This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

PlaneSeg: Building a Plug-In for Boosting Planar Region Segmentation

Zhicheng Zhang, *Graduate Student Member, IEEE*, Song Chen, Zichuan Wang, and Jufeng Yang, *Member, IEEE*

***Abstract*— Existing methods in planar region segmentation suffer the problems of vague boundaries and failure to detect small-sized regions. To address these, this study presents an end-to-end framework, named PlaneSeg, which can be easily integrated into various plane segmentation models. Specifically, PlaneSeg contains three modules, namely, the edge feature extraction module, the multiscale module, and the resolution-adaptation module. First, the edge feature extraction module produces edge-aware feature maps for finer segmentation boundaries. The learned edge information acts as a constraint to mitigate inaccurate boundaries. Second, the multiscale module combines feature maps of different layers to harvest spatial and semantic information from planar objects. The multiformity of object information can help recognize small-sized objects to produce more accurate segmentation results. Third, the resolution-adaptation module fuses the feature maps produced by the two aforementioned modules. For this module, a pairwise feature fusion is adopted to resample the dropped pixels and extract more detailed features. Extensive experiments demonstrate that PlaneSeg outperforms other state-of-the-art approaches on three downstream tasks, including plane segmentation, 3-D plane reconstruction, and depth prediction. Code is available at https://github.com/nku-zhichengzhang/PlaneSeg.**

***Index Terms*— Deep learning, depth prediction, planar region segmentation, plane reconstruction, plug-in.**

I. Introduction

IN THIS article, we study the problem of planar region segmentation, which is an essential subproblem of 3-D plane reconstruction. Planar regions in a scene provide vital information for many vision-based tasks, including augmented reality [1], [2], [3], stereo vision [4], [5], [6], and simultaneous localization and mapping (SLAM) [7], [8]. After extracting all planes from a single image, people can then select the ones of their interest and design useful and attractive applications based on these planar regions. For example, one could decorate plain walls with a favorite texture [9], or advertisers might make the best use of less-informative regions [10] (e.g., desks, walls, and boards) in a promotional video to market their products more efficiently. Moreover, planar features are also crucial cues for autonomous robots to perceive surroundings and build a map through camera views [11].

Plane segmentation has been extensively explored with the aid of additional 3-D structure information. In some cases, the structure information of an image is explicitly provided, (e.g., point cloud [14], [15] or depthmap [16], [17], [18]), which contains essential information of a 3-D space that can be directly used to fit a plane. However, accurately segmenting planar instances from a single image still remains a challenge. Many traditional works adopt bottom-up approaches [19], [20], which first detect structure priors, such as vanishing points, intersections, lines, and image patches, in man-made scenes. Later, the geometric relationships of these primitives are explored. These approaches describe man-made environments with low-level features, but handcrafted priors are not robust when encountering images with low resolution and minor spatial malposition.

Recently, PlaneNet [9] and PlaneRecover [21] propose straightforward CNN-based architectures to detect planar regions. Both regard this task as a pixelwise segmentation problem. However, the maximum of detected planes is limited. Specifically, given a value K that represents the maximum plane number in an image, PlaneNet produces a $K \times 3$ plane parameters vector by a regression branch. Consequently, this method can only produce no more than K plane detection results. Similarly, in PlaneRecover, there is also a conv layer with a kernel size of 3m, which limits the maximum detection number to m. Recently, PlaneEmbedding [22] and PlaneRCNN [13] was designed to address this issue. PlaneEmbedding applies a two-stage bottom-up strategy with the mean shift clustering algorithm, while PlaneRCNN leverages mask R-CNN [23] to generate an arbitrary number of planes. In PlaneRCNN, a refinement module is proposed to refine plane masks by integrating features of all masks extracted by mask R-CNN.

However, it is an ill-posed problem to segment planes from a single image [13]. With the class-agnostic setting, the plane generally includes a large set of classes and varying-sized objects for the cupboard, lavatory cover, and even piece of rubbish bin. Thus, there are two challenges that still remain, as illustrated in Fig. 1. First, the predicted segmentation mask is inaccurate, especially for the area around the plane. Unlike the clear boundary of a generic object, the plane boundary area is vague due to the high-level abstract of structure information like depth and normal information [13], [22]. As a result, the abstract raises the degree of challenge which requires the introduction of knowledge for a promising performance. Second, the small-sized planes are erroneously detected

Manuscript received 1 October 2022; revised 21 February 2023; accepted 14 March 2023. This work was supported in part by the National Key Research and Development Program of China Grant 2018AAA0100400, in part by the Natural Science Foundation of Tianjin, China, under Grant 20JCJQJC00020, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Jufeng Yang.)*

Zhicheng Zhang, Song Chen, and Jufeng Yang are with the Tianjin Key Laboratory of Visual Computing and Intelligent Perception (VCIP) and the College of Computer Science, Nankai University, Tianjin 300350, China (e-mail: gloryzzc6@sina.com; songcheney@mail.nankai.edu.cn; yangjufeng@nankai.edu.cn).

Zichuan Wang is with the School of Mathematical Sciences, Nankai University, Tianjin 300071, China (e-mail: zichuan.wang@mail.nankai.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3262544.

Digital Object Identifier 10.1109/TNNLS.2023.3262544

2162-237X © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

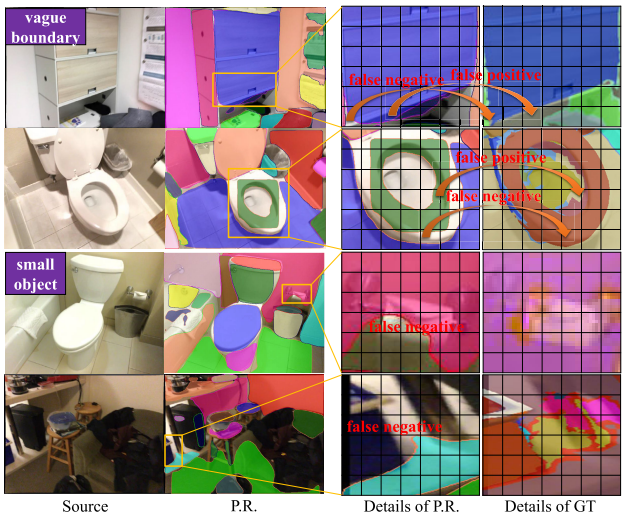

Fig. 1. Illustration of the two challenges for planar region segmentation. We show four examples selected from the ScanNet [12] dataset. The vague boundaries challenge (the first two rows) arises from insufficiency to capture edge-aware features, while the small-sized regions challenge (the last two rows) comes from the loss of multiscale information. P.R. and GT denote PlaneRCNN [13] and the ground truth, respectively.

(see Fig. 1). In planar region segmentation, planes are built in a piecewise manner to represent surfaces in the real world [9]. However, existing methods focusing on high-level features largely throw away the spatial information of small-sized objects, which is easily overlooked. Therefore, prevenient works neglect the aforementioned challenges [9], [21], [22] or only take into account one [13].

Considering these challenges, this study proposes PlaneSeg, which is a CNN-based network that consists of three corresponding modules, namely, the edge feature extraction module, the multiscale module, and the resolution-adaptation module. Since the proposed PlaneSeg takes in the rich information mentioned above, when it is integrated into existing state-of-the-art PlaneRCNN, PlaneEmbedding, and PlaneRecNet, PlaneSeg improves its performance both qualitatively and quantitatively. Following tradition, the proposed PlaneSeg has trained end-to-end on the recent ScanNet [12] dataset from which the study uniformly selected 32 000 frames as training data. Similar to [24], the edge module training signal was generated from the ground truth of plane masks. These edge features were fused with multiscale context information in a pairwise style and restored dropped pixels to produce better features with which the original plane segmentation models can perform better.

On the whole, our contributions are summarized as follows.

1) We design PlaneSeg that leverages edge and multiscale information to conduct plane segmentation in a simple, efficient manner. We propose a pairwise fusion strategy to integrate the edge information and multiscale context information, as well as recover the feature map of small planes. With the help of the PlaneSeg, more accurate segmentation results with finer boundaries can be produced.
2) We develop a plug-in component, namely, PlaneSeg, which can be flexibly integrated into various plane segmentation models and brings a significant performance boost to existing state-of-the-art methods. Extensive experiments on two benchmark datasets demonstrate the superiority of the proposed method.

II. Related Works

A. Instance Segmentation

Instance segmentation [25], [26], [27], [28] contains two types, i.e., bottom-up methods and top-down ones. Bottom-up approaches predict the instance label of each pixel through clustering. For example, PersonLab [29] first recognizes individual keypoints and then groups these into pose instances. Li et al. [30] takes salient information into consideration and further constructs the segmentation model. Unlike the bottom-up paradigm, top-down approaches first predict bounding box proposals based on detection models and then refine the bounding boxes to obtain segmentation results. Mask R-CNN [23] adds a network branch for predicting instance mask, built on top of faster R-CNN [31] for object detection. This backbone has been widely combined in state-of-the-art instance segmentation pipelines [32]. Zhang et al. [33] proposed a MaskSSD that successfully extended single shot multibox detector (SSD) [34] to an instance segmentation method. Therefore, this study presents PlaneSeg which consists of three modules and can assist plane segmentation models from both categories in dealing with the ambiguity of boundaries.

B. Planar Region Analysis

The analysis of planar regions has been widely studied due to the various geometric cues provided for downstream tasks, such as scene understanding and scene reconstruction. With the help of analyzing plane regions, it is possible to develop applications in 3-D reconstruction [35], [36], [37], [38], [39], [40] and depth estimation [41], [42]. In the static image, for image segmentation, the prior information provided by planes can help segment common objects [43], [44]. For example, Zhang et al. [43] extracted view-independent 3-D features (e.g., planarity of surface), which are useful for splitting planar and nonplanar objects. The features are then used to segment and recognize different objects in the image. For plane segmentation, much effort has been placed into designing effective deep neural network architectures [13], [22], analyzing the theory of networks [45], [46], boosting mountains of downstream applications in geometric understanding [47], [48], and indoor layout estimation [49], [50]. In the dynamic video, for manipulating planes [51], [52], [53], tracking manipulation tasks (TMT) [52] builds a hardware system consisting of a robot arm and planar objects that guide the manipulator in moving planar objects into the desired position. Furthermore, Li et al. [51] proposed a novel adaptive neural network for developing multiple planar manipulators, where an admittance model is developed to generate reference which is then used in manipulators. For tracking planes [54], [55], [56], Valeiras et al. [54] proposed tracking objects based on their projection onto the focal plane and then leveraging a virtual mechanical system to introduce high temporal elasticity for adaptation on the change of plane. Recently, Gracker [56] introduces a graph-matching algorithm into tracking, that is, building the graph between the initial planar region and the searched image. Then, the planar regions are associated across frames by computing the similarity between nodes of planes.

C. Plane Segmentation From a Single Image

Plane segmentation has been a long-standing problem that is difficult to solve [57], [58], [59]. Previously, Hoiem et al. [58]

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

and Saxena et al. [59] explored this problem with different approaches. The former first divides the image into superpixels and then employs a boosted decision tree algorithm to classify these into different planes. Meanwhile, the latter predicts per-pixel depth and then produces a 3-D model of a given image. However, the methods mentioned earlier can only detect coarse planes, such as building exteriors or streets, and mask boundaries are unable to properly coincide with the actual boundaries. As such, accurate detection of small and complex object surfaces remains unsolved.

