LayoutNet: Reconstructing the 3D Room Layout from a Single RGB Image

Chuhang Zou[†] Alex Colburn[‡] Qi Shan[‡] Derek Hoiem[†]
[†]University of Illinois at Urbana-Champaign
{czou4, dhoiem}@illinois.edu
[‡]Zillow Group
{alexco, qis}@zillow.com

Abstract

We propose an algorithm to predict room layout from a single image that generalizes across panoramas and perspective images, cuboid layouts and more general layouts (e.g. "L"-shape room). Our method operates directly on the panoramic image, rather than decomposing into perspective images as do recent works. Our network architecture is similar to that of RoomNet [15], but we show improvements due to aligning the image based on vanishing points, predicting multiple layout elements (corners, boundaries, size and translation), and fitting a constrained Manhattan layout to the resulting predictions. Our method compares well in speed and accuracy to other existing work on panoramas, achieves among the best accuracy for perspective images, and can handle both cuboid-shaped and more general Manhattan layouts.

1. Introduction

Estimating the 3D layout of a room from one image is an important goal, with applications such as robotics and virtual/augmented reality. The room layout specifies the positions, orientations, and heights of the walls, relative to the camera center. The layout can be represented as a set of projected corner positions or boundaries, or as a 3D mesh. Existing works apply to special cases of the problem, such as predicting cuboid-shaped layouts from perspective images or from panoramic images.

We present LayoutNet, a deep convolution neural network (CNN) that estimates the 3D layout of an indoor scene from a single perspective or panoramic image (Figure. 1). Our method compares well in speed and accuracy on panoramas and is among the best on perspective images. Our method also generalizes to non-cuboid Manhattan layouts, such as "L"-shaped rooms. Code is available at: https://github.com/zouchuhang/LayoutNet.

Our LayoutNet approach operates in three steps (Figure. 2). First, our system analyzes the vanishing points and aligns the image to be level with the floor (Sec. 3.1). This alignment ensures that wall-wall boundaries are vertical lines and substantially reduces error according to our experiments. In the second step, corner (layout junctions) and boundary probability maps are predicted directly on the image using a CNN with an encoder-decoder structure and skip connections (Sec. 3.2). Corners and boundaries each provide a complete representation of room layout. We find that jointly predicting them in a single network leads to better estimation. Finally, the 3D layout parameters are optimized to fit the predicted corners and boundaries (Sec. 3.4). The final 3D layout loss from our optimization process is difficult to back-propagate through the network, but direct regression of the 3D parameters during training serves as an effective substitute, encouraging predictions that maximize accuracy of the end result.

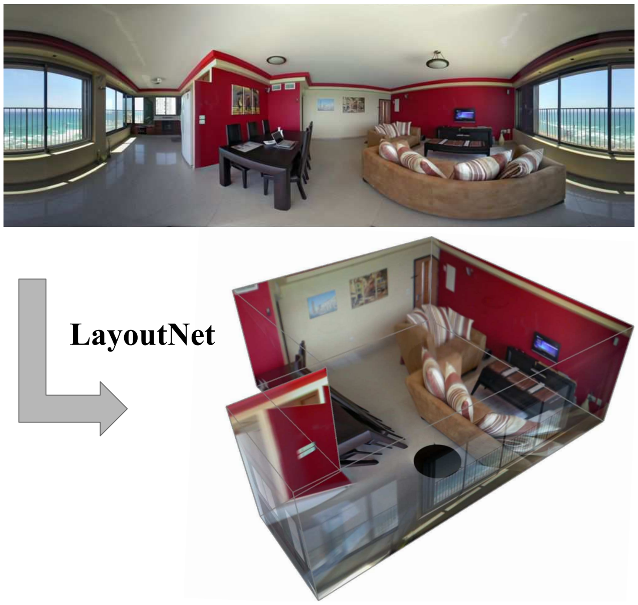

Figure 1. **Illustration.** Our LayoutNet predicts a non-cuboid room layout from a single panorama under equirectangular projection.

Our contributions are:

- We propose a more general RGB image to layout algorithm that is suitable for perspective and panoramic

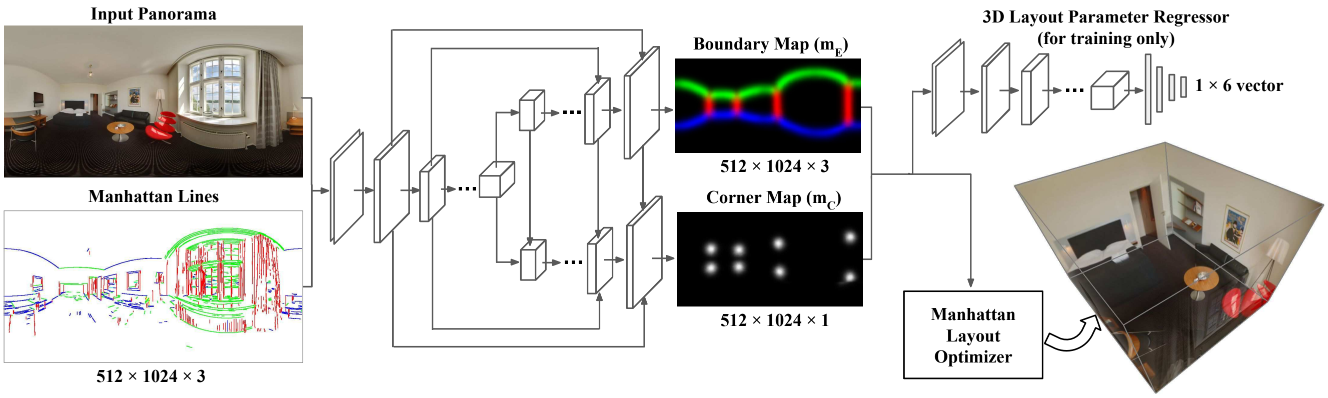

Figure 2. **Overview.** Our LayoutNet follows the encoder-decoder strategy. The network input is a concatenation of a single RGB panorama and Manhattan line map. The network jointly predicts layout boundaries and corner positions. The 3D layout parameter loss encourages predictions that maximize accuracy. The final prediction is a Manhattan constrained layout reconstruction. Best viewed in color.

images with Manhattan layouts. Our system compares well in speed and accuracy for panoramic images and achieves the second best for perspective images, while also being the fastest.

