Seg2Reg: Differentiable 2D Segmentation to 1D Regression Rendering for 360 Room Layout Reconstruction

Supplementary Material

The coordinate system used in this work is explained in detail in Sec. 1. In Sec. 2, we show the results under various setups of the baseline models reproduced by our PanoLayoutStudio. Additional technical details about layout 3D warping and Seg2Reg can be found in Sec. 3 and Sec. 4. Finally, we present an extensive qualitative comparison in Sec. 5.

1. Coordinate system

As we only consider a single image, the camera position is set as the world origin. We use a z-down positive world coordinate system where the positive z points toward the floor.

Image to world. Let i, j be the image row and column index of a pixel. We can transform them into spherical coordinates by

$$u = \left(\frac{(j + 0.5)}{W} - 0.5 \right) \cdot 2\pi \ , \tag{1a}$$

$$v = \left(\frac{(i + 0.5)}{H} - 0.5 \right) \cdot \pi \ , \tag{1b}$$

where u is azimuthal angle and v is the angel with respect to the xy-plane. We can then lift the pixel to 3D by

$$x = d \cdot \cos(v) \sin(u) \ , \tag{2a}$$

$$y = d \cdot \cos(v) \cos(u) \ , \tag{2b}$$

$$z = d \cdot \sin(v) \ , \tag{2c}$$

where d is the pixel depth.

World to image. The inverse transformation of Eq. (2) is

$$u = \arctan2(x, y) \ , \tag{3a}$$

$$v = \arctan2\left(z, \sqrt{x^2 + y^2}\right) \ , \tag{3b}$$

which is used in the operation EqProj (main paper Eq. (2d)) to project sampled 3D points on the floor or the ceiling planes back to the equirectangular image for density interpolation.

2. Baseline tuning

We implement HorizonNet [8], HoHoNet [9], LED2Net [11], and LGTNet [4] into our codebase—PanoLayoutStudio. In this section, we explore various setups of these reproduced baselines and evaluate on MatterportLayout [13] valid set. We use a unified training recipe for all the methods—Adam optimizer with $1e{-}4$ learning rate trained for 1k epochs. We activate Stochastic Weight Averaging [3] in the last 200 epochs to stabilize training. Except stated otherwise, we use the basic setup with LGT-Net [4]—ResNet-34 as backbone; standard left-right flip, circular shifting, PanoStretch [8], and luminance jittering as data augmentation. We accumulate the results from different training seeds as box plots.

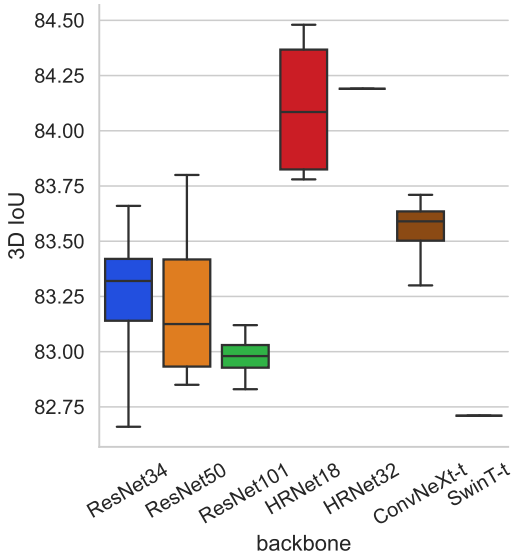


Figure 1. **Results of different backbones.** The results of HRNet32 and SwinT-t are obtained from only one training seed.

Backbones. We show the results with different backbones, including ResNet [2] and the more advanced HRNet [12], ConvNeXt [6], and SwinTransformer [5], in Fig. 1. Overall, HRNet performs especially well in this task; ConvNeXt is slightly above ResNet; the SwinTransformer backbone seems to be unsuitable for this task. Increasing the number of backbone layers offers limited merits. The result with ResNet101 is even worse than ResNet34. Please note that all the results are trained with the same recipe. It could be possible that larger models or different network architectures need different training recipes. More future research about the different network architectures for this task would be valuable.

