Deep Floor Plan Recognition Using a Multi-Task Network with Room-Boundary-Guided Attention

Zhiliang Zeng Xianzhi Li Ying Kin Yu Chi-Wing Fu
The Chinese University of Hong Kong
{zlzeng,xzli,cwfu}@cse.cuhk.edu.hk ykyu.hk@gmail.com

Abstract

This paper presents a new approach to recognize elements in floor plan layouts. Besides walls and rooms, we aim to recognize diverse floor plan elements, such as doors, windows and different types of rooms, in the floor layouts. To this end, we model a hierarchy of floor plan elements and design a deep multi-task neural network with two tasks: one to learn to predict room-boundary elements, and the other to predict rooms with types. More importantly, we formulate the room-boundary-guided attention mechanism in our spatial contextual module to carefully take room-boundary features into account to enhance the room-type predictions. Furthermore, we design a cross-and-within-task weighted loss to balance the multi-label tasks and prepare two new datasets for floor plan recognition. Experimental results demonstrate the superiority and effectiveness of our network over the state-of-the-art methods.

(a) Floor plan #1 (b) Our result
(c) Floor plan #2 (d) Our result

Figure 1. Our network is able to recognize walls of nonuniform thickness (see boxes 2, 4, 5), walls that meet at irregular junctions (see boxes 1, 2), curved walls (see box 3), and various room types in the layout; see Figure 2 for the legend of the color labels.

1. Introduction

To recognize floor plan elements in a layout requires the learning of semantic information in the floor plans. It is not merely a general segmentation problem since floor plans present not only the individual floor plan elements, such as walls, doors, windows, and closets, *etc.*, but also how the elements relate to one another, and how they are arranged to make up different types of rooms. While recognizing semantic information in floor plans is generally straightforward for humans, automatically processing floor plans and recognizing layout semantics is a very challenging problem in image understanding and document analysis.

Traditionally, the problem is solved based on low-level image processing methods [14, 2, 7] that exploit heuristics to locate the graphical notations in the floor plans. Clearly, simply relying on hand-crafted features is insufficient, since it lacks generality to handle diverse conditions.

Recent methods [11, 5, 20] for the problem has begun to explore deep learning approaches. Liu *et al.* [11] designed a convolutional neural network (CNN) to recognize junction points in a floor plan image and connected the junctions to locate walls. The method, however, can only locate walls of uniform thickness along XY-principal directions in the image. Later, Yamasaki *et al.* [20] adopted a fully convolutional network to label pixels in a floor plan; however, the method simply uses a general segmentation network to recognize pixels of different classes and ignores the spatial relations between floor plan elements and room boundary.

This paper presents a new method for floor plan recognition, with a focus on recognizing diverse floor plan elements, *e.g.*, walls, doors, rooms, closets, *etc.*; see Figure 1 for two example results and Figure 2 for the legend. These elements are inter-related graphical elements with structural semantics in the floor plans. To approach the problem, we model *a hierarchy of labels* for the floor plan elements and design a *deep multi-task neural network* based on the hierarchy. Our network learns shared features from the input floor plan and refines the features to learn to recognize individual elements. Specifically, we design the *spatial contextual module* to explore the spatial relations between

elements via the *room-boundary-guided attention mechanism* to avoid feature blurring, and formulate the *cross-and-within-task weighted loss* to balance the labels across and within tasks. Hence, we can effectively explore the spatial relations between the floor plan elements to maximize the network learning; see again the example results shown in Figure 1, which exhibit the capability of our network.

Our contributions are threefold. First, we design a deep multi-task neural network to learn the spatial relations between floor plan elements to maximize network learning. Second, we present the spatial contextual module with the room-boundary-guided attention mechanism to learn the spatial semantic information, and formulate the cross-and-within-task weighted loss to balance the losses for our tasks. Lastly, we take the datasets from [11] and [10], collect additional floor plans, and prepare two new datasets with labels on various floor plan elements and room types.

2. Related Work

Traditional approaches recognize elements in floor plan based on low-level image processing. Ryall *et al.* [16] applied a semi-automatic method for room segmentation. Other early methods [1, 6] locate walls, doors, and rooms by detecting graphical shapes in the layout, *e.g.*, line, arc, and small loop. Or *et al.* [15] converted bitmapped floor plans to vector graphics and generated 3D room models. Ahmed *et al.* [2] separated text from graphics and extracted lines of various thickness, where walls are extracted from the thicker lines and symbols are assumed to have thin lines; then, they applied such information to further locate doors and windows. Gimenez *et al.* [7] recognized walls and openings using heuristics, and generated 3D building models based on the detected walls and doors.

Using heuristics to recognize low-level elements in floor plans is error-prone. This motivates the development of machine learning methods [4], and very recently, deep learning methods [5, 11, 20] to address the problem. Dodge *et al.* [5] used a fully convolutional network (FCN) to first detect the wall pixels, and then adopted a faster R-CNN framework to detect doors, sliding doors, and symbols such as kitchen stoves and bathtubs. Also, they employed a library tool to recognize text to estimate the room size.