- We demonstrate gains from using precomputed vanishing point cues, geometric constraints, and post-process optimization, indicating that deep network approaches still benefit from explicit geometric cues and constraints. We also show that adding an objective to directly regress 3D layout parameters leads to better predictions of the boundaries and corners that are used to solve for the final predicted layout.
- We extend the annotations for the Stanford 2D-3D dataset [1], providing room layout annotations that can be used in future work.

2. Related Work

Single-view room layout estimation has been an active topic of research for the past ten years. Delage et al. [7] fit floor/wall boundaries in a perspective image taken by a level camera to create a 3D model under "Manhattan world" assumptions [3]. The Manhattan world assumptions are that all walls are at right angles to each other and perpendicular to the floor. A special case is the cuboid model, in which four walls, ceiling, and floor enclose the room. Lee et al. [17] produce Orientation Maps, generate layout hypotheses based on detected line segments, and select a best-fitting layout from among them. Hedau et al. [10] recover cuboid layouts by solving for three vanishing points, sampling layouts consistent with those vanishing points, and selecting the best layout based on edge and Geometric Context [12] consistencies. Subsequent works follow a similar approach, with improvements to layout generation [26, 27, 22], features for scoring layouts [27, 22], and incorporation of object hypotheses [11, 16, 5, 6, 33] or other context. The most recent methods train deep network features to classify pixels into layout surfaces (walls, floor, ceiling) [4, 13], boundaries [21], corners [15], or a combination [24].

Nearly all of these works aim to produce cuboid-shaped layouts from perspective RGB images. A few works also operate on panoramic images. Zhang et al. [32] propose the PanoContext dataset and method to estimate room layout from 360° panoramic images (more on this later). Yang et al. [30] recover layouts from panoramas based on edge cues, Geometric Context, and other priors. Xu et al. [29] estimate layout based on surface orientation estimates and object hypotheses. Other works recover indoor layout from multiple images (e.g., [2]) or RGBD images (e.g., [28, 31, 9, 18]), where estimates rely heavily on 3D points obtained from sensors or multiview constraints. Rent3D [19] takes advantage of a known floor plan. Our approach simplifies reconstruction by estimating layout directly on a single RGB equirectangular panorama. Our final output is a sparse and compact planar Manhattan layout parameterized by each wall's distance to camera, height, and the layout rotation.

Our work is most similar in goal to PanoContext [32] and in approach to RoomNet [15]. PanoContext extends the frameworks designed for perspective images to panoramas, estimating vanishing points, generating hypotheses, and scoring hypotheses according to Orientation Maps, Geometric Context, and object hypotheses. To compute these features, PanoContext first projects the panoramic image into multiple overlapping perspective images, and then combines the feature maps back into a panoramic image. Our approach is more direct: after aligning the panoramic image based on vanishing points, our system uses a deep network to predict boundaries and corners directly on the panoramic image. In this regard, we are similar to RoomNet, which uses a deep network to directly predict layout corners in perspective images, as well as a label that indicates which corners are visible. Our method differs from RoomNet in several ways. Our method applies to panoramic images. Our method also differs in the alignment step (RoomNet performs none) and in our multitask prediction of boundaries, corners, and 3D cuboid parameters. Our final inference is constrained to produce a Manhattan 3D layout. RoomNet uses an RNN to refine 2D corner position predictions, but those predictions might not be consistent with any 3D cuboid layout. Our experiments show that all of these differences improve results.

More generally, we propose the first method, to our knowledge, that applies to both perspective and panoramic images. We also show that our method extends easily to non-cuboid Manhattan layouts. Thus, our method is arguably the most general and effective approach to date for indoor layout estimation from a single RGB image.

3. Approach

We first describe our method for predicting cuboid-shaped layouts from panoramas: alignment (Sec. 3.1), corner and boundary prediction with a CNN (Sec. 3.2 and 3.3), and optimization of 3D cuboid parameters (Sec. 3.4). Then, we describe modifications to predict on more general (non-cuboid) Manhattan layouts and perspective images (Sec. 3.5).

3.1. Panoramic image alignment

Given the input as a panorama that covers a 360° horizontal field of view, we first align the image by estimating the floor plane direction under spherical projection, rotate the scene, and reproject it to the 2D equirectangular projection. Similar to Zhang et al.'s approach [32], we select long line segments using the Line Segment Detector (LSD) [23] in each overlapped perspective view, then vote for three mutually orthogonal vanishing directions using the Hough Transform. This pre-processing step eases our network training. The detected candidate Manhattan line segments also provide additional input features that improve the performance, as shown in Sec. 4.

3.2. Network structure

An overview of the LayoutNet network is illustrated in Fig. 2. The network follows an encoder-decoder strategy.

