. 6. Qualitative comparisons with prior plane recovery networks using unseen testing datasets from SYNTHIA [39]×2, KITTI [2]×2, and CityScape [10]×2 (from left to right). From top to bottom: input images, ground truths, results of PlaneRecover [52], results of PlaneRCNN [29], and results of ours.

TABLE II
QUANTITATIVE COMPARISONS WITH PLANE SEGMENTATION NETWORKS. NOTE, FOR THE REASONS DISCUSSED IN SEC. IV-B2, WE HAVE TO USE THE PRE-TRAINED MODELS OF PLANERECOVER [52] ON SYNTHIA [39], PLANERCNN [29] ON SCANNET [12], AND OURS ON OUR OWN TRAINING DATASET, AND TEST ON SYNTHIA [39], CITYSCAPE [10], AND KITTI [2].

		PlaneRecover	PlaneRCNN	Ours
F_β	SYNTHIA	**0.8863**	0.7543	0.8223
	CityScape	0.8569	0.7460	**0.8729**
	KITTI	0.7376	0.6696	**0.7798**

Fig. 6 presents some examples in the qualitative comparison results. Note, PlaneRCNN [29] could wrongly recognize skies in KITTI [2] as planes. This is probably because PlaneRCNN [29] is built upon Mask R-CNN [19], which means that PlaneRCNN [29] relies more on local texture features for plane recognition. PlaneRecover [52] produces better results, as it considers more geometric information, *e.g.*, relation between depth and normal. However, the network performance drops much in the KITTI [2] and CityScape [10] datasets, as depth prediction is challenging in complex scenes of urban streets. On the other hand, our method utilizes both geometric features of surface normals and general visual cues, thus the predictions outperform prior methods in complex urban street views.

C. Ablation Analysis

We evaluate the contribution of individual components in our network using the pixel-wise segmentation metrics presented in Sec. IV-B1. All the models in this ablation study were trained from scratch on the training dataset and evaluated on the test dataset using the same set of hyper-parameters. Specifically, we compare four ablated network models:

TABLE III
ABLATION ANALYSIS ON THE CONTRIBUTIONS OF INDIVIDUAL COMPONENTS IN OUR PLANE SEGMENTATION NETWORK.

	pixel accuracy	mean accuracy	mean IoU
Baseline	0.7970	0.7366	0.6466
+ Normal	0.7972	0.7789	0.6552
+ Normal + GC	0.8040	0.8026	0.6721
Ours (+ Normal + GC + DS)	**0.8130**	**0.8280**	**0.6910**

- *Baseline*: The comparison is performed with respect to a baseline network that is essentially a VGG-16 [47] model built on the general visual cues;
- *Baseline + Normal*: We enrich the feature map by simply concatenating two-stream features of general visual cues and surface normals;
- *Baseline + Normal + GC*: Instead of simple concatenation, we combine the features of general visual cues and surface normals using the gated convolutional (GC) layers (Sec. III-B2); and
- *Our Full Model (Baseline + Normal + GC + DS)*: Last, we further encapsulate multi-scale convolutional features using the deeply supervised loss (Sec. III-B3).

Table III presents the ablation study results, showing that all components have positive contributions. Specifically, we can notice that a simple concatenation of the general visual cues and surface normals generate marginal improvements. The improvements become larger, if we combine the two-stream features using gated convolutional layers. Compared to the baseline, our full model improves the overall pixel accuracy by 2.00%, mean pixel accuracy by 12.41%, and mean IoU by 6.87%. Fig. 7 shows some qualitative examples

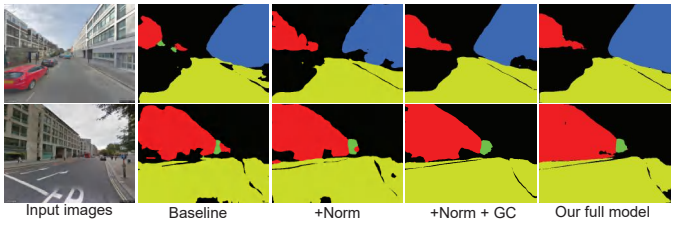


Fig. 7. Effects of individual component in the plane segmentation network.