Liu *et al.* [11] trained a deep neural network to first identify junction points in a given floor plan image, and then used integer programming to join the junctions to locate the walls in the floor plan. Due to the Manhattan assumption, the method can only handle walls that align with the two principal axes in the floor plan image. Hence, it can recognize layouts with only rectangular rooms and walls of uniform thickness. Later, Yamasaki *et al.* [20] trained a FCN to label the pixels in a floor plan with several classes. The classified pixels formed a graph model and were taken to retrieve houses of similar structures. However, their method

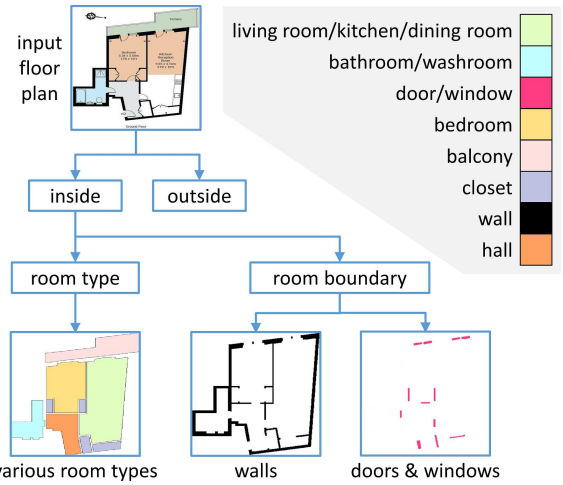


Figure 2. Floor plan elements organized in a hierarchy.

adopts a general segmentation network, where it simply recognizes pixels of different classes independently, thus ignoring the spatial relations among classes in the inference.

Compared with the recent works, our method has several distinctive improvements. Technical-wise, our method simultaneously considers multiple floor plan elements in the network; particularly, we take their spatial relationships into account and design a multi-task approach to maximize the learning of the floor plan elements in the network. Result-wise, our method is more general and capable of recognizing nonrectangular room layouts and walls of nonuniform thickness, as well as various room types; see Figure 2.

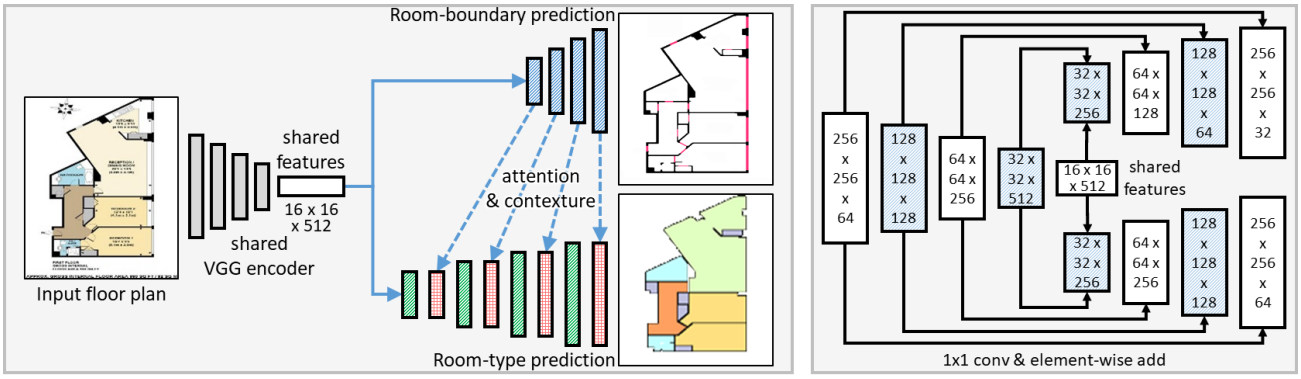
Recently, there are several other works [22, 9, 24, 21, 18] related to room layouts, but they focus on a different problem, *i.e.*, to reconstruct 3D room layouts from photos.

3. Our Method

3.1. Goals and Problem Formulation

The objectives of this work are as follows. First, we aim to recognize various kinds of floor plan elements, which are not only limited to walls but also include doors, windows, room regions, *etc.* Second, we target to handle rooms of nonrectangular shapes and walls of nonuniform thickness. Last, we aim also to recognize the rooms types in floor plans, *e.g.*, dining room, bedroom, bathroom, *etc.*

Achieving these goals requires the ability to process the floor plans and find multiple nonoverlapping but spatially-correlated elements in the plans. In our method, we first organize the floor plan elements in a hierarchy (see Figure 2), where pixels in a floor plan can be identified as inside or outside, while the inside pixels can be further identified as *room-boundary pixels* or *room-type pixels*. Moreover, the room-boundary pixels can be *walls*, *doors*, or *windows*, whereas room-type pixels can be the *living room, bathroom, bedroom, etc.*; see the legend in Figure 2. Based on the hi-