Deep panorama encoder: The input is a 6-channel feature map: the concatenation of single RGB panorama with resolution of 512×1024 (or 512×512 for perspective images) and the Manhattan line feature map lying on three orthogonal vanishing directions using the alignment method in Sec. 3.1. The encoder contains 7 convolution layers with kernel size of 3×3. Each convolution is followed by a ReLU operation and a max pooling layer with the down-sampling factor of 2. The first convolution contains 32 features, and we double size after each convolution. This deep structure ensures a better feature learning from high resolution images and help ease the decoding step. We tried Batch Normalization after each convolution layer but observe lower accuracy. We also explored an alternative structure that applies a separate encoder for the input image and the Manhattan lines, but observe no increase in performance compared to our current simpler design.

2D layout decoder: The decoder consists of two branches as shown in Fig. 2. The top branch, the layout boundary map ($\boldsymbol{m_E}$) predictor, decodes the bottleneck feature into the 2D feature map with the same resolution as the input. $\boldsymbol{m_E}$ is a 3-channel probability prediction of wall-wall, ceiling-wall and wall-floor boundary on the panorama, for both visible and occluded boundaries. The boundary predictor contains 7 layers of nearest neighbor up-sampling operation, each followed by a convolution layer with kernel size of 3×3, and the feature size is halved through layers from 2048. The final layer is a Sigmoid operation. We add skip connections to each convolution layer following the spirit of the U-Net structure [25], in order to prevent shifting of predictions results from the up-sampling step. The lower branch, the 2D layout corner map ($\boldsymbol{m_C}$) predictor, follows the same structure as the boundary map predictor and additionally receives skip connections from the top branch for each convolution layer. This stems from the intuition that layout boundaries imply corner positions, especially for the case when a corner is occluded. We show in our experiments (Sec. 4) that the joint prediction helps improve the accuracy of the both maps, leading to a better 3D reconstruction result. We experimented with fully convolutional layers [20] instead of the up-sampling plus convolutions structure, but observed worse performance with checkerboard artifacts.

3D layout regressor: The function to map from 2D corners and boundaries to 3D layout parameters is simple mathematically, but difficult to learn. So we train a regressor for 3D layout parameters with the purpose of producing better corners and boundaries, rather than for its own sake. As shown in Fig. 2, the 3D regressor gets as input the concatenation of the two predicted 2D maps and predicts the parameters of the 3D layout. We parameterize the layout with 6 parameters, assuming the ground plane is aligned on the $x - z$ axis: width s_w, length s_l, height s_h, translation $T = (t_x, t_z)$ and rotation r_θ on the $x - z$ plane. The regressor follows an encoder structure with 7 layers of convolution with kernel size 3×3, each followed by a ReLU operation and a max pooling layer with the down sampling factor of 2. The convolution feature size doubles through layers from the input 4 feature channel. The next four fully-connected layers have sizes of 1024, 256, 64, and 6, with ReLU in between. The output 1×6 feature vector $d = \{s_w, s_l, s_h, t_x, t_z, r_\theta\}$ is our predicted 3D cuboid parameter. Note that the regressor outputs the parameters of the 3D layout that can be projected back to the 2D image, presenting an end-to-end prediction approach. We observed that the 3D regressor is not accurate (with corner error of

3.36% in the PanoContext dataset compared with other results in Table 1), but including it in the loss objective tends to slightly improve the predictions of the network. The direct 3D regressor fails due to the fact that small position shifts in 2D can have a large difference in the 3D shape, making the network hard to train.

Loss function. The overall loss function of the network is in Eq. 1:

$$\begin{aligned} L(\boldsymbol{m_E}, \boldsymbol{m_C}, \boldsymbol{d}) = &-\alpha\frac{1}{n}\sum_{p\in \boldsymbol{m_E}}\big(\hat{p}\log p + (1-\hat{p})\log(1-p)\big) \\ &-\beta\frac{1}{n}\sum_{q\in \boldsymbol{m_C}}\big(\hat{q}\log q + (1-\hat{q})\log(1-q)\big) \\ &+\tau\|\boldsymbol{d}-\hat{\boldsymbol{d}}\|_2 \end{aligned} \tag{1}$$

The loss is the summation over the binary cross entropy error of the predicted pixel probability in $\boldsymbol{m_E}$ and $\boldsymbol{m_C}$ compared to ground truth, plus the Euclidean distance of regressed 3D cuboid parameters d to the ground truth $\hat{\boldsymbol{d}}$. p is the probability of one pixel in $\boldsymbol{m_E}$, and $\hat{\boldsymbol{p}}$ is the ground truth of p in $\boldsymbol{m_E}$. q is the pixel probability in $\boldsymbol{m_C}$, and $\hat{\boldsymbol{q}}$ is the ground truth. n is the number of pixels in $\boldsymbol{m_E}$ and $\boldsymbol{m_C}$ which is the image resolution. Note that the RoomNet approach [15] uses L2 loss for corner prediction. We discuss the performance using two different losses in Sec. 4. α, β and τ are the weights for each loss term. In our experiment, we set $\alpha = \beta = 1$ and $\tau = 0.01$.

3.3. Training details

Our LayoutNet predicts pixel probabilities for corners and boundaries and regresses the 3D layout parameters. We find that joint training from a randomly initialized network sometimes fails to converge. Instead, we train each subnetwork separately and then jointly train them together. For the 2D layout prediction network, we first train on the layout boundary prediction task to initialize the parameters of the network. For the 3D layout regressor, we first train the network with ground truth layout boundaries and corners as input, and then connect it with the 2D layout decoder and train the whole network end-to-end.

The input Manhattan line map is a 3 channel 0-1 tensor. We normalize each of the 3D cuboid parameter into zero mean and standard deviation across training samples. We use ADAM [14] to update network parameters with a learning rate of e^{-4}, $\alpha = 0.95$ and $\epsilon = e^{-6}$. The batch size for training the 2D layout prediction network is 5 and changes to 20 for training the 3D regressor. The whole end-to-end training uses a batch size of 20.