TABLE IV
QUANTITATIVE COMPARISON OF PIXEL ACCURACY BEFORE AND AFTER THE PLANE RECTIFICATION. AGAIN, LDN, HK, AND NYC DENOTE THE CITY OF LONDON, HONG KONG, AND NEW YORK CITY, RESPECTIVELY.

	Type	Before Rectification	After Rectification
LDN	Left	0.812	**0.881**
	Right	0.791	**0.850**
	Central	0.815	**0.865**
	Overall	0.806	**0.865**
HK	Left	0.786	**0.794**
	Right	0.735	**0.792**
	Central	0.867	**0.893**
	Overall	0.796	**0.825**
NYC	Left	0.833	**0.857**
	Right	0.681	**0.707**
	Central	**0.753**	0.724
	Overall	0.756	**0.767**

of incrementally adding the four ablated components. Better results of more complete planar regions with less missing parts and holes can be generated, which is particularly important for the plane rectification module.

D. Results of Plane Rectification Module

Table IV presents the quantitative comparison results before (*i.e.*, solely by the plane segmentation network) and after the plane rectification (*i.e.*, our final results) in terms of pixel accuracy. In the camera views, we consider only left-, right-, and central-oriented building planes, which are suitable for placing virtual images or objects. From Table IV, we can see that the accuracy generally improves after rectifying planes in different orientations, indicating that our plane rectification module is robust. Such improvements are contributed by the filling of small holes and gaps that are hard to avoid for pixel-wise plane segmentation networks, as shown by the qualitative results in Fig. 8. Nevertheless, for New York, the accuracy is very low for right and central planes, and it even drops after rectification for central planes. After probing the results, we suspect this is probably because many street views in New York are taken at crossroads, making it difficult to classify the plane orientations; see Fig. 8 (bottom) for examples.

E. Runtime Analysis

Next, we present a runtime analysis of our method performed on a workstation with a single NVidia Titan Xp GPU and eight-core 2.90GHz Intel Xeon E5 CPUs. We consider input images in three different resolutions: 240×180, 360×270, and 480×360. For the plane rectification module, we further recorded the runtime of its sub-modules: line segment detection, line grouping, vanishing point detection, and rectification (see Fig. 4). The plane segmentation module is implemented

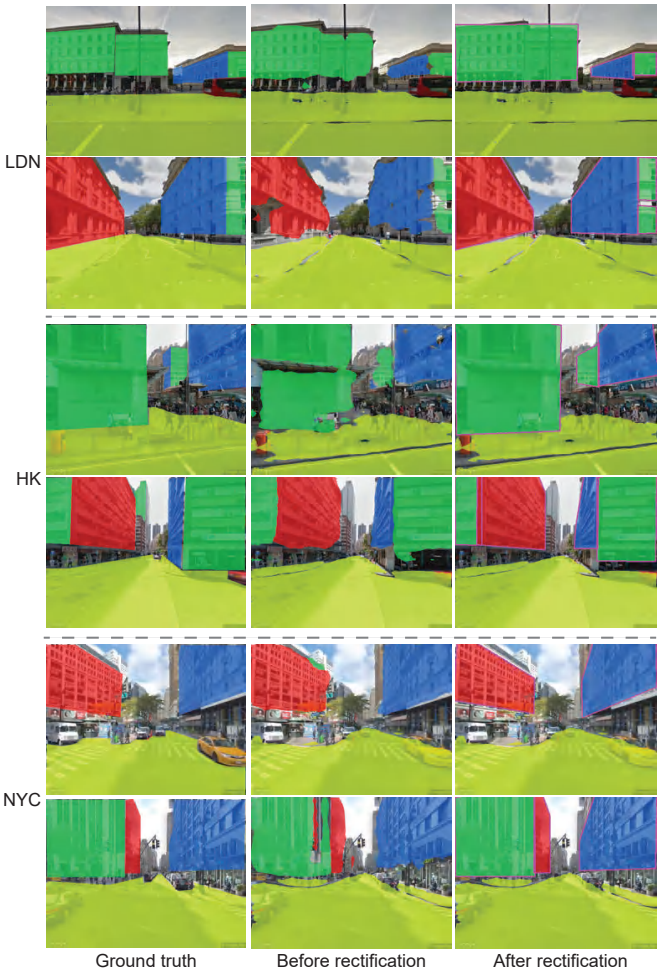


Fig. 8. Qualitative examples of plane rectification effects.