1D decoder and model head. We decouple the regression-based decoder into the architecture and the layout representations. The results are presented in Fig. 2. The 1D network architecture of HorizonNet [8], HoHoNet [9], and LGTNet [4] are RNN [7], Transformer [10],

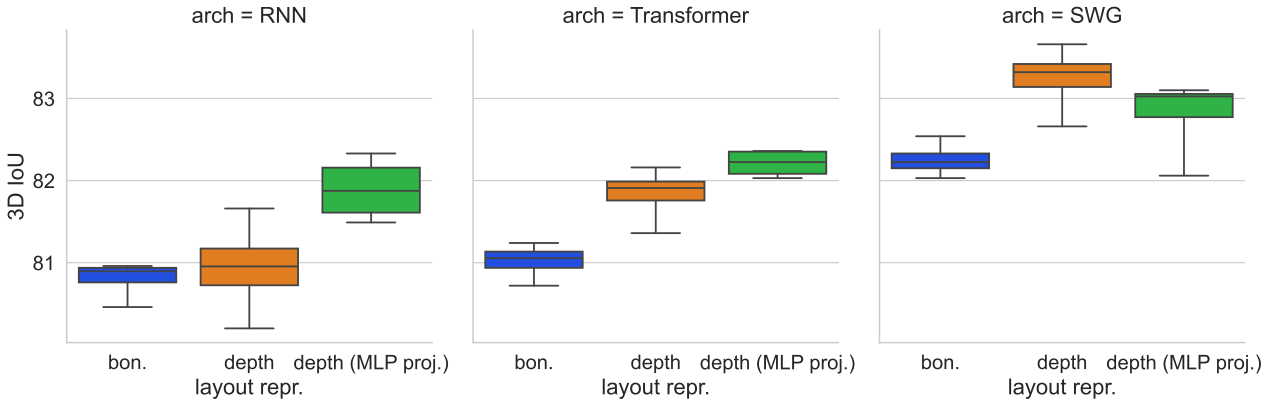


Figure 2. **Results of the regression-based methods.** Originally, RNN is employed by HorizonNet, Transformer is employed by HoHoNet, and SWG is proposed by LGT-Net as the 1D decoder. The layout representation 'bon' indicates predicting per-column layout boundary on the image space while 'depth' is for per-column layout depth. 'MLP proj.' indicates two non-linear layers are added at the very end of the decoder before predicting the layout.

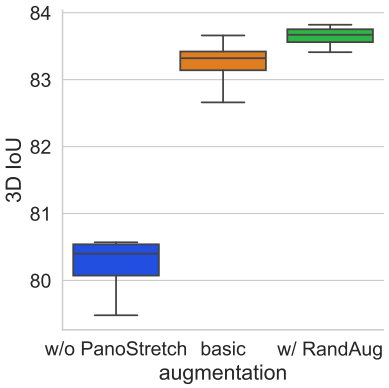


Figure 3. **Results of modified basic data augmentations.** PanoStretch is crucial to recent methods to achieve state-of-the-art results. Employing RandAug can further improve the result.

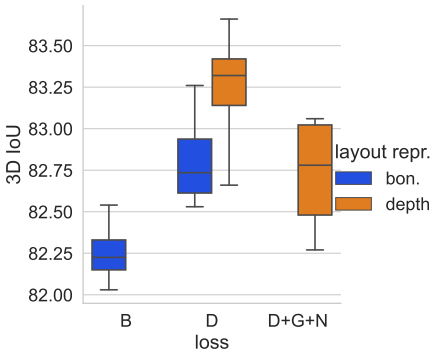


Figure 4. **Results of regression training loss.** Training with layout depth loss improves results even when the model predicts layout boundary on image space. Directly predicting layout depth achieves the best result. Adding gradient and normal losses to regularize layout depth does not improve.

and SWG (an adaptation of Swin-transformer [5]), respectively. HorizonNet proposes to predict per-column layout boundary on image space, while LGT-Net proposes to predict per-column layout depth with a layout height. For the architecture, SWG significantly outperforms Transformer and RNN. Directly predicting layout depth consistently improves the results from all architectures compared to predicting layout boundary on the image space. Originally, the features from all the architecture are just followed by a linear projection layer to predict layout. We also try to add two additional non-linear MLP layers, which improve RNN and Transformer decoder but degrade SWG results.

