Supplementary Materials for 3D Room Layout Estimation from a Cubemap of Panorama Image via Deep Manhattan Hough Transform

Yining Zhao[1], Chao Wen[2], Zhou Xue[2], and ✉Yue Gao[1]

[1] BNRist, THUIBCS, BLBCI, KLISS, School of Software, Tsinghua University
[2] Pico IDL, ByteDance

A Proof of the Manhattan Lines Orientation in Cubemap

Proposition. Given a panoramic image of a 3D room, if the following two assumptions holds:

1. The Manhattan world assumption, *i.e.*, all of the walls, the ceiling and the floor must be perpendicular to each other, and all of the intersection lines of them must be parallel with one of the coordinate axes of some orthogonal coordinate space (named Manhattan space).
2. The input image must be aligned, *i.e.*, the camera of each cubemap tile faces precisely to one of the walls, and its optical axis is parallel with one of the coordinate axes of the Manhattan space.

and the cubemap of the panoramic images is generated by making E2P transform six times with azimuth angle $u = 0°, 90°, 180°, -90°, 0°, 0°$ and elevation angle $v = 0°, 0°, 0°, 0°, 90°, -90°$ respectively and FoV$= 90°$ for all cubemap tiles, then for any of the lines in the wireframe of the 3D room, it must be either a horizontal line ($\theta = 0$), a vertical line ($\theta = \pi/2$) or a line passing the center ($\rho = 0$) in the cubemap tiles.

(i) Preliminary. Coordinate Definition. Two coordinate space is defined: the 3D space and the image space, as Fig. 1 shows. In the 3D space, the position of the panoramic camera is the origin, the y axis is along the optical axis of the camera, the z axis points to the ceiling vertically, and the x axis is orthogonal to the y axis points to the right. In the image space, the center of the image is the origin, the x axis points to the right and the y axis points to the bottom.

✉Corresponding author.

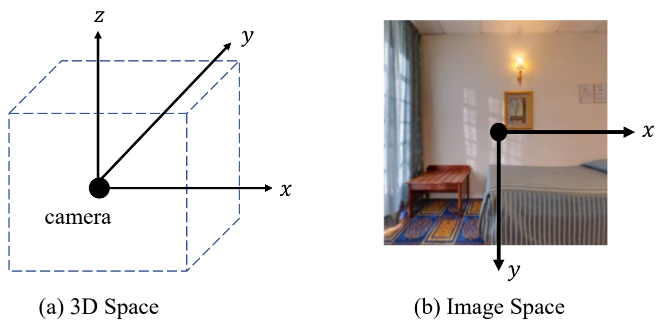


Fig. 1: Coordinate space definition of 3D space and image space.

(ii) Proof. Coordinate Transform from the 3D Space to the Image Space. Given a coordinate p in the 3D space, to get its coordinate q in the image space of a specific cubemap view, we can first apply the observation transform:

$$r = (M_u M_v)^{-1} \cdot p \tag{1}$$

in which M_u is the rotation matrix representing rotation along the z axis by angle u, and M_v is the rotation matrix representing rotation along the x axis by angle v:

$$\begin{aligned} M_u &= \begin{bmatrix} \cos(u) & -\sin(u) & 0 \\ \sin(u) & \cos(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ M_v &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v) & -\sin(v) \\ 0 & \sin(v) & \cos(v) \end{bmatrix} \end{aligned} \tag{2}$$

Then we can apply the projection:

$$\begin{aligned} q_x &= \frac{r_x}{r_y} \cdot \cot(\alpha_H) \cdot \frac{w}{2} \\ q_y &= \frac{-r_z}{r_y} \cdot \cot(\alpha_V) \cdot \frac{h}{2} \end{aligned} \tag{3}$$

in which $q = (q_x, q_y)^\top$ is the coordinate in the image space, r_x, r_y is the x, y components of r, α_H, α_V is the FoV in horizontal and vertical direction respectively, and h, w is the height and width of the image.

Since $\alpha_H, \alpha_V = 90°$, from Eq. (1) (2) (3), we have

$$\begin{aligned} q_x &= \frac{w}{2} \cdot \frac{p_x \cos(u) + p_y \sin(u)}{-p_x \cos(v)\sin(u) + p_y \cos(v)\cos(u) + p_z \sin(v)} \\ q_y &= \frac{h}{2} \cdot \frac{-p_x \sin(v)\sin(u) + p_y \sin(v)\cos(u) - p_z \cos(v)}{-p_x \cos(v)\sin(u) + p_y \cos(v)\cos(u) + p_z \sin(v)} \end{aligned} \tag{4}$$

(iii) Conclusion. Manhattan Line Orientation. Take the cubemap tile at the front of the camera, which corresponds to $u = 0°, v = 0°$, as an example. By substituting u and v into Eq. (4), we get

$$q_x = \frac{w}{2} \cdot \frac{p_x}{p_y}$$
$$q_y = \frac{h}{2} \cdot \frac{-p_z}{p_y} \tag{5}$$

For a horizontal wireframe line which is along the x axis in the 3D space, take any two points $p_1 = (p_{x1}, p_{y1}, p_{z1})^\top$ and $p_2 = (p_{x2}, p_{y1}, p_{z1})^\top$ on the line (Notice that the two points has the same y and z coordinates). By Eq. (5), we have $q_1 = \left(\frac{w}{2} \cdot \frac{p_{x1}}{p_{y1}}, \frac{h}{2} \cdot \frac{-p_{z1}}{p_{y1}}\right)$, $q_2 = \left(\frac{w}{2} \cdot \frac{p_{x2}}{p_{y1}}, \frac{h}{2} \cdot \frac{-p_{z1}}{p_{y1}}\right)$. q_1 and q_2 have the same y and different x coordinate in the image space, so it is a horizontal line in the cubemap tile.