In recent years, many works have proved that it is possible to reconstruct both depth [60], [61] and normal [62], [63] information accurately from a single RGB image. These works provide sufficient information to recover the 3-D structure of a given image and generate accurate plane detection results [49], [64], [65], [66], [67]. PlaneNet [9], PlaneRecover [21], PlaneRCNN [13], PlaneEmbedding [22], and PlaneRecNet [66] have further proved the validity of CNN-based methods to segment and reconstruct more detailed plane instances. These models combine the segmented planes and predicted plane parameters for reconstructing the 3-D version of planes to contain a depth prediction module as an auxiliary. PlaneNet and PlaneRecover propose an end-to-end architecture. However, both suffer the drawback wherein the output is limited to a fixed number of planes at 10 and 5, respectively. Most recently, PlaneRCNN, PlaneRecNet, and PlaneEmbedding successfully solved this problem with the ability to segment an arbitrary number of plane instances. Specifically, PlaneRCNN employs mask R-CNN, which includes a region proposal network and can be applied to instance segmentation tasks. PlaneRCNN also improves segmentation quality through a refinement network and enhances segmentation consistency with a warping loss. To further exploit the power of the instance segmentation network, PlaneRecNet employs SOLOv2 [68] which reformulates the parallel generation of instances as mask prediction and feature learning. Moreover, PlaneRecNet further corrects the error of predicted masks by geometric constraint, and thus, avoids generating masks at the edge area. Meanwhile, PlaneEmbedding proposes a two-stage strategy with associative embedding, which first assigns pixels that belong to the same plane with similar embedding and then clusters the embedding vectors through mean shift clustering. Compared with the other three CNN-based methods, PlaneEmbedding runs in real time (i.e., 32.26 fps on a single graphics processing unit (GPU)).

Nevertheless, the above-mentioned approaches still have several shortcomings as these methods ignore vital information in the image for plane segmentation tasks (e.g., object edges and loss of resolution). Consequently, it is difficult to generate accurate mask edges or detect objects of small size [24]. The PlaneSeg proposed by this study integrates traditional CNN architecture with three new modules to effectively combine the learned edge features and multiscale information of an image. In this manner, vague boundaries and difficulties in detecting small-sized regions can be alleviated.

D. Edge-Aware CNN Models

In some dense prediction tasks (e.g., salient object detection (SOD) [69], scene segmentation [70], and parsing [71]), boundary localization may be inaccurate between different classes due to the complex semantic relations of multiple instances [71]. Currently, edge information has been used in various vision tasks as a possible solution for improving classification performance [72]. To leverage edge features, a common pipeline is that the encoder or feature extractor first learns low-level features. Then, the edge module predicts the boundary according to these outputs. Finally, the feature fusion process is applied to the edge features and the extracted high-level features related to specific tasks. However, most traditional methods suffer from lost information and do not sufficiently learn the relationship between edge and context features. This study, therefore, proposes PlaneSeg which learns edge and multiscale context information, integrates them in a pairwise manner and recovers dropped pixels in high-level features.

The primary purpose of incorporating an edge module in CNN models is to simultaneously decrease ambiguity between neighboring objects and increase feature similarity within the same class. Also, the negative influences caused by noisy labels can be mitigated as a byproduct. For example, SOD [73], [74], [75] is a large body of work that benefits from edge-aware CNN models. With the aid of edge features, salient objects with narrow stripes are detected with finer boundaries, and thus, are more distinguishable from the background. Edge guidance network (EGNet) [69] and SCRN [76] optimize the corresponding tasks jointly with a one-to-one guidance module and stacked cross-refinement units, respectively.

Besides the prevailing use of edge stream in SOD, researchers aim to preserve boundary structures in other vision tasks. With the restriction of boundaries, Ding et al. [70] strengthens feature similarity within the same object in terms of scene segmentation. For human parsing, context embedding with edge perceiving (CE2P) [24] examines the availability of edge details while ignoring the negative influences caused by noisy labels. Meanwhile, by leveraging the performance boost of the edge module, A-CE2P [77] proposes an iterative self-correction strategy to purify the training labels. A comparison of existing works with the proposed PlaneSeg is detailed in Section III-D.

III. Method

The proposed PlaneSeg incorporates edge and multiscale context information in a pairwise manner and then recovers dropped pixels in high-level feature maps to mitigate vague boundaries of plane masks. Specifically, three key modules constitute the proposed PlaneSeg: the edge feature extraction module, the multiscale module, and the resolution-adaptation module (see Fig. 2). Here, PlaneSeg is demonstrated with the ResNet-101 [78] backbone. All the losses are further combined to make the proposed method end-to-end trainable. Finally, we list the difference between our PlaneSeg to similar works for distinguishment.

A. Revisiting SOTA Methods

PlaneRCNN is a well-performing framework for the planar segmentation task and consists of four modules. First, the Mask R-CNN backbone takes an image as input and generates feature maps, bounding boxes, and plane masks. Next, a depth prediction module predicts the depth map based on image features. Also, plane normals and plane offsets are estimated with k-means clustering and simple algebra. Finally, the refinement network integrates the preliminary plane masks and depth map to produce finer segmentation masks. Unlike PlaneRCNN which can be trained end-to-end, PlaneEmbedding is a two-stage framework. First, a CNN is trained to extract per-pixel embedding, where the pixels

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

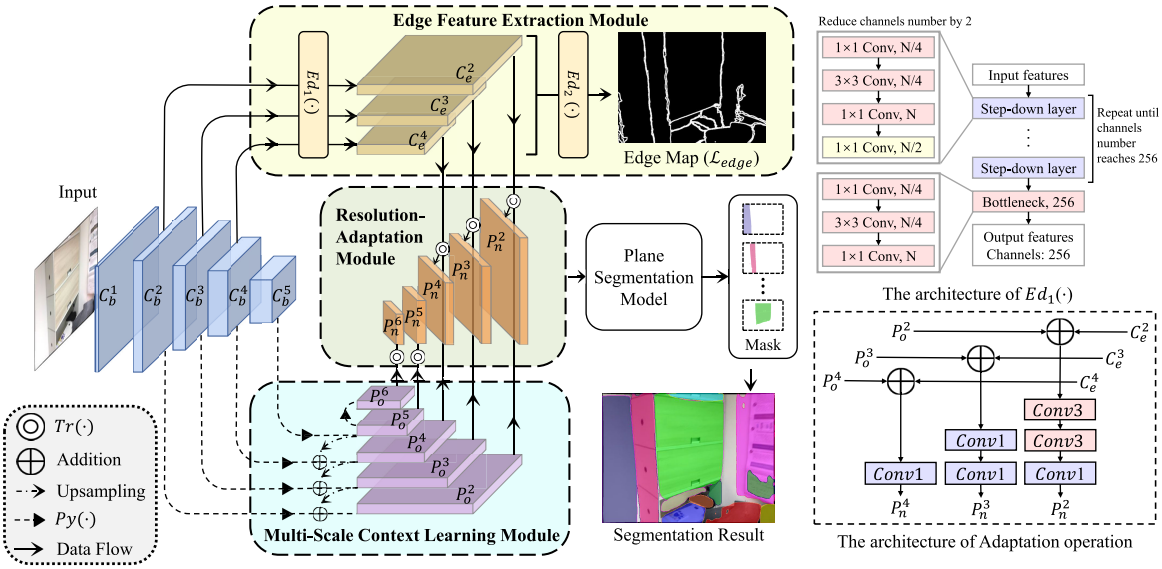

Fig. 2. **Left:** Network architecture of PlaneSeg based on ResNet-101 [78]. Image features are extracted by ResNet blocks (denoted by C_b^i, $i = 1, 2, 3, 4, 5$). These features then flow parallel into the multiscale module (lower) and the edge feature extraction module (upper). The multiscale module fuses features of different sizes and generates context features P_o^i ($i = 2, 3, 4, 5, 6$). The edge feature extraction module generates edge features (denoted by C_e^i, $i = 2, 3, 4$) and edge probability map through $\mathrm{Ed}_1(\cdot)$ and $\mathrm{Ed}_2(\cdot)$, respectively. Next, the resolution-adaptation module (middle) integrates the context features and edge features pairwisely and restores dropped pixels from high-level feature maps. The resolution-adaptation module output features are denoted as P_n^i ($i = 2, 3, 4, 5, 6$). **Top Right:** $\mathrm{Ed}_1(\cdot)$ consists of step-down layers and a bottleneck layer. The step-down layer is responsible for reducing the channel number by 2. It consists of a bottleneck layer and a 1×1 conv layer. N denotes the channel number of the input feature of the step-down layer. The step-down layer is repeated according to the input feature shape to produce an output feature map of 256 channels. **Bottom Right:** The adaptation operation differs for the features in different resolution levels ($i = 2, 3, 4$). Each adaptation operation takes features of the same resolution (e.g., P_o^4 and C_e^4) as input and produces the refined feature.

of the same planar instance are expected to have similar embedding representations. Then, plane instances are obtained by using the mean shift algorithm to cluster the embedding vectors. Third, considering the consistency between pixel-level and instance-level, the parameters for each plane instance are estimated. Recently, PlaneRecNet leverages the power of single-stage instance segmentation network SOLOv2 [68]. First, the ResNet backbone extracts feature from the given image, which is then fed into the instance segmentation branch and the depth decoder branch separately. In the instance segmentation branch, plane instances are predicted through the dynamic convolution operation. In the depth decoder branch, the reconstructed depth map is obtained from upsampled features and can be used for computing plane parameters by RANSAC.

Although existing planar region segmentation methods have already incorporated important geometric constraints to address the limitations in previous works, such as predicting the limited number of planes, however, these state-of-the-art models could be further strengthened from the feature level. For instance, the boundary of the plane is still not explicitly modeled. That is, the complementarity between edge and plane information is yet to be noticed. To this end, this study designed PlaneSeg to learn boundary features. Moreover, the PlaneSeg extracted multiscale context information cooperates with edge features in a pairwise manner. Also, the dropped pixels are resampled for higher level feature maps.

B. Network Architecture

1) Multiscale Module: Several recent studies show that richer multiscale information can help convolutional networks to extract more useful and robust features [79], [80]. Each ResNet block is formally denoted as blocks block_i ($i = 1, 2, \ldots, 5$), and the output feature maps of each block as C_b^i. Traditional methods [24] first learn multiscale context from C_b^5 and then concatenate it with C_b^2. We improved the traditional method by leveraging all context information between C_b^2 and C_b^5. Our method not only learns features of multiple scales and levels but also preserves the different sizes of ResNet feature maps, which is crucial for the pairwise feature fusion process in the resolution-adaptation module. In our experiments, feature pyramid network (FPN) [81] is adopted to assist in multiscale context extraction. We define the functions of FPN by $Py_i(\cdot)$ ($i = 2, 3, 4, 5, 6$), and outputs by P_o^i. Then, we have the relationship between FPN and ResNet

$$P_o^i = \begin{cases} Py_i\left(C_b^i\right) + \mathrm{Up}\left(P_o^{i+1}\right), & i = 2, 3, 4 \\ Py_i\left(C_b^i\right), & i = 5 \\ Py_i\left(P_o^{i-1}\right), & i = 6 \end{cases} \tag{1}$$

where $\mathrm{Up}(\cdot)$ refers to upsampling by bilinear interpolation. $Py_i(\cdot)$ is a combination of two convolution layers with kernel sizes of 1 and 3. It is easily noticed that in (1), feature maps with two different perceptive fields are added. This operation combines multiscale contexts in a simple yet effective manner.

2) Edge Feature Extraction Module: Similarly, in this part, the edge feature extraction module shares the features learned by ResNet to reduce computation costs. We consider the edge module as a function of C_b^2, C_b^3, and C_b^4, denoted by $\mathrm{Ed}(\cdot)$. To define $\mathrm{Ed}(\cdot)$, we exclude the feature maps from the first layer C_b^1 and the last layer C_b^5 of ResNet blocks. The reason is that for C_b^1, its receptive field is small because block_1 directly acts on the input image [82]. Even though C_b^5 has the largest receptive field among the five feature maps, when we perform upsampling by bilinear interpolation in the edge module, details of edge information may be lost heavily. Here, $\mathrm{Ed}(\cdot)$ consists of two phases denoted by $\mathrm{Ed}_1(\cdot)$ and $\mathrm{Ed}_2(\cdot)$.

Now, we have $\mathrm{Ed}(\cdot) = \mathrm{Ed}_2(\mathrm{Ed}_1(\cdot))$. The first phase calculates the corresponding edge features w.r.t. C_b^i ($i = 2, 3, 4$),

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

denoted by C_e^i. $\mathrm{Ed}_1(\cdot)$ learns edge features and decreases the number of feature map channels. Different from the traditional method [77] that reduces channels in a brute way (with only one conv layer), we employ a smooth and step-down approach. For each C_b^i, we decrease the number of channels by a factor of 2 in each step until we obtain the desired number. For clarity, this step is described as

$$C_e^i = \mathrm{Ed}_1\left(C_b^i\right). \tag{2}$$

Here, $\mathrm{Ed}_1(\cdot)$ consists of multiple convolutional layers. For different i, the input channel number may vary. The detailed structure of $\mathrm{Ed}_1(\cdot)$ is shown in Fig. 2. We adopt the bottleneck architecture proposed by [78] to help generate the edge features. The bottleneck architecture consists of three conv layers. The first 1×1 conv layer reduces the channel number to a quarter, which is then restored by the last 1×1 conv layer. The 3×3 convolution in the middle is responsible for extracting information from the compressed features. We adopt this method as the basic feature extraction unit in $\mathrm{Ed}_1(\cdot)$.