TABLE V
RUNNING TIME OF MAIN MODULES AND THEIR SUB-MODULES FOR INPUT IMAGES IN VARIOUS RESOLUTIONS.

Main Module	Sub-Module	240×180	360×270	480×360
Plane Segmentation	-	31 ms	53 ms	92 ms
Plane Rectification	Line Segment Detection	62 ms	68 ms	110 ms
	Line Grouping	27 ms	58 ms	91 ms
	Vanishing Point Detection	26 ms	45 ms	49 ms
	Rectification	16 ms	36 ms	68 ms
Overall	-	162 ms	260 ms	410 ms

using TensorFlow and runs on the GPU, whereas the plane rectification module is implemented in Python and runs on a single-core CPU, in our current implementation.

Table V reports the method runtime. Overall, it takes around 410 ms for regular 480×360 input images, and after looking into individual modules, we find that the plane rectification module takes up over 75% of the overall time. In the future, we will optimize it by harassing the multi-core CPUs, as the computations in sub-modules can be processed in parallel. Another promising direction is to reduce the image resolution, as we can see the overall execution time can drop to 260 ms for 360×270 input images and further to 162 ms for 240×180 input images. We can readily project quad planes detected on

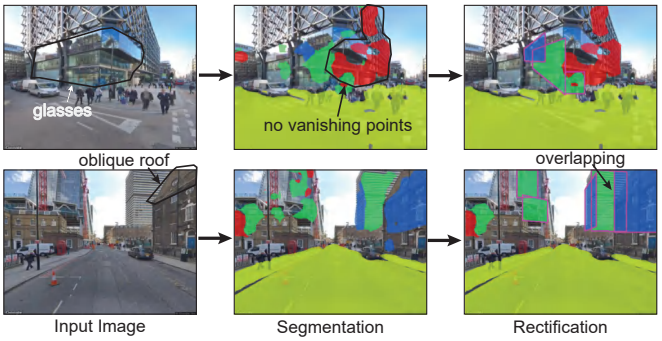


Fig. 9. Our method can fail for glasses (top) and oblique roofs (bottom).

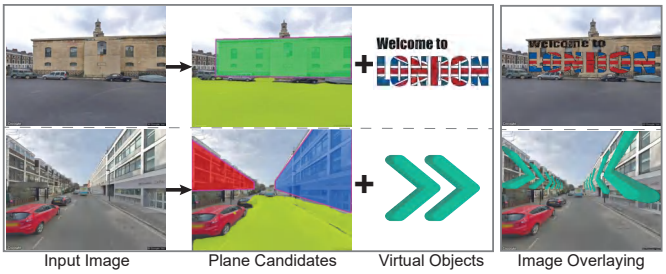


Fig. 10. The procedure of overlaying virtual contents on street-view images. After identifying plane candidates from a street-view image, we align suitable virtual objects on the planes based on their geometric properties.

low-resolution images to high-resolution images for the application for faster estimation of quad planes. Experimentally, we found that the overall pixel accuracy drops only by 12% for 360×270 inputs and 14% for 240×180 inputs.

F. Failure Cases

Our method is prone to produce inaccurate results due to glasses and oblique roofs in the street-view images, as demonstrated in Fig. 9. First, glass material may show illusive reflections (Fig. 9 (top-left)) that interfere the plane segmentation network. The predictions are in the cracked segments (Fig. 9 (top-middle)). In such case, the plane rectification module deduces no vanishing points, and fails to recover a quad plane (Fig. 9 (top-right)). Second, our current implementation omits the oblique roofs, as they are regarded as unsuitable for placing virtual objects, yet the roofs can occlude the building facades behind (Fig. 9 (bottom-left)). The roofs are planar, and our plane segmentation network may mis-predict their pixels as plane regions (Fig. 9 (bottom-middle)). Hence, the prediction results can further mislead the plane rectification module to generate overlapping quad planes (Fig. 9 (bottom-right)).