For a vertical wireframe line which is along the z axis in the 3D space, take any two points $p_3 = (p_{x3}, p_{y3}, p_{z3})^\top$ and $p_4 = (p_{x3}, p_{y3}, p_{z4})^\top$. Similarly by Eq. (5), we have $q_3 = \left(\frac{w}{2} \cdot \frac{p_{x3}}{p_{y3}}, \frac{h}{2} \cdot \frac{-p_{z3}}{p_{y3}}\right)$, $q_4 = \left(\frac{w}{2} \cdot \frac{p_{x3}}{p_{y3}}, \frac{h}{2} \cdot \frac{-p_{z4}}{p_{y3}}\right)$, which have the same x and different y coordinate in the image space, so it is a vertical line in the cubemap tile.

Finally, for a horizontal wireframe line which is along the y axis in the 3D space, take any two points $p_5 = (p_{x5}, p_{y5}, p_{z5})^\top$ and $p_6 = (p_{x5}, p_{y6}, p_{z5})^\top$. Similarly by Eq. (5), we have $q_5 = \left(\frac{w}{2} \cdot \frac{p_{x5}}{p_{y5}}, \frac{h}{2} \cdot \frac{-p_{z5}}{p_{y5}}\right)$, $q_6 = \left(\frac{w}{2} \cdot \frac{p_{x5}}{p_{y6}}, \frac{h}{2} \cdot \frac{-p_{z5}}{p_{y6}}\right)$. Naturally, we have $\frac{q_{y5}-0}{q_{x5}-0} = \frac{-h \cdot p_{z5}}{w \cdot p_{x5}} = \frac{q_{y6}-0}{q_{x6}-0}$, so the three points q_5, q_6 and $(0, 0)$ lie on the same line, *i.e.* the line passing q_5 and q_6 must be a line that passes the center of the cubemap tile.

Similarly, the proposition can also be proved for the other cubemap tiles by substituting the u and v into Eq. (4), and derive the point coordinates in the image space for each of the three directions of lines in the 3D space.

B Implementation Details

B.1 Hough Voting for Lines Passing the Center

Given a channel of the features of a cubemap tile $\mathbf{X} \in \mathbb{R}^{h \times w}$, in our implementation, the vector generated by Manhattan Hough Transform $\mathbf{C} \in \mathbb{R}^{2(h+w)}$, with each of the bins in $\mathbf{C}$ representing a line which starts at the center $(x = 0, y = 0)$ and ends at a bin on the border of the image $(x = x_0, y = y_0)$, in which either $x_0 = \pm\frac{w}{2}, y_0 \in \mathbb{Z}$ or $x_0 \in \mathbb{Z}, y_0 = \pm\frac{h}{2}$. Note that in our experiment, $h = w = 512$.

For each line, if it intersects the border of the image on the left or the right, *i.e.* $\Delta y < \Delta x$, the value in the bin of the Hough vector $\mathbf{C}$ is the sum of $\frac{w}{2}$ values corresponding to $\frac{w}{2}$ points whose coordinate x is integer. For each integer x, we calculate the y coordinate (which may not be integer). If $y \in \mathbb{Z}$, the value of this point is naturally defined as the pixel $\mathbf{X}[x, y]$. (The brackets "[ ]" denote

Block	Output Channels	Output Size
`block1`	32	256×256
`block2`	64	128×128
`block3`	128	64×64
`block4`	256	64×64
`block5`	512	64×64

Table 1: Summary of the feature maps output by each block of the encoder.

the index of the feature map.) If $y \notin \mathbb{Z}$, the value of this point is defined as the linear interpolation of the two pixel $\mathbf{X}[x, \lfloor y \rfloor]$ and $\mathbf{X}[x, \lceil y \rceil]$.

When the line intersects the border of the image on the top or the bottom, *i.e.* $\Delta x < \Delta y$, the definition is similar except that the points used for summation is selected by coordinate y being integer.

B.2 Network Structure

Here we introduce our network structure in detail. Our Network can be divided into three parts: feature extractor, Manhattan Hough Head and output modules.

Feature Extractor. In the following example, we take DRN-38[10] as the encoder. The structure of the encoder blocks are summarized in Tab. 1.

Manhattan Hough Head. We extract multi-scale features from the encoder. We obtain five feature maps of different scales and feed them into the corresponding Manhattan Hough Head module. The network contains five separate Manhattan Hough Heads. In Tab. 2, we take the output of block1 of the encoder, *i.e.* `encoder.block1`, as an example to introduce the structure of the Manhattan Hough Head. The final three layers, `conv1d_H.2`, `conv1d_V.2` and `conv1d_C.2` generates the three output feature vectors $\mathbf{H}$, $\mathbf{V}$ and $\mathbf{C}$ of the Manhanttan Hough Head respectively.

Output Modules. The final module processes the three types of feature vectors of $\mathbf{H}$, $\mathbf{V}$ and $\mathbf{C}$ to obtain probabilities. We take the output module corresponding to $\mathbf{H}$ as an example to introduce the detailed structure. Since feature vectors from each Manhattan Hough Head have different size, they must be upsampled before being concatenated. Specifically, the channels of the output by each Manhattan Hough Head are $16, 32, 64, 128, 256$ respectively. Denote $\mathbf{H}_n^{\uparrow}$ is the $\mathbf{H}$ feature vectors output by the n-th Manhattan Hough Head after being upsampled, the structure of the output module is shown in Tab. 3. Note that there are another two separate output modules for $\mathbf{V}$ and $\mathbf{C}$, whose structures are the same except for output size.

