arXiv:2104.07986v1 [cs.CV] 16 Apr 2021

Learning to Reconstruct 3D Non-Cuboid Room Layout from a Single RGB Image

Cheng Yang[1*] Jia Zheng[2*] Xili Dai[1,3] Rui Tang[2] Yi Ma[3] Xiaojun Yuan[1†]
[1]University of Electronic Science and Technology of China
[2]KooLab, Manycore [3]University of California, Berkeley

https://github.com/CYang0515/NonCuboidRoom

Figure 1. This paper tackles the room layout reconstruction from a single RGB image *without* cuboid-shape prior or Manhattan World assumption. From left to right: input image, our estimated 2D room layout, reconstructed depth map, and corresponding 3D room model.

Abstract

Single-image room layout reconstruction aims to reconstruct the enclosed 3D structure of a room from a single image. Most previous work relies on the cuboid-shape prior. This paper considers a more general indoor assumption, i.e., the room layout consists of a single ceiling, a single floor, and several vertical walls. To this end, we first employ Convolutional Neural Networks to detect planes and vertical lines between adjacent walls. Meanwhile, estimating the 3D parameters for each plane. Then, a simple yet effective geometric reasoning method is adopted to achieve room layout reconstruction. Furthermore, we optimize the 3D plane parameters to reconstruct a geometrically consistent room layout between planes and lines. The experimental results on public datasets validate the effectiveness and efficiency of our method.

1. Introduction

Room layout estimation aims to reconstruct the enclosed structure of an indoor scene, consisting of walls, ceiling, and floor (Figure 1). Estimating a 3D room layout from a single image plays a vital role in many applications such as robotics, Virtual Reality (VR), and Augmented Reality (AR).

In an early learning-based attempt, Hedau et al. [4] assume a simple cuboid model for the room structure (ceiling, floor, and three walls). Zhang et al. [35] further define 11 types of cuboid-shape layouts to cover most of the possible situations under typical camera poses. Almost all existing methods [1, 4, 5, 10, 15, 20, 21, 22, 32, 37] follow these two cuboid-based definitions [4, 35] of the room layout, thereby being not flexible enough to handle variations in the real-world scenario.

Recent approaches [6, 25] attempt to relax these assumptions by casting room layout estimation as a plane detection problem. For example, Planar R-CNN [6] modifies Faster R-CNN [19] to detects 3D planes and Render-and-Compare (in short, RaC) [25] builds upon the advanced plane detection method PlaneRCNN [13]. To correctly infer room layout with relaxed assumptions, two core challenges must be addressed properly. One challenge is how to infer the connectivity relations between planes in 3D space. Planar R-CNN [6] does not reason such relation and only reconstructs the piece-wise planar surfaces. RaC [25] defines a constrained discrete optimization problem to reason the relations. However, the optimization step is computationally expensive and maybe less robust due to the hand-crafted heuristics. The other challenge is how to deal with the occlusions, *i.e.*, two adjacent walls in 2D space may be physically disconnected in 3D space. Planar R-CNN [6] directly uses a bounding box to locate the boundary of each plane coarsely. RaC [25] adopts RANSAC algorithm to fit the occlusion line to the points with the largest depth discrepancy changes in an analysis-by-synthesis fashion. This method requires several iterations, and the hyper-

*: Equal contribution.
†: Corresponding author.

parameters of RANSAC algorithm should be carefully chosen.

Hence, to address the above challenges more effectively and efficiently, we assume that the room layout consists of a single ceiling, a single floor, and several vertical walls. Instead of only relying on 3D plane detection results, we introduce the 2D vertical lines of adjacent walls. This allows us to fully utilize the geometric relationship between planes and lines to solve the above challenges. Two adjacent walls in 2D image space are either physically connected or disconnected in 3D space: (i) Two physically connected walls form a vertical line segment in 2D image space. We can detect the line segment to adjust the 3D plane parameter estimations. (ii) Two physically disconnected walls form an occlusion line. We can directly detect the occlusion line to bound the planar surface.

To this end, we first train Convolutional Neural Networks (CNNs) to detect planes and vertical lines in the input RGB image. Meanwhile, we also estimate the 3D parameters (*i.e.*, surface normal and offset) for each plane. Then, we explore the underlying geometric relationship between planes and lines to achieve room layout reconstruction. Specifically, we first calculate the intersection line of two adjacent walls and project it into 2D image space with the known camera intrinsic matrix. Depending on whether the projected intersection line lies between two adjacent walls or not, we classify the geometric relationship of these two walls as physically connected or disconnected in 3D space. Furthermore, if two adjacent walls are physically connected in 3D space, their projected intersection line should align with the corresponding detected line. Thus, we use the detected line as the geometric cues to optimize the 3D plane parameters, which enables our method to reconstruct a geometrically consistent 3D room layout. Otherwise, if two adjacent walls are not physically connected and the occlusion occurs, we directly use the corresponding detected line to separate them accurately.

In summary, **our contributions** are as follows: (i) We present a simple yet effective framework for 3D room layout reconstruction from a single RGB image. (ii) We propose to jointly detect 3D planes and vertical lines, and use vertical lines as complementary cues to assist the layout reconstruction. (iii) Experimental results on two challenging datasets, namely Structured3D dataset [38] and NYUv2 303 dataset [23, 32], validate the effectiveness and efficiency of our method.