V. Applicability

In this section, we demonstrate the applicability of the recovered quad planes and their orientations in image overlay applications, and discuss the possible usage in the estimated poses. Figure 10 illustrates the procedure for overlaying virtual contents on real-world street-view images. In detail, we first identify the plane candidates through the proposed plane segmentation network and rectification module. Next, we search for suitable virtual objects in a pool, and overlay them on the plane. In the top, a single building plane orthogonal to the viewpoint is identified, which can be covered by a welcome flag. In the bottom, two building planes along the road are identified, which are suitable for put arrow icons to show the direction and support the navigation.

A key requirement here is to align the selected image or virtual object with the quad plane boundary and orientation. As discussed in Sec. III, each plane S_i is represented as a 5-tuple: $\{L_1^i, L_2^i, L_3^i, L_4^i, o^i\}$. The line segments define four intersections. On the other hand, the virtual objects utilized in this work are 2D images of visual context, which can also be specified by four vertices. We can then compute a homography matrix that maps the vertices of a virtual object to the intersections in the corresponding quad plane. Specifically for virtual objects with directional information, we need to consider the plane orientation as well. For instance, the arrows are flipped on the right building plane in Fig. 10 (bottom), so both arrows can point to the same direction. Also, notice that the building planes are long narrow rectangles, so we further cut them into parts to fit with the arrow marks.

The pool of virtual objects can be categorized into four main groups based on their usage scenarios, *i.e.*, entertainment, navigation, tourism, and greenery. Figure 11 presents more examples of image overlay applications in these usage scenarios. We choose street-view images from London, New York, and Hong Kong, as listed on the left, middle, and right columns of the figure, respectively. Further, we include a side view and a road view for each city. The seamless blending effect demonstrates the effectiveness of our detected quad planes.

VI. Conclusion and Future Work

This paper presents a new neural network architecture to segment a single street-view image into per-pixel orientations. In the network, we adopted a new gating mechanism to connect the general visual cues and surface normals, and formulated a holistic loss function that encapsulates multi-scale convolutional features and enables deeply-supervised network training. Our network outperforms competing methods on a newly-compiled benchmark with fine-grained plane annotations of outdoor street views collected in three metropolis.

Also, we rectified the plane segmentations into vanishing-point-constrained planes, and demonstrated the applicability of overlaying virtual contents on the detected quad planes to support various image overlay applications.

The applications demonstrate the potentials of our method for extension to handle outdoor AR applications and seamlessly blend virtual and real objects, as a building component in AR systems [3]. Nevertheless, well-designed AR systems further require real-time performance and various 3D pose estimation, which are not yet supported in our current implementation. In the future, we plan to improve the runtime performance, e.g., by designing lightweight networks and by multiprocessing implementations.

Also, the twofold processes are realized as two separate modules. Another direction is to encode the plane rectification module in the plane segmentation network, and design an end-to-end trainable network for the challenging yet rewarding task. This adaption could also improve the overall runtime.

Fig. 11. Demonstration of adding virtual contents to urban scene: images in the first row show the final results with quad plane regions; the rows below give examples of possible image overlay applications with different virtual contents, including entertainment, navigation, tourism, and greenery.

Furthermore, our implementation detects line segments and employs them to rectify building planes. As demonstrated in [33], the quad planes recovered by our work can be utilized in a series of applications, such as wide baseline stereo matching and planar 3D reconstruction. Nevertheless, empirical studies (*e.g.*, [33], [28]) on recovering rectilinear structures in single images often struggle with rectangle parsing, which requires the processing of pairwise graph relations among a large set of line segments. Our approach can help to ease the problem by considering the spatial proximities between line segments only with relevant plane segmentations (see Sec. III-C). This has good potential for supporting high-performance applications in practice.