(a) Overall network architecture (b) VGG encoder and decoder architecture

Figure 3. (a) Schematic diagram illustrating our deep multi-task neural network. We have a VGG encoder to extract features from the input floor plan image. These features are shared for two subsequent tasks in the network: one for predicting the room-boundary pixels (wall, door, and windows) and the other for predicting the room-type pixels (dining room, bedroom, etc.). Most importantly, these two tasks have separate VGG decoders. We design the room-boundary-guided attention mechanism (blue arrows) to make use of the room-boundary features from the decoder in the upper branch to help the decoder in the lower path to learn the contextual features (red boxes) for predicting the room-type pixels. (b) Details of the VGG encoder and decoders. The dimensions of the features in the network are shown.

erarchy, we design a deep multi-task network with one task to predict room-boundary elements and the other to predict room-type elements. In particular, we formulate the spatial contextual module to explore the spatial relations between elements, *i.e.*, using the features learned for the room boundary to refine the features for learning the room types.

3.2. Network Architecture

Overall network architecture. Figure 3(a) presents the overall network architecture. First, we adopt a shared VGG encoder [17] to extract features from the input floor plan image. Then, we have two main tasks in the network: one for predicting the room-boundary pixels with three labels, *i.e.*, wall, door, and window, and the other for predicting the room-type pixels with eight labels, *i.e.*, dining room, washroom, *etc.*; see Figure 2 for details. Here, *room boundary* refers to the floor-plan elements that separate room regions in floor plans; it is not simply low-level edges nor the outermost border that separates the foreground and background.

Specifically, our network first learns the shared feature, common for both tasks, then makes use of two separate VGG decoders (see Figure 3(b) for the connections and feature dimensions) to perform the two tasks. Hence, the network can learn additional features for each task. To maximize the network learning, we further make use of the room-boundary context features to bound and guide the discovery of room regions, as well as their types; here, we design the spatial contextual module to process and pass the room-boundary features from the top decoder (see Figure 3(a)) to the bottom decoder to maximize the feature integration for room-type predictions.

Spatial contextual module. Figure 4 shows the network architecture of the spatial contextual module. It has two branches. The input to the top branch is the room-boundary features from the top VGG decoder (see the blue boxes in Figures 3(a) & 4), while the input to the bottom branch is the room-type features from the bottom VGG decoder (see the green boxes in Figures 3(a) & 4). See again Figure 3(a): there are four levels in the VGG decoders, and the spatial contextual module (see the dashed arrows in Figure 3(a)) is applied four times, once per level, to integrate the room-boundary and room-type features from the same level (*i.e.*, in the same resolution) and generate the spatial contextual features; see the red boxes in Figures 3(a) & 4.

- In the top branch, we apply a series of convolutions to the room-boundary feature and reduce it to a 2D feature map as the *attention weights*, denoted as $a_{m,n}$ at pixel location m, n. The attention weights are learned through the convolutions rather than being fixed.
- Furthermore, we apply the attention weights to the bottom branch twice; see the "X" operators in Figure 4. The first attention is applied to compress the noisy features before the four convolutional layers with direction-aware kernels, while the second attention is applied to further suppress the blurring features. We call it the *room-boundary-guided attention mechanism* since the attention weights are learned from the room-boundary features. Let $f_{m,n}$ as the input feature for the first attention weight $a_{m,n}$ and $f'_{m,n}$ as the output, the X operation can be expressed as
$$f'_{m,n} = a_{m,n} \cdot f_{m,n} \,. \tag{1}$$
- In the bottom branch as shown in Figure 4, we first apply a 3×3 convolution to the room-type features and then reduce it into a 2D feature map. After that, we apply the first attention to the 2D feature map followed by four separate direction-aware kernels (horizontal, vertical, diagonal, and flipped diagonal) of k unit size to

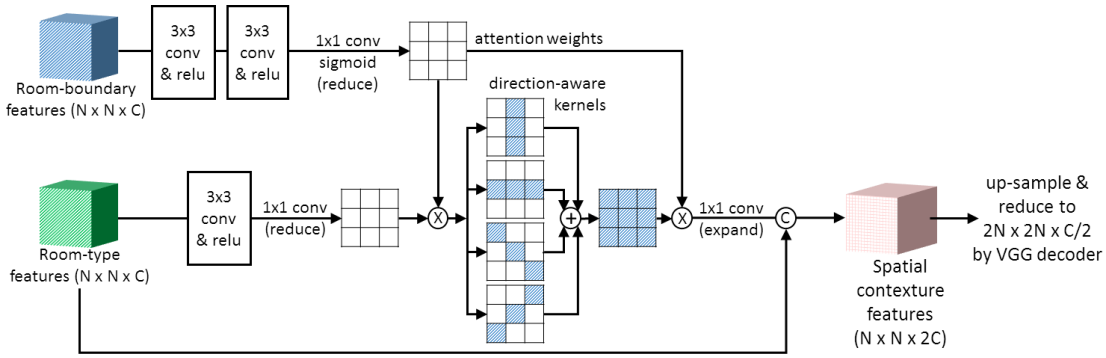


Figure 4. Our *spatial contextual module* with the *room-boundary-guided attention mechanism*, which leverages the room-boundary features to learn the attention weights for room-type prediction. In the lower branch, we use convolutional layers with four different direction-aware kernels to generate features for integration with the attention weights and produce the spatial contextual features (in red; see also Figure 3). Here "C" denotes concatenation, while "X" and "+" denote element-wise multiplication and addition, respectively.

