[Supplementary Material] Extreme Floorplan Reconstruction by Structure-Hallucinating Transformer Cascades

Sepidehsadat Hosseini
Simon Fraser University
sepidh@sfu.ca

Yasutaka Furukawa
Simon Fraser University
furukawa@sfu.ca

The supplementary document provides the full specifications of our neural architecture in Section 1, more details on dataset and refinement procedure 2, and 3, results on different IOUs provided in 5 and more experimental results in Figs. 1, 2, 3, and 4. Figures 1 and 2 show more qualitative comparisons in the same format as Figure 4 in the main paper. Figures 3 and 4 show reconstructed floorplans before and after the final refinement in the same format as Figure 5 in the main paper.

1. Architecture Specifications

Category Wise CNN takes an input image of the resolution $800 \times 800 \times 14$. For the house/apartment with X number of visible rooms, We will have X number of C-wise CNN for rooms category were the output will be $X \times 256 \times 25 \times 25$, then we apply max function which result output of that to be $256 \times 25 \times 25$. Same will be applied to door and room (both) category. For doors only category there is no need for max, as we give all doors at same time to one C-wise CNN block. Then we concatenate them resulting output of all to be 3×256, then we reshape them to get input sequence for encoder to be 1875×256 let's call it *src*.

Encoder has 6 layers (See Table 2), it receives *src*, position, and type embedding as input. As positional embedding shape is 625×256, we concatenate it by itself three times to produce 1875×256 positional embedding. Type embedding shape is 3×256, we concatenate each embedding related to each category by itself 625 times and concatenate all them together to produce 1875×256 type embedding. Encoder input is summation of *src* and those two embeddings.

Final output of encoder will be 1875×256. We only use the ones for room-door (both) category as decoder input which will be 625×256 lets call this *memory (Mem)*.

Decoder has three cascaded blocks and 6 layers in each block. Table 3 shows one layer in one block. Let us assume that we have Y dangling doors. then number of queries for first decoder will be Y, second will be Y+15 and third will be Y+30. Lets show each cascade input queries as *tgt*. The first cascade input is Y dummy queries and encoder output (*Mem*). Then first cascade output will be concatenate with 15 dummy queries the result is input queries for second decoder cascade, second cascade also receives *Mem* as input. For last cascade, second cascade output will be concatenate with 15 dummy queries and same as other two cascades, third cascade also receives *Mem* too.

2. Datasets Detail

In this section, we first give more detail on our proposed dataset, then provide some details on the RPLAN dataset. In our proposed dataset, the number of panoramas per house ranges from 1 to 7. Specifically, (31, 129, 222, 199, 118, 2) houses contain (1, 2, 3, 4, 5, 7) panoramas, respectively. The test set contains (9, 19, 15, 7) houses containing (2, 3, 4, 5) panorama images, respectively. The number of invisible rooms varies between 1 and 17. The number of invisible doors varies between 0 and 11.

As it has been stated in paper in order to solve class imbalance problem we use a weighing constant ($w(i)$) that is inversely proportional to the number of samples in the training set, which is (1.135, 3.7. 0.6. 0.63, 0.47, 1.2, 2.2, 0.76, 0.48, 0.18, 0.763, 0.524, 0.33, 0.7, and 0.1) for (living room, kitchen, bedroom, toilet, balcony, corridor, tatami , washroom. bathroom, closet, closet door, open door, door, entrance, and no-room), respectively.

In the RPLAN case, we use the Housegan++ [2] data parser, which results in 60K houses; the number of rooms per house varies between 5 to 8. We divide the dataset into 55K for training and 5K for testing. Using our dataset's statistics, we flag the number of rooms per house as invisible. The number of visible rooms per house varies between 2 to 7, the number of invisible rooms varies between 1 to 6, and the number of invisible doors varies between 0 to 5.

Table 1. Category Wise CNN (C-Wise CNN) block

Layer name/type	Specification	Output size
Conv1	7×7, 64, stride 2	$64 \times 400 \times 400$
Conv block2	3×3, maxpool, stride 2 $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$256 \times 200 \times 200$
Conv block3	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$512 \times 100 \times 100$
Conv block4	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$1024 \times 50 \times 50$
Conv block5	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$2048 \times 25 \times 25$
Conv6	1×1, 256	$256 \times 25 \times 25$

Table 2. One layer of encoder, we have 6 layers

Layer name/type	input	Specification	Output size
Multi head attention	src	embed_dim=256, num_heads=8	1875×256 (src2)
Dropout	src2	dropout=0.1	1875×256 (src3)
LayerNorm	src3+src	normalized_shape=256	1875×256 (src4)
Linear1, ReLU	src4	in_features=256, out_features=2048	1875×2048 (src5)
Dropout	src5	dropout=0.1	1875×2048 (src6)
Linear2	src6	in_features=2048, out_features=256	1875×256(src7)
Dropout	src7	dropout=0.1	1875×256 (src8)
LayerNorm	src8+src4	normalized_shape=256	1875×256 (src9)