Interestingly, we find directly training SWG decoder to predict layout boundary fails to converge. We find it is because *i)* the output variance of SWG is high, and *ii)* the layout boundary prediction uses sigmoid to constraint the output range. The combination of these two causes the last layer to 'die' as predicting a large absolute value receives zero gradient from the sigmoid function. As a workaround, we multiply the output value of SWG by 0.1 for the layout boundary prediction. The result suggests that predicting layout boundary may be more unstable compared to predicting layout depth, which may be one of the reason for the superiority of predicting layout depth.

Augmentations. The basic augmentation consists of left-right flip, circular shifting, PanoStretch [8], and luminance jittering. In Fig. 3, we show the results of ablating PanoStretch and replacing luminance jittering with the modified modified RandAug [1]. The results show that PanoStretch is crucial to the quality, highlighting the importance of geometric-based data augmentation. Using RandAug introduces a more diverse image appearance and improves results further.

Losses. We explore existing training losses for regression-based methods. The result is presented in Fig. 4. LED^2-Net [11] proposes to render layout depth from the predicted layout boundary on the image space, which we find indeed can improve boundary prediction quality. We try the gradient and normal losses employed by LGT-Net [4] but do not observe improvement in our reproduction.

3. Layout 3D warping—more details

The proposed LayoutWarp (main paper Eq. (16)) enables us to produce more diverse geometrically augmented views by crafting layout polygon transformation functions $\mathrm{T}_{\mathbf{v}}$ and layout height transformation functions T_h. In the following, we detail how we compute the source image coordinates from the destination image based on the given transformation so that we can apply backward warping to form the transformed image.

Source image coordinate computation. Recap that LayoutWarp takes a source image $\mathcal{I}$ and it's layout polygon $\{\mathbf{v}_i\}_{i=1}^K$, layout height h, and their transformation functions $\mathrm{T}_{\mathbf{v}}, \mathrm{T}_h$ as input to form the warped image $\mathcal{I}'$:

$$\mathcal{I}' = \text{LayoutWarp}\left(\mathcal{I}, \{\mathbf{v}_i\}_{i=1}^K, h, \mathrm{T}_{\mathbf{v}}, \mathrm{T}_h\right) .$$

Let

$$\{\mathbf{v}'_i\}_{i=1}^K = \mathrm{T}_{\mathbf{v}}\left(\{\mathbf{v}_i\}_{i=1}^K\right) , \tag{4a}$$

$$h' = \mathrm{T}_h\left(h\right) , \tag{4b}$$

where $\{\mathbf{v}'\}_{i=1}^K$ and h' are the transformed layout polygon coordinates and layout height. To do backward warping, for each target pixel (i', j') at the destination image, we want to compute its corresponding coordinate (i, j) on the source image following the given layout transformation. We first use Eq. (1) to compute the spherical coordinate (u', v') of the target pixel. By casting a 2D ray following the azimuthal angle u', we can find the ray-polygon intersection point $\hat{\mathbf{v}}$ on the k-th layout polygon edge $\overline{\mathbf{v}'_k\mathbf{v}'_{k+1}}$ (let $\mathbf{v}'_{K+1} = \mathbf{v}'_1$ as the polygon is closed). The depth term in Eq. (2) is

$$d' = \begin{cases} z^{(\text{floor})} \cdot \csc(v'), & \text{if } (i', j') \in \text{floor} \\ \left(z^{(\text{floor})} - h'\right) \cdot \csc(v'), & \text{if } (i', j') \in \text{ceiling} \\ \|\hat{\mathbf{v}}\| \cdot \sec(v'), & \text{if } (i', j') \in \text{wall} , \end{cases} \tag{5}$$