Layer	**Input**	**Channels**	**Output Size**	**k**	**p**	**BN**	**Act.**
`conv2d_H`	`encoder.block1`	32→16	256 × 256	1 × 1	0	✓	ReLU
`conv2d_V`	`encoder.block1`	32→16	256 × 256	1 × 1	0	✓	ReLU
`conv2d_C`	`encoder.block1`	32→16	256 × 256	1 × 1	0	✓	ReLU
$\mathcal{MH}$_H	`conv2d_H`	16→16	256				
$\mathcal{MH}$_V	`conv2d_V`	16→16	256				
$\mathcal{MH}$_C	`conv2d_C`	16→16	1024				
`conv1d_H.1`	$\mathcal{MH}$_H	16→16	256	3	1	✓	ReLU
`conv1d_V.1`	$\mathcal{MH}$_V	16→16	256	3	1	✓	ReLU
`conv1d_C.1`	$\mathcal{MH}$_C	16→16	1024	3	1	✓	ReLU
`conv1d_H.2`	`conv1d_H.1`	16→16	256	3	1	✓	ReLU
`conv1d_V.2`	`conv1d_V.1`	16→16	256	3	1	✓	ReLU
`conv1d_C.2`	`conv1d_C.1`	16→16	1024	3	1	✓	ReLU

Table 2: **Manhattan Hough Head Architecture.** For all layers we show the input and the number of channels. For convolution layers, we additionally show the kernel size (**k**), the padding (**p**), batch normalization (**BN**), and the activation function (**Act.**). The stride of convolution is set to 1.

Layer	**Input**	**Channels**	**Output Size**	**k**	**p**	**BN**	**Act.**
`concat_H`	$\mathbf{H}_1^{\uparrow}, \mathbf{H}_2^{\uparrow}, \mathbf{H}_3^{\uparrow}, \mathbf{H}_4^{\uparrow}, \mathbf{H}_5^{\uparrow}$	16+32+64+128+256=496	512				
`conv1d_H.1`	`concat_H`	496→248	512	3	1	✓	ReLU
`conv1d_H.2`	`conv1d_H.1`	248→248	512	1	0	✓	ReLU
`conv1d_H.3`	`conv1d_H.2`	248→1	512	1	0		Sigmoid

Table 3: **Output Modules Architecture.** For all layers we show the input and the number of channels. For convolution layers, we additionally show the kernel size (**k**), the padding (**p**), batch normalization (**BN**), and the activation function (**Act.**). The stride of convolution is set to 1.

B.3 Post-processing

We explain the pipeline in Fig. 2 that can handle cuboid and non-cuboid rooms.

1. We convert probabilities to lines by finding the prominent peaks using `scipy.signal.find_peaks`.
2. Each camera-to-wall distance and room height, *i.e.* T, are optimized here. The parameterized layout T, is transformed by projection π onto each tile to

maximize the overall probability according to the network's line probability output S, *i.e.*, $E(T;S) = \|\pi(T) - S\|$.

3. Necessity: Each wall is overdetermined by the raw output (*i.e.* three types of lines). We convert lines to room height and the camera-to-wall distances T, to make the problem determined.
4. For a MW room, we consider the room walls oriented in an x-y alternation. As the input is axis-aligned, the orientations of other walls can be determined successively, so there is no need for the additional plane normal parameter.
5. In Tab. 4, we add an ablation of different optimization steps to demonstrate the necessity of post-processing.

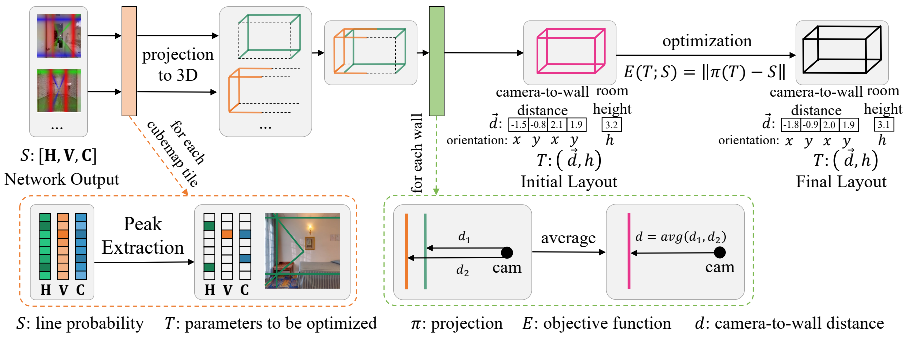


Fig. 2: Post-processing pipeline.

#Opt. Steps	3D IOU	
	PanoContext	Stanford 2D-3D
0	83.65	81.66
10	84.72	83.04
50	85.30	84.36
100	**85.48**	**84.93**

Table 4: Ablation on number of post-processing optimize steps.

C More Baseline

C.1 Naive Baseline

An intuitive idea is to apply the classical Hough transform to line detection and then estimate the room layout. However, our experiments demonstrate that this trivial solution cannot solve the room layout estimation problem effectively.

We propose two baselines based on classical Hough transform: Standard Hough Transform ($\mathcal{HT}$-S) [4] and Probabilistic Hough Line Transform ($\mathcal{HT}$-P) [6]. $\mathcal{HT}$-P improves $\mathcal{HT}$-S, its output format is line segment instead of the whole line. Specifically, after getting the cubemap of the panoramic image with E2P transform, we first turn the cubemap into a grayscale image and use Canny [2] to detect edge. Then, the two classical Hough transform methods are performed on the egde detection result to detect lines, with voting threshold 100 for $\mathcal{HT}$-S and 50 for $\mathcal{HT}$-P. Though classical Hough transform detect lines of every direction and offset, we only keep those which are horizontal ($|\tan(\theta)| < 0.05$), vertical ($|\cot(\theta)| < 0.05$) or passing the center ($\rho < 5$). Since the lines may be too many to perform post-processing, we filter the lines by categorizing them into 8 groups according to their possible position in the 3D space, and keeping only the line with the highest Hough voting value for each group.