2. Related Work

Room layout estimation. Hedau et al. [4] propose to model the room by a parametric box (cuboid). They generate layout hypotheses by sampling rays from the detected vanishing points and then select the best layout hypothesis. The following methods [21, 22, 32] follow this paradigm and improve this method. Inspired by the recent success of CNN on semantic segmentation, several approaches train CNNs to classify pixels into boundaries [15, 20, 37], surfaces [1], or corners [10].

Recently, several approaches [6, 25, 33] tackle the room layout estimation beyond the cuboid shape assumption. Howard-Jenkins et al. [6] leverage advanced detection methods [19] to detect each plane instance and then reconstruct the layout from multiple posed images by a voting scheme. Built upon PlaneRCNN [13], Stekovic et al. [25] formulate the layout reconstruction as constrained discrete optimization. However, this method requires several seconds to process a single frame, which is very time-consuming. Zhang et al. [33] propose to regress the plane parameters and then utilize mean-shift clustering to get the plane segmentation. However, this method does not take the occlusion into account. All these methods merely consider plane cues to achieve reconstruction. In contrast, we jointly consider planes and lines, and show how the geometric relationship between them can assist the layout reconstruction.

Due to the limited field-of-view of the standard camera, another line of work [27, 30, 34, 41] proposes to exploit more contextual information from the panoramic images. For example, Zou et al. [41] predict the corner maps and boundary maps directly. Sun et al. [27] propose to encode the layout as three 1D vectors, including ceiling-wall boundary, wall-wall boundary, and floor-wall boundary. Yang et al. [30] predict the floorplan probability in the ceiling view and floor view converted from the panorama.

Piece-wise planar reconstruction. Piece-wise planar reconstruction [13, 14, 29, 31] aims to use multiple planes to represent the scene. All these approaches focus on the visible part of the planar regions and do not need to address the occlusion reasoning problem. In contrast, the room layout estimation is much more challenging due to hard occlusion by the foreground objects. Recently, Qian and Furukawa [18] leverage the pairwise relationships between planes to optimize the plane parameters and segmentation masks. In contrast, we propose to leverage the relationships between planes and lines to assist the room layout reconstruction.

Line segment and wireframe detection. Line segment detection is a classical problem in computer vision. Conventional approaches to this problem involve grouping the low-level features in the image domain [26] or global accumulation in the Hough domain [3]. Recently, [12] trains deep networks with the Hough transformation priors to tackle this problem. Another line of work [7, 36, 40] proposes wireframe detection, *i.e.*, jointly detecting straight line segments and how these lines connect to each other. In contrast, we only focus on a particular type of line segment, *i.e.*, the vertical lines between adjacent wall planes.

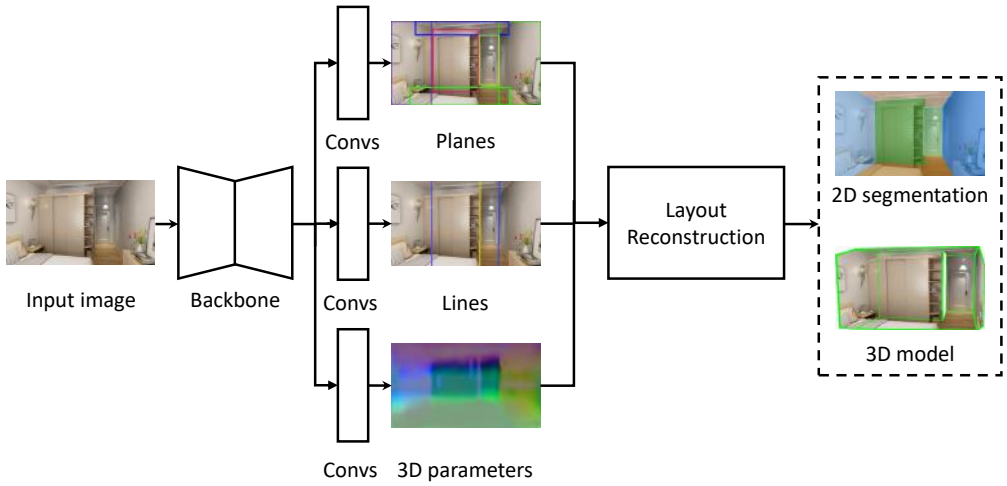


Figure 2. Pipeline. The network first takes a single RGB image as input and predicts planes and vertical lines. Meanwhile, we also estimate the 3D plane parameters for each plane. Then, a simple yet effective geometric reasoning method is adopted to reconstruct a geometrically consistent 3D room layout.

3. Our Method

Our goal is to reconstruct the 3D room layout from a single RGB image. We first detect planes $\mathbf{P} = \{p_1, p_2, \ldots\}$ and vertical lines $\mathbf{L} = \{l_1, l_2, \ldots\}$. Meanwhile, we estimate the 3D parameters of the planes. Then, a simple yet effective geometric reasoning method is employed to reconstruct a geometrically consistent 3D room layout. Figure 2 shows the overall pipeline of our method.

3.1. Plane and Line Detection

Given an input RGB image $\boldsymbol{I} \in \mathbb{R}^{H \times W \times 3}$, we first adopt HRNet-W32 [28] as our backbone to extract the visual features:

$$\boldsymbol{F} = \textsc{Backbone}(\boldsymbol{I}), \tag{1}$$

where $\boldsymbol{F} \in \mathbb{R}^{\hat{H} \times \hat{W} \times C}$. The resolution of the feature map is 4 times less than that of the input image, *i.e.*, $H = 4\hat{H}, W = 4\hat{W}$.