Next, we build a step-down layer by combining a bottleneck with a 1×1 conv layer. The 1×1 conv is responsible for reducing the channel number by a factor of 2. In $\mathrm{Ed}_1(\cdot)$, multiple step-down layers are stacked, and each layer halves the number of channels. These step-down layers smoothly reduce the channel number to 256. Finally, we attach another bottleneck layer to the last step-down layer to generate edge features. These features will be fused with the output of the multiscale module in the next resolution-adaptation module.

The second phase is edge prediction

$$\begin{aligned}\hat{e} &= \mathrm{Ed}_2\left(C_e^2, C_e^3, C_e^4\right) \\ &= \mathrm{Conv1}\left(\mathrm{Concat}\left\{\mathrm{Up}\left(\mathrm{Conv3}\left(C_e^i\right)\right)\right\}_{i=2,3,4}\right)\end{aligned} \tag{3}$$

where $\hat{e}$ is the predicted edge probability map. Each value of $\hat{e}$ denotes the probability of the corresponding pixel being on an edge. $\mathrm{Up}(\cdot)$ refers to bilinear upsampling. This operation upsamples all feature maps to the same size, making subsequent concatenation operation reasonable. $\mathrm{Conv1}(\cdot)$ and $\mathrm{Conv3}(\cdot)$ denote convolution layers with a kernel size of 1 and 3, respectively. In order to generate a probability map, all Conv layers have an output channel dimension of 2. In detail, we first use 3×3 convolution layers to reduce the channel number of C_e^i $(i = 2, 3, 4)$ to 2. Next, we upsample these three features to the same size and concatenate them. Finally, we apply a 1×1 conv layer to fuse three edge probability maps. To train this module end-to-end, we generate the ground truth edge labels from plane mask labels and supervise the learning process with the edge loss $\mathcal{L}_{\text{edge}}$ described in Section III-C. The gradient of $\mathcal{L}_{\text{edge}}$ will be backpropagated to update the parameters of this module, and the quality of C_e^i will be further improved. In the next process, we combine the edge feature with the multiscale context feature pairwisely and recover dropped pixels.

3) Resolution-Adaptation Module: In order to segment planar instances, especially the ones of small sizes features with a high resolution are critical. However, high-level features have a much smaller size compared with the original input image due to downsampling. Therefore, we design a resolution-adaptation module that can integrate the features learned by the previous two modules and adaptively capture the dropped features at higher levels. Compared with the common approach that fuses features by a trivial concatenation, we design a pairwise feature fusion method in this module based on some observations of the aforementioned two modules. Intuitively, this pairwise fusion help to sample pixels from two aspects, i.e., edge-related area, and body-aware area. With such, the pixels dropped by downsampling operations [83] can be resampled, where a 2×2 max pooling operation can dropout three-quarters of pixels from the original resolution. In our edge feature extraction module, we obtain feature maps of three different sizes (e.g., 160×160, 80×80, and 40×40 in PlaneRCNN). Also, the features P_o^i $(i = 2, 3, 4)$ extracted from the multiscale module happen to share the same sizes. Consequently, based on this observation, we fuse the corresponding features of the same size. First, our resolution-adaptation module aggregates feature from the aforementioned two modules by lateral connection, which keeps information flowing at the same resolution level. To further recover dropped pixels, the adaptation operation is parameterized by compositions of convolutional filters.

In this way, not only does this module integrate the two types of features but it can also learn the relationship between edge and multiscale context information. Now, we define the renewed features as

$$P_n^i = \begin{cases} \mathrm{Tr}_i\left(P_o^i + C_e^i\right), & i = 2, 3, 4 \\ P_o^i, & i = 5, 6 \end{cases} \tag{4}$$

in which

$$\begin{aligned}\mathrm{Tr}_2 &= \mathrm{Conv3}(\cdot) \circ \mathrm{Conv3}(\cdot) \circ \mathrm{Conv1}(\cdot) \\ \mathrm{Tr}_3 &= \mathrm{Conv1}(\cdot) \circ \mathrm{Conv1}(\cdot) \\ \mathrm{Tr}_4 &= \mathrm{Conv1}(\cdot)\end{aligned} \tag{5}$$

where $\circ$ denotes function composition. The number of output channels is set to 256 among all conv layers. In $\mathrm{Tr}_i(\cdot)$ $(i = 2, 3, 4)$, we use filters to recover the dropped features. Since features from earlier layers have a relatively large size, we use multiple conv layers with larger kernels for these features to adjust the receptive field.

This module generates five features (P_n^2 to P_n^6), each of which has exactly the same shape as the corresponding one in the features generated by ResNet (C_b^1 to C_b^5). Thus, we can easily forward the newly generated features to the plane segmentation model smoothly as it could treat each P_n^i as the original backbone C_b^{i-1}. Therefore, the way the resolution-adaption module gets plugged into the segmentation model varies with different segmentation models. For example, PlaneRCNN is based on MaskRCNN, so after our PlaneSeg learns edge features, P_n^i are fed into the following RPN (region proposal network) of MaskRCNN, where these features communicate with each other. And for PlaneEmbedding, based on the model structure of the public code, only P_n^2 is used for plane segmentation. Therefore, to integrate all edge features into P_n^i, we add each P_o^i and $C_e^i (i = 2, 3, 4)$ followed by the transformation function (two 1×1 conv layers) before P_o^i are upsampled in the multiscale module. For instance, in PlaneEmbedding, $P_o^2 = \mathrm{Up}(\mathrm{Tr}(P_o^3 + C_e^3))$. To sum up, it is relatively flexible to design the pipeline to get the output of the resolution-adaption module and plug it into the following segmentation model, and this depends on the structure of the segmentation network. Fig. 2 shows the network architecture based on PlaneRCNN, but the structure of PlaneSeg may vary in different conditions. In the following steps, the features will facilitate the original plane segmentation model.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

C. Loss Function

The hybrid loss function of each model generally contains two terms, namely, the edge loss that assists in preserving boundary information, and the loss of the original plane segmentation model. Formally, the hybrid loss is

$$\mathcal{L} = \lambda_e \mathcal{L}_{\text{edge}} + \lambda_s \mathcal{L}_{\text{seg}}. \tag{6}$$

Here, $\mathcal{L}_{\text{edge}}$ is the edge loss. $\mathcal{L}_{\text{seg}}$ denotes the segmentation loss from the original model. λ_e and λ_e are hyperparameters.

1) Edge Loss: We hope to learn boundary-aware features, which can further facilitate the performance of downstream modules. Intuitively, this loss should explicitly preserve the edges of ground truth plane masks. Thus, the mismatch between predicted edges and labels is penalized by a weighted cross-entropy loss

$$\mathcal{L}_{\text{edge}} = \frac{1}{N}\left[\sum_{e_i=1}\frac{N^-}{N}\log \hat{e}_i + \sum_{e_i=0}\frac{N^+}{N}\log\left(1-\hat{e}_i\right)\right]. \tag{7}$$

Here, N denotes the number of pixels in an image, N^+ and N^- denote the number of edge pixels and other pixels, respectively, e_i denotes the edge label of the ith pixel, and $\hat{e}_i$ denotes the probability of the ith pixel being on edge. Also, $e_i = 1$ indicates edge region, and $e_i = 0$ otherwise. In A-CE2P [77], only edge pixels contribute to $\mathcal{L}_{\text{edge}}$. However, our experiments show that this does not improve model performance (see Table I).

2) Loss Related to PlaneRCNN: In the content of PlaneRCNN, $\mathcal{L}_{\text{seg}}$ is defined as

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{bbox}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{refine}} + \mathcal{L}_{\text{warp}}. \tag{8}$$

The first three terms are for training its mask R-CNN backbone, while the remaining terms are for other modules in PlaneRCNN. That is, $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{bbox}}$, and $\mathcal{L}_{\text{mask}}$ are for proposal ID classification, bounding box regression, and mask prediction, respectively. $\mathcal{L}_{\text{depth}}$ is for depth prediction which further facilitates the mask refinement module. $\mathcal{L}_{\text{refine}}$ is the cross-entropy loss that forces the refinement module to improve the accuracy of plane masks. And $\mathcal{L}_{\text{warp}}$ denotes the warping loss that preserves the consistency of depth map outputs between nearby video frames. Please refer to [13] and [23] for further explanations.

3) Loss Related to PlaneEmbedding: In the content of PlaneEmbedding, $\mathcal{L}_{\text{seg}}$ is defined as

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{emb}} + \mathcal{L}_{\text{pln}} + \mathcal{L}_{\text{pp}} + \mathcal{L}_{\text{ip}}. \tag{9}$$

In this equation, $\mathcal{L}_{\text{emb}}$ denotes the loss to train the plane embedding module. $\mathcal{L}_{\text{pln}}$ is for planar/nonplanar region prediction. $\mathcal{L}_{\text{pp}}$ and $\mathcal{L}_{\text{ip}}$ are for pixelwise and instance-level plane parameter prediction, respectively.

4) Loss Related to PlaneRecNet: In the content of PlaneRecNet, $\mathcal{L}_{\text{seg}}$ is defined as

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{ins}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{norm}}. \tag{10}$$

In this equation, $\mathcal{L}_{\text{ins}}$ denotes the loss to train the instance segmentation module, and $\mathcal{L}_{\text{depth}}$ is for pointwise depth prediction. $\mathcal{L}_{\text{grad}}$ and $\mathcal{L}_{\text{norm}}$ are constraints using gradient and normal planes, respectively.

D. Comparison With Similar Works

Here, we discuss the difference between the use of edge information in PlaneSeg and most similar works in planar region segmentation, semantic segmentation, and SOD. Current planar region segmentation works address the limitations in previous works, such as requiring a maximal number of planes. However, these tend to simply stack multilayer features, which could be further strengthened from the feature level. To this end, we design PlaneSeg to learn boundary features. Moreover, the multiscale context information PlaneSeg extracts can cooperate with edge features pairwisely. Also, the dropped pixels are restored in higher level feature maps. That is, PlaneSeg can be easily integrated into them to improve their performance at a negligible cost, which also proves its efficiency.

Compared with methods leveraging edge cues in semantic segmentation and SOD, the major differences are: 1) feature maps where the receptive field is small or big are excluded; 2) a smooth and step-down approach for edge feature extraction; and 3) weighted cross-entropy is used to take into account both edge and nonedge regions. First, feature maps with a small receptive field directly act on the input image and originally contain many low-level features, such as edges and corners. For feature maps with a large receptive field, when performing upsampling by bilinear interpolation, edge information details may be heavily lost. These details are ignored by most works in semantic segmentation and SOD. For example, recent gated-shape CNN (SCNN) [84] and PAGE [85] utilize all feature maps generated by the backbone. Second, different from the traditional method [77], which reduces channels in a brute way (with only one conv layer) that could result in information loss, a smooth and step-down approach is employed. Third, for computing loss function, most works [69], [84], [86] simply use binary cross-entropy while ignoring the extreme imbalance between the number of edge pixels and nonedge pixels. Besides, PAGE [85] directly uses L2 loss, A-CE2P [77] only considers edge pixels and CGBNet [87] does not calculate the loss of edge information but treats it as the weighting coefficient of the feature maps, while we use weighted cross-entropy that considers the imbalance.