further process the feature. Taking the horizontal kernel as an example, our equation is as follows:

$$h_{m,n} = \sum_{k} (\alpha_{m-k,n} \cdot f'_{m-k,n} + \alpha_{m,n} \cdot f'_{m,n} + \alpha_{m+k,n} \cdot f'_{m+k,n}), \quad (2)$$

where $h_{m,n}$ is the contextual features along the horizontal direction; $f'_{m,n}$ is the input feature (see Eq. (1)); and α is the weight. In our experiments, we set α to 1.

- In the second attention, we further apply the attention weights ($a_{m,n}$) to integrate the aggregated features:

$$f''_{m,n} = a_{m,n} \cdot (h_{m,n} + v_{m,n} + d_{m,n} + d'_{m,n}), \quad (3)$$

where $v_{m,n}$, $d_{m,n}$, and $d'_{m,n}$ denotes the contextual features along the vertical, diagonal, and flipped diagonal directions, respectively, after the convolutions with the direction-aware kernels.

3.3. Network Training

Datasets. As there are no public datasets with pixel-wise labels for floor plan recognition, we prepared two datasets, namely R2V and R3D. Specifically, R2V has 815 images, all from Raster-to-Vector [11], where the floor plans are mostly in rectangular shapes with uniform wall thickness. For R3D, besides the original 214 images from [10], we further added 18 floor plan images of round-shaped layouts to the data. Compared with R2V, most room shapes in R3D are irregular with nonuniform wall thickness. Here, we used Photoshop to manually label the image regions in R2V and R3D for walls, doors, bedrooms, etc. Note that we used the same label for some room regions, *e.g.*, living room and dining room (see Figure 2), since they usually locate just next to one another without walls separating them. Such a situation can be observed in both datasets. Second, we followed the GitHub code in Raster-to-Vector [11] to group room regions, so that we can compare with their results.

For the train-test split ratio, we followed the original paper [11] to split R2V into 715 images for training and 100 images for testing. For R3D, we randomly split it into 179 images for training and 53 images for testing.

Cross-and-within-task weighted loss. Each of the two tasks in our network involves multiple labels for various room-boundary and room-type elements. Since the number of pixels varies for different elements, we have to balance their contributions within each task. Also, there are generally more room-type pixels than room-boundary pixels, so we have to further balance the contributions of the two tasks. Therefore, we design a *cross-and-within-task weighted loss* to balance between the two tasks as well as among the floor plan elements within each task.

- *Within-task weighted loss*. Here, we define the within-task weighted loss in an entropy style as

$$\mathcal{L}_{task} = w_i \sum_{i=1}^{C} -y_i \log p_i, \quad (4)$$

where y_i is the label of the i-th floor plan element in the floor plan and C is the number of floor plan elements in the task; p_i is the prediction label of the pixels for the i-th element ($p_i \in [0, 1]$); and w_i is defined as follows:

$$w_i = \frac{\hat{\mathcal{N}} - \hat{N}_i}{\sum_{j=1}^{C} (\hat{\mathcal{N}} - \hat{N}_j)}, \quad (5)$$

where $\hat{N}_i$ is the total number of ground-truth pixels for the i-th floor plan element in the floor plan, and $\hat{\mathcal{N}} = \sum_{i=1}^{C} \hat{N}_i$, which means the total number of ground-truth pixels over all the C floor plan elements.

- *Cross-and-within-task weighted loss*: L_{rb} and L_{rt} denotes the within-task weighted losses for the room-boundary and room-type prediction tasks computed from Eq. (4), respectively. N_{rb} and N_{rt} are the total number of network output pixels for room boundary

Figure 5. Visual comparison of floor plan recognition results produced by our method (c&d) and by others (e-g) on the R2V dataset; note that we have to use rectangular floor plans for comparison with Raster-to-Vector [11]. Symbol † indicates the postprocessing step.