Then, we use different CNN-based heads to detect the planes, vertical lines between adjacent walls and regress 3D parameters of planes, respectively.

Planes. Following CenterNet [39], we represent each 2D plane p_i using a bounding box including its center position $\boldsymbol{c}_i = (x_i, y_i)$ and size $\boldsymbol{s}_i = (w_i, h_i)$.

We use three branches to predict a plane center likelihood map $\boldsymbol{C} \in \mathbb{R}^{\hat{H} \times \hat{W} \times 3}$, a center offset map $\boldsymbol{O}^p \in \mathbb{R}^{\hat{H} \times \hat{W} \times 2}$, and a plane size map $\boldsymbol{S} \in \mathbb{R}^{\hat{H} \times \hat{W} \times 2}$. Each channel of the center likelihood map $\boldsymbol{C}$ represents semantic different categories, *i.e.*, wall, floor, and ceiling. The corresponding ground truths are:

$$\boldsymbol{C}(\boldsymbol{u}) = \exp\left(-\|\boldsymbol{u} - \lfloor \boldsymbol{c} \rfloor\|^2 / (2\delta^{p2})\right), \tag{2}$$

$$\boldsymbol{O}^p(\boldsymbol{u}) = \begin{cases} \boldsymbol{c} - \boldsymbol{u}, & \boldsymbol{u} = \lfloor \boldsymbol{c} \rfloor, \\ \boldsymbol{0}, & \text{otherwise}, \end{cases} \tag{3}$$

$$\boldsymbol{S}(\boldsymbol{u}) = \begin{cases} \boldsymbol{s}, & \boldsymbol{u} = \lfloor \boldsymbol{c} \rfloor, \\ \boldsymbol{0}, & \text{otherwise}, \end{cases} \tag{4}$$

where $\boldsymbol{u} = (u_x, u_y)$ is the pixel coordinate on the output map and δ^p is an object size-adaptive standard deviation [9].

We use the focal loss [11] to supervise the plane center likelihood map. For other maps, we use the standard L_1 loss and only calculate at the plane center locations.

Lines. For a 2D vertical line l_j between two adjacent walls, we represent it by its angle θ_j of the line inclination and points $\mathcal{T}_j$ lying on the line:

$$\mathcal{T}_j = \{\boldsymbol{t}_i = (t_{i,x}, t_{i,y}) \mid t_{i,y} \in [y_{\text{min}}, y_{\text{max}}], \\ t_{i,y} \in \mathbb{N}, t_{i,x} \in \mathbb{R}\}, \tag{5}$$

where y_{min} and y_{max} represent the upper and lower bounds along the y-axis in the output map, respectively.

We adopt another three branches to predict a line likelihood map $\boldsymbol{L} \in \mathbb{R}^{\hat{H} \times \hat{W}}$, an offset map $\boldsymbol{O}^l \in \mathbb{R}^{\hat{H} \times \hat{W}}$, and an orientation map $\boldsymbol{\Theta} \in \mathbb{R}^{\hat{H} \times \hat{W}}$:

$$\boldsymbol{L}(\boldsymbol{u}) = \begin{cases} \exp\left(-\|\boldsymbol{u} - \lfloor \boldsymbol{t}_i \rfloor\|^2 / (2\delta^{l2})\right), & u_y = t_{i,y}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

$$\boldsymbol{O}^l(\boldsymbol{u}) = \begin{cases} t_{i,x} - u_x, & \boldsymbol{u} = \lfloor \boldsymbol{t}_i \rfloor, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

$$\boldsymbol{\Theta}(\boldsymbol{u}) = \begin{cases} \theta_j, & \boldsymbol{u} = \lfloor \boldsymbol{t}_i \rfloor, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

(a) (b) (c) (d)

Figure 3. Layout Reconstruction. (a) The input image. (b) Planes and lines candidates: planes (*i.e.*, p_1, p_2, p_3, p_{floor}, and p_{ceiling}) and lines (*i.e.*, the intersection line l_1 (solid yellow line), the occlusion line l_2 (solid red line)). The yellow dotted line is formed by the intersection of predicted p_1 and p_2. The red dotted line is formed by the intersection of predicted p_2 and p_3. $R_{1,2}$ and $R_{2,3}$ in the white shading represent the potential intersection line region of p_1 and p_2, p_2 and p_3, respectively. (c) 2D layout segmentation before optimization. (d) 2D layout segmentation after optimization.

where $\delta^l = 5/6$. Specifically, the offset along y-axis is always 0, so we only predict the offset along the x-axis.

We use the focal loss [11] to supervise the line region likelihood map. For other maps, we use the standard L_1 loss and only calculate at the line locations.

3D plane parameters. To reconstruct the 3D room layout, we further estimate the 3D parameters for each plane. The 3D plane parameters include its surface normal $\boldsymbol{n} \in \mathbb{S}^2$ and offset d. For a 3D point $\boldsymbol{q} \in \mathbb{R}^3$ lying on the plane, we have $\boldsymbol{n}^T\boldsymbol{q} + d = 0$. Let $\boldsymbol{v} = [\boldsymbol{n}, d]$, we predict a plane parameter map $\boldsymbol{V} \in \mathbb{R}^{\hat{H}\times\hat{W}\times 3}$ at the plane centers:

$$\boldsymbol{V}(\boldsymbol{u}) = \begin{cases} \boldsymbol{v}, & \boldsymbol{u} = \lfloor \boldsymbol{c} \rfloor, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

We use the standard smooth L_1 loss [2] to supervise the learning of the plane parameters. Inspired by PlaneRecover [29] and PlaneAE [31], we also use the standard smooth L_1 loss to supervise the depth map inferred by plane parameters.