Furthermore, we make a detailed exploration of our resolution-adaption module, which utilizes the convolutional layer to fuse features of the edge module and the context module in a pairwise manner. Compared with other works, the major differences are: 1) our resolution-adaptation module adopts a learnable form to adaptively fuse features of the edge module and the context module, which is regarded as a more efficient data-driven approach. However, most works [77], [88] simply perform elementwise addition or channelwise concatenation, which is unsuitable for the task of recovering dropped pixels. For example, decouple segmentation network (DSN) [86] simply adopts the elementwise addition between edge features and body features to produce the final segmentation map. Owing to the adoption of multiple downsampling operations like the max-pooling layer, simple elementwise addition is incapable of correcting heavy boundary distortion and, thus, provides trivial help in recovering dropped pixels. Therefore, we design architecture composites of convolutional layers to perform recovery. Specifically, the resolution-adaptation module adaptively selects the features for finer feature maps in different resolution levels.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

2) Our resolution-adaptation module conducts a pairwise matching to integrate the edge feature and context feature of the same resolution level. On the contrary, CE2P [24], pyramidical gather-excite context (PGEC) [89], and CGBNet [87] simply upsample edge features into the same resolution and then concatenate them with other features. To this end, the proposed pairwise fusion is capable of effectively restoring dropped pixels in higher level feature maps thereby outperforming current methods.

IV. Experiments

A. Implementation Details

For a fair comparison, we choose ResNet-101 as the feature extractor in all experiments. We train our model on the ScanNet dataset. Since the authors of PlaneEmbedding and PlaneRCNN use different data divisions for training and testing and both of them are not made public, we build a training set with 32K images and a test set with 800 images for a fair comparison. We retrain PlaneEmbedding and PlaneRCNN on the built training set using the official code. Also, to improve the quality of the training data, we exclude similar views and filter out blurry frames. Specifically, from every 50 continuous video frames, we choose the one with the highest variation of the Laplacian [91] and finally acquire a training set with 32 000 images. We train models for ten epochs on a 1080Ti GPU, and it takes 2.5 days to train PlaneRCNN and five hours to train PlaneEmbedding. During training, the hyperparameters remain the same as the official implementation. As to the weight parameters in the hybrid loss function (6), λ_e and λ_s are both set to 1.

B. Datasets and Metrics

We have evaluated PlaneSeg on two public benchmark datasets: ScanNet [12] and NYUv2 [90]. ScanNet is a large-scale indoor RGB and depth (RGBD) video dataset with 1513 scenes and 2.5M views. In this article, we use the plane labels generated by Liu et al. [13] and randomly select 800 images as the test set. The NYUv2 dataset for indoor instance segmentation consists of 1449 RGB and depth image pairs. We use the public test set of 654 images and the plane labels produced by Yu et al. [22] for evaluation.

For quantitative evaluation, we adopt average precision (AP) with intersection over union (IOU) threshold 0.5 and 3 depth thresholds (0.3, 0.6, and 0.9 m), per-pixel recall, per-plane recall, and three other metrics that are designed for clustering comparison [92], [93] as follows.

1) Variance of Information: The variance of information (VoI) measures the difference between two clusters S and G by

$$\mathrm{VoI}(S, G) = H(S) + H(G) - 2\ I(S, G) \tag{11}$$

where H and I are the entropy and mutual information between two clusters. S and G refer to the predicted plane segmentation result and ground truth, respectively.

2) Rand Index: The rand index (RI) metric compares the compatibility of planar instance assignments of S and G as

$$\mathrm{RI}(S, G) = \frac{2(N_s + N_d)}{N(N-1)} \tag{12}$$

TABLE I

Comparison With Different Methods for Calculating $\mathcal{L}_{\text{edge}}$ on ScanNet. *P-Pixels* Denotes Only Using Positive Pixels [77], While *A-Pixels* Denotes Using All Pixels (Ours)

Edge Loss	Plane Segmentation			Plane Detection		
	VoI↓	RI↑	SC↑	$AP^{0.3m}$ ↑	$AP^{0.6m}$ ↑	$AP^{0.9m}$ ↑
P-Pixels	1.996	0.861	0.613	0.252	0.332	0.350
A-Pixels	**1.964**	**0.863**	**0.620**	**0.255**	**0.343**	**0.363**

where N represents the number of pixels in an image, and N_s and N_d denote the number of pixel pairs that have the same/different plane IDs both in S and G.

3) Segmentation Covering: We first define the covering $C(\cdot, \cdot)$ of S by G by

$$C(S \to G) = \frac{1}{N} \sum_{P_S \in S} |P| \max_{P_G \in G} \mathrm{IoU}(P_S, P_G) \tag{13}$$

where P_S and P_G are planar regions of S by G, respectively, and $\mathrm{IoU}(\cdot, \cdot)$ is the intersection of union of two planar regions. Then, segmentation covering (SC) is defined as

$$\mathrm{SC}(S, G) = \frac{1}{2}(C(S \to G) + C(G \to S)). \tag{14}$$

These metrics are suitable for this task because we only focus on whether different plane instances are assigned with different plane IDs instead of what the IDs are. Besides, we can easily derive that if we apply a permutation to plane IDs, then the numeric results of these three metrics will not change.

Furthermore, to evaluate the segmentation accuracy around plane boundaries, we adopt the methodology proposed by [94]. More precisely, we count the relative number of incorrectly classified pixels within a trimap of different bandwidths. Here, trimap refers to a narrowband around the ground truth boundaries, and the bandwidth of a trimap represents the width of the narrowband. This metric is suitable to evaluate how well an algorithm performs near the object boundaries.

Moreover, we evaluate depth prediction while six standard metrics following [22], including absolute relative error (Rel), log 10 error ($\log_{10}$), linear root-mean-square error (RMS), and accuracy under three thresholds (σ_1, σ_2, and σ_3).

C. Experimental Results

1) Comparison With State of the Art: In the loss function, we calculate the edge loss over all pixels instead of only on positive ones as A-CE2P does. Table I shows the comparison between both methods based on PlaneRCNN, and it can be seen that the model using all pixels for edge loss provides a better result. The reason is that the area of edge regions is relatively small. Thus, the positive pixel strategy only constrains a small portion of pixels, leading to the problem that the features are only partly updated. In such cases, the nonedge pixels are ignored and may contain many more false positive predictions. However, when learning through the loss of each pixel of an image, the edge features of the whole image will be updated, and it brings performance improvement.

We then evaluate PlaneSeg on ScanNet and NYUv2 datasets quantitatively. Besides, we follow the tradition [13], [22] of not measuring AP metrics on the NYUv2 dataset. As is shown in Table II, PlaneSeg improves PlaneRCNN and PlaneEmbedding in terms of all evaluation metrics. We calculate the performance gain by dividing the difference by the original

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

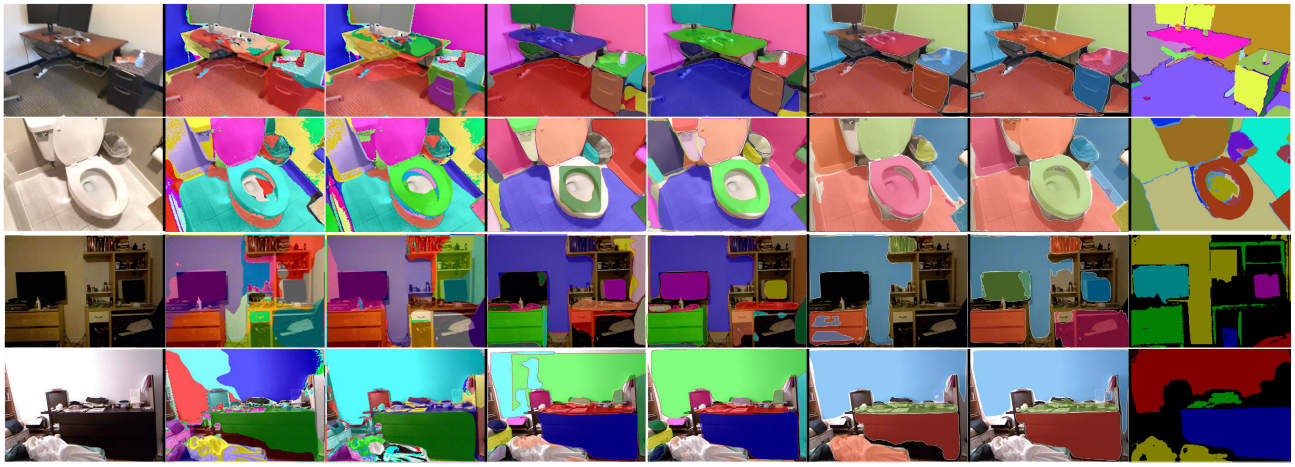

Fig. 3. Plane segmentation results in ScanNet (the first two rows) and NYUv2 (the last two rows) datasets. Source denotes the input RGB image. P.E., P.R., P.N., and P.S. denote PlaneEmbedding, PlaneRCNN, PlaneRecNet, and PlaneSeg, respectively. Please note that ground truth segmentations are generated by graph cut methods [12], [90] instead of human annotation, and thus contain errors (e.g., in row 1, the graph cut method mistakenly connects part of the wall with the whiteboard). Nevertheless, even though there are incorrect instance segmentation annotations in the datasets, our method can still successfully learn plane features during training to generate correct plane segmentation results.

TABLE II

PLANE SEGMENTATION RESULTS ON SCANNET AND NYUV2 DATASETS. P.E., P.R., P.N., AND P.S. DENOTE PLANEEMBEDDING, PLANERCNN, PLANERECNET, AND PLANESEG, RESPECTIVELY. AR^{plane} AND AR^{pixel} DENOTE PER-PLANE RECALL AND PER-PIXEL RECALL UNDER A DEPTH THRESHOLD OF 1.0 M, RESPECTIVELY. BESIDES, WE FOLLOW THE TRADITION [13], [22] OF NOT MEASURING AP METRICS ON THE NYUV2 DATASET. SEG, DET, REC, AND DEP REPRESENT PLANE SEGMENTATION, PLANE DETECTION, PLANE RECONSTRUCTION, AND DEPTH PREDICTION, RESPECTIVELY

Dataset	Task	Metrics	P.E. [22]	P.E.+P.S.	P.R. [13]	P.R.+P.S.	P.N. [66]	P.N.+P.S.
ScanNet	Seg	VoI↓	1.633	**1.360**	2.011	**1.964**	2.534	**2.458**
		RI↑	0.849	**0.886**	0.853	**0.863**	0.823	**0.833**
		SC↑	0.680	**0.757**	0.605	**0.620**	0.541	**0.549**
	Det	$AP^{0.3m}$ ↑	0.072	**0.128**	0.253	**0.255**	0.419	**0.432**
		$AP^{0.6m}$ ↑	0.165	**0.250**	0.331	**0.343**	0.497	**0.513**
		$AP^{0.9m}$ ↑	0.224	**0.297**	0.348	**0.363**	0.510	**0.528**
	Rec	AR^{plane} ↑	0.249	**0.323**	0.367	**0.380**	0.599	**0.613**
		AR^{pixel} ↑	0.373	**0.466**	0.633	**0.643**	0.743	**0.766**
NYUv2	Seg	VoI↓	1.815	**1.684**	2.243	**1.160**	1.961	**1.497**
		RI↑	0.838	**0.856**	0.818	**0.824**	0.812	**0.816**
		SC↑	0.700	**0.741**	0.380	**0.385**	0.371	**0.379**
	Dep	Rel ↓	0.203	**0.191**	0.203	**0.191**	0.194	**0.178**
		log_{10}↓	0.084	**0.081**	0.084	**0.081**	0.082	**0.079**
		RMS↓	0.691	**0.645**	0.691	**0.645**	0.679	**0.634**
		σ_1↑	0.673	**0.692**	0.673	**0.692**	0.685	**0.709**
		σ_2↑	0.912	**0.925**	0.912	**0.925**	0.919	**0.931**
		σ_3↑	0.978	**0.990**	0.978	**0.982**	0.980	**0.984**

value. Performance gains (calculated by the relative change) on the ScanNet dataset over PlaneRCNN [13] are 2.3%, 1.2%, 2.5%, 0.8%, 3.6%, 4.3%, 3.4%, and 1.5%, and the gains over PlaneEmbedding [22] are 16.7%, 4.4%, 11.3%, 77.8%, 51.5%, 32.6%, 29.9%, and 25.2%, for the metrics VOI, RI, SC, $AP^{0.3\,m}$, $AP^{0.6\,m}$, $AP^{0.9\,m}$, per-plane recall, and per-pixel recall (depth threshold = 1), respectively.