Ground truth smoothing: Our target 2D boundary and corner map is a binary map with a thin curve or point on the image. This makes training more difficult. For example, if the network predicts the corner position slightly off the ground truth, a huge penalty will be incurred. Instead, we dilate the ground truth boundary and corner map with a factor of 4 and then smooth the image with a Gaussian kernel of 20 × 20. Note that even after smoothing, the target image still contains $\tilde{9}5\%$ zero values, so we re-weight the back propagated gradients of the background pixels by multiplying with 0.2.

Data augmentation: We use horizontal rotation, left-right flipping and luminance change to augment the training samples. The horizontal rotation varies from $0^o - 360^o$. The luminance varies with γ values between 0.5-2. For perspective images, we apply $\pm 10^\circ$ rotation on the image plane.

Algorithm 1 3D layout optimization

```
Given panorama $I$, layout corner prediction $\boldsymbol{m_C}$, and boundary prediction $\boldsymbol{m_E}$;
Initialize 3D layout $L_0$ based on Eq. 2;
$E_{best} = Score(L_0)$ by Eq. 3, $L_{best} = L_0$;
for $i = 1$ :wallNum do
    Sample candidate layouts $L_i$ by varying wall position $w_i$ in 3D, fix other wall positions;
    for $j = 1 : |L_i|$ do
        Sample candidate Layouts $L_{ij}$ by varying floor and ceiling position in 3D;
        Rank the best scored Layout $L_B \in \{L_{ij}\}$ based on Eq. 3;
        if $E_{best} < Score(L_B)$ then
            $E_{best} = Score(L_B)$, $L_{best} = L_B$;
    Update $w_i$ from $L_{best}$, fix it for following sampling
return $L_{best}$
```

3.4. 3D layout optimization

The initial 2D corner predictions are obtained from the corner probability maps that our network outputs. First, the responses are summed across rows, to get a summed response for each column. Then, local maxima are found in the column responses, with distance between local maxima of at least 20 pixels. Finally, the two largest peaks are found along the selected columns. These 2D corners might not satisfy Manhattan constraints, so we perform optimization to refine the estimates.

Given the predicted corner positions, we can directly recover the camera position and 3D layout, up to a scale and translation, by assuming that bottom corners are on the same ground plane and that the top corners are directly above the bottom ones. We can further constrain the layout shape to be Manhattan, so that intersecting walls are perpendicular, e.g. like a cuboid or "L"-shape in a top-down view. For panoramic images, the Manhattan constraints can be easily incorporated, by utilizing the characteristic that the columns of the panorama correspond to rotation angles of the camera. We parameterize the layout coordinates in the top-down view as a vector of 2D points $\boldsymbol{L_v} = \{\boldsymbol{v_1} = (0,0), \boldsymbol{v_2} = (x_1, y_1), \ldots, \boldsymbol{v_N} = (x_N, y_N)\}$.

$\boldsymbol{v_1}$ resolves the translation ambiguity, and $|\boldsymbol{v_1} - \boldsymbol{v_2}| = 1$ sets the scale. Because the layout is assumed to be Manhattan, neighboring vertices will share one coordinate value, which further reduces the number of free parameters. We recover the camera position $\boldsymbol{v_c} = \{x_c, y_c\}$ and $\boldsymbol{L_v}$ based on the following generalized energy minimization inspired by Farin et al. [8]:

$$E(\boldsymbol{L_v}, \boldsymbol{v_c}) = \min_{\boldsymbol{v_c}, \boldsymbol{L_v}} \sum_{(i,j) \in \boldsymbol{L_v}} |\beta(\boldsymbol{v_i}, \boldsymbol{v_j}) - \alpha(\boldsymbol{v_i}, \boldsymbol{v_j})| \quad (2)$$

where $\boldsymbol{v_i}, \boldsymbol{v_j}$ are pairs of neighboring vertices, and $\beta_{ij} = arccos \frac{\boldsymbol{v_i} - \boldsymbol{v_c} \cdot \boldsymbol{v_j} - \boldsymbol{v_c}}{\|\boldsymbol{v_i} - \boldsymbol{v_c}\| \|\boldsymbol{v_j} - \boldsymbol{v_c}\|}$ is the rotation angle of the camera $\boldsymbol{v_c}$ between $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$. We denote α_{ij} as the pixel-wise horizontal distance on the image between $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ divided by the length of the panorama. Note that this $L2$ minimization also applies to general Manhattan layouts. We use L-BFGS [34] to solve for Eq. 2 efficiently.

We initialize the ceiling level as the average (mean) of 3D upper-corner heights, and then optimize for a better fitting room layout, relying on both corner and boundary information using the following score to evaluate 3D layout candidate L:

$$\begin{aligned} Score(L) = w_{junc} & \sum_{l_c \in C} \log P_{\text{corner}}(l_c) \\ + w_{ceil} & \sum_{l_e \in L_e} \max \log P_{\text{ceil}}(l_e) \\ + w_{floor} & \sum_{l_f \in L_f} \max \log P_{\text{floor}}(l_f) \end{aligned} \quad (3)$$

where C denotes the 2D projected corner positions of L. Cardinality of L is #walls× 2. We connect the nearby corners on the image to obtain L_e which is the set of projected wall-ceiling boundaries, and L_f which is the set of projected wall-floor boundaries (each with cardinality of #walls). $P_{\text{corner}}(\cdot)$ denotes the pixel-wise probability value on the predicted $\boldsymbol{m_C}$. $P_{\text{ceil}}(\cdot)$ and $P_{\text{floor}}(\cdot)$ denote the probability on $\boldsymbol{m_E}$. The 2nd and 3rd term take the maximum value of log likelihood response in each boundary $l_e \in L_e$ and $l_f \in L_f$. w_{junc}, w_{ceil} and w_{floor} are the term weights, we set to 1.0, 0.5 and 1.0 respectively using grid search. This weighting conforms with the observation that wall-floor corners are often occluded, and the predicted boundaries could help improve the layout reconstruction. We find that adding wall-wall boundaries in the scoring function helps less, since the vertical pairs of predicted corners already reveals the wall-wall boundaries information.