Inference. During inference, we extract plane and line candidates from the outputs. Following CenterNet [39], we first find all peaks (x_i, y_i) in the likelihood map for each semantic category of the planes. The corresponding offset, bounding box size and 3D plane parameter can be obtained:

$$\{o_{i,x}, o_{i,y}\} = \boldsymbol{O}^p(x_i, y_i), \tag{10}$$

$$\{w_i, h_i\} = \boldsymbol{S}(x_i, y_i), \tag{11}$$

$$\boldsymbol{v} = \boldsymbol{V}(x_i, y_i). \tag{12}$$

Then, the bounding box of the detected plane is $(x_i + o_{i,x}, y_i + o_{i,y}, w_i, h_i)$.

For lines, we use the normal form of the line, *i.e.*, $x\cos\theta + y\sin\theta - b = 0$. Similarly, we first extract the peaks (x_i, y_i) in the likelihood map. The corresponding offset and line inclination can be obtained:

$$o_{i,x} = \boldsymbol{O}^l(x_i, y_i), \tag{13}$$

$$\theta = \boldsymbol{\Theta}(x_i, y_i). \tag{14}$$

Then, we obtain $b = (x_i + o_{i,x})\cos\theta + y_i \sin\theta$.

We use the non-maximum suppression (NMS) to remove the duplicated candidates. For planes, we use IoU-based NMS among all categories. For lines, if they intersect in the image or the maximal distance along x coordinate with each row is less than a threshold, we discard the one with the lower confidence.

3.2. 3D Layout Reconstruction

Once we have detected planes, vertical lines, and 3D parameters of planes, we further perform geometric reasoning to reconstruct the 3D room layout under the assumption that the room layout consists of a single ceiling, a single floor, and several vertical walls. In such an assumption, any two non-adjacent walls in image space must be physically disconnected in 3D space, so we only need to infer the connectivity relations of the adjacent walls. Once we know the connectivity relations of all walls, we combine the floor and the ceiling to achieve reconstruction.

Specifically, the detected planes contain several vertical walls, a ceiling, and a floor. We first order all walls from the lowest to the highest by the x-coordinates of plane centers in image space. Next, we calculate the intersection line of two adjacent walls by their 3D parameters and project it into image space with the known camera intrinsic matrix. Then, we classify the geometric relationship in 3D space of these two walls into two types: physically connected or disconnected, depending on whether the projected line lies between two adjacent walls or not. More specifically, we define a potential intersection line region by the bounding boxes of the adjacent walls in image space, as shown in Figure 3(b). Each potential intersection line region has at most one detected line, and we choose the one with the highest confidence when there are multiple detected lines. Due to space limitations, we refer readers to supplementary material for more details.

If two walls are physically connected in 3D space, we directly calculate the boundary with their 3D parameters. Furthermore, we expect to construct a geometrically consistent 3D room layout between detected planes and lines. We optimize the 3D plane parameters to align the calculated boundary with the detected intersection line. Specifically,

we construct a list of triplets $\mathcal{T} = \{(p_i, l_j, p_k)\}$ by identifying all detected intersection lines l_j, and the wall planes p_i, p_j on its two sides. Then, we optimize the predicted 3D plane parameters $(\boldsymbol{n}, d)$ by the following objective function:

$$\begin{aligned}\min_{\boldsymbol{n},d} \sum_{(i,j,k)\in\mathcal{T}} &\|(\frac{\boldsymbol{n}_i}{d_i} - \frac{\boldsymbol{n}_k}{d_k})^T \boldsymbol{K}^{-1} - \boldsymbol{t}_j^T\| \\ &+\alpha\|\boldsymbol{n}_i - \tilde{\boldsymbol{n}}_i\| + \beta\|d_i - \tilde{d}_i\| \\ &+\alpha\|\boldsymbol{n}_k - \tilde{\boldsymbol{n}}_k\| + \beta\|d_k - \tilde{d}_k\|,\end{aligned} \tag{15}$$

where $\boldsymbol{t}_j = [\cos\theta, \sin\theta, -b]^T$ is the detected intersection line parameter in the homogeneous coordinate, $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is the known camera intrinsic matrix, $\tilde{\boldsymbol{n}}$ and $\tilde{d}$ are the plane parameters estimated by the neural network, α and β are balance weights. In our experiment, we set $\alpha = 1$ and $\beta = 0.01$. The first term enforces the two connected wall planes to fit the detected line, and the rest keeps the solution close to the initial estimated plane parameters. We use BFGS optimization method [16] from the SciPy library to solve this problem. Figures 3(c) and (d) show the qualitative comparison with or without the proposed layout optimization. As expected, the results with optimization preserve the boundary of the walls very well.