Also, following [22], to validate the generalization ability of PlaneSeg, we directly use all models trained on ScanNet for evaluation on NYUv2. On the unseen NYUv2 dataset, our PlaneSeg can improve PlaneRCNN by 3.7%, 0.73%, and 1.3%, and improve PlaneEmbedding by 7.2%, 2.1%, and 5.9% for VOI, RI, and SC, respectively. Again, in both quantitative and qualitative evaluation, PlaneSeg brings a significant performance boost to them, which suggests that our PlaneSeg maintains the generalization ability of models trained on ScanNet.

Besides, we show the segmentation results of our proposed method in Fig. 3. From these illustrations, the existing three methods fail to detect and segment some objects with complex boundaries, such as the closestool in the second row and the irregular wall in the fourth row. However, PlaneSeg is able to constrain the contours of planar regions even for small objects by utilizing the information of edge features and multiscale context, and by recovering dropped pixels. We can observe major improvements in Fig. 3, e.g., the problem of incorrect segmentation of wall is solved by ours, and the mask of closestool is corrected with the help of the edge information (the second row). Also, our proposed method is capable of detecting small planar regions compared with the original methods, e.g., the monitor in the third row.

Based on Table II and Fig. 3, we analyze how PlaneSeg improves the two models in different ways. That is, why PlaneSeg brings a relatively large improvement to PlaneEmbedding than PlaneRCNN? The reason is that our PlaneSeg contributes to PlaneEmbedding by merging scattered regions. The segmentation results for PlaneEmbedding have remarkable characteristics, wherein even though the edges of masks are finer compared with PlaneRCNN, the mean shift clustering algorithm in PlaneEmbedding introduces small dispersive clusters in the plane instance, which explains why VoI, RI, and SC perform better, while other metrics are lower on ScanNet. From the definitions of VoI, SC, and RI, it can be known that when a plane is divided into two regions with one occupying a large region and the other a small region, these three metrics will suffer less influence than APs and recalls do. APs and recalls will evidently decrease because the smaller region does not satisfy an IoU of 0.5 (the threshold in our experiments). When adding our PlaneSeg, the small regions merge together because the model is fed with edge information, and thus, can better distinguish features within a plane and near boundaries. Consequently, PlaneSeg improves PlaneEmbedding in terms of all metrics, especially for APs and recalls. Second, when integrated into PlaneRCNN, PlaneSeg is conducive to learning accurate edges. With the region

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

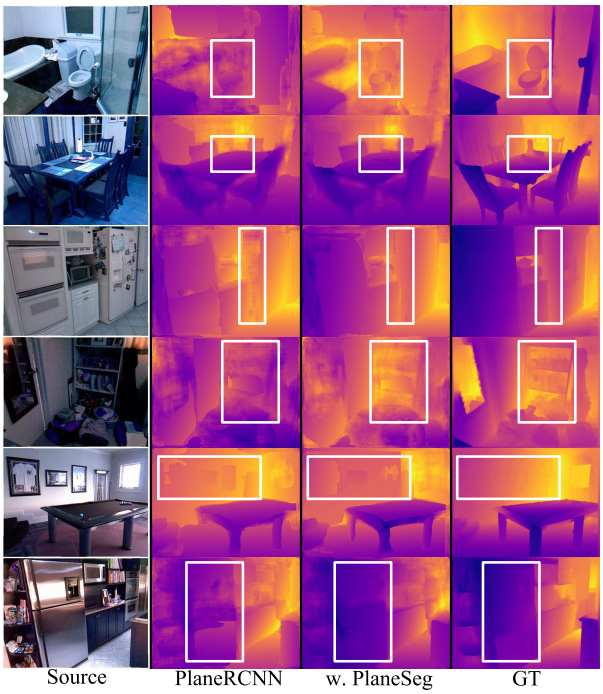

Fig. 4. Results of depth prediction on NYUv2 dataset. The selected samples include cluttered (first), small (second), as well as narrow planar regions (third) and are collected from overdark and overlight scenes (fourth, fifth, and sixth).

proposal technique, PlaneRCNN does not produce annoying scattered points as PlaneEmbedding does, which is part of the reason why PlaneRCNN obtains higher APs and recalls. However, the masks tend to have irregular curving edges. From the image pairs in Fig. 3, PlaneSeg successfully extracts edge features that further constrain the contours of planar regions. For example, in the first row, the black side of the drawer and the floor with shadow are grouped together by PlaneRCNN, but they are separated by the edge features learned by PlaneSeg. When integrated into PlaneEmbedding, PlaneRecNet, and PlaneRCNN, PlaneSeg is connected at different positions. In PlaneEmbedding, PlaneSeg is placed at a higher level, while in PlaneRCNN it is placed at a lower level. We believe this difference can partially explain why the performance gains are smaller in PlaneRCNN. We also infer that PlaneSeg will bring more performance boosts if it is placed at a higher level.

Furthermore, we demonstrate how our PlaneSeg improves 3-D reconstruction and depth prediction. We verify the improvement of PlaneSeg for depth prediction. As shown in Fig. 4, we see a significant improvement in solving difficult scenarios with PlaneSeg, even in overdark and overlight scenes, for detecting cluttered, small, and narrow planes. These results prove the effectiveness of our proposed method in an intuitive way. Meanwhile, the 3-D reconstruction results on ScanNet are shown in Fig. 5. By incorporating edge information, our PlaneSeg produces much finer results near edges, e.g., in the second column of the top group, a small part of the wall and table are no longer indiscernible. Also, benefitting from the multiscale information, our PlaneSeg achieves better results on both small and large objects, e.g., the large door and wall in the first column of the top group are better detected. The small white pillar in the fourth column of the bottom group, and the small planar objects in the third column of the top group, can now be correctly segmented. The broken segmentation of the wall in the third column of the bottom group is also fixed.

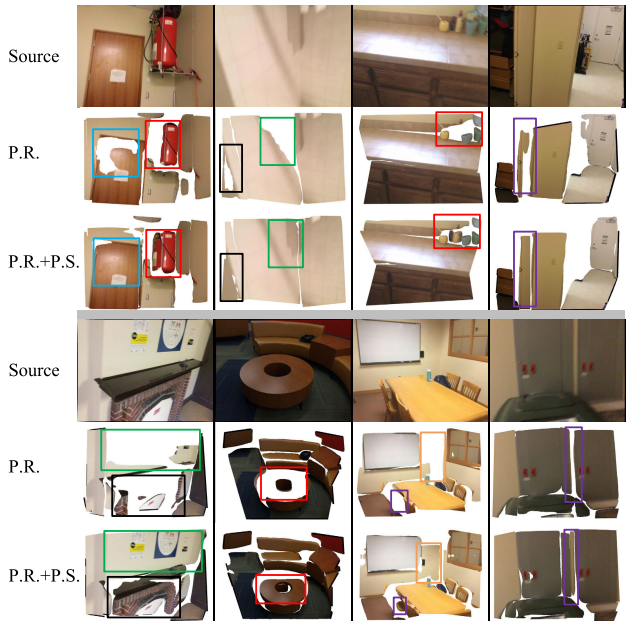

Fig. 5. Reconstruction results on ScanNet dataset. We follow [13] of providing 3-D rendering from segmentation results.

TABLE III

INFERENCE TIME RESULTS ON SCANNET AND NYUV2 USING A SINGLE NVIDIA GEFORCE GTX 1080 TI GPU. ALL RESULTS ARE MEASURED IN MILLISECONDS. P.E., P.R., P.N., AND P.S. DENOTE PLANEEMBEDDING [22], PLANERCNN [13], PLANERECNET [66] AND OUR PROPOSED PLANESEG, RESPECTIVELY

	P.E. [22]	P.E.+P.S.	P.R. [13]	P.R.+P.S.	P.N. [66]	P.N.+P.S.
ScanNet	80.9	87.4	303.9	317.0	174.7	176.9
NYUv2	79.6	95.2	287.1	294.3	160.0	164.9
Average	80.3	91.3	295.5	305.7	167.3	170.9
Diff.	11.0		10.2		3.5	

Next, we conduct experiments testing the inference time. All models run on a single NVIDIA GeForce GTX 1080 Ti graphic card. Table III shows the average inference time measured in milliseconds. On average, PlaneSeg takes an additional 10.6 ms for inference. Compared with their original inference times, PlaneEmbedding, PlaneRCNN, and PlaneRecNet methods integrated with PlaneSeg run 13.69%, 3.45%, and 2.09% slower, respectively. Also, the statistic composition of the network parameters in Table IV fairly exploits the flexibility of our modules. As can be seen, only 8.38%, 7.06%, and 9.80% of parameters come from PlaneSeg when it is integrated with PlaneRCNN, PlaneEmbedding, and PlaneRecNet respectively. Compared with the number of whole network parameters in P.S. + P.E. (i.e., 45.90 M), our resolution-adaptation module only owns 0.09 M parameters, which accounts for 0.2% but provides a performance improvement. The reason lies in that the resolution-adaptation module helps recover dropped pixels in small feature maps. Taking the edge feature and context feature as input, the resolution-adaptation module efficaciously fuses feature pairs of the same resolution. Meanwhile, our resolution-adaptation module only consists of a few convolution layers with elementwise addition, producing a few computation costs and parameters.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV
COMPOSITION OF THE PROPOSED NETWORK PARAMETERS. E, C, AND R DENOTE THE EDGE FEATURE EXTRACTION MODULE, MULTISCALE MODULE, AND RESOLUTION-ADAPTATION MODULE, RESPECTIVELY. *: P.E. [22] DECODES FEATURE MAPS ONLY BY A FEW PARALLELED CONVOLUTION LAYERS. AS CAN BE SEEN, PLUG-IN MODULES TAKE UP THE MINORITY

PlaneSeg+PlaneRCNN: 89.17M					
Modules	PlanSeg: 7.48M (8.38%)			original: 81.69M (91.61%)	
	E	C	R	Backbone	Decoder
Parameters	2.68M	3.34M	1.45M	42.55M	39.14M
Proportion	3.01%	3.75%	1.62%	47.72%	43.89%
PlaneSeg+PlaneEmbedding: 45.90M					
Modules	PlanSeg: 3.24M (7.06%)			original: 42.66M (92.94%)	
	E	C	R	Backbone	Decoder*
Parameters	2.87M	0.28M	0.09M	42.62M	0.04M
Proportion	6.25%	0.61%	0.20%	92.87%	0.08%
PlaneSeg+PlaneRecNet: 63.56M					
Modules	PlanSeg: 6.23M (9.80%)			original: 57.33M (90.19%)	
	E	C	R	Backbone	Decoder
Parameters	2.68M	3.34M	0.21M	43.18M	14.15M
Proportion	4.21%	5.25%	0.33%	67.93%	22.26%

2) Comparison With Model Variants: In PlaneSeg, several minor architectures are utilized to make the model more effective. Three of them are considered of utmost significance: the exclusion of C_b^5, step-down layers in the edge feature extraction module, and the pairwise fusion operation in the resolution adaption module. To prove these minor architectures are effective, we conduct comparative experiments with some model variants. We build the model variants by varying one or more of the above-mentioned architectures. For convenience, we named the changes as: *with* C_e^5, *without step-down layer*, and *without pairwise fusion*, respectively. *With* C_e^5 denotes generating an additional edge feature from C_b^5 through the function of $\mathrm{Ed}_1(\cdot)$. Like all other edge features, the extra feature is used to produce the edge probability map, we then generate P_n^5 by attaching an addition operation between P_o^5 and C_e^5. P_n^6 is also renewed as the $2\times$ downsample of P_n^5. *Without step-down layer* denotes replacing the step-down layer with a 1×1 conv layer in the edge feature extraction module (the brute way). *Without pairwise fusion* denotes replacing the $\mathrm{Tr}_i(\cdot)$ with a trivial concatenate with 1×1 conv layer in the multiscale module.

