MonteFloor: Extending MCTS for Reconstructing Accurate Large-Scale Floor Plans – Supplementary Material

Sinisa Stekovic[1], Mahdi Rad[1], Friedrich Fraundorfer[1], Vincent Lepetit[2,1]
[1]Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria
[2]Université Paris-Est, École des Ponts ParisTech, Paris, France
{sinisa.stekovic, rad, fraundorfer}@icg.tugraz.at, vincent.lepetit@enpc.fr

This document provides additional information about our MonteFloor method. We also provide a video demonstrating it in action.

1. MCTS Implementation

In this section, we provide more details about our implementation of Monte Carlo Tree Search (MCTS) with our refinement step.

Tree nodes. As explained in the main paper, the nodes contain several attributes. Every node is associated with some proposal P_i, but the same proposal P_i is associated with one or more nodes along different sub-paths of the tree. Every node N also keeps track of its "node score" $Q(N)$ and number of visits $n(N)$ that will be used later to guide the search towards the most promising directions. A node can have multiple children nodes children(N) and has a single parent node.

Initialization. Initially, the search tree contains only a root node and is built online by expanding the tree in the most promising direction at every iteration.

During the search, we follow the standard Select-Expand-Simulate-Update strategy:

Selection step. At each iteration, and starting from the root node, we traverse existing nodes at each level of the tree by following the Upper Confidence Bound (UCB) criterion: From current node N_{curr}, we select its child node $N \in \text{children}(N_{\text{curr}})$ that maximizes the criterion

$$\arg \max_{N \in \text{children}(N_{\text{curr}})} Q(N) + \lambda_{UCB} \cdot \sqrt{\frac{\log n(N_{curr})}{n(N)}}, \quad (1)$$

where λ_{UCB} is a hyperparameter that balances exploitation and the exploration during the search.

Expansion step. If during selection, there is a node with a child node $N \in \text{children}(N_{\text{curr}})$ that has not been created yet, we do not follow the selection criterion, but instead, we add N to the tree. We set $Q(N) = -\infty$ and $n(N) = 0$ at this stage.

Simulation step. Immediately after the expansion step, we perform a simulation step and randomly create and traverse the children nodes until we reach a leaf node of the tree.

Refinement step. Whenever a leaf node is reached during simulation, we run 1 iteration of the Adam optimizer [2] on the selected proposals P to minimize $\mathcal{L}(P)$ (Eq. (1) of the main paper) and calculate the score $S = -\mathcal{L}(P)$: Running more iterations decreases the number of necessary MCTS iterations, but still results in longer run-times. Instead, we allow the tree search to select already visited solutions. This results in faster run-time and accurate solutions, as we will run multiple refinement steps on the most promising solutions, but few on the less likely ones.

Update step. Once we know the score S for a set of proposals P after the refinement step, for every traversed node N, we update the node score $Q(N) \leftarrow \max(Q(N), S)$ and increment $n(N)$.

During experiments, for each scene we set $\lambda_{UCB} = 1$ initially, and linearly decay it such that the last MCTS iteration uses $\lambda_{UCB} = 0.01$.

2. Generating Floor Plans for Training the Metric Network

The metric network is trained on the Structured3D dataset [3] with input density maps and floor plans of size 256×256. During training, we generate input floor plans directly from the ground truth annotations to simulate a large variety of possible settings. More exactly, with probability of 30%, we select the ground truth floor plan as input. Otherwise, each individual room is added with probability of 50%. A single room is randomly rotated with 50%

chance by either $90°, 180°$, or $270°$, and randomly translated by $[-50, 50]$ pixels with probability of 10%. With 1% probability, a vertex inside a room polygon is translated in range $[-10, 10]$ pixels. Labels for individual rooms are shuffled to make sure the network does not overfit to some specific label ordering. Similarly to input floor plans, we augment the dataset by rotating and translating the input density map and the corresponding ground truth floor plan. We match the generated room shapes with the ground truth shapes, as in Section 4.1 of the main paper, and calculate the Intersection-Over-Union (IOU) between the matches to obtain the final ground truth score.

The network is then trained by minimizing the Root-Mean-Square-Error (RMSE) between predicted and ground truth scores. We use the Adam optimizer [2] with learning rate set to 10^{-3}.

3. Choice of Hyper-Parameters for the Objective Function

During refinement step, we set $\lambda_{\text{ang}} = 0.01, \lambda_0 = 0.01, \lambda_{\text{glob}} = 0.2, \lambda_f = 0.1$ and normalize $p(\alpha)$ to $[0, 1]$. We have found empirically that this set of hyper-parameters balances the influence of the corresponding loss terms. We have found that setting $\lambda_{\text{glob}} = 0$ and $\lambda_f = 1$ in the score calculation step increases convergence speed.

For the test set from [1], our metric network is less stable as it was not trained on this dataset. Hence, we have observed that setting $\lambda_{\text{ang}} = 0.1, \lambda_0 = 0.1, \lambda_{\text{glob}} = 0.2, \lambda_f = 0.05$, during refinement step, improves performance. During the score calculation step, we still set $\lambda_{\text{glob}} = 0$ and $\lambda_f = 1$.

4. Multiple Optimizers

As the number of room proposals increases, optimizer has to deal with much larger number of learning parameters. In turn, this leads to larger computation times during optimization step. To overcome this issue, we employ multiple optimizers: For each leaf node in the tree, we create an optimizer that optimizes the set of proposals along corresponding path in the tree. Such approach improves computation times considerably.