In contrast, if two walls are physically disconnected, and the occlusion occurs. Instead of fitting the occlusion line to the points with the largest depth discrepancy changes [25], we directly use the detected line as the occlusion line to handle the occlusion problem. Specifically, when a detected line is located between two disconnected walls, we regard it as the occlusion line. However, when failing to detect the occlusion line, we coarsely locate it by the bounding boxes of wall planes. Then, same as RaC [25], we also introduce a virtual plane p, *i.e.*, back-projection of the occlusion line, defined by the camera center and the occlusion line. Since the virtual plane passes through the camera center, we obtain the offset $d = 0$. The surface normal $\boldsymbol{n}$ of the virtual plane can be obtained as:

$$\boldsymbol{n} = \frac{\boldsymbol{K}^T\boldsymbol{t}}{\|\boldsymbol{K}^T\boldsymbol{t}\|}. \tag{16}$$

By introducing the virtual plane passing through the occlusion line, all adjacent walls in the image space are physically connected in the 3D space. Thus, we directly calculate the boundary of adjacent walls with their 3D parameters. Specifically, we calculate the 3D corner $\boldsymbol{q} \in \mathbb{R}^3$ of the room layout by solving a system of linear equations:

$$\begin{cases} \boldsymbol{n}_i^T\boldsymbol{q} + d_i = 0, \\ \boldsymbol{n}_j^T\boldsymbol{q} + d_j = 0, \\ \boldsymbol{n}_k^T\boldsymbol{q} + d_k = 0, \end{cases} \tag{17}$$

where i, j are indices of the two adjacent wall planes, and k represents either a floor or a ceiling plane.

Then, we obtain the 2D corner $\boldsymbol{p}$ by projecting the 3D point $\boldsymbol{q}$ to the 2D image space:

$$\boldsymbol{p} \sim \boldsymbol{K}\boldsymbol{q}. \tag{18}$$

4. Experiments

In this section, we conduct experiments to evaluate the performance of the proposed method over two public benchmarks: Structured3D dataset [38] and NYUv2 303 dataset [23, 32].

4.1. Implementation Details

We implement our network with PyTorch [17]. The batch size is set to 24. We use color jittering as data augmentation. For Structure3D dataset, we train the model for 50 epochs. We use Adam optimizer [8] with learning rate 1×10^{-4} and 5×10^{-4} weight decay. The learning rate is decayed by a factor of 10 at 30^{th} and 40^{th} epoch. For NYUv2 303 dataset, we adopt the model trained on Structure3D dataset and fine-tune it on the SUN RGB-D dataset [24] for 50 epochs. Then, we fine-tune the model on NYUv2 subset for 10 epochs. The learning rate is set to 1×10^{-4}. Since NYUv2 303 dataset is a subset of the SUN RGB-D dataset, we exclude 101 test images from the training set.

4.2. Results on Structured3D Dataset

Structured3D dataset [38] is a large-scale photo-realistic synthetic dataset with ground-truth 3D room structure annotations. We divide the dataset at the scene level into train/validation/test, which contain 3000/250/250 scenes and 68 096/6579/6280 images. The resolution of input image is 640×384.

Methods for comparison. We compare our method with the following two methods: (i) Planar R-CNN [6]: Although Planar R-CNN can reconstruct the piece-wise planar model from a single image, it does not reason about the extents of the planar surface, which is vital in the room layout estimation task. Since the source code of Planar R-CNN is unavailable, we reimplement it closely following the given implementation details. (ii) RaC [25][1]: We use the plane detection results by our methods as the input. Additionally, this method requires plane instance segmentation. To this end, we further predict the plane parameters for each pixel. During inference, we compare the pixel-level plane parameters with the instance-level plane parameters to get the plane instance segmentation. Since the above two methods build on 3D plane detection results, and the key contribution is the different post-process steps to reconstruct the room layout according to the plane detection results, for a fair comparison, we use the same plane detection results as our method.