Table V shows the segmentation performance of an individual or composed model variants. The experimental results illustrate that all model variants bring significant performance degradation. Here, we will give our explanations. First, the *With* C_e^5 variant suffers from the feature size of C_e^5 being too small. Even though going through the same process as other edge features do, detailed information is lost heavily due to the small feature map size, making the edge lines in the generated edge probability map ambiguous. Second, the performance degradation of *without step-down layer* variant is related to the loss function. In a planar region segmentation model, the backbone network is constrained by a combination of multiple loss functions, not merely edge loss. Consequently, the backbone feature maps themselves do not contain sufficient edge information for an edge prediction task. That is to say, with only one conv layer, the "without step-down layer" variant fails to extract effective edge information from ResNet blocks. Last, the *without pairwise fusion* variant adopts a traditional feature fusion method: a concatenation with 1×1 conv layer. It is worth noticing that this structure is actually a superset of an additional layer. However, when fusing edge features with other features of the same image, they should be interpreted as "correction." The addition operation naturally has an advantage in expressing the concept of "correction," while the traditional fusion method is more suitable for merging two features that are less relevant.

D. Ablation Studies

We perform ablation experiments based on PlaneRCNN to evaluate each module in our PlaneSeg. All experiments are conducted on the ScanNet dataset, and the results are shown in Table VI. In our basic model (we name it plain CNN), we do not add any proposed modules and simply use ResNet-101 as the backbone. Since the channel numbers of C_b^i $(i = 3, 4, 5)$ do not match those of P_o^i, we cannot directly remove the three modules. To reduce feature channels, we apply simple 1×1 *conv* layers to C_b^i. Also, to get P_o^6, we follow the original FPN function, i.e., $P_o^6 = Py_6(P_o^5)$. Results verify the positive contribution of each component, and we give the details of applying the three modules as follows.

1) Edge Feature Extraction Module: First, we analyze the performance of the edge feature extraction module that learns edge information. Based on the basic model (plain CNN), we add a network branch that produces an edge probability map from C_b^2, C_b^3, and C_b^4. This network branch incorporates learned edge information into C_b^i. The connection between this model and the segmentation model is the same as the basic model (plain CNN). To train this module, we use the edge loss $\mathcal{L}_{\text{edge}}$ described in Section III-C. Because we have not added the multiscale module, the contribution of $\mathcal{L}_{\text{edge}}$ to the basic model is that it updates the ResNet-101 backbone parameters. From Table VI, this module can bring about an average improvement of 4.4% on the six metrics.

Next, experimental results of quantitative comparisons with more existing edge modules are provided in Table VII, including CE2P [24], boundary-guided aggregation network (BFAN) [95] and PAGE-Net [85]. CE2P is for the human parsing task. Since CE2P has a similar structure compared with our proposed PlaneSeg, we did not change the structure when combining its edge module with the two plane segmentation models. Meanwhile, BFAN is designed for SOD. It conducts attention mechanisms for fusing features and boundary cues. PAGE-Net is also a deep network proposed for SOD, which deploys a pyramid attention technique. For larger feature maps, PAGE-Net performs more attention operations. Considering the number of parameters introduced by conv layers with the kernel size of 3×3, we modify the kernel size to 1×1 in attention modules.

We also conduct trimap evaluation over the plain PlaneRCNN as well as the PlaneRCNN integrated with edge modules. Trimap evaluates the classification accuracy near the actual object boundaries, and thus, is useful for measuring whether a method can produce masks with accurate boundaries. Following [96], we evaluate on ScanNet with bandwidth values no greater than 20. As shown in Fig. 6, PlaneSeg delivers better results compared with previous works, especially when the bandwidth exceeds 3. This result proves our edge module can effectively learn accurate edges and utilize this information to resolve the problem of ambiguous edges.

2) Multiscale Module: Second, we conduct experiments by introducing a multiscale module. Specifically, based on the

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE V

SEGMENTATION RESULTS OF MODEL VARIANTS ON SCANNET DATASET. THE LAST ROW REFLECTS THE AVERAGE DEGRADATION OF THE RELATIVE CHANGES BETWEEN EACH VARIANT AND OURS (FIRST COLUMN)

Module	With C_e^5	-	✓	-	-	✓	✓	-	✓
	W/o pair-wise fusion	-	-	✓	-	-	✓	✓	✓
	W/o stepdown layer	-	-	-	✓	✓	-	✓	✓
Segmentation	VoI↓	1.964	1.968	1.980	1.969	2.005	1.990	2.041	2.030
	RI↑	0.863	0.863	0.863	0.863	0.859	0.861	0.856	0.855
	SC↑	0.620	0.618	0.615	0.617	0.610	0.613	0.602	0.603
Detection	$AP^{0.3m}$ ↑	0.255	0.181	0.180	0.165	0.180	0.166	0.164	0.156
	$AP^{0.6m}$ ↑	0.343	0.250	0.247	0.235	0.253	0.237	0.233	0.228
	$AP^{0.9m}$ ↑	0.363	0.332	0.328	0.320	0.335	0.323	0.314	0.320
Average	Difference	-0.00%	-10.87%	-11.44%	-13.32%	-11.25%	-13.25%	-14.81%	-15.20%

TABLE VI

ABLATIONS OF DIFFERENT COMPONENTS OF PLANESEG, INCLUDING THE EDGE FEATURE EXTRACTION MODULE (E), MULTISCALE MODULE (C), AND RESOLUTION-ADAPTATION MODULE (R)

Module			Segmentation			Detection		
E	C	R	VoI↓	RI↑	SC↑	$AP^{0.3m}$ ↑	$AP^{0.6m}$ ↑	$AP^{0.9m}$ ↑
-	-	-	2.196	0.846	0.569	0.170	0.244	0.261
✓	-	-	2.187	0.848	0.572	0.196	0.258	0.272
✓	✓	-	1.980	0.862	0.615	0.222	0.328	0.355
✓	✓	✓	**1.964**	**0.863**	**0.620**	**0.255**	**0.343**	**0.363**

TABLE VII

SEGMENTATION RESULTS ON SCANNET AND NYUV2 DATASETS USING DIFFERENT EDGE MODULES. BESIDES, WE FOLLOW THE TRADITION [13], [22] OF NOT MEASURING AP METRICS ON THE NYUV2 DATASET

Base	Edge	ScanNet						NYUv2		
		VoI	RI	SC	$AP^{0.3m}$	$AP^{0.6m}$	$AP^{0.9m}$	VoI	RI	SC
P.E. [22]	-	1.633	0.849	0.680	0.072	0.165	0.224	1.815	0.838	0.700
	CE2P [24]	1.924	0.827	0.592	0.068	0.147	0.201	1.959	0.830	0.646
	BFAN [95]	1.818	0.848	0.642	0.066	0.151	0.207	2.542	0.782	0.505
	PAGE [85]	1.612	0.864	0.694	0.078	0.178	0.246	1.917	0.834	0.670
	Ours	**1.360**	**0.886**	**0.757**	**0.128**	**0.250**	**0.297**	**1.684**	**0.856**	**0.741**
P.R. [13]	-	2.011	0.853	0.605	0.253	0.331	0.348	2.243	0.818	0.380
	CE2P	2.033	0.856	0.605	0.250	0.329	0.347	2.168	**0.835**	**0.395**
	BFAN [95]	2.178	0.842	0.573	0.194	0.269	0.285	3.011	0.748	0.316
	PAGE [85]	2.046	0.855	0.603	0.239	0.328	0.347	2.164	0.829	0.389
	Ours	**1.964**	**0.863**	**0.620**	**0.255**	**0.343**	**0.363**	**2.160**	0.824	0.385

previous model with only the edge feature extraction module, we add the multiscale module to produce P_o^i, $(i = 2, 3, 4, 5, 6)$, which are then fed to the segmentation model. In Table VI, we can find that using this module brings an average improvement of 14.9% on the six metrics. It is possible that this is because by learning and integrating multiscale context information, the model can detect not only distinct large planes but also small-sized planar regions.

In order to prove that the multiscale module facilitates detecting small-sized planes, we next provide experimental results with regard to different plane sizes. Planes are divided into three categories: small, medium, and large. To achieve a fair division, we counted the size values of all the planes in the test set. Then sort them from small to large, and divide them into three parts that contain the same amount of planes. Each category contains precisely 1/3 of the number of planes in the test set. On our selected test set of ScanNet, the size thresholds between small, medium, and large are 2267 and 10 500 pixels, respectively.

We compare PlaneSeg with existing multiscale methods. PSPNet [97] proposes a hierarchical strategy of pooling to grasp multiscale information. Atrous spatial pyramid pooling (ASPP) [98] employs dilated convolution [80] to harvest multiscale context.

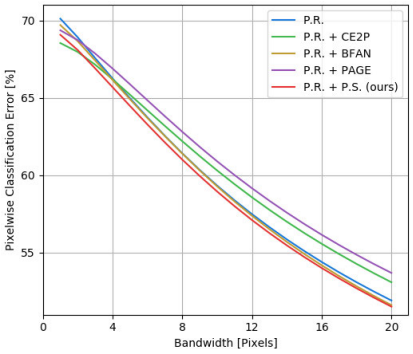

Fig. 6. Segmentation accuracy within trimap (a narrowband surrounding actual segmentation boundaries). P.R. and P.S. denote PlaneRCNN and PlaneSeg, respectively. Bandwidth represents the width of the trimap. Pixelwise classification error is the percentage of misclassified pixels within trimap.

TABLE VIII

PLANE SEGMENTATION AND DETECTION RESULTS WITH REGARD TO DIFFERENT PLANE SIZES. EVALUATION IS CONDUCTED WITH PLANERCNN ON SCANNET DATASET. "−" DENOTES PLANERCNN WITHOUT MULTISCALE MODULE. AP UNDER THREE THRESHOLDS ARE USED FOR PLANE DETECTION PERFORMANCE MEASUREMENT. MAP, MAR, AND MIOU ARE USED FOR IMAGE SEGMENTATION PERFORMANCE MEASUREMENT BECAUSE THE PREVIOUS METRICS CAN NOT APPLY TO A SUBSET OF A SEGMENTATION MAP

Size	Metrics	Multi-scale Module				Size	Multi-scale Module			
		-	PSPNet	ASPP	Ours		-	PSPNet	ASPP	Ours
All	$AP^{0.3m}$	0.196	0.208	0.241	**0.255**	Medium	0.042	0.074	0.076	**0.083**
	$AP^{0.6m}$	0.258	0.300	0.317	**0.323**		0.058	0.105	0.109	**0.121**
	$AP^{0.9m}$	0.272	0.326	0.330	**0.363**		0.058	0.105	0.109	**0.121**
	mAP	0.559	**0.663**	0.642	0.661		0.195	0.249	0.240	**0.250**
	mAR	0.475	0.520	0.523	**0.535**		0.526	0.617	0.617	**0.633**
	mIOU	0.352	0.431	0.416	**0.439**		0.106	0.141	0.137	**0.145**
Small	$AP^{0.3m}$	0.001	0.006	0.007	**0.008**	Large	0.404	0.361	0.433	**0.441**
	$AP^{0.6m}$	0.001	0.010	0.010	**0.012**		0.518	0.505	0.537	**0.555**
	$AP^{0.9m}$	0.001	0.010	0.010	**0.013**		0.540	0.541	0.552	**0.581**
	mAP	0.050	0.084	0.081	**0.091**		0.397	**0.429**	0.411	0.415
	mAR	0.095	0.142	0.114	**0.154**		0.872	0.880	0.881	**0.887**
	mIOU	0.019	0.037	0.036	**0.042**		0.250	**0.280**	0.267	0.278

Per-pixel recall and per-plane recall under different plane sizes are evaluated. The recall values are collected on the ScanNet dataset. As shown in Fig. 7, ours provides a significant performance improvement across all sizes, especially for small planar regions. Among small-sized planar regions, there is a leap between ours and the plain one without a multiscale module, while the gap is gradually reduced as plane size increases. This result shows our multiscale module can improve the performance when detecting small-scale planes.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

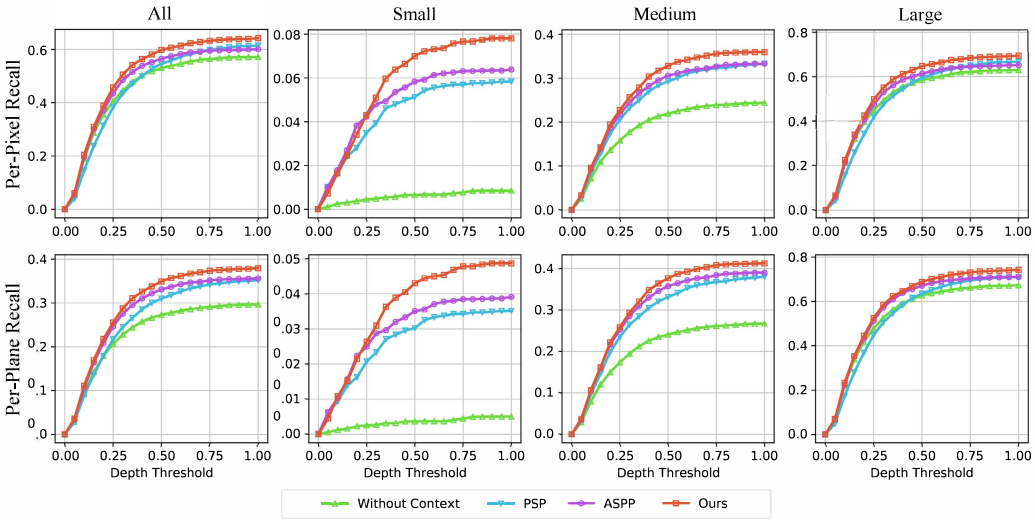

Fig. 7. Per-pixel and per-plane recall curve with regard to different plane sizes on ScanNet. We obtain different recall values under different models and depth thresholds. The first row shows per-pixel recall curves, while the second row shows per-plane recall curves. Each column in the figure represents experimental results under a specific plane size range.