which lifts the pixel to a 3D point (x', y', z'). The corresponding source 3D point is

$$x = ax' + by' , \tag{6a}$$

$$y = cx' + dy' , \tag{6b}$$

$$z = z^{(\text{floor})} - \frac{h}{h'} \cdot \left(z^{(\text{floor})} - z'\right) , \tag{6c}$$

where the backward transformation parameters a, b, c, d align $\overline{\mathbf{v}'_k\mathbf{v}'_{k+1}}$ to $\overline{\mathbf{v}_k\mathbf{v}_{k+1}}$. Let $\mathbf{v}'_k = (x'_k, y'_k)$ and $\mathbf{v}_k = (x_k, y_k)$, we solve the following linear system:

$$\begin{bmatrix} x'_k & y'_k & 0 & 0 \\ 0 & 0 & x'_k & y'_k \\ x'_{k+1} & y'_{k+1} & 0 & 0 \\ 0 & 0 & x'_{k+1} & y'_{k+1} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ x_{k+1} \\ y_{k+1} \end{bmatrix} , \tag{7}$$

where the solution is

$$a = \frac{y'_{k+1} \cdot x_k - y'_k \cdot x_{k+1}}{y'_{k+1} \cdot x'_k - y'_k \cdot x'_{k+1}} , \tag{8a}$$

$$b = \frac{x'_{k+1} \cdot x_k - x'_k \cdot x_{k+1}}{x'_{k+1} \cdot y'_k - x'_k \cdot y'_{k+1}} , \tag{8b}$$

$$c = \frac{y'_{k+1} \cdot y_k - y'_k \cdot y_{k+1}}{y'_{k+1} \cdot x'_k - y'_k \cdot x'_{k+1}} , \tag{8c}$$

$$d = \frac{x'_{k+1} \cdot y_k - x'_k \cdot y_{k+1}}{x'_{k+1} \cdot y'_k - x'_k \cdot y'_{k+1}} . \tag{8d}$$

Note that we assume $\mathbf{v}_k \neq \mathbf{v}_{k+1}$ and $\mathbf{v}'_k \neq \mathbf{v}'_{k+1}$ for all layout polygon edges. Finally, the corresponding source 3D point (x, y, z) from Eq. (6) is projected to the source image using Eq. (3) to interpolate the color.

Effect	$\mathrm{T}_{\mathbf{v}}$	T_h
Left-right circular shifting	$\mathbf{v}'_k = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \mathbf{v}_k$	$h' = h$
† The rotation θ is shared by all the K vertices.		
Left-right flip	$\mathbf{v}'_k = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v}_k$	$h' = h$
PanoStretch [8]	$\mathbf{v}'_k = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \mathbf{v}_k$	$h' = h$
† The scaling factors s_x, s_y are shared by all the K vertices.		
Camera height adjustment	$\mathbf{v}'_k = s\mathbf{v}_k$	$h' = sh$
† The s scales camera height by $\frac{1}{s}$ and is shared by all vertices and height.		
Random perturbation	$\mathbf{v}'_k = s_k\mathbf{v}_k$	$h' = h$
† Each of the K vertex has it's own scaling factor.		

Table 1. **Geometric-based data augmentation by** LayoutWarp**.**

Crafting the transformation functions. We can craft $\mathrm{T}_{\mathbf{v}}, \mathrm{T}_h$ to produce various geometric augmentations. We summarize the realization of the commonly-used left-right circular shifting, left-right flip, PanoStretch [8], and the new camera height adjustment and random layout polygon perturbation in Tab. 1. We visualize more augmented views in Fig. 5.