[1] https://github.com/vevenom/RoomLayout3D_RandC

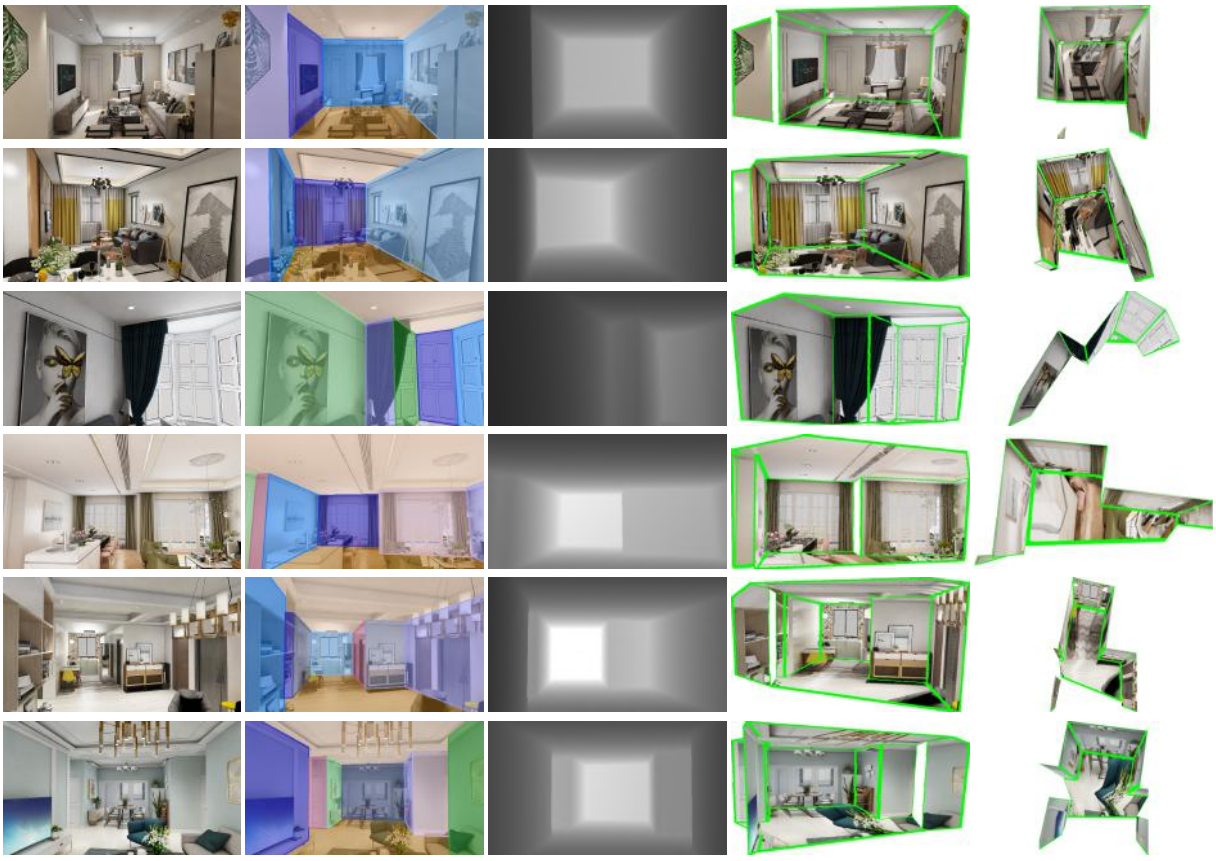


Figure 4. 3D room layout reconstruction results on Structured3D dataset [38]. The ceiling is ignored in the top view of the 3D model. More reconstruction results can be found in the supplementary material.

Method	IoU (%) ↑	PE (%) ↓	EE ↓	RMSE ↓	Runtime (s) ↓
Planar R-CNN [6]†	79.64	7.04	6.58	0.4013	0.11
RaC [25]	76.29	8.07	7.19	0.3465	5.35
Ours (w/o optimization)	79.94	6.40	6.80	**0.2827**	**0.07**
Ours	**81.40**	**5.87**	**5.78**	0.2905	0.24

Table 1. Quantitative results on Structured3D dataset [38]. †: our implementation.

Evaluation metric. Following RaC [25], we adopt four standard evaluation metrics: (i) IoU: intersection over the union between the predicted plane layout and the ground truth, (ii) Pixel Error (PE): pixel-wise error between predicted 2D plane segmentation and the ground truth, (iii) Edge Error (EE): the symmetrical Chamfer distance between predicted layout boundary and the ground truth, (iv) Root Mean Square Error (RMSE) between the predicted layout depth map and the ground truth. For the 2D metrics (*i.e.*, IoU and PE), we match the predicted plane segmentation to the ground truths. Starting from the largest ground-truth segmentation, we iteratively find the predicted segmentation with the highest IoU score. Each ground truth is only allowed to be matched at most once.

Quantitative evaluation. Table 1 shows the quantitative results and runtime of all methods. All times are measured on the same computing platform with an Intel Xeon Gold 6128 @ 3.4GHz (24 cores) and a single NVIDIA TITAN Xp GPU. As one can see, our proposed method achieves state-of-the-art performance without optimization. Compared with RaC [25], our method reasons the connectivity relations between planes and handles with occlusions through introducing room layout assumption and vertical line detection, and avoids complex optimization as well as decreasing the running time from 5.35 s to 0.07 s. Furthermore, when using the proposed layout optimization algorithm to reconstruct a geometrically consistent room layout between planes and lines, our method provides a good