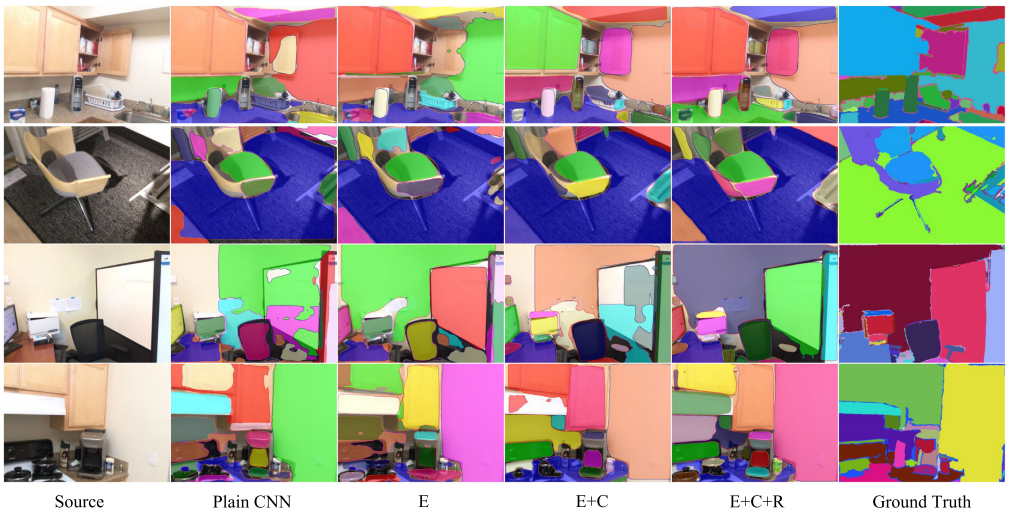

Fig. 8. Visualization results of different ablation models, including the edge feature extraction module (E), multiscale module (C), and resolution-adaptation module (R). The images come from the ScanNet dataset.

Moreover, we provide plane segmentation and detection results of different plane sizes in Table VIII. Because previously used metrics VoI, RI, and SC are not suitable to evaluate a subset of the segmentation result, we adopt three metrics that are commonly used in image segmentation tasks: mAP, mAR, and mIOU to measure the quality of produced plane masks. The experimental results show that, over the six metrics, the multiscale module can not only bring a great improvement compared with the plain PlaneRCNN method but also outperform other similar modules.

Performance improvements are mainly due to the effective integration of shallow and deep features by the multiscale module. Generally speaking, only the coarsest features are utilized in traditional CNNs, while the superficial features are often ignored. A considerable amount of detailed information is lost in deep features due to their small size. However, detailed information is extremely important when detecting and segmenting small planar regions. By integrating feature maps of different layers, our approach preserves the details in shallow features without losing semantic information learned in deeper layers.

3) Resolution-Adaptation Module: Finally, we evaluate the performance of the resolution-adaptation module. As shown in Table VI, with the above-mentioned modules, this module leads to an average improvement of 3.9% on the six metrics. The reason is threefold. First, the resolution-adaptation module primarily utilizes elementwise addition to correct the feature maps of small-sized planar objects. Second, our module adopts the form of learnable to perform recovery. Third, there are many small planar regions in an image, which can be treated as a fine-grained pixelwise classification. Therefore, much detailed information extracted by the resolution-adaptation module assists the network in accurately identifying various small-sized planes. Essentially, the three aforementioned modules are coupled and supportive of each other. The edge module and the multiscale module provide fundamental contours and context features, respectively. Next, the resolution-adaptation module integrates the above-mentioned features pairwisely and recovers the dropped pixels at higher network levels.

4) Qualitative Evaluation With Different Modules: The qualitative comparisons with different modules on ScanNet

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

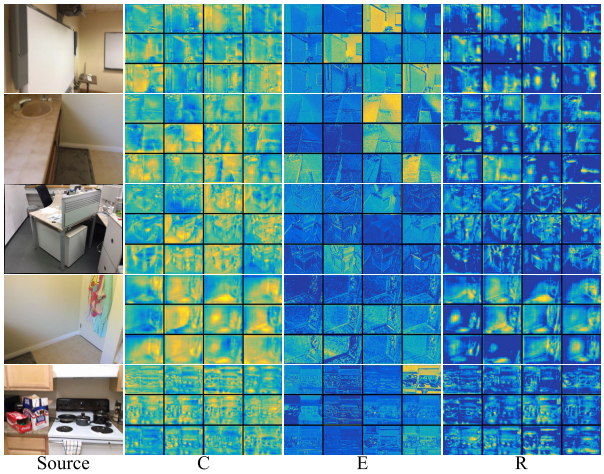

Fig. 9. Visualization of feature maps. Obtained from the multiscale module (C), edge feature extraction module (E), and resolution-adaptation module (R), respectively. Given the features obtained from C and E, our proposed resolution-adaptation module mitigates the gap between them and extracts semantic information for detecting planes, where the plane areas are assigned with high activations.

are shown in Fig. 8. We use E, C, and R to denote the edge feature extraction module, multiscale module, and resolution-adaptation module, respectively. As observed in Fig. 8, the planar region detection accuracy increases as more modules are attached. First, the edge feature extraction module helps differentiate the plane areas more clearly. The edge information of planar regions is an underlying constraint that mitigates the vague boundaries of nearby planes. It resolves the stickiness problem of different planar regions in the original method, as well as that of a planar region being incorrectly divided into multiple small ones. As shown in the third row, in the original method, the upper part of the whiteboard is mixed with the wall, which led to the whole whiteboard being erroneously divided into several small areas. At the same time, the lower part of the wall is also mistakenly divided into another small area. With the edge feature extraction module, the whiteboard and the wall are correctly separated by utilizing the edge information. Second, the multiscale module enhances the ability of the model to detect small-scale planes. For example, the cabinet doors in the first row, the cabinets on the table in the third row, and the objects on the table in the fourth row are difficult to segment correctly without this module. However, these small and messy objects can be clear as well as accurately separated if the multiscale module is added. Finally, the resolution-adaption module integrates the above-mentioned two modules to generate high-quality planar inspection results.

Based on Fig. 8, Fig. 9, and the aforementioned analysis, we make the following observations: 1) plain CNN has some issues, such as vague boundaries, while the model of E can mitigate this effect by utilizing the edge information. 2) The model of $E+C$ captures more detailed information than the model of E. 3) The proposed PlaneSeg (i.e., $E+C+R$) generates better segmentation results due to the resolution-adaption module that integrates edge features and multiscale context information pairwisely, then recovers dropped pixels in small feature maps.

V. Conclusion

In this article, we propose a universal planar region segmentation framework, PlaneSeg, which is end-to-end trainable and can be easily integrated into various plane segmentation models to improve performance. PlaneSeg consists of three key modules, namely, the edge feature extraction module, the multiscale module, and the resolution-adaptation module. Specifically, we use the edge feature extraction module to learn edge features and constrain object contours, use the multiscale module for extracting multiscale semantic information, and use the resolution-adaptation module to integrate the former features pairwisely and capture the lost details of features in higher layers. The experimental results demonstrate that the proposed PlaneSeg can boost existing methods to achieve the new state-of-the-art on the ScanNet and NYUv2 datasets by a large margin.

References

[1] D. Kim, S. Chae, J. Seo, Y. Yang, and T.-D. Han, "Realtime plane detection for projection augmented reality in an unknown environment," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 5985–5989.

[2] Y. Kim and H. Woo, "Integrating a deep learning-based plane detector in mobile AR systems for improvement of plane detection," in *Proc. 8th Int. Conf. Comput. Artif. Intell.*, Mar. 2022, pp. 597–602.

[3] Z. Fu and W. Hu, "Dynamic point cloud inpainting via spatial–temporal graph learning," *IEEE Trans. Multimedia*, vol. 23, pp. 3022–3034, 2021.

[4] H. Park, H. Lee, and S. Sull, "Efficient viewer-centric depth adjustment based on virtual fronto-parallel planar projection in stereo 3D images," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 326–336, Feb. 2014.

[5] J. Liu et al., "PlaneMVS: 3D plane reconstruction from multi-view stereo," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8665–8675.

[6] H. Zhu et al., "VPFNet: Improving 3D object detection with virtual point based LiDAR and stereo data fusion," *IEEE Trans. Multimedia*, early access, Jul. 11, 2022, doi: 10.1109/TMM.2022.3189778.

[7] S. Yang and S. Scherer, "Monocular object and plane SLAM in structured environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3145–3152, Oct. 2019.

[8] V. Patil, C. Sakaridis, A. Liniger, and L. Van Gool, "P3Depth: Monocular depth estimation with a piecewise planarity prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1610–1621.

[9] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "PlaneNet: Piece-wise planar reconstruction from a single RGB image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2579–2588.

[10] Y. Li, K. W. Wan, X. Yan, and C. Xu, "Real time advertisement insertion in baseball video based on advertisement effect," in *Proc. 13th Annu. ACM Int. Conf. Multimedia*, Nov. 2005, pp. 343–346.

[11] P. Kim, B. Coltin, and H. J. Kim, "Linear RGB-D SLAM for planar environments," in *Proc. ECCV*, 2018, pp. 333–348.

[12] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[13] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz, "PlaneRCNN: 3D plane detection and reconstruction from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4450–4459.

[14] L. Lin, W. Zhang, M. Cheng, C. Wen, and C. Wang, "Planar primitive group-based point cloud registration for autonomous vehicle localization in underground parking lots," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[15] A. Roychoudhury, M. Missura, and M. Bennewitz, "Plane segmentation in organized point clouds using flood fill," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 13532–13538.

[16] L. Huynh, P. Nguyen-Ha, J. Matas, E. Rahtu, and J. Heikkilä, "Guiding monocular depth estimation using depth-attention volume," in *Proc. ECCV*, 2020, pp. 581–597.

[17] Z. Yu, L. Jin, and S. Gao, "P^{2}Net: Patch-match and plane-regularization for unsupervised indoor depth estimation," in *Proc. ECCV*, 2020, pp. 206–222.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[18] L. He et al., "Rethinking supervised depth estimation for 360° panoramic imagery," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 1–9.

[19] E. Delage, H. Lee, and A. Y. Ng, "Automatic single-image 3D reconstructions of indoor Manhattan world scene," in *Proc. ISRR*, 2007, pp. 305–321.

[20] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 1849–1856.

[21] F. Yang and Z. Zhou, "Recovering 3D planes from a single image via convolutional neural networks," in *Proc. ECCV*, 2018, pp. 85–100.

[22] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, "Single-image piecewise planar 3D reconstruction via associative embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1029–1037.

[23] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, 2017, pp. 2961–2969.