Table 3. One layer of decoder, we have 6 layers, Y is number of input queries, in first cascade it is equal to number of dangling doors in second it is number of dangling doors+15, and third number of dangling doors+30

Layer name/type	input	Specification	Output size
Multi head attention	tgt	embed_dim=256, num_heads=8	$Y \times 256$ (tgt2)
Dropout	tgt2	dropout=0.1	$Y \times 256$ (tgt3)
LayerNorm	tgt3+tgt	normalized_shape=256	$Y \times 256$ (tgt4)
Multi head attention	Mem, tgt	embed_dim=256, num_heads=8	$Y \times 256$ (tgt5)
Dropout	tgt6	dropout=0.1	$Y \times 256$ (tgt7)
LayerNorm	tgt7+tgt4	normalized_shape=256	$Y \times 256$ (tgt8)
Linear1, ReLU	tgt8	in_features=256, out_features=2048	$Y \times 2048$ (tgt9)
Dropout	tgt9	dropout=0.1	$Y \times 2048$ (tgt10)
Linear2	tgt10	in_features=2048, out_features=256	$Y \times 256$(tgt11)
Dropout	tgt11	dropout=0.1	$Y \times 256$ (tgt12)
LayerNorm	tgt8+tgt12	normalized_shape=256	$Y \times 256$ (tgt13)

3. Refinement Detail

Here we give a few more details on our refinement process. RPLAN dataset [4] was initially used to train House-GAN++ but is synthetic and may exhibit incompatible database bias. We use LIFULL HOME'S dataset [1] instead. Since the goal is the refinement without overall arrangement changes, we use fully-connected graphs for training and testing. The reconstructed floorplan from our cascading decoder is specified as the input constraint. In the first iteration, we give our last cascade output as inputs

Figure 1. Main qualitative evaluations. The figure shows reconstruction results before the final refinement for four houses. For each method, we show two results, when the input partial reconstruction is from the ground-truth or the extreme SfM system [3]. At the right, we show both our ground-truth vector-graphics floorplans as well as the original raster floorplan images by an architect, samples are same houses in Figure 5 in main paper. Sometimes SfM room types and GT rooms types are different which are result of error in room type classification in SfM

Figure 2. Main qualitative evaluations. The figure shows reconstruction results for more samples before the final refinement for four houses. For each method, we show two results, when the input partial reconstruction is from the ground-truth or the extreme SfM system [3]. At the right, we show both our ground-truth vector-graphics floorplans as well as the original raster floorplan images by an architect, samples are same houses in Figure 5 in main paper

Figure 3. Reconstructed floorplans before and after the refinement by House-GAN++ and several heuristics for more samples. The same refinement process is used for all the methods. The input partial reconstructions are ground-truth in this figure

Figure 4. Reconstructed floorplans before and after the refinement by House-GAN++ and several heuristics for more samples. The same refinement process is used for all the methods. The input partial reconstructions are derived using SfM system [3] in this figure

mask to Housegan++. The predicted mask of the previous iteration will be input for each node from the second iteration, and we continue this to the 10th iteration. By doing this, as we are limiting Housegan++, masks only get refined and curvy edges will change to manhattan shapes.

4. Result on different Threshold

In Figure 5 we are providing recall and precision graph for different IOUs starting from 0 going up to 1, increasing by 0.05. Both the precision and the recall decrease as we increase the IOU threshold in Figure 5. As our plots are based on the IOU threshold on the metric computation, by decreasing the IOU threshold number of True positive increases in both precision and recall which cause the increase in both of them, in all IOU thresholds our method has the highest performance.

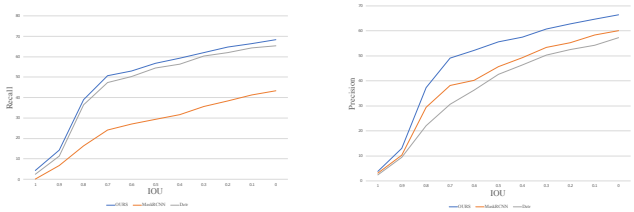

Figure 5. Both the precision and the recall decrease as we increase the IOU threshold in Fig5, which is not what a normal precision/recall curve shows, for example in a detection task. This is because, our plots are based on the IOU threshold on the metric computation, as opposed to a confidence threshold of a detection in a normal precision/recall curve.

References

[1] Liful dataset. https://www.nii.ac.jp/dsc/idr/lifull. 2

[2] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. Housegan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13632–13641, 2021. 1

[3] Mohammad Amin Shabani, Weilian Song, Makoto Odamaki, Hirochika Fujiki, and Yasutaka Furukawa. Extreme structure from motion for indoor panoramas without visual overlaps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5703–5711, 2021. 3, 4, 6

[4] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6), 2019. 2