More qualitative comparison of random perturbation. In the main paper, we show random perturbation can improve the results when generalizing from a Manhattan-aligned dataset to a general layout dataset. We show more

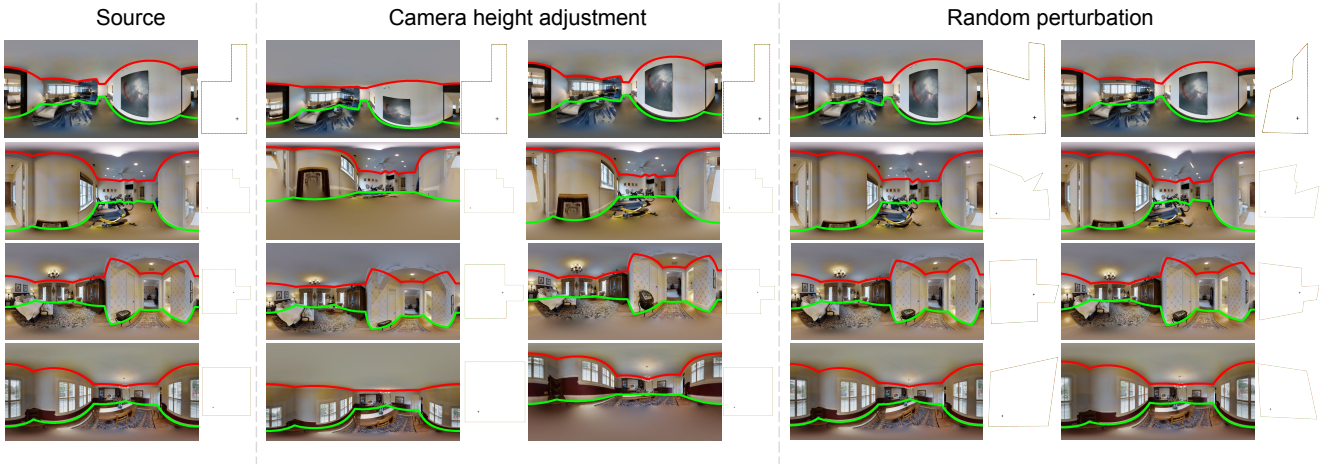
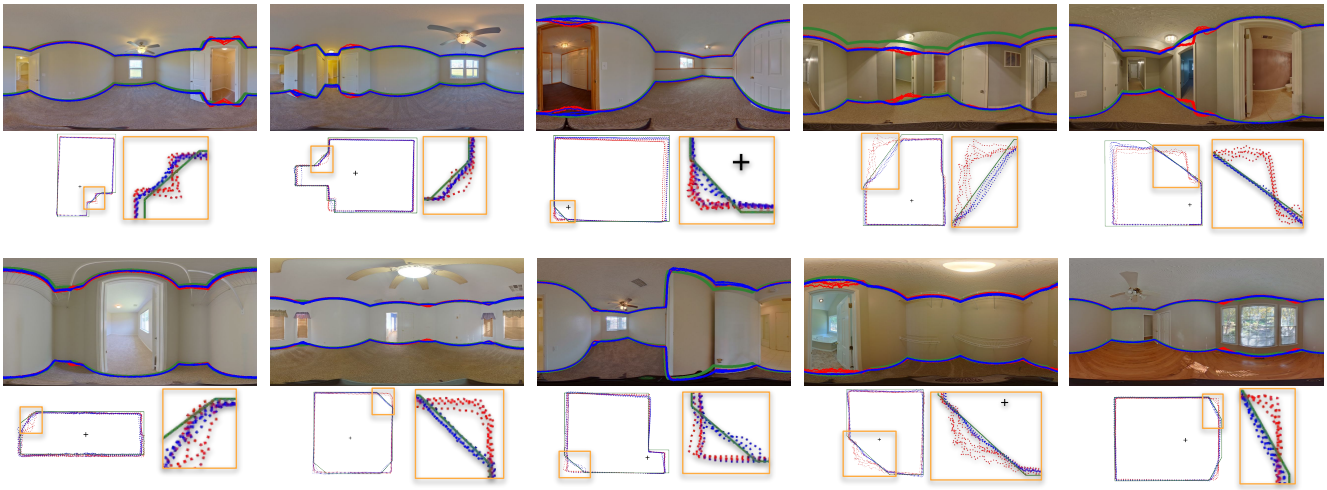


Figure 5. **More visualization of the new data augmentations by** LayoutWarp



† The result variations are due to the four different training seeds.