[24] T. Ruan et al., "Devil in the details: Towards accurate single and multiple human parsing," in *Proc. AAAI*, 2018, pp. 4814–4821.

[25] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 328–335.

[26] J. Yao, S. Fidler, and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 702–709.

[27] S. Hao, Y. Zhou, Y. Guo, R. Hong, J. Cheng, and M. Wang, "Real-time semantic segmentation via spatial-detail guided context propagation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 8, 2022, doi: 10.1109/TNNLS.2022.3154443.

[28] L. Yu, X. Liu, and J. Van De Weijer, "Self-training for class-incremental semantic segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 17, 2022, doi: 10.1109/TNNLS.2022.3155746.

[29] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "PersonLab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *Proc. ECCV*, 2018, pp. 269–286.

[30] T. Li, K. Zhang, S. Shen, B. Liu, Q. Liu, and Z. Li, "Image co-saliency detection and instance co-segmentation using attention graph clustering based graph convolutional network," *IEEE Trans. Multimedia*, vol. 24, pp. 492–505, 2022.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 1–9.

[32] P. Luc, C. Couprie, Y. LeCun, and J. Verbeek, "Predicting future instance segmentation by forecasting convolutional features," in *Proc. ECCV*, 2018, pp. 584–599.

[33] H. Zhang, Y. Tian, K. Wang, W. Zhang, and F.-Y. Wang, "Mask SSD: An effective single-stage approach to object instance segmentation," *IEEE Trans. Image Process.*, vol. 29, pp. 2078–2093, 2019.

[34] W. Liu et al., "SSD: Single shot MultiBox detector," in *Proc. ECCV*, 2016, pp. 21–37.

[35] S. Li, Y. Liu, and J. Gall, "Rethinking 3-D LiDAR point cloud segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 16, 2021, doi: 10.1109/TNNLS.2021.3132836.

[36] S. P. Lim and H. Haron, "Cube Kohonen self-organizing map (CKSOM) model with new equations in organizing unstructured data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1414–1424, Sep. 2013.

[37] J.-G. Wu et al., "Machine learning for structure determination in single-particle cryo-electron microscopy: A systematic review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 452–472, Feb. 2022.

[38] J. Ma, J. Wu, J. Zhao, J. Jiang, H. Zhou, and Q. Z. Sheng, "Nonrigid point set registration with robust transformation learning under manifold regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3584–3597, Dec. 2019.

[39] X. Zhang, Y. Zhuang, H. Hu, and W. Wang, "3-D laser-based multiclass and multiview object detection in cluttered indoor scenes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 177–190, Jan. 2017.

[40] T. Wan et al., "RGB-D point cloud registration based on salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3547–3559, Aug. 2022.

[41] J. Liu, Y. Wang, Y. Li, J. Fu, J. Li, and H. Lu, "Collaborative deconvolutional neural networks for joint depth estimation and semantic segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5655–5666, Nov. 2018.

[42] X. Chen, X. Chen, Y. Zhang, X. Fu, and Z.-J. Zha, "Laplacian pyramid neural network for dense continuous-value regression for complex scenes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 5034–5046, Nov. 2021.

[43] C. Zhang, L. Wang, and R. Yang, "Semantic segmentation of urban scenes using dense depth maps," in *Proc. ECCV*. Cham, Switzerland: Springer, 2010, pp. 708–721.

[44] A. Alush and J. Goldberger, "Hierarchical image segmentation using correlation clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1358–1367, Jun. 2016.

[45] Y. Xie, M. Gadelha, F. Yang, X. Zhou, and H. Jiang, "PlanarRecon: Realtime 3D plane detection and reconstruction from posed monocular videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6219–6228.

[46] K.-N. Lianos, L. Puig, A. Unagar, and S. Jiddi, "Robust planar optimization for general 3D room layout estimation," in *Proc. IEEE Int. Symp. Mixed Augmented Reality Adjunct (ISMAR-Adjunct)*, Oct. 2022, pp. 875–880.

[47] M. Dahnert, J. Hou, M. Nießner, and A. Dai, "Panoptic 3D scene reconstruction from a single RGB image," in *Proc. NeurIPS*, 2021, pp. 8282–8293.

[48] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang, "Total3DUnderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image," in *Proc. CVPR*, 2020, pp. 55–64.

[49] C. Sun, C.-W. Hsiao, N.-H. Wang, M. Sun, and H.-T. Chen, "Indoor panorama planar 3D reconstruction via divide and conquer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11338–11347.

[50] C. Wang and X. Guo, "Efficient plane-based optimization of geometry and texture for indoor RGB-D reconstruction," in *Proc. CVPRW*, 2019, pp. 49–53.

[51] Y. Li, C. Yang, W. Yan, R. Cui, and A. Annamalai, "Admittance-based adaptive cooperative control for multiple manipulators with output constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3621–3632, Dec. 2019.

[52] A. Roy, X. Zhang, N. Wolleb, C. P. Quintero, and M. Jagersand, "Tracking benchmark and evaluation for manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2448–2453.

[53] R. Balakrishnan, T. Baudel, G. Kurtenbach, and G. Fitzmaurice, "The Rockin'Mouse: Integral 3D manipulation on a plane," in *Proc. ACM SIGCHI Conf. Hum. Factors Comput. Syst.*, Mar. 1997, pp. 311–318.

[54] D. R. Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S.-H. Ieng, and R. Benosman, "An asynchronous neuromorphic event-driven visual part-based shape tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3045–3059, Mar. 2015.

[55] F. Ornelas-Tellez, E. N. Sanchez, and A. G. Loukianov, "Discrete-time neural inverse optimal control for nonlinear systems via passivation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1327–1339, Aug. 2012.

[56] T. Wang and H. Ling, "Gracker: A graph-based planar object tracker," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1494–1501, Jun. 2018.

[57] O. Haines and A. Calway, "Recognising planes in a single image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1849–1861, Sep. 2015.

[58] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *Int. J. Comput. Vis.*, vol. 75, pp. 0920–5691, Oct. 2007.

[59] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2008.

[60] A. Chuchvara, A. Barsi, and A. Gotchev, "Fast and accurate depth estimation from sparse light fields," *IEEE Trans. Image Process.*, vol. 29, pp. 2492–2506, 2020.

[61] J. Tan, W. Lin, A. X. Chang, and M. Savva, "Mirror3D: Depth refinement for mirror surfaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15990–15999.

[62] X. Wang, D. F. Fouhey, and A. Gupta, "Designing deep networks for surface normal estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 539–547.

[63] H. Ikoma, C. M. Nguyen, C. A. Metzler, Y. Peng, and G. Wetzstein, "Depth from defocus with learned optics for imaging and occlusion-aware depth estimation," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, May 2021, pp. 1–12.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[64] Z. Yang, L. E. Li, and Q. Huang, "StruMonoNet: Structure-aware monocular 3D prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7413–7422.

[65] Y. Xie, J. Rambach, F. Shu, and D. Stricker, "PlaneSegNet: Fast and robust plane estimation using a single-stage instance segmentation CNN," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 13574–13580.

[66] Y. Xie, F. Shu, J. Rambach, A. Pagani, and D. Stricker, "PlaneRecNet: Multi-task learning with cross-task consistency for piece-wise plane detection and reconstruction from a single RGB image," in *Proc. BMVC*, 2021, pp. 1–14.

[67] B. Tan, N. Xue, S. Bai, T. Wu, and G.-S. Xia, "PlaneTR: Structure-guided transformers for 3D plane recovery," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4186–4195.

[68] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and fast instance segmentation," in *Proc. NeurIPS*, 2020, pp. 17721–17732.

[69] J.-X. Zhao, J.-J. Liu, D.-P. Fan, Y. Cao, J. Yang, and M.-M. Cheng, "EGNet: Edge guidance network for salient object detection," in *Proc. ICCV*, 2019, pp. 8779–8788.

[70] H. Ding, X. Jiang, A. Q. Liu, N. M. Thalmann, and G. Wang, "Boundary-aware feature propagation for scene segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6819–6829.

[71] Y. Zhao, J. Li, Y. Zhang, and Y. Tian, "Multi-class part parsing with joint boundary-semantic awareness," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9177–9186.

[72] W. Zhang, X. Wang, W. You, J. Chen, P. Dai, and P. Zhang, "RESLS: Region and edge synergetic level set framework for image segmentation," *IEEE Trans. Image Process.*, vol. 29, pp. 57–71, 2019.

[73] X. Li, F. Yang, H. Cheng, W. Liu, and D. Shen, "Contour knowledge transfer for salient object detection," in *Proc. ECCV*, 2018, pp. 355–370.

[74] D. Fan, Z. Lin, Z. Zhang, M. Zhu, and M. Cheng, "Rethinking RGB-D salient object detection: Models, data sets, and large-scale benchmarks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2075–2089, Jun. 2021.

[75] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, "BASNet: Boundary-aware salient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7479–7489.

[76] Z. Wu, L. Su, and Q. Huang, "Stacked cross refinement network for edge-aware salient object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7264–7273.

[77] P. Li, Y. Xu, Y. Wei, and Y. Yang, "Self-correction for human parsing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 3260–3271, Jun. 2022.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[79] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1520–1528.

[80] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[81] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[82] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Proc. NIPS*, 2016, pp. 1–9.

[83] L. Cui et al., "Context-aware block net for small object detection," *IEEE Trans. Cybern.*, vol. 52, no. 4, pp. 2300–2313, Apr. 2022.

[84] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-SCNN: Gated shape CNNs for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5229–5238.

[85] W. Wang, S. Zhao, J. Shen, S. C. H. Hoi, and A. Borji, "Salient object detection with pyramid attention and salient edges," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1448–1457.

[86] X. Li et al., "Improving semantic segmentation via decoupled body and edge supervision," in *Proc. ECCV*, 2020, pp. 435–452.

[87] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, and G. Wang, "Semantic segmentation with context encoding and multi-path decoding," *IEEE Trans. Image Process.*, vol. 29, pp. 3520–3533, 2020.

[88] D. Cheng, G. Meng, S. Xiang, and C. Pan, "FusionNet: Edge aware deep convolutional networks for semantic segmentation of remote sensing harbor images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 12, pp. 5769–5783, Dec. 2017.

[89] S. Zhang, G.-J. Qi, X. Cao, Z. Song, and J. Zhou, "Human parsing with pyramidical gather-excite context," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 1016–1030, Mar. 2021.

[90] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. ECCV*, 2012, pp. 1–14.

[91] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martínez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: A comparative study," in *Proc. 15th Int. Conf. Pattern Recognit. (ICPR)*, 2000, pp. 314–317.

[92] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, Aug. 2011.

[93] M. Meilă, "Comparing clusterings—An information based distance," *J. Multivariate Anal.*, vol. 98, no. 5, pp. 873–895, May 2007.

[94] P. Kohli, L. Ladicky, and P. H. S. Torr, "Robust higher order potentials for enforcing label consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 302–324.

[95] Y. Zhuge, G. Yang, P. Zhang, and H. Lu, "Boundary-guided feature aggregation network for salient object detection," *IEEE Signal Process. Lett.*, vol. 25, no. 12, pp. 1800–1804, Dec. 2018.

[96] K. Philipp and K. Vladlen, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. NIPS*, 2011, pp. 1–9.

[97] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.

[98] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.

Zhicheng Zhang (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the College of Computer Science, Nankai University, Tianjin, China.

His current research interests include deep learning and computer vision, with an emphasis on video affective computing and video tracking.

Song Chen received the master's degree from Nankai University, Tianjin, China, in 2022.

His current research interests include image understanding, including object detection and object segmentation.

Zichuan Wang received the bachelor's degree in mathematics from Nankai University, Tianjin, China, in 2020, and the master's degree in computer science from Brown University, Providence, RI, USA, in 2023.

His research interests include deep learning and computer graphics.

Jufeng Yang (Member, IEEE) received the Ph.D. degree from Nankai University, Tianjin, China, in 2009.

He was a Visiting Scholar with the Vision and Learning Laboratory, University of California at Merced, Merced, CA, USA, from 2015 to 2016. He is currently a full Professor with the Department of Computer Science, Nankai University. His research interests include computer vision and machine learning.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on April 09,2023 at 07:16:03 UTC from IEEE Xplore. Restrictions apply.