The quantitative result in the PanoContext dataset[11] is shown in Tab. 5. The qualitative result is shown in Fig. 3. Though classical Hough transform methods are possible to detect the intersection lines of the walls in the room, they may detect more false-positive lines. The lines not only have no benefits to layout estimation but also make the post-processing module fail to give correct estimation result. In contrast, our proposed DMH-Net uses Deep Manhattan Hough Transform which can learn semantic information from the image and make more accurate detection of the intersection lines of the walls in the room rather than detecting all lines equally.

Method	PanoContext		
	3DIoU	CE	PE
$\mathcal{HT}$-S [4]	23.83	7.01	21.49
$\mathcal{HT}$-P [6]	21.78	6.50	17.20
DMH-Net	**85.48**	**0.73**	**1.96**

Table 5: Quantitative results of cuboid room layout estimation evaluated on the PanoContext dataset [11]. CE means Corner Error, and PE means Pixel Error.

C.2 Baseline Discussion

In the main text, we compare multiple baseline algorithms. To ensure fair comparisons, we use the train/test split that most algorithms use commonly. Since the algorithms proposed in the Zou *et al.* [13] and LGT-Net [5] both use additional data in PanoContext and Stanford 2D-3D experiments, we do not quantitatively compare such methods for fairness.

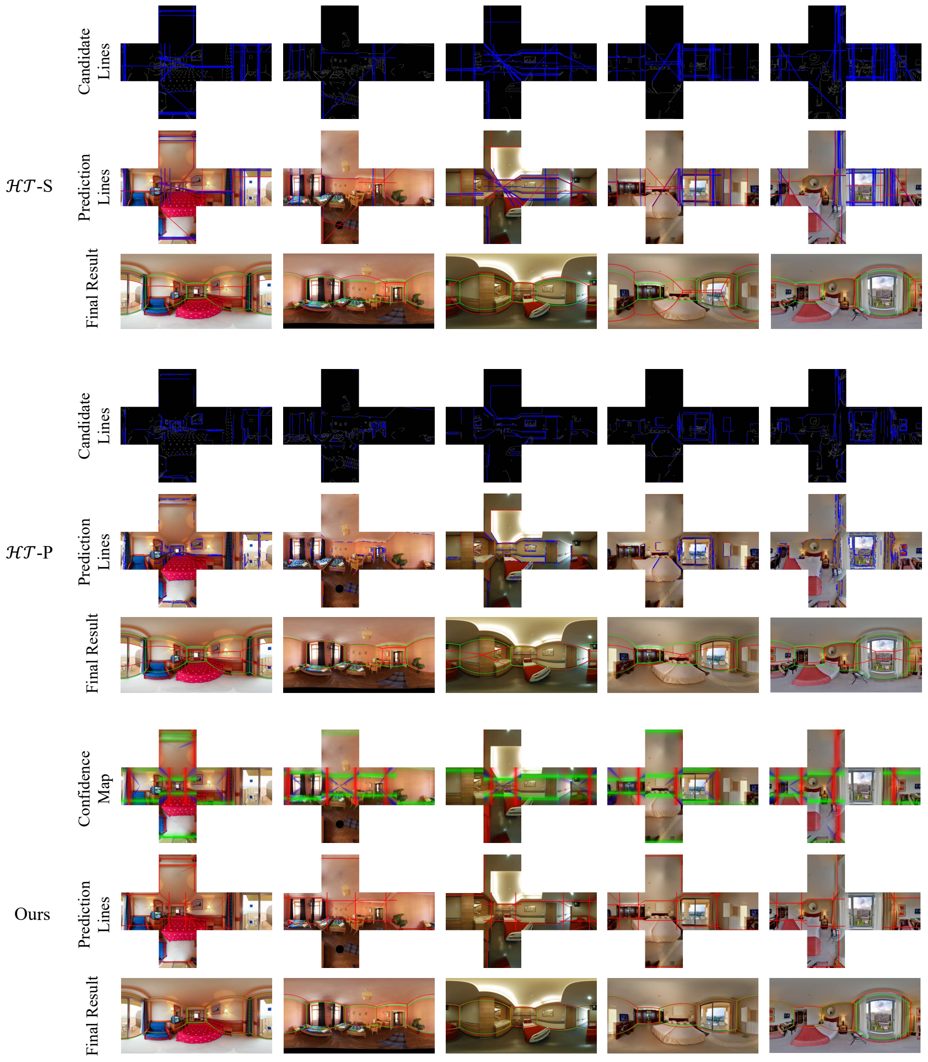


Fig. 3: Qualitative results. For $\mathcal{HT}$-S and $\mathcal{HT}$-P: "Candidate Lines" visualizes the raw output, and the blue lines are the outputs of Hough transform on the Canny edge map. "Prediction Lines" visualizes the lines filtered by the threshold. The blue lines represent all the lines, and the red lines represent the line that meets the threshold. For our method: "Confidence Map" visualizes the raw output of network. The colored heatmap represents the probability of the Manhattan lines (described in Sec. A). "Final Result" visualizes the room layout. The green lines are ground truth layout while the red lines are estimated.

D More Model Analysis

D.1 Voting bin size

The number of bins, *i.e.* bin size, affects the performance. The line predictions will be coarser when the number of bins is reduced. We add an ablation study with 1/2 and 1/4 bins in Tab. 6.

#Bins	3D IOU	
	PanoContext	Stanford 2D-3D
$\times 1$	**85.48**	**84.93**
$\times 1/2$	84.73	83.43
$\times 1/4$	83.77	78.81

Table 6: Ablation on number of bins.