Table 1. Comparison with Raster-to-Vector [11] on the R2V dataset. Symbol † indicates our method with postprocessing (see Section 4.1).

	overall_accu	*class_accu*							
		Wall	Door & Window	Closet	Bathroom & *etc.*	Living room & *etc.*	Bedroom	Hall	Balcony
Raster-to-Vector [11]	0.84	0.53	0.58	0.78	0.83	0.72	**0.89**	0.64	0.71
Ours	0.88	**0.88**	**0.86**	0.80	0.86	0.86	0.75	0.73	0.86
Ours†	**0.89**	**0.88**	**0.86**	**0.82**	**0.90**	**0.87**	0.77	**0.82**	**0.93**

and room type, respectively. Then, the overall cross-and-within-task weighted loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = w_{rb}\mathcal{L}_{rb} + w_{rt}\mathcal{L}_{rt}, \tag{6}$$

where w_{rb} and w_{rb} are weights given by

$$w_{rb} = \frac{N_{rt}}{N_{rb}+N_{rt}} \quad \text{and} \quad w_{rt} = \frac{N_{rb}}{N_{rb}+N_{rt}}. \tag{7}$$

4. Experiments

4.1. Implementation Details

Network training. We trained our network on an NVIDIA TITAN Xp GPU and ran 40k iterations in total. We employed Adam optimizer to update the parameters and used a fixed learning rate of 1e-4 to train the network. The resolution of the input floor plan is 512×512, for keeping the thin and short lines (such as the walls) in the floor plans. Moreover, we used a batch size of one without using batch normalization, since it requires at least 32 batch size [19]. Also, we did not use any other normalization method. For other existing methods in our comparison, we used the original hyper-parameters reported in their original papers to train their networks. To obtain the best recognition results, we further evaluated the result every five training epochs and reported only the best one.

Network testing. Given a test floor plan image, we feed it to our network and obtain its output. However, due to the per-pixel prediction, the output may contain certain noise, so we further find connected regions bounded by the predicted room-boundary pixels to locate room regions, count the number of pixels of each predicted room type in each bounded region, and set the overall predicted type as the type of the largest frequency (see Figure 5(c) & (d)). Our code and datasets are available at: https://github.com/zlzeng/DeepFloorplan.

4.2. Qualitative and Quantitative Comparisons

Comparing with Raster-to-Vector. First, we compared our method with Raster-to-Vector [11], the state-of-the-art method for floor plan recognition. Specifically, we used images from the R2V dataset to train its network and also our network. To run Raster-to-Vector, we used its original labels (which are 2D corner coordinates of rectangular bound-

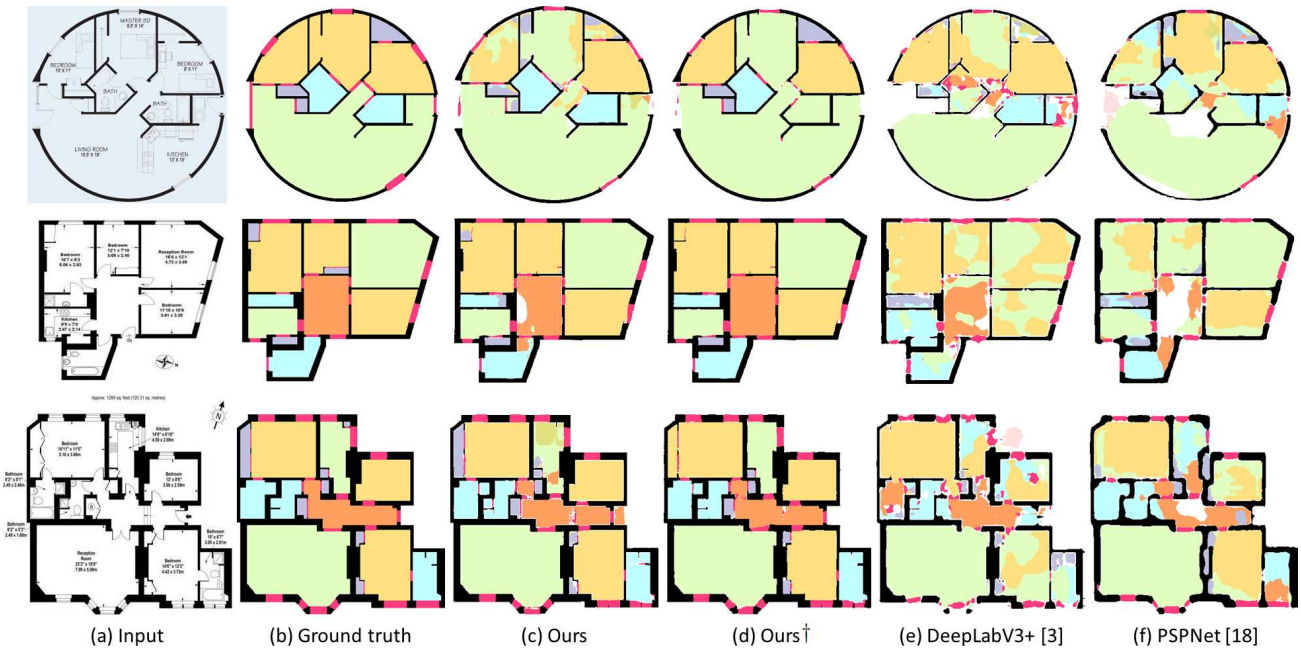


Figure 6. Visual comparison of floor plan recognition results produced by our method (c&d) and others (e-f) on the R3D dataset. Symbol † indicates our method with postprocessing (see Section 4.1).

ing boxes), while for our network, we used per-pixel labels. Considering that the Raster-to-Vector network can only output 2D corner coordinates of bounding boxes, we followed the procedure presented in [11] to convert its bounding box outputs to per-pixel labels to facilitate comparison with our method; please refer to [11] for the procedural details.