5. Qualitative Results

Figures 1 and 2 show additional qualitative results on the Structured3D dataset 1 and the test set from [1]. As shown in Figure 3, from the floor plans reconstructed by our MonteFloor method, we can directly reconstruct 3D floor plans simply by assuming constant wall height. Our 3D floor plans are consistent with underlying 3D scene.

6. Limitations and Future Work

We demonstrate some failure cases in Figure 4. Even though our method achieves state-of-the-art results on existing datasets we do believe there is still a large potential for improvements:

- The number of vertices of polygonal proposals is determined strictly by the polygonized outputs of the masks from Mask R-CNN. It would be interesting to consider possibility of dynamically adding, or removing, vertices of polygonal proposals. Similarly, it would be interesting to dynamically generate new rooms in places where large discrepancies are detected.
- It is clear from initialization that some proposals are unlikely to be part of the final layout. Using prior information about the quality of proposals to initialize node scores in the tree could make our approach run even faster.
- Density maps are a useful representation, but in practice, they can still be ambiguous in some cases. Hence, considering additional information such as perspective views, and camera trajectory, could be very helpful to deal with such ambiguities.

References

[1] Jiacheng Chen, Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floor-SP: Inverse CAD for Floorplans by Sequential Room-Wise Shortest Path. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2, 4

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

[3] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling. In *European Conference on Computer Vision*, 2020. 1, 3

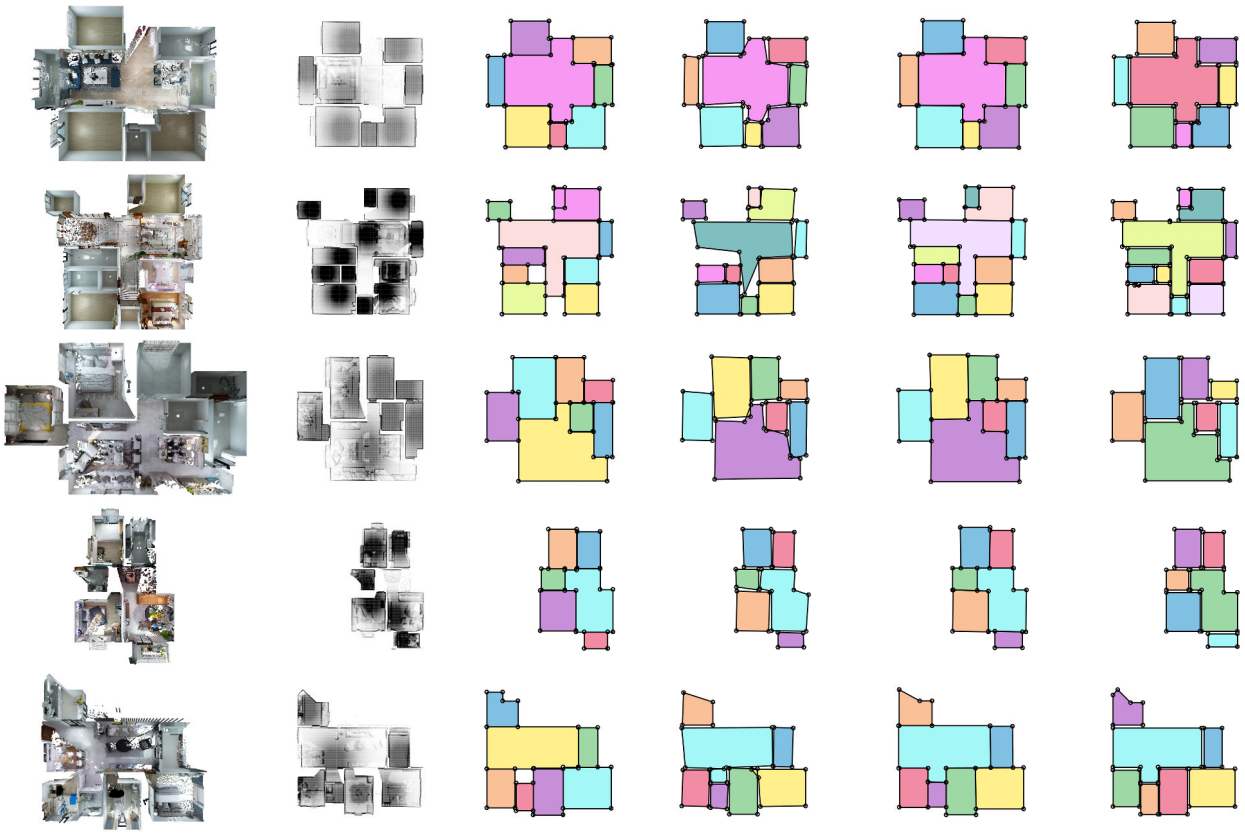

Figure 1: Qualitative results on Structured3D [3]. In the first example, both Floor-SP and our MonteFloor retrieve good floor plans. In the second example, Floor-SP misses some rooms and retrieves self-intersecting floor plans. In the third example, the light blue room reconstructed by Floor-SP that is slightly more complex than reality. In the fourth example, both methods perform well. In the fifth example, Floor-SP oversimplifies the yellow and dark blue rooms.