Directly optimizing Eq. 3 is computationally expensive, since we penalize on 2D projections but not direct 3D properties. In this case, we instead sample candidate layout shapes and select the best scoring result based on Eq. 3. We use line search to prune the candidate numbers to speed up the optimization. Algorithm 1 demonstrates the procedure. In each step, we sample candidate layouts by shifting one of the wall position within $\pm\%10$ of its distance to the camera center. Each candidate's ceiling and floor level is then optimized based on the same sampling strategy and scored based on Eq. 3. Once we find the best scored layout by moving one of the walls, we fix this wall position, move to the next wall and perform the sampling again. We start from the least confident wall based on our boundary predictions. In total, $\sim$ 1000 layout candidates are sampled. The optimization step spends less then 30 sec for each image and produces better 3D layouts as demonstrated in Sec. 4.

3.5. Extensions

With small modifications, our network, originally designed to predict cuboid layouts from panoramas, can also predict more general Manhattan layouts from panoramas and cuboid-layouts from perspective images.

General Manhattan layouts: To enable more general layouts, we include training examples that have more than four walls visible (e.g. "L"-shaped rooms), which applies to about 10% of examples. We then determine whether to generate four or six walls by thresholding the score of the sixth strongest wall-wall boundary. Specifically, the average probability along the sixth strongest column of the corner map is at least 0.05. In other words, if there is evidence for more than four walls, our system generates additional walls; otherwise it generates four. Since the available test sets do not have many examples with more than four walls, we show qualitative results with our additional captured samples in Sec. 4.2 and in the supplemental material.

Note that there will be multiple solutions given non-cuboid layout when solving Eq. 2. We experimented with predicting a concave/convex label as part of the corner map prediction to obtain single solution, but observed degraded 2D prediction. We thus enumerate all possible shapes (e.g. for room with six walls, there will be six variations) and choose the one with the best score. We found this heuristic search to be efficient as it searches in a small discrete set. We do not train with the 3D parameter regressor for the non-cuboid layout.

Perspective images: When predicting on perspective images, we skip the alignment and optimization steps, instead directly predicting corners and boundaries on the image. We also do not use the 3D regressor branch. The network predicts a 3-channel boundary layout map with ceiling-wall, wall-wall and wall-floor boundaries, and the corner map has eight channels for each possible corner. Since perspective images have smaller fields of view and the number of visible corners varies, we add a small decoding branch that predicts the room layout type, similar to RoomNet [15]. The predictor has 4 fully-connected (fc) layers with 1024, 256, 64 and 11 nodes, with ReLU operations in between. The predicted layout type then determines which corners are detected, and

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
PanoContext [32]	67.23	1.60	4.55
ours (corner)	73.16	1.08	4.10
ours (corner+boundary)	73.26	1.07	**3.31**
ours full (corner+boundary+3D)	**74.48**	**1.06**	3.34
ours w/o alignment	69.91	1.44	4.39
ours w/o cuboid constraint	72.56	1.12	3.39
ours w/o layout optimization	73.25	1.08	3.37
ours w/ $L2$ loss	73.55	1.12	3.43
ours full w/ Stnfd. 2D-3D data	75.12	1.02	3.18

Table 1. Quantitative results on cuboid layout estimation from panorama using PanoContext dataset [32]. We compare the PanoContext method, and include an ablation analysis on a variety of configurations of our method. Bold numbers indicate the best performance when training on PanoContext data.

Method	Average CPU time (s)
PanoContext [32]	> 300
ours full (corner+boundary+3D)	44.73
ours w/o alignment	31.00
ours w/o cuboid constraint	13.75
ours w/o layout optimization	14.23

Table 2. Average CPU time for each method. We evaluate the methods on the PanoContext dataset [32] using Matlab on Linux machine with an Intel Xeon 3.5G Hz (6 cores).

the corners are localized as the most probable positions in the corner maps. We use cross entropy loss to jointly train the layout boundary and corner predictors. To ease training, similar to the procedure in Sec. 3.3, we first train the boundary/corner predictors, and then add the type predictor branch and train all components together.

4. Experiments

We implement our LayoutNet with Torch and test on a single NVIDIA Titan X GPU. The layout optimization is implemented with Matlab R2015a and is performed on Linux machine with Intel Xeon 3.5G Hz in CPU mode.

We demonstrate the effectiveness of our approach on the following tasks: 1) predict 3D cuboid layout from a single panorama, 2) estimate 3D non-cuboid Manhattan layout from a single panorama, and 3) estimate layout from a single perspective image. We train only on the training split of each public dataset and tune the hyper-parameters on the validation set. We report results on the test set. Our final corner/boundary prediction from the LayoutNet is averaged over results with input of the original panoramas/images and the left-right flipped ones. Please find more results in the supplemental materials.