Figure 5 (c-e) shows visual comparisons between our method and Raster-to-Vector. For our method, we provide both results with (denoted with †) and w/o postprocessing. For Raster-to-Vector, it has already contained a simple postprocessing step to connect room regions. Comparing the results with the ground truths in (b), we can see that Raster-to-Vector tends to have poorer performance on room-boundary predictions, *e.g.*, missing even some room regions. Our results are more similar to the ground truths, even without postprocessing. For the R3D dataset, it contains many nonrectangular room shapes, so Raster-to-Vector performed badly with many missing regions, due to its Manhattan assumption; thus, we did not report the comparisons on R3D.

For quantitative evaluation, we adopted two widely-used metrics [13], *i.e.*, the overall pixel accuracy and the per-class pixel accuracy:

$$overall\_accu = \frac{\sum_i N_i}{\sum_i \hat{N}_i} \text{ and } class\_accu(i) = \frac{N_i}{\hat{N}_i}, \quad (8)$$

where $\hat{N}_i$ and N_i are the total number of the ground-truth pixels and the correctly-predicted pixels for the i-th floor plan element, respectively. Table 1 shows the quantitative comparison results on the R2V dataset. From the results, we can see that our method achieves higher accuracies for most floor plan elements, and the postprocessing could further improve our performance.

Comparing with segmentation networks. To evaluate how general segmentation networks perform for floor plan recognition, we further compare our method with two recent segmentation networks, DeepLabV3+ [3] and PSPNet [23].

For a fair comparison, we trained their networks, as well as our network, on the R2V dataset and also on the R3D dataset, and adjusted their hyper-parameters to obtain the best recognition results. Figures 5 & 6 present visual comparisons with PSPNet and DeepLabV3+ on testing floor plans from R2V and R3D, respectively. Due to space limitation, please see our supplementary material for results of PSPNet and DeepLabV3+ with postprocessing. From the figures, we can see that their results tend to contain noise, especially for complex room layouts and small elements like doors and windows. Since these elements are usually the room boundary between room regions, so the results further affect the room-type predictions. Please see the supplementary material for more visual comparison results.

Table 2 reports the quantitative comparison results for various methods with and without postprocessing, in terms of the overall and per-class accuracy, on both R2V and R3D datasets. Comparing with DeepLabV3+ and PSPNet, our method performs better for most floor plan elements, even without postprocessing, showing its superiority over these general-purpose segmentation networks. Note that, our postprocessing step assumes plausible room-boundary predictions, so it typically fails to enhance results with poor room-boundary predictions; see the results in Figure 6.

Table 2. Comparison with DeepLabV3+ and PSPNet. Besides the class accuracy, we further followed the GitHub code of [13] to compute the $mean\_IoU$ metric; see the last row. The values inside () indicate the performance after postprocessing. Note that the R2V dataset contains floor plans that are mostly in rectangular shapes, while the R3D dataset contains a much richer variety shape of floor plans.

		R3D						R2V					
		Ours		DeepLabV3+ [3]		PSPNet [23]		Ours		DeepLabV3+ [3]		PSPNet [23]	
$overall\_accu$		0.89	(**0.90**)	0.85	(0.83)	0.84	(0.81)	0.89	(**0.90**)	0.88	(0.87)	0.88	(0.88)
$class\_accu$	wall	0.98	(**0.98**)	0.93	(0.93)	0.91	(0.91)	0.89	(**0.89**)	0.80	(0.80)	0.84	(0.84)
	door-and-window	0.83	(**0.83**)	0.60	(0.60)	0.54	(0.54)	0.89	(**0.89**)	0.72	(0.72)	0.76	(0.76)
	closet	**0.61**	(0.54)	0.24	(0.048)	0.45	(0.086)	0.81	(**0.92**)	0.78	(0.85)	0.80	(0.71)
	bathroom & *etc.*	**0.81**	(0.78)	0.76	(0.57)	0.70	(0.50)	0.87	(**0.93**)	0.90	(0.90)	0.90	(0.84)
	living room & *etc.*	0.87	(**0.93**)	0.76	(0.90)	0.76	(0.89)	0.88	(**0.91**)	0.85	(0.84)	0.83	(0.90)
	bedroom	0.75	(**0.79**)	0.56	(0.40)	0.55	(0.40)	0.83	(**0.91**)	0.82	(0.65)	0.86	(0.92)
	hall	0.59	(0.68)	**0.72**	(0.44)	0.61	(0.23)	0.68	(0.84)	0.55	(**0.87**)	0.78	(0.81)
	balcony	0.44	(**0.49**)	0.08	(0.0027)	0.41	(0.11)	0.90	(**0.92**)	0.87	(0.45)	0.87	(0.82)
$mean\_IoU$		0.63	(**0.66**)	0.50	(0.44)	0.50	(0.41)	0.74	(**0.76**)	0.69	(0.67)	0.70	(0.69)