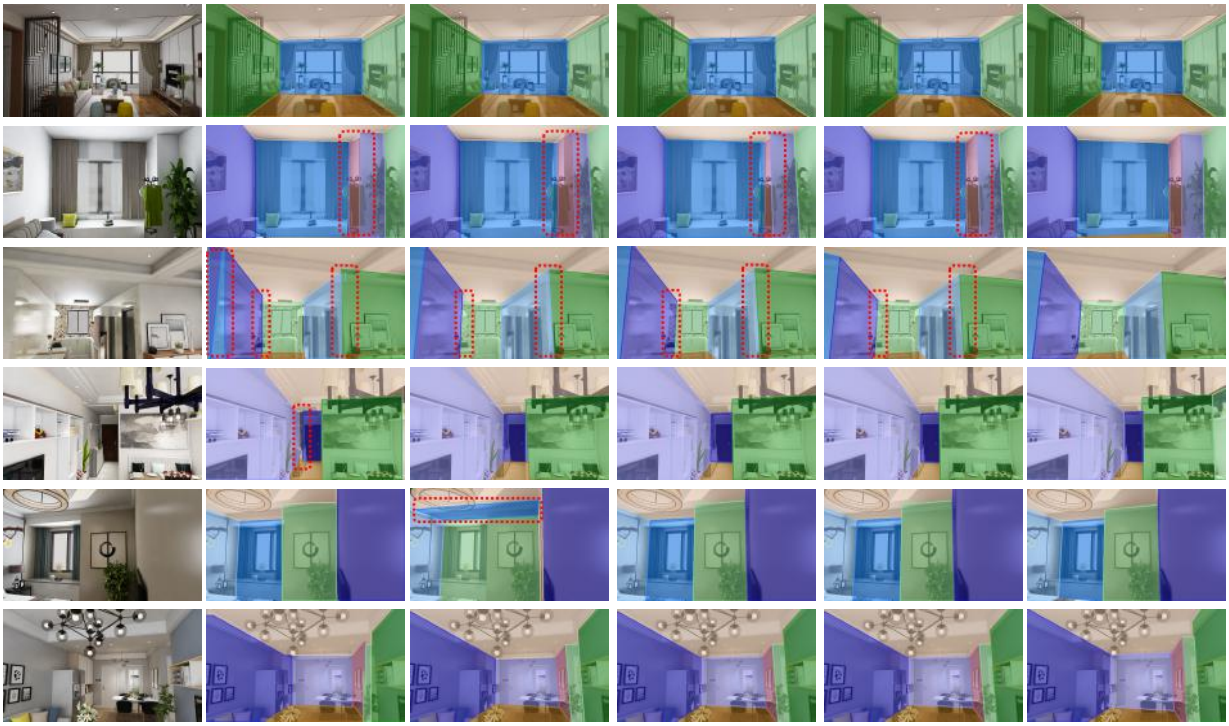


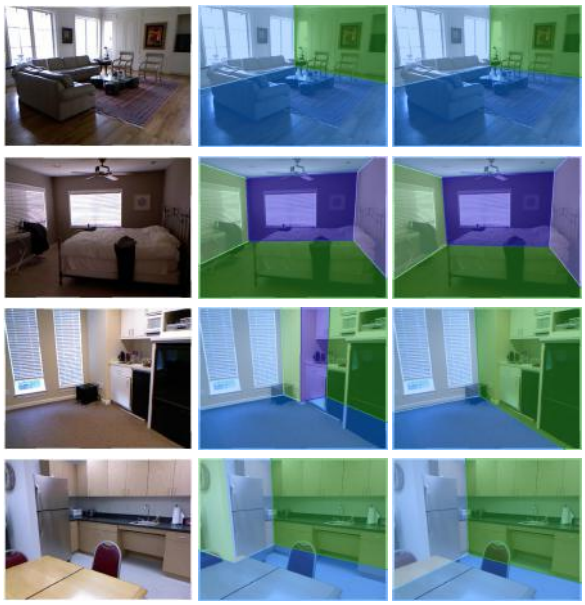
Figure 5. Qualitative results on Structured3D dataset [38]. The correspondences are marked in a common color. We highlight the major differences in the dashed red bounding boxes.

trade-off between accuracy and speed. This clearly demonstrates the effectiveness of the proposed method.

Qualitative evaluation. Figure 4 shows our 3D room layout reconstruction results for a variety of scenes. The qualitative comparisons against existing non-cuboid room layout estimation methods show in Figure 5. We make the following observations: (i) All methods perform well in simple scenarios (*e.g.*, the first row). (ii) Since we explicitly use detected lines as geometric cues to optimize the room layout, our results can preserve the boundary of two adjacent walls well (*e.g.*, the second and third row). (iii) Planar R-CNN does not reason how the adjacent walls connect and bounds each plane segmentation by their bounding boxes. As a result, two adjacent walls may inter-penetrate each other (*e.g.*, the third row) or do not touch each other (*e.g.*, the fourth row). (iv) The carefully designed heuristics of RaC are not robust in every configuration (*e.g.*, the fifth row). Furthermore, when the number of plane candidates is large, RaC is very computationally expensive (*e.g.*, about 8 minutes for the sixth row).

Figure 6 shows our detection results of planes and vertical lines between adjacent walls. Thanks to the advent of powerful detection technology, we can accurately detect planes and lines. Meanwhile, we only consider the lines that locate in the potential intersection line region, which reduces the false positives, such as texture lines.

Figure 6. Plane detections and vertical lines detections between adjacent walls on Structured3D dataset [38]. The lines with red cross are filtered by the potential intersection line region.



Input image Ours Ground truth

Figure 7. Qualitative results on NYU 303 dataset [32]. The correspondences are marked in a common color.

Method	PE (%) ↓
Schwing et al. [22]	13.66
Zhang et al. [32]	13.94
RoomNet [10]†	12.31
PlaneNet [14]	12.64
Hirzer et al. [5]	**8.49**
Planar R-CNN [6]	12.19
RaC [25]	13.00
Ours (w/o optimization)	11.25
Ours	**10.61**

Table 2. Quantitative results on NYUv2 303 dataset [32]. †: The results are reported by [5].

4.3. Results on NYUv2 303 Dataset

We further evaluate the performance of our approach on a real but much smaller dataset, *i.e.*, NYUv2 303 dataset [32], which contains 303 images from NYUv2 dataset [23]. The resolution of input image is 640×480. Note that the dataset only provides cuboid-based layout annotation (hence evaluation on this dataset may favor cuboid-based methods).

Quantitative evaluation. Table 2 shows the quantitative comparisons against both cuboid-based and non-cuboid methods on the NYUv2 303 dataset. To validate our approach, we follow two recent non-cuboid methods [6, 25] and use the Hungarian algorithm to match detected planes to the ground truths. As one can see, our methods outperform all non-cuboid layout methods and almost all cuboid-based methods (with the exception of Hirzer et al. [5]). Nevertheless, such cuboid-based methods cannot be applied to the more extensive and flexible Structured3D dataset.

Input image Ours Ground truth

Figure 8. Failure Cases. The correspondences are marked in a common color.