4.1. Cuboid layout for panorama

We evaluate our approach on three standard metrics:

1. 3D Intersection over Union (IoU), calculated between our predicted 3D layout and the ground truth and averaged across all images;

Method	3D IoU (%)	Corner error (%)	Pixel error (%)
ours (corner)	72.50	1.27	3.44
ours (corner+boundary)	75.26	1.03	**2.68**
ours full (corner+boundary+3D)	75.39	**1.01**	2.70
ours w/o alignment	68.56	1.56	3.70
ours w/o cuboid constraint	74.13	1.08	2.87
ours w/o layout optimization	74.47	1.07	2.92
ours w/ $L2$ loss	**76.33**	1.04	2.70
ours full w/ PanoContext data	77.51	0.92	2.42

Table 3. Evaluation on our labeled Stanford 2D-3D annotation dataset. We evaluate our LayoutNet approach with various configurations for ablation study. Bold numbers indicate best performance when training only on Stanford 2D-3D training set.

2. Corner error, the $L2$ distance between predicted room corner and the ground truth, normalized by the image diagonal and averaged across all images;
3. Pixel error, the pixel-wise accuracy between the layout and the ground truth, averaged across all images.

We perform our method using the same hyper-parameter on the following two datasets.

PanoContext dataset: The PanoContext dataset [32] contains 500 annotated cuboid layouts of indoor environments such as bedrooms and living rooms. Since there is no existing validation set, we carefully split 10% validation images from the training samples so that similar rooms do not appear in the training split. Table 1 shows the quantitative comparison of our method, denoted as "ours full (corner+boundary+3D)", compared with the state-of-the-art cuboid layout estimation by Zhang et al. [32], denoted as "PanoContext". Note that PanoContext incorporates object detection as a factor for layout estimation. Our LayoutNet directly recovers layouts and outperforms the state-of-the-art on all the three metrics. Figure 3 shows the qualitative comparison. Our approach presents better localization of layout boundaries, especially for a better estimate on occluded boundaries, and is much faster in time as shown in Table 2.

Our labeled Stanford 2D-3D annotation dataset: The dataset contains 1413 equirectangular RGB panorama collected in 6 large-scale indoor environment including office and classrooms and open space like corridors. Since the dataset does not contain applicable layout annotations, we extend the annotations with carefully labeled 3D cuboid shape layout, providing 571 RGB panoramas with room layout annotations. We evaluate our LayoutNet quantitatively in Table 3 and qualitatively in Figure 4. Although the Stanford 2D-3D annotation dataset is more challenging with smaller vertical field of view (FOV) and more occlusions on the wall-floor boundaries, our LayoutNet recovers the 3D layouts well.

Ablation study: We show, in Table 1 and Table 3, the performance given the different configurations of our approach: 1) with only room corner prediction, denoted as

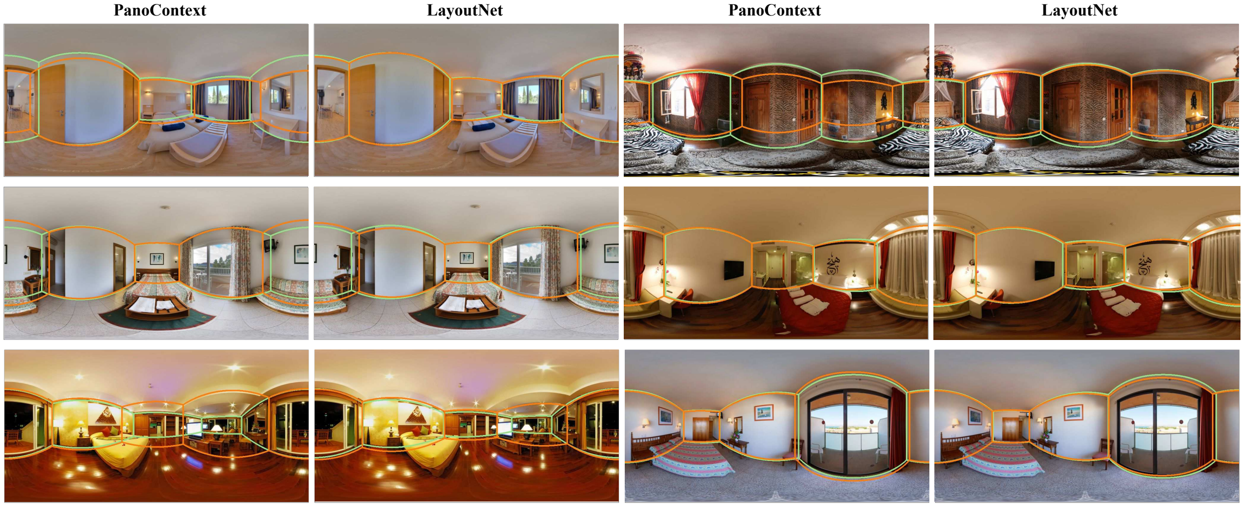

Figure 3. **Qualitative results (randomly sampled) for cuboid layout prediction on PanoContext dataset [32].** We show both our method's performance (even columns) and the state-of-the-art [32] (odd columns). Each image consists predicted layout from given method (orange lines) and ground truth layout (green lines). Our method is very accurate on the pixel level, but as the IoU measure shows in our quantitative results, the 3D layout can be sensitive to even small 2D prediction errors. Best viewed in color.