D.2 Speed analysis

We provide time profiling per cuboid room sample on a PC with an Intel i7-8700 CPU and a single NVIDIA 2080Ti GPU. Pre-processing: 0.297s. Network inference: 0.155s. Post-processing: 0.643s.

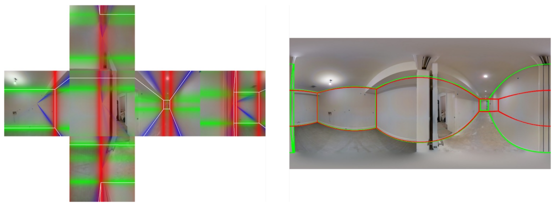
E Failure Cases and Limitations

Two failure cases are given in Fig. 4. Though our method can effectively estimate the 3D room layout for most cases, it also has some limitations. Firstly, our model predict confidence vectors which varies continuously in the Hough space. Thus, it is challenge for our method to distinguish two very near parallel lines in the cubemap tile, as is shown in the cumbmap tile at line 2, column 4 of Fig. 4 (a). Secondly, our method is still lack of interaction between different cubemap tiles during training. Finally, when the wall-wall intersection line is less salient than other lines on the wall, our method may predict incorrect lines, which leads to a layout estimation error. For example, in Fig. 4 (b), our method predict the lines between the blue part and the white part as the wall-ceiling intersection lines of the room.

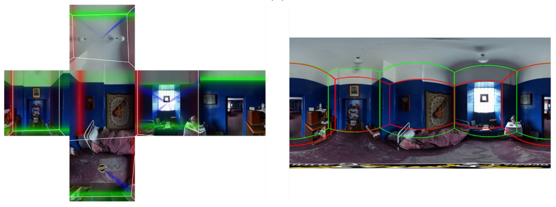
F More Qualitative Results

F.1 Qualitative Results Compared with Baselines

For thorough comparison with our baselines, we show additional qualitative results. As illustrated in Fig. 5 and Fig. 6, we provide the qualitative comparison results with LayoutNet [12,13] and HorizonNet [8] on Stanford 2D-3D dataset [1], PanoContext dataset [11] and Matterport3D dataset [3].



(a)



(b)

Fig. 4: Failure cases of our model. (a) is a non-cuboid sample in the Matterport3D dataset [3], (b) is a cuboid sample in the PanoContext dataset [11]. The green, red and blue lines heatmap in cubemap represents the probabilities of Manhattan line as described in Sec. A. The thin white line in cubemap represents ground truth. The green lines in equirectangular panorama are ground truth layout and red lines are predictions.

F.2 Room Layout Comparison with Other Baselines

In addition, we provide the qualitative comparison results with HoHoNet [9] and AtlantaNet [7] on the Matterport3D dataset [3], as illustrated in Fig. 7.

F.3 3D Visualization

We also show 3D layout visualizations of our predictions to demonstrate the performance of our proposed method. Specifically, Fig. 8 shows the results of Matterport3D dataset [3], and Fig. 9 shows the results of Stanford 2D-3D dataset [1] and PanoContext dataset [11].

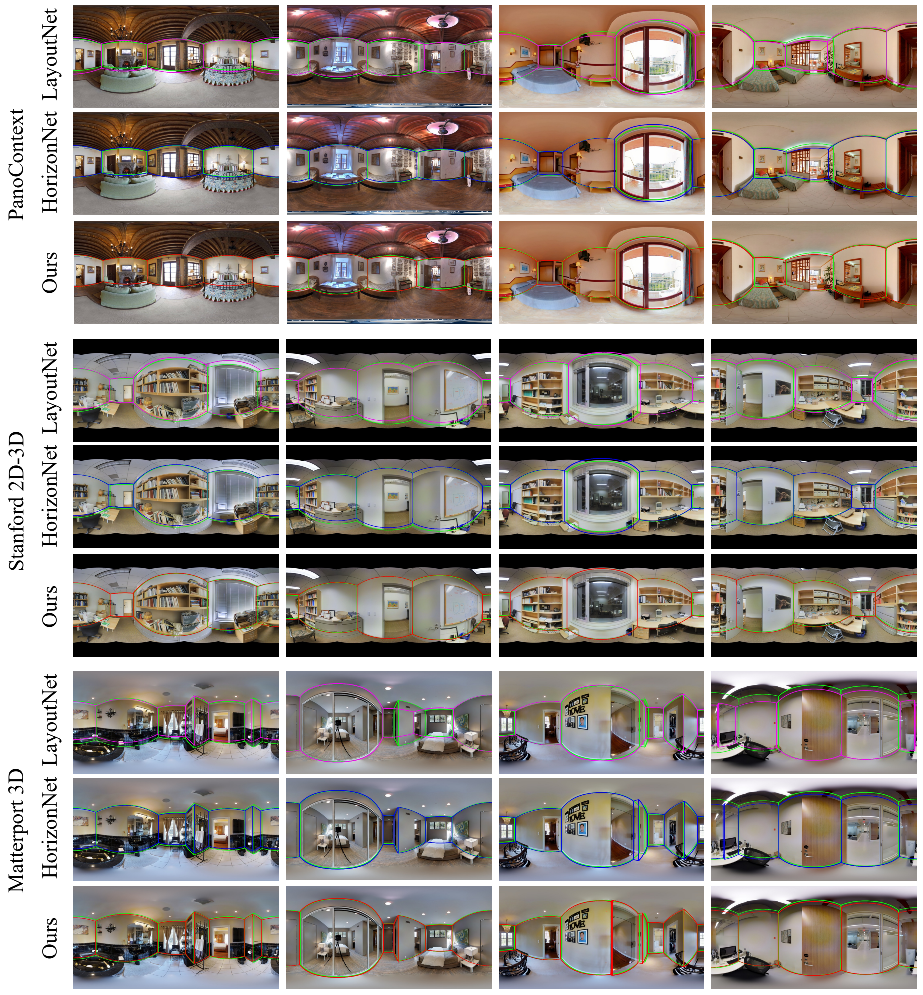


Fig. 5: More qualitative results of both cuboid and non-cuboid layout estimation (1). The green lines are ground truth layout while the pink, blue and red lines are estimated by LayoutNet [12], HorizonNet [8] and our DMH-Net.