Figure 6. **Training with random perturbation generalizes better from Manhattan to general layout.** We can observe that the models trained with random perturbation are less 'panics' when the input is not Manhattan-aligned.

visual comparison in Fig. 6.

4. Seg2Reg—more details

We provide more technical details and analyses of the proposed Seg2Reg in this section.

Training details. Our Seg2Reg enables differentiable layout depth (*i.e.*, distance to layout wall on the floor plan) rendering from the 2D density map prediction. The 'flattened' volume rendering is detailed in the main paper Sec. 3.1. Directly supervising the rendering by ground-truth layout depth leads to ambiguity. Recap that the depth rendering equation of a ray is (main paper Eq.(3)):

$$d = \sum_{i=1}^{K} T_i \alpha_i t_i \text{ , where } T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \ ,$$

where K is the number of sampled points on the ray. We can see that there are an infinite number of weight distributions on the ray that can render to a specific ground-truth depth d^*. To resolve ambiguity, we directly derive a compact weight distribution w^* for the ground-truth depth d^*. Let t_k, t_{k+1} be the two depth values nearest to d^* on the

ray. Our ground truth weight distribution is:

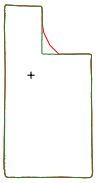
$$w_k = \frac{t_{k+1} - d^*}{t_{k+1} - t_k} \,, \tag{9a}$$

$$w_{k+1} = 1 - w_k \,, \tag{9b}$$

$$w_i = 0 \,, \text{where } i = 1, \cdots, k-1, k+2, \cdots, K+1 \,. \tag{9c}$$

There is a special case when the far clipping distance of the ray does not reach the layout $t_K < d^*$. In this special case, the weights are all zero except $W_{K+1} = 1$.

To render the primary layout, we directly use the $\frac{H}{2}$ points in the image column as the sampled points. For the secondary layouts, we sample $N^{(\text{secondary})} = 32$ secondary camera centers. To render layout depth for the secondary cameras, we uniformly sampled 1,024 points on a ray with the farthest distance $t_K = 16$ (the camera-to-floor distance is $z^{(\text{floor})} = 1.6$). The loss weights for $\mathcal{L}^{(\text{pri.})}$, $\mathcal{L}^{(\text{2nd})}$, and $\mathcal{L}^{(\text{seg.})}$ are $w_1 = 1.0, w_2 = 0.1, w_3 = 1.0$, respectively.



Robust polygons merging algorithm. To merge the primary layout polygon and secondary layout polygons, the simplest way is to directly take the polygon union. Such a naive strategy may have polygon edges crossing the region outside the layout where the model actually predicts high floor plan density (the red polygon in the figure), especially when the rendered secondary polygons are in lower resolution (fewer number of vertices). This prompts us to design a more robust algorithm. Our idea is to connect all the rendered vertices with the minimum perimeter. We first compute the pair-wise distance matrix of all the rendered vertices from primary and secondary views. The pair-wise distance matrix represents a complete graph. We then construct a minimum spanning tree from the complete graph via the Kruskal algorithm. The final merged polygon (the green polygon in the figure) is determined by the trajectory connecting the farthest two points on the tree, which can be realized by running the depth-first tree search two times.

We show the results of the two polygon merging methods and the result after polygon simplification in Tab. 2.

Union	MST	MST + simplification
87.22%	87.33%	87.13%

Table 2. **Results of different polygon merging algorithms.** We report the 2DIoU↑ comparison on MatterportLayout valid set. Polygon merging via minimum spanning tree (MST) achieves slightly better results. Applying polygon simplification causes the number result to drop slightly.

Raw prediction visualizations. In the main paper's Table.3, we show that our Seg2Reg achieves better quantitative results than a purely segmentation-based method. In Fig. 7, we visualize the raw predictions activated by Sigmoid. Segmentation-based method only supervises per-pixel classification. The proposed Seg2Reg introduces rendering loss, which emphasizes the rendered polygon position, while the density far outside the layout polygon is less important as they do not affect the rendering results.