Qualitative evaluation. Figure 7 shows room layout estimation results on the NYU 303 dataset. The first two examples show that our approach can predict cuboid-shape layout even without such an assumption. The last two examples demonstrate that our method can predict the more refined correct layouts than the ground truth (cuboids).

4.4. Failure Cases

We show some failure cases of our method in Figure 8. In the first example, our method mistakenly detects the foreground furniture as the wall. A possible reason is that most of the wall is occluded by the foreground furniture. In the second example, our approach fails to detect the floor, which leads to an inaccurate wall-floor boundary. In the third example, the ceiling-wall and floor-wall boundaries of the light pink plane are not precisely localized due to the inaccurate 3D parameter estimations of the small plane.

5. Conclusion

This paper proposes a simple yet effective 3D room layout reconstruction approach assuming that the room layout consists of a single ceiling, a single floor, and several vertical walls. Specifically, we first employ CNNs to detect 3D planes and vertical lines. Then, we adopt a geometric reasoning method to achieve the room layout reconstruction. Finally, to reconstruct a geometrically consistent layout, we optimize the 3D parameters of the planes to align detected lines. The proposed method achieves state-of-the-art performance on the largest non-cuboid room layout benchmark. In the future, we will consider introducing more geometric cues, such as wall-ceiling/wall-floor intersection lines, and relax the assumption to handle multi-tiered ceilings/floors.

Acknowledgements. We would like to thank Zihan Zhou for valuable comments on this paper.

References

[1] Saumitro Dasgupta, Kuan Fang, Kevin Chen, and Silvio Savarese. Delay: Robust spatial layout estimation for cluttered indoor scenes. In *CVPR*, pages 616–624, 2016. 1, 2

[2] Ross B. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015. 4

[3] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(4):722–732, 2010. 2

[4] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, pages 1849–1856, 2009. 1, 2

[5] Martin Hirzer, Peter M. Roth, and Vincent Lepetit. Smart hypothesis generation for efficient and robust room layout estimation. In *WACV*, pages 2912–2020, 2020. 1, 8

[6] Henry Howard-Jenkins, Shuda Li, and Victor Prisacariu. Thinking outside the box: Generation of unconstrained 3d room layouts. In *ACCV*, pages 432–448, 2018. 1, 2, 5, 6, 8

[7] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *CVPR*, pages 626–635, 2018. 2

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

[9] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *IEEE Int. J. Comput. Vis.*, 128(3):642–656, 2020. 3

[10] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. In *ICCV*, pages 4875–4884, 2017. 1, 2, 8

[11] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3, 4

[12] Yancong Lin, Silvia L Pintea, and Jan C. van Gemert. Deep hough-transform line priors. In *ECCV*, pages 323–340, 2020. 2

[13] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *CVPR*, pages 4450–4459, 2019. 1, 2

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

[15] Arun Mallya and Svetlana Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, pages 936–944, 2015. 1, 2

[16] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006. 5

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

[18] Yiming Qian and Yasutaka Furukawa. Learning pairwise inter-plane relations for piecewise planar reconstruction. In *ECCV*, pages 330–345, 2020. 2

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015. 1, 2

[20] Yuzhuo Ren, Shangwen Li, Chen Chen, and C-C Jay Kuo. A coarse-to-fine indoor layout estimation (cfile) method. In *ACCV*, pages 36–51, 2016. 1, 2

[21] Alexander G Schwing, Sanja Fidler, Marc Pollefeys, and Raquel Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *ICCV*, pages 353–360, 2013. 1, 2

[22] Alexander G Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *CVPR*, pages 2815–2822, 2012. 1, 2, 8

[23] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 2, 5, 8

[24] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015. 5

[25] Sinisa Stekovic, Friedrich Fraundorfer, and Vincent Lepetit. General 3d room layout from a single view by render-and-compare. In *ECCV*, pages 187–203, 2020. 1, 2, 5, 6, 8

[26] Richard S Stephens. Probabilistic approach to the hough transform. *Image and Vision Computing*, 9(1):66–71, 1991. 2

[27] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *CVPR*, pages 1047–1056, 2019. 2

[28] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 3

[29] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *ECCV*, pages 87–103, 2018. 2, 4

[30] 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 *CVPR*, pages 3363–3372, 2019. 2

[31] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *CVPR*, pages 1029–1037, 2019. 2, 4

[32] Jian Zhang, Chen Kan, Alexander G Schwing, and Raquel Urtasun. Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In *ICCV*, pages 1273–1280, 2013. 1, 2, 5, 8

[33] Weidong Zhang, Wei Zhang, and Yinda Zhang. Geolayout: Geometry driven room layout estimation based on depth maps of planes. In *ECCV*, pages 632–648, 2020. 2

[34] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *ECCV*, pages 668–686, 2014. 2

[35] Yinda Zhang, Fisher Yu, Shuran Song, Pingmei Xu, Ari Seff, and Jianxiong Xiao. Large-scale scene understanding challenge: Room layout estimation, 2016. 1

[36] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua

Gao. Ppgnet: Learning point-pair graph for line segment detection. In *CVPR*, pages 7105–7114, 2019. 2

[37] Hao Zhao, Ming Lu, Anbang Yao, Yiwen Guo, Yurong Chen, and Li Zhang. Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation. In *CVPR*, pages 10–18, 2017. 1, 2

[38] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *ECCV*, pages 519–535, 2020. 2, 5, 6, 7

[39] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019. 3, 4

[40] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *ICCV*, pages 962–971, 2019. 2

[41] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *CVPR*, pages 2051–2059, 2018. 2