Comparing with an edge detection method. To show that room boundaries (*i.e.*, wall, door, and window) are not merely edges in the floor plans but structural elements with semantics, we further compare our method with a state-of-the-art edge detection network [12] (denoted as RCF) on detecting wall elements in floor plans. Here, we re-trained RCF using our wall labels, separately on the R2V and R3D datasets; since RCF outputs a per-pixel probability ($\in [0, 1]$) on wall prediction, we need a threshold (denoted as t_{RCF}) to locate the wall pixels from its results. In our method, we extract a binary map from our network output for walls pixels; see Figure 2 (bottom) for an example.

To quantitatively compare the binary maps produced by RCF and our method, we employ F-measure [8], a commonly-used metric, which is expressed as

$$F_\beta = \frac{(1+\beta^2) Precision \times Recall}{\beta^2 Precision + Recall}, \tag{9}$$

where $Precision$ and $Recall$ are the ratios of the correctly-predicted wall pixels over all the predicted wall pixels and over all the ground-truth wall pixels, respectively. To account for the fact that we need t_{RCF} to threshold RCF's results, we extend F_β into F_β^{max} and F_β^{mean} in the evaluations:

$$F_\beta^{\text{max}} = \frac{1}{M}\sum_{p=1}^{M} \tilde{F}_\beta^p \text{ and } F_\beta^{\text{mean}} = \frac{1}{MT}\sum_{p=1}^{M}\sum_{t=0}^{T-1} F_\beta^p(\frac{t}{T-1}),$$

where M is the total number of testing floor plans; $\tilde{F}_\beta^p$ is the best F_β on the p-th test input over T different t_{RCF} ranged in [0,1]; and $F_\beta^p(\frac{t}{T-1})$ is F_β on the p-th test input using $t_{\text{RCF}} = \frac{t}{T-1}$. In our implementation, as suggested by previous work [8], we empirically set β^2=0.3 and T=256. Note that F_β^{max} and F_β^{mean} are the same for the binary maps produced by our method, since they do not require t_{RCF}. Table 3 reports the results, clearly showing that our method outperforms RCF on detecting the walls. Having said that, simply detecting edges in the floor plan images is inefficient to floor plan recognition.

Table 3. Comparison with a state-of-the-art edge detection network (RCF [12]) on detecting the walls in floor plans.

	R2V		R3D	
	F_β^{max}	F_β^{mean}	F_β^{max}	F_β^{mean}
RCF [12]	0.62	0.56	0.68	0.58
Ours	**0.85**	**0.85**	**0.95**	**0.95**

Table 4. A comparison of our full network with *Baseline network #1* and *Baseline network #2* using the R3D dataset.

Metrics	Methods		
	Baseline #1	Baseline #2	Our full network
$overall\_accu$	0.82	0.85	**0.89**
$average\ class\_accu$	0.72	0.72	**0.80**

4.3. Architecture Analysis on our Network

Next, we present an architecture analysis on our network by comparing it with the following two baseline networks:

- *Baseline #1: two separate single-task networks.* The first baseline breaks the problem into two separate single-task networks, one for room-boundary prediction and the other for room-type prediction, with two separate sets of VGG encoders and decoders. Hence, there are no shared features and also no spatial contextual modules compared to our full network.
- *Baseline #2: without the spatial contextual module.* The second baseline is our full network with the shared features but without the spatial contextual module.

Table 4 shows the comparison results, where we trained and tested each network using the R3D dataset [10]. From the results, we can see that our full network outperforms the two baselines, indicating that the multi-task scheme with the shared features and the spatial contextual module both help improve the floor plan recognition performance.

4.4. Analysis on the Spatial Contextual Module

An ablation analysis of the spatial contextual module (see Figure 4 for details) is presented here.

Figure 7. Reconstructed 3D models from our recognition results.

Table 5. Ablation study on the spatial contextual module.

Metrics	Methods		
	No attention	No direction -aware kernels	Our complete version
overall_accu	0.86	0.87	**0.89**
average class_accu	0.74	0.77	**0.80**

- *No attention*: the room-boundary-guided attention mechanism (see the top branch in Figure 4) is removed from the spatial contextual module.
- *No direction-aware kernels*: the convolution layers with the four direction-aware kernels in the spatial contextual module are removed. Only the room-boundary-guided attention mechanism is applied.

Table 5 shows the comparison results between the above schemes and the full method (*i.e.*, with both attention and direction-aware kernels). Again, we trained and tested on the R3D dataset [10]. From Table 5, we can see that the spatial contextual module performs the best when equipped with the attention mechanism and direction-aware kernels.