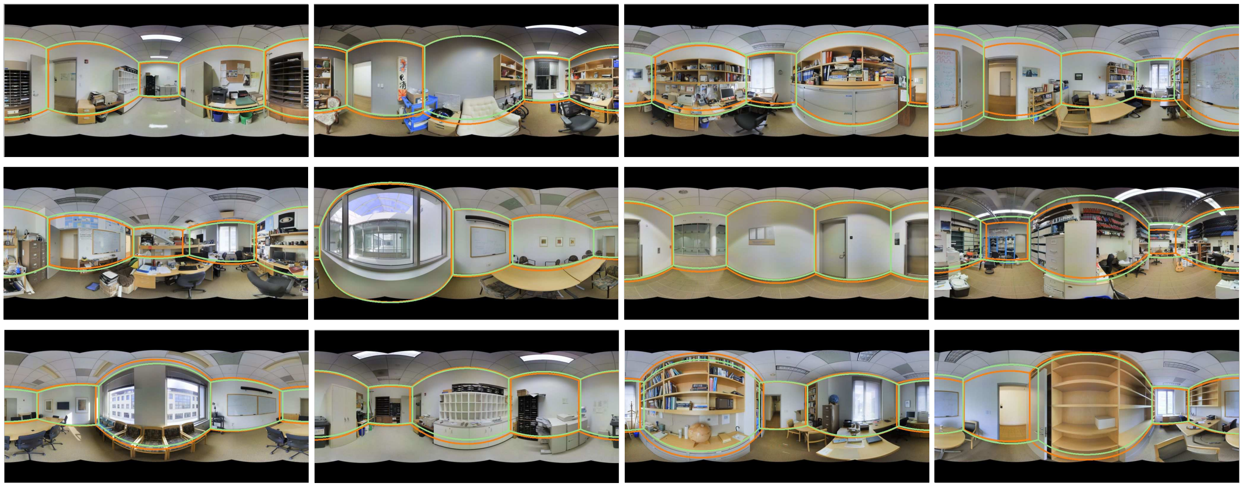

Figure 4. **Qualitative results (randomly sampled) for cuboid layout prediction on the Stanford 2D-3D annotation dataset.** This dataset is more challenging than the PanoContext dataset, due to a smaller vertical field of view and more occlusion. We show our method's predicted layout (orange lines) compared with the ground truth layout (green lines). Best viewed in color.

"ours (corner)"; 2) joint prediction of corner and boundary, denoted as "ours (corner+boundary)"; 3) our full approach with 3D layout loss, denoted as "ours full (corner+boundary+3D)"; 4) our full approach trained on a combined dataset; 5) our full approach without alignment step; 6) our full approach without cuboid constraint; 7) our full approach without layout optimization step; and 8) our full approach using $L2$ loss for boundary/corner prediction instead of cross entropy loss. Our experiments show that the full approach that incorporates all configurations performs better across all the metrics. Using cross entropy loss appears to have a better performance than using L2. Training with 3D regressor has a small impact, which is the part of the reason we do not use it for perspective images. Table 2 shows the average runtimes for different configurations.

Method	L2 dist	cosine dist
Yang et al. [30]	27.02	**4.27**
Ours	**18.51**	5.85

Table 4. Depth distribution error compared with Yang et al. [30].

Comparison to other approaches: We compare with Yang et al. based on their depth distribution metric. We directly run our full cuboid layout prediction (deep net trained on PanoContext + optimization) on 88 indoor panoramas collected by Yang et al. As shown in Table 4, our approach outperforms Yang et al. in L2 distance and is slightly worse in cosine distance. Another approach, Pano2CAD [29], has not made their source code available and has no evaluation on layout, making direct comparison difficult. For time

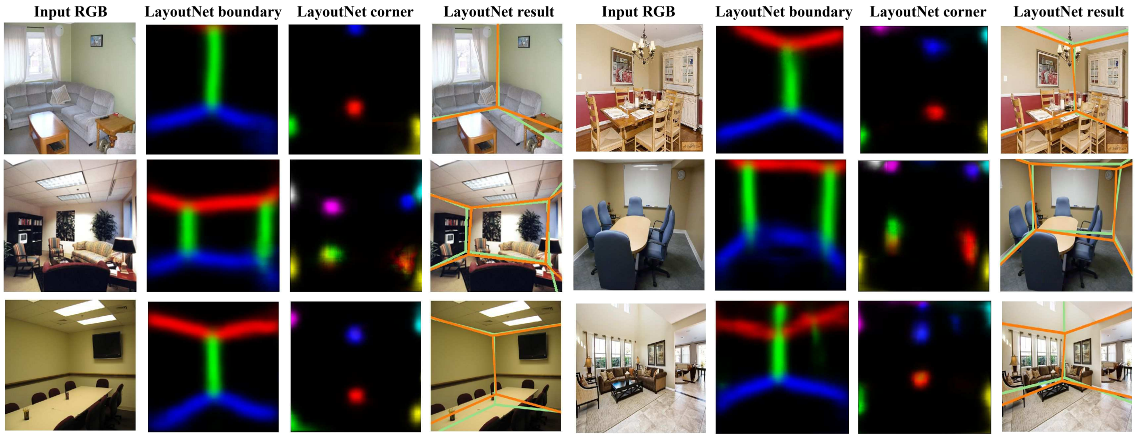

Figure 5. **Qualitative results for perspective images**. We show the input RGB image, our predicted boundary/corner map and the final estimated layout (orange lines) compared with ground truth (green lines). Best viewed in color.

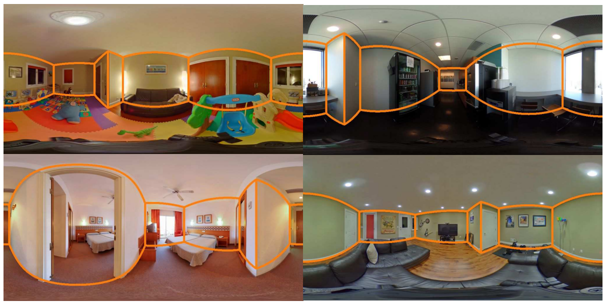

Figure 6. **Qualitative results for non-cuboid layout prediction.** We show our method's predicted layout (orange lines) for non-cuboid layouts such as "L"-shaped rooms. Best viewed in color.

consumption, Yang et al. report to be less than 1 minute, Pano2CAD takes 30s to process one room. One forward pass of LayoutNet takes 39ms. In CPU mode (w/o parallel for loop) using Matlab R2015a, our cuboid constraint takes 0.52s, alignment 13.73s, and layout optimization 30.5s.