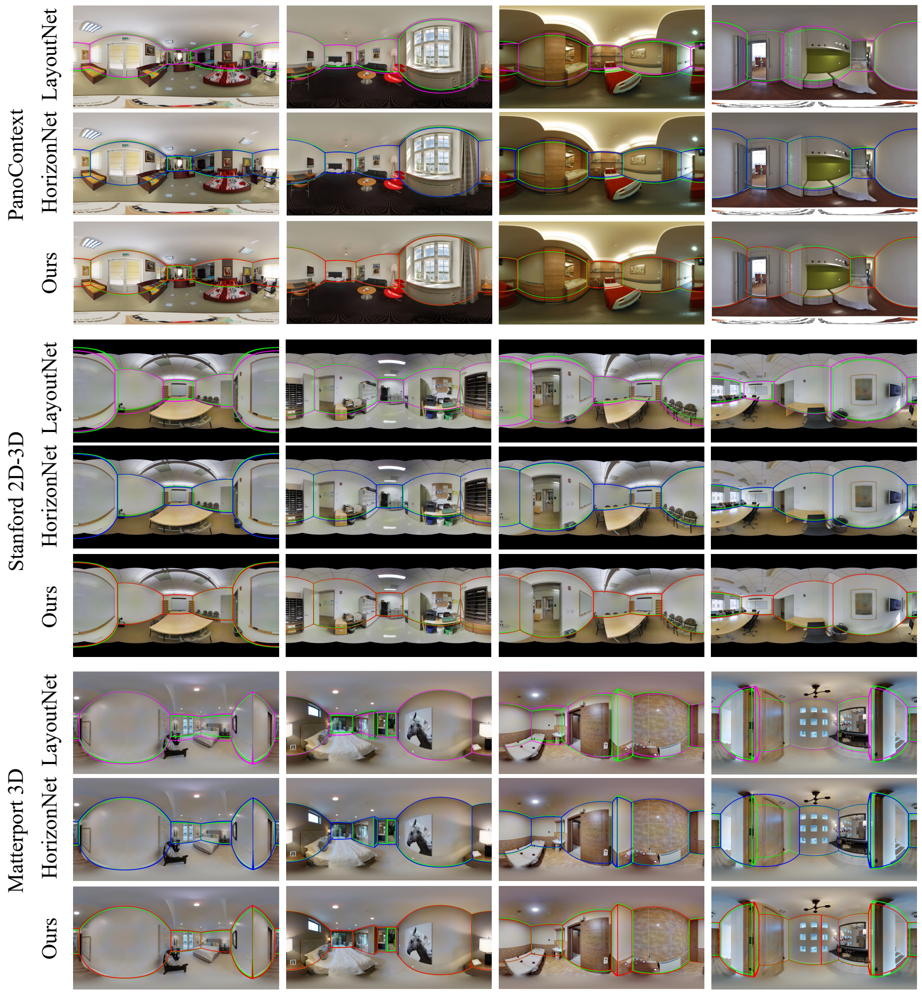


Fig. 6: More qualitative results of both cuboid and non-cuboid layout estimation (2). The green lines are ground truth layout while the pink, blue and red lines are estimated by LayoutNet [12], HorizonNet [8] and our DMH-Net.