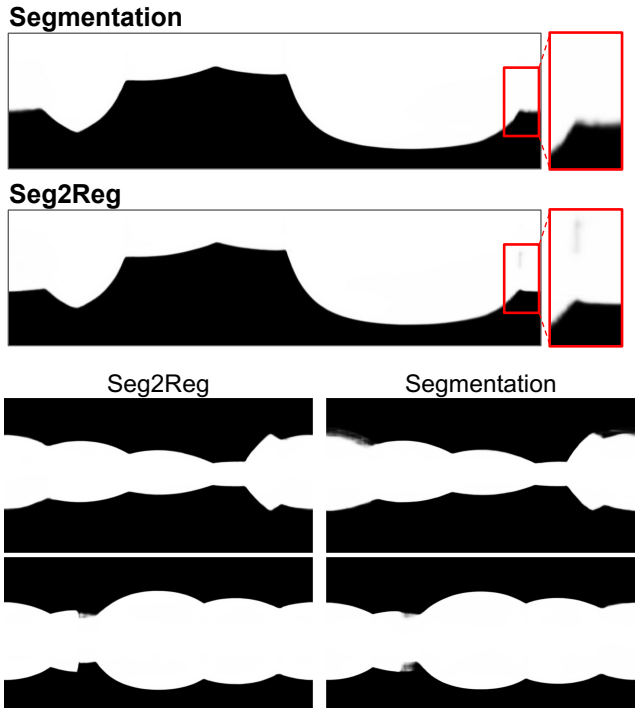


Figure 7. **Visualization of the raw predictions.** Segmentation-only method and our Seg2Reg converge to different results.

5. Qualitative comparison

We provide an qualitative comparisons in Figs. 8 and 9. We run LGT-Net's official repo to show the performance of the existing state-of-the-art.