4.5. Discussion

Application: 3D model reconstruction. Here, we take our floor plan recognition results to reconstruct 3D models. Figure 7 shows several examples of the constructed 3D floor plans. Our method is able to recognize walls of nonuniform thickness and a wide variety of shapes. It thus enables us to construct 3D room-boundary of various shapes, *e.g.*, curved walls in floor plan. One may notice that we only reconstruct the walls in 3D in Figure 7. In fact, we may further reconstruct the doors and windows, since our method has also recognized them in the layouts. For more reconstruction results, please refer to our supplementary material.

Limitations. Here, we discuss two challenging situations, for which our method fails to produce plausible predictions. First, our network may fail to differentiate inside and outside regions, in case there are some special room structures in the floor plan, *e.g.*, long and double-bended corridors. Second, our network may wrongly recognize large icons (*e.g.*, compass icon) in floor plans as wall elements. To address these issues, we believe that more data is needed for the network to learn more variety of floor plans and the semantics. Also, we may explore weakly-supervised learning for the problem to avoid the tedious annotations; please see the supplemental material for example failure cases.

5. Conclusion

This paper presents a new method for recognizing floor plan elements. There are three key contributions in this work. First, we explore the spatial relationship between floor plan elements, model a hierarchy of floor plan elements, and design a multi-task network to learn to recognize room-boundary and room-type elements in floor plans. Second, we further take the room-boundary features to guide the room-type prediction by formulating the spatial contextual module with the room-boundary-guided attention mechanism. Further, we design a cross-and-within-task weighted loss to balance the losses within each task and across tasks. In the end, we prepared also two datasets for floor plan recognition and extensively evaluated our network in various aspects. Results show the superiority of our network over the others in terms of the overall accuracy and F_β metrics. In the future, we plan to further extract the dimension information in the floor plan images, and learn to recognize the text labels and symbols in floor plans.

Acknowledgments. We thank reviewers for valuable comments, and Chen Liu, Chenxi Liu and Alexander Schwing for providing their code and data. This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region (CUHK 14203416 & 14201717).

References

[1] Christian Ah-Soon and Karl Tombre. Variations on the analysis of architectural drawings. In *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1997. 2

[2] Sheraz Ahmed, Marcus Liwicki, Markus Weber, and Andreas Dengel. Improved automatic analysis of architectural floor plans. In *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2011. 1, 2

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018. 6, 7

[4] Lluís-Pere de las Heras, Joan Mas, Gemma Sanchez, and Ernest Valveny. Wall patch-based segmentation in architectural floorplans. In *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2011. 2

[5] Samuel Dodge, Jiu Xu, and Björn Stenger. Parsing floor plan images. In *International Conference on Machine Vision Applications (MVA)*. IEEE, 2017. 1, 2

[6] Philippe Dosch, Karl Tombre, Christian Ah-Soon, and Gérald Masini. A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3(2):102–116, 2000. 2

[7] Lucile Gimenez, Sylvain Robert, Frédéric Suard, and Khaldoun Zreik. Automatic reconstruction of 3D building models from scanned 2D floor plans. *Automation in Construction*, 63:48–56, 2016. 1, 2

[8] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip H. S. Torr. Deeply supervised salient object detection with short connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4):815–828, 2018. 7

[9] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. RoomNet: End-to-end room layout estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[10] Chenxi Liu, Alex Schwing, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Rent3D: Floor-plan priors for monocular layout estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 4, 7, 8

[11] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-Vector: Revisiting floorplan transformation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 4, 5, 6

[12] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6, 7

[14] Sébastien Macé, Hervé Locteau, Ernest Valveny, and Salvatore Tabbone. A system to detect rooms in architectural floor plan images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, 2010. 1

[15] Siu-Hang Or, Kin-Hong Wong, Ying-Kin Yu, and Michael Ming-Yuan Chang. Highly automatic approach to architectural floorplan image understanding and model generation. In *Proc. of Vision, Modeling, and Visualization 2005 (VMV-2005)*, pages 25–32, 2005. 2

[16] Kathy Ryall, Stuart Shieber, Joe Marks, and Murray Mazer. Semi-automatic delineation of regions in floor plans. In *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1995. 2

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[18] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. HorizonNet: Learning room layout with 1D representation and pano stretch data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[19] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018. 5

[20] Toshihiko Yamasaki, Jin Zhang, and Yuki Takada. Apartment structure estimation using fully convolutional networks and graph model. In *Proceedings of the 2018 ACM Workshop on Multimedia for Real Estate Tech*, 2018. 1, 2

[21] Shang-Ta Yang, Fu-En Wang, Chi-Han Peng, Peter Wonka, Min Sun, and Hung-Kuo Chu. DuLa-Net: A dual-projection network for estimating room layouts from a single RGB panorama. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[22] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. PanoContext: A whole-room 3D context model for panoramic scene understanding. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[23] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[24] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. LayoutNet: Reconstructing the 3D room layout from a single RGB image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2