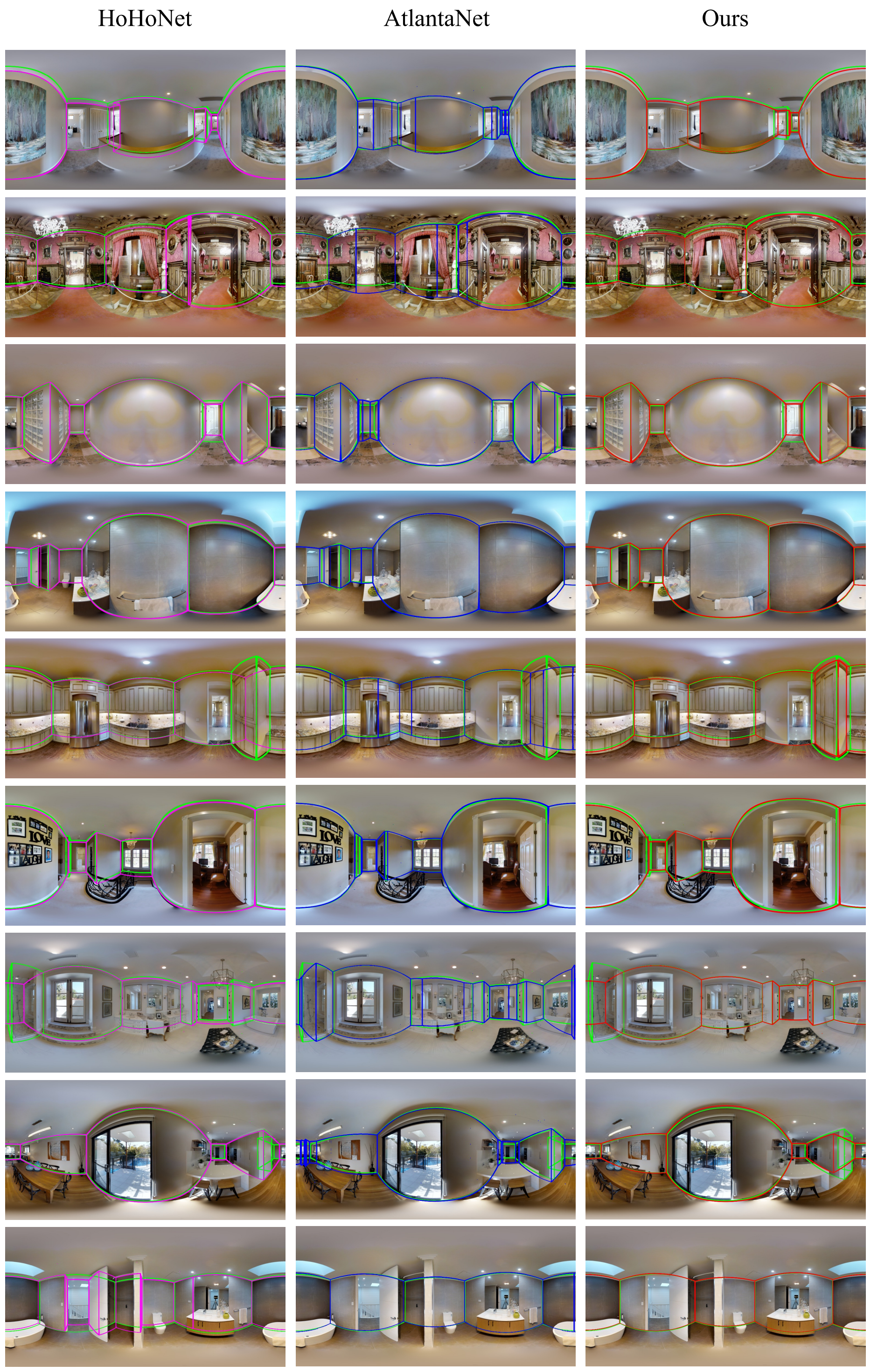


Fig. 7: The qualitative results of Matterport3D [3]. The green lines are ground truth layout while the pink, blue and red lines are estimated by HoHoNet [9], AtlantaNet [7] and our DMH-Net.

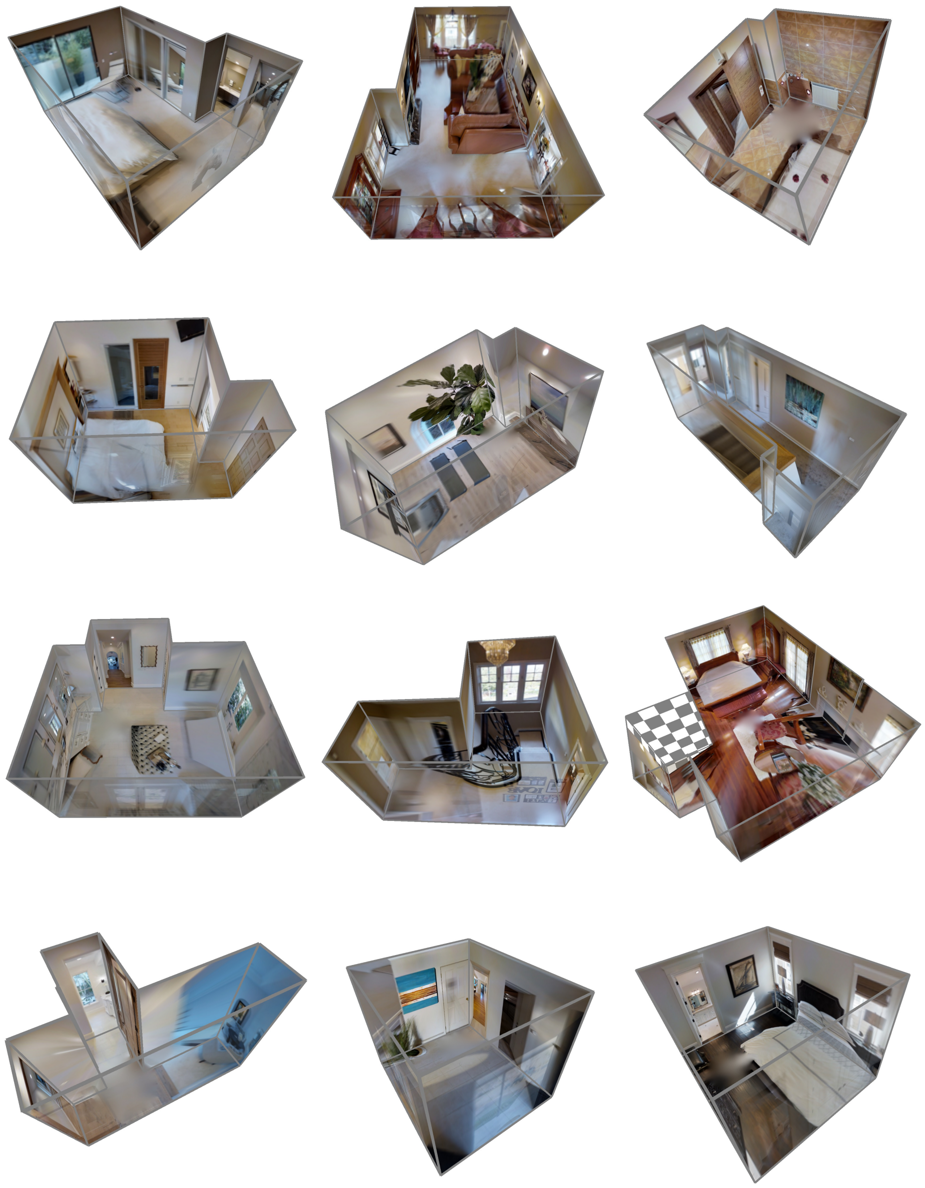


Fig. 8: The 3D visualization results of Matterport3D dataset [3].