ACKNOWLEDGMENT

We thank the anonymous reviewers for the helpful comments that help us to improve the clarity in this paper, and the street-view images from the Google Street View service. This work is supported partially by the Research Grants Council of the Hong Kong Special Administrative Region (CUHK 14203416 & 14201717) and National Natural Science Foundation of China (Grant No. 61802388).

REFERENCES

[1] Google's street view static API. https://developers.google.com/maps/documentation/streetview. Accessed: 2020-02-19. 5

[2] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 126(9):961–972, 2018. 2, 6, 7

[3] R. T. Azuma. A survey of augmented reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355—-385, 1997. 2, 9

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 39(12):2481–2495, 2017. 2

[5] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 2

[6] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *TPAMI*, 8(4):425–455, 1986. 2

[7] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, volume 33, pages 8001–8008, 2019. 6

[8] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *TPAMI*, 40(4):834–848, 2018. 2

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 833–851, 2018. 5, 6

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 2, 6, 7

[11] J. M. Coughlan and A. L. Yuille. Manhattan world: compass direction from a single image by Bayesian inference. In *ICCV*, pages 941–947, 1999. 2

[12] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, pages 2432–2443, 2017. 6, 7

[13] S. Dasgupta, K. Fang, K. Chen, and S. Savarese. DeLay: Robust spatial layout estimation for cluttered indoor scenes. In *CVPR*, pages 616–624, 2016. 2

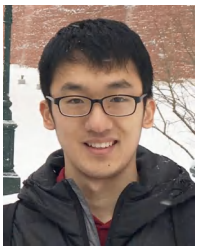
[14] A. Fond, M. Berger, and G. Simon. Facade proposals for urban augmented reality. In *ISMAR*, pages 32–41, 2017. 1, 2

[15] R. Gal, L. Shapira, E. Ofek, and P. Kohli. FLARE: Fast layout for augmented reality applications. In *ISMAR*, pages 207–212, 2014. 1, 2

[16] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall.