4.2. Non-cuboid layout for panorama

Figure 6 shows qualitative results of our approach to reconstruct non-cuboid Manhattan layouts from single panorama. Due to the limited number of non-cuboid room layouts in the existing datasets, we captured several images using a Ricoh Theta-S 360° camera. Our approach is able to predict 3D room layouts with complex shape that are difficult for existing methods.

4.3. Perspective images

We use the same experimental setting as in [4, 15]. We train our modified approach to jointly predict room type on the training split of the LSUN layout estimation challenge. We do not train on the validation split.

Table 5 shows our performance compared with the state-of-the-art on Hedau's dataset [10]. Our method ranks second among the methods. Our method takes 39ms (25 FPS) to process a perspective image, faster than the 52ms (19 FPS) of RoomNet basic [15] or 168ms (6 FPS) of RoomNet recurrent, under the same hardware configuration. We report the result on LSUN dataset in the supplemental material. Figure 5 shows qualitative results on the LSUN validation split. Failure cases include room type prediction error (last row, right column) and heavy occlusion from limited field of view (last row, left column).

Method	Pixel Error (%)
Schwing et al. [26]	12.8
Del Pero et al. [6]	12.7
Dasgupta et al. [4]	9.73
LayoutNet (ours)	9.69
RoomNet recurrent 3-iter [15]	**8.36**

Table 5. Performance on Hedau dataset [10]. We show the top 5 results, LayoutNet ranks second to RoomNet recurrent 3-iter in Pixel Error (%).

5. Conclusion

We propose LayoutNet, an algorithm that predicts room layout from a single panorama or perspective image. Our approach relaxes the commonly assumed cuboid layout limitation and works well with non-cuboid layouts (e.g. "L"-shape room). We demonstrate how pre-aligning based on vanishing points and Manhattan constraints substantially improve the quantitative results. Our method operates directly on panoramic images (rather than decomposing into perspective images) and is among the state-of-the-art for the perspective image task. Future work includes extending to handle arbitrary room layouts, incorporating object detection for better estimating room shapes, and recovering a complete 3D indoor model recovered from single images.

Acknowledgements

This research is supported in part by NSF award 14-21521, ONR MURI grant N00014-16-1-2007, and Zillow Group. We thank Zongyi Wang for his invaluable help with panorama annotation.

References

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv:1702.01105*, 2017.

[2] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *CVPR*, pages 628–635, 2014.

[3] J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *ICCV*, volume 2, pages 941–947. IEEE, 1999.

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

[5] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *CVPR*, pages 2719–2726, 2012.

[6] L. Del Pero, J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard. Understanding bayesian rooms using composite 3d object models. In *CVPR*, pages 153–160, 2013.

[7] E. Delage, H. Lee, and A. Y. Ng. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *CVPR*, volume 2, pages 2418–2428. IEEE, 2006.

[8] D. Farin, W. Effelsberg, et al. Floor-plan reconstruction from panoramic images. In *ACM Multimedia*, pages 823–826. ACM, 2007.

[9] R. Guo, C. Zou, and D. Hoiem. Predicting complete 3d models of indoor scenes. *arXiv:1504.02437*, 2015.

[10] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[11] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. *ECCV*, pages 224–237, 2010.

[12] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, volume 1, pages 654–661. IEEE, 2005.

[13] H. Izadinia, Q. Shan, and S. M. Seitz. IM2CAD. In *CVPR*, 2017.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[15] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. Roomnet: End-to-end room layout estimation. *arXiv:1703.06241*, 2017.

[16] D. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, pages 1288–1296, 2010.

[17] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, pages 2136–2143. IEEE, 2009.

[18] C. Liu, P. Kohli, and Y. Furukawa. Layered scene decomposition via the occlusion-crf. In *CVPR*, pages 165–173, 2016.

[19] C. Liu, A. G. Schwing, K. Kundu, R. Urtasun, and S. Fidler. Rent3d: Floor-plan priors for monocular layout estimation. In *CVPR*, pages 3413–3421, 2015.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[21] A. Mallya and S. Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, pages 936–944, 2015.

[22] S. Ramalingam, J. K. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *CVPR*, pages 3065–3072, 2013.

[23] G. Randall, J. Jakubowicz, R. G. von Gioi, and J.-M. Morel. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 32:722–732, 2008.

[24] Y. Ren, C. Chen, S. Li, and C. J. Kuo. A coarse-to-fine indoor layout estimation (CFILE) method. *arXiv:1607.00598*, 2016.

[25] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[26] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *CVPR*, pages 2815–2822, 2012.

[27] A. G. Schwing and R. Urtasun. Efficient exact inference for 3d indoor scene understanding. In *ECCV*, pages 299–313, 2012.

[28] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[29] J. Xu, B. Stenger, T. Kerola, and T. Tung. Pano2CAD: Room layout from a single panorama image. *WACV*, pages 354–362, 2017.

[30] H. Yang and H. Zhang. Efficient 3d room shape recovery from a single panorama. In *CVPR*, 2016.

[31] J. Zhang, C. Kan, A. G. Schwing, and R. Urtasun. Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In *ICCV*, pages 1273–1280, 2013.

[32] Y. Zhang, S. Song, P. Tan, and J. Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *ECCV*, pages 668–686, 2014.

[33] Y. Zhao and S.-C. Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, pages 3119–3126, 2013.

[34] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.