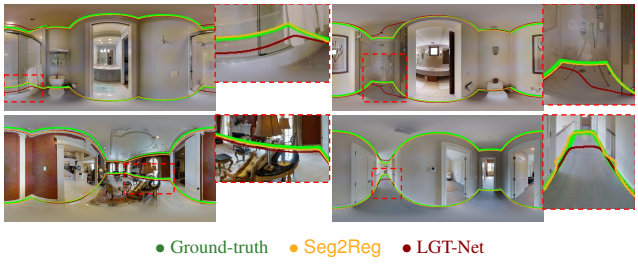


Figure 8. **Qualitative comparisons on the unseen data from MatterportLayout.**

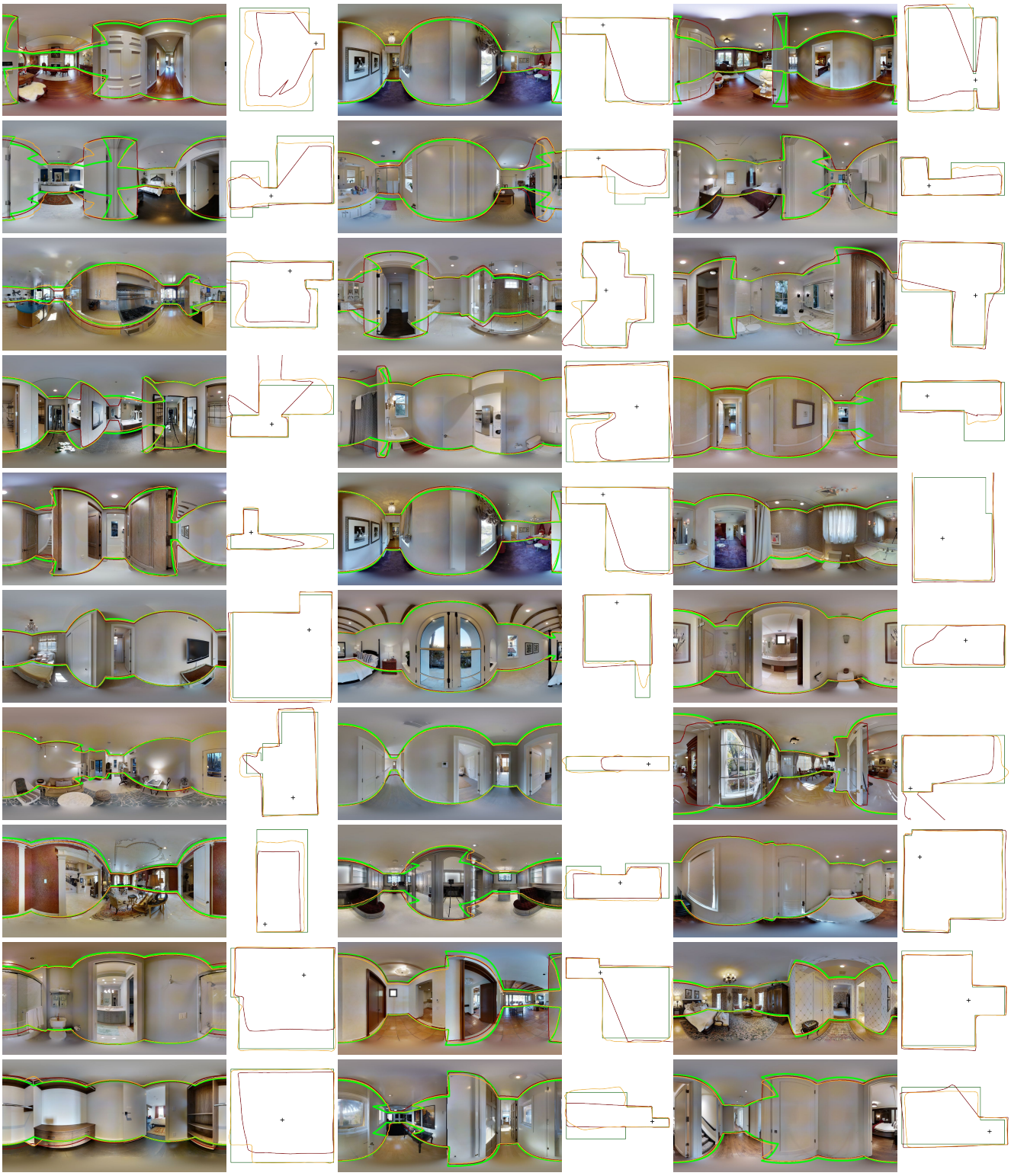


Figure 9. **Qualitative comparisons on the unseen data from MatterportLayout.** We compare it with the state-of-the-art model from LGT-Net's official repo. The test set 2DIoU is 83.52% for the official LGT-Net and 85.14% for our model.

References

[1] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 3008–3017. Computer Vision Foundation / IEEE, 2020. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 1

[3] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press, 2018. 1

[4] Zhigang Jiang, Zhongzheng Xiang, Jinhua Xu, and Ming Zhao. Lgt-net: Indoor panoramic room layout estimation with geometry-aware transformer network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 1644–1653. IEEE, 2022. 1, 3

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021. 1, 2

[6] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11966–11976. IEEE, 2022. 1

[7] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. In *IEEE Transactions on Signal Processing*, 1997. 1

[8] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1047–1056. Computer Vision Foundation / IEEE, 2019. 1, 2, 3

[9] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Hohonet: 360 indoor holistic understanding with latent horizontal features. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 2573–2582. Computer Vision Foundation / IEEE, 2021. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 1

[11] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. Led2-net: Monocular 360deg layout estimation via differentiable depth rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 12956–12965. Computer Vision Foundation / IEEE, 2021. 1, 3

[12] 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.*, 43(10):3349–3364, 2021. 1

[13] Chuhang Zou, Jheng-Wei Su, Chi-Han Peng, Alex Colburn, Qi Shan, Peter Wonka, Hung-Kuo Chu, and Derek Hoiem. Manhattan room layout reconstruction from a single 360° image: A comparative study of state-of-the-art methods. *Int. J. Comput. Vis.*, 129(5):1410–1431, 2021. 1