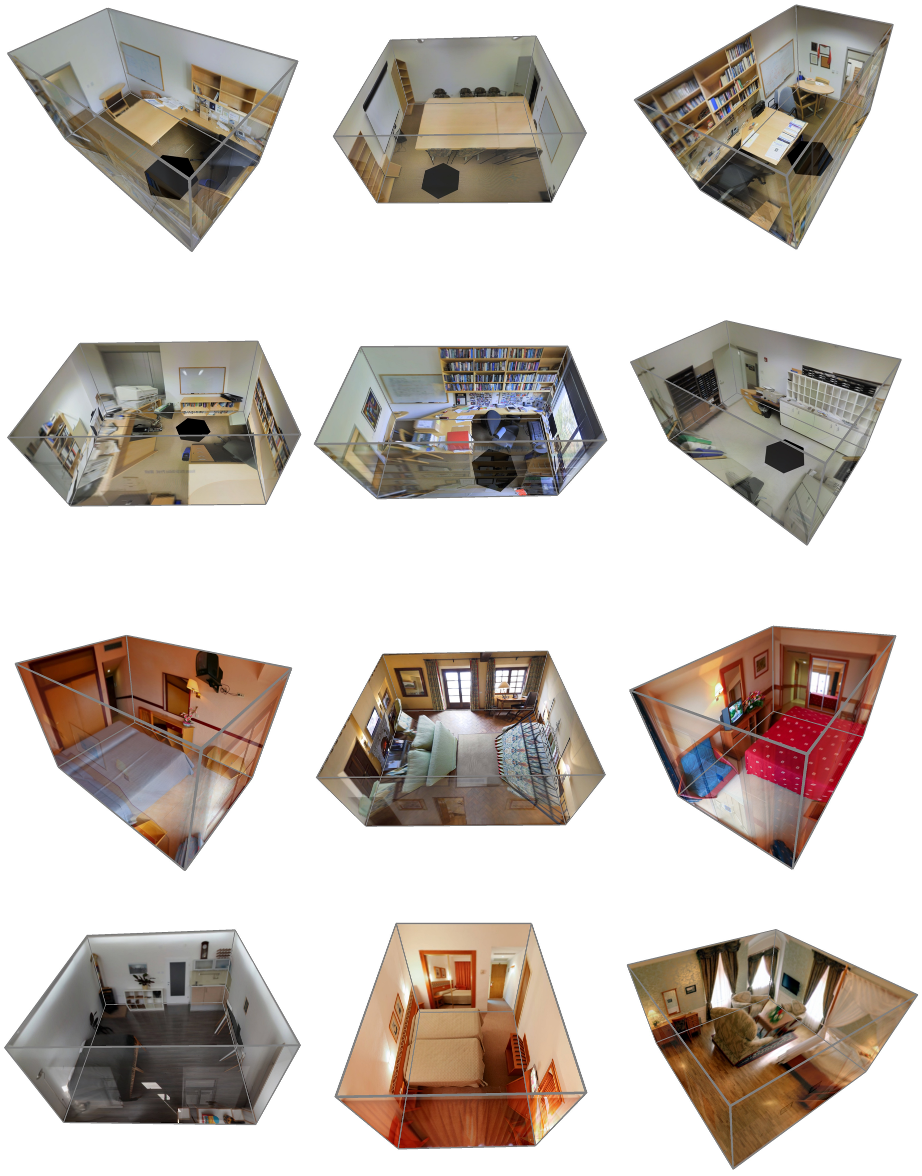


Fig. 9: The 3D visualization results of PanoContext dataset [11] and Stanford 2D-3D dataset [1].

References

1. Armeni, I., Sax, S., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105 (2017) 9, 10, 15
2. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(6), 679–698 (1986) 7
3. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3D: Learning from RGB-D data in indoor environments. Proc. of the International Conf. on 3D Vision (3DV) pp. 667–676 (2017) 9, 10, 13, 14
4. Hough, P.V.: Method and means for recognizing complex patterns (Dec 18 1962), uS Patent 3,069,654 7
5. Jiang, Z., Xiang, Z., Xu, J., Zhao, M.: Lgt-net: Indoor panoramic room layout estimation with geometry-aware transformer network. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 1654–1663 (2022) 7
6. Matas, J., Galambos, C., Kittler, J.: Robust detection of lines using the progressive probabilistic hough transform. Computer vision and image understanding **78**(1), 119–137 (2000) 7
7. Pintore, G., Agus, M., Gobbetti, E.: Atlantanet: Inferring the 3d indoor layout from a single 360° image beyond the manhattan world assumption. In: Proc. of the European Conf. on Computer Vision (ECCV). pp. 432–448 (2020) 10, 13
8. Sun, C., Hsiao, C.W., Sun, M., Chen, H.T.: Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 1047–1056 (2019) 9, 11, 12
9. Sun, C., Sun, M., Chen, H.T.: Hohonet: 360 indoor holistic understanding with latent horizontal features. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 2573–2582 (2021) 10, 13
10. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 636–644 (2017) 4
11. Zhang, Y., Song, S., Tan, P., Xiao, J.: Panocontext: A whole-room 3d context model for panoramic scene understanding. In: Proc. of the European Conf. on Computer Vision (ECCV). pp. 668–686 (2014) 7, 9, 10, 15
12. Zou, C., Colburn, A., Shan, Q., Hoiem, D.: Layoutnet: Reconstructing the 3d room layout from a single rgb image. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 2051–2059 (2018) 9, 11, 12
13. Zou, C., Su, J.W., Peng, C.H., Colburn, A., Shan, Q., Wonka, P., Chu, H.K., Hoiem, D.: Manhattan room layout reconstruction from a single 360° image: A comparative study of state-of-the-art methods. International Journal of Computer Vision (IJCV) **129**(5), 1410–1431 (2021) 7, 9