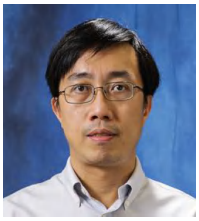
LSD: a Line Segment Detector. *Image Processing On Line*, 2:35–55, 2012. 2, 3, 4
[17] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, pages 482–496, 2010. 2
[18] O. Haines and A. Calway. Recognising planes in a single image. *TPAMI*, 37(9):1849–1861, 2015. 1, 2, 3
[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, pages 2961–2969, 2017. 7
[20] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, pages 1849–1856, 2009. 2, 3
[21] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, pages 654–661, 2005. 1, 2, 3
[22] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, pages 151–172, 2007. 1, 2
[23] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. S. Torr. Deeply supervised salient object detection with short connections. *TPAMI*, 41(4):815–828, 2018. 6
[24] J. Karlekar, S. Z. Zhou, W. Lu, Z. C. Loh, Y. Nakayama, and D. Hii. Positioning, tracking and mapping for outdoor augmentation. In *ISMAR*, pages 175–184, 2010. 2
[25] Y.-C. Kung, Y.-L. Huang, and S.-Y. Chien. Efficient surface detection for augmented reality on 3D point clouds. In *CGI*, pages 89–92, 2016. 1, 2
[26] T. Langlotz, T. Nguyen, D. Schmalstieg, and R. Grasset. Next-generation augmented reality browsers: Rich, seamless, and adaptive. *Proceedings of the IEEE*, 102(2):155–169, 2014. 2
[27] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. Roomnet: End-to-end room layout estimation. In *ICCV*, pages 4875–4884, 2017. 2
[28] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, pages 2136–2143, 2009. 2, 10
[29] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *CVPR*, 2019. 1, 2, 3, 5, 6, 7
[30] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. PlaneNet: Piece-wise planar reconstruction from a single RGB image. In *CVPR*, pages 2579–2588, 2018. 1, 2, 3
[31] Y. Liu, M. Cheng, X. Hu, J. Bian, L. Zhang, X. Bai, and J. Tang. Richer convolutional features for edge detection. *TPAMI*, 41(8):1939–1946, 2019. 2
[32] A. Mallya and S. Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, pages 936–944, 2015. 2
[33] B. Micusik, H. Wildenauer, and J. Kosecka. Detection and matching of rectilinear structures. In *CVPR*, pages 1–7, 2008. 2, 10
[34] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, pages 1520–1528, 2015. 2
[35] B. Nuernberger, E. Ofek, H. Benko, and A. D. Wilson. SnapToReality: Aligning augmented reality to the real world. In *CHI*, pages 1233–1244, 2016. 1, 2
[36] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *NIPS*, pages 4797–4805, 2016. 2
[37] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. GeoNet: Geometric neural network for joint depth and surface normal estimation. In *CVPR*, pages 283–291, 2018. 2, 3
[38] G. Reitmayr and T. W. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *ISMAR*, pages 109–118, 2006. 2
[39] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, pages 3234–3243, 2016. 6, 7
[40] C. Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9):647–655, 2002. 2
[41] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D scene structure from a single still image. *TPAMI*, 31(5):824–840, 2009. 1, 2
[42] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *TPAMI*, pages 640–651, 2017. 2
[43] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, pages 1–8, 2008. 2
[44] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009. 2
[45] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, pages 746–760, 2012. 1, 2, 3
[46] G. Simon, A. Fond, and M.-O. Berger. A-contrario horizon-first vanishing point detection using second-order grouping laws. In *ECCV*, pages 323–338, 2018. 2, 5
[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 7
[48] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler. Gated-SCNN: Gated shape CNNs for semantic segmentation. In *ICCV*, pages 5229–5238, 2019. 2
[49] K. Wada. LabelMe: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. 5
[50] Y. Wu and K. He. Group normalization. In *ECCV*, pages 3–19, 2018. 5
[51] J. Xiao and Y. Furukawa. Reconstructing the world's museums. *IJCV*, 110(3):243–258, 2014. 2
[52] F. Yang and Z. Zhou. Recovering 3D planes from a single image via convolutional neural networks. In *ECCV*, pages 87–103, 2018. 1, 2, 3, 5, 6, 7
[53] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, pages 6230–6239, 2017. 5, 6
[54] F. Zhou, H. B.-L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *ISMAR*, pages 193–202, 2008. 1, 2

Zhiliang Zeng received the B.S. degree from the Beijing Normal University, Zhuhai, and the M.Sc. degree in Computer Science and Engineering from the Chinese University of Hong Kong. He is currently a PhD student in Computer Science and Engineering of Chinese University of Hong Kong. His research interests include deep neural network and image segmentation, using neural network for indoor/outdoor scene analysis and application.



Mengyang Wu received the B.S. degree from University College London. He is currently a PhD student in Computer Science and Engineering of Chinese University of Hong Kong. His recent research interests include deep learning for 3D vision, scene understanding, and outdoor augmented reality.

Wei Zeng is currently an associate researcher at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He received the PhD degree in computer science from Nanyang Technological University in 2015. He served as co-chair of ChinaVis'20 International Forum and PacificVis'19 Poster session, and program committee members in various research conferences, including ChinaVis, IVAPP, etc. His research interests include data visualization, AR/VR, human–computer interaction, and urban computing.



Chi-Wing Fu is currently an associate professor in the Chinese University of Hong Kong. He served as the co-chair of SIGGRAPH ASIA 2016's Technical Brief and Poster program, associate editor of IEEE Computer Graphics & Applications and Computer Graphics Forum, panel member in SIGGRAPH 2019 Doctoral Consortium, and program committee members in various research conferences, including SIGGRAPH Asia Technical Brief, SIGGRAPH Asia Emerging tech., IEEE visualization, CVPR, IEEE VR, VRST, Pacific Graphics, GMP, etc. His recent research interests include computation fabrication, point cloud processing, 3D computer vision, user interaction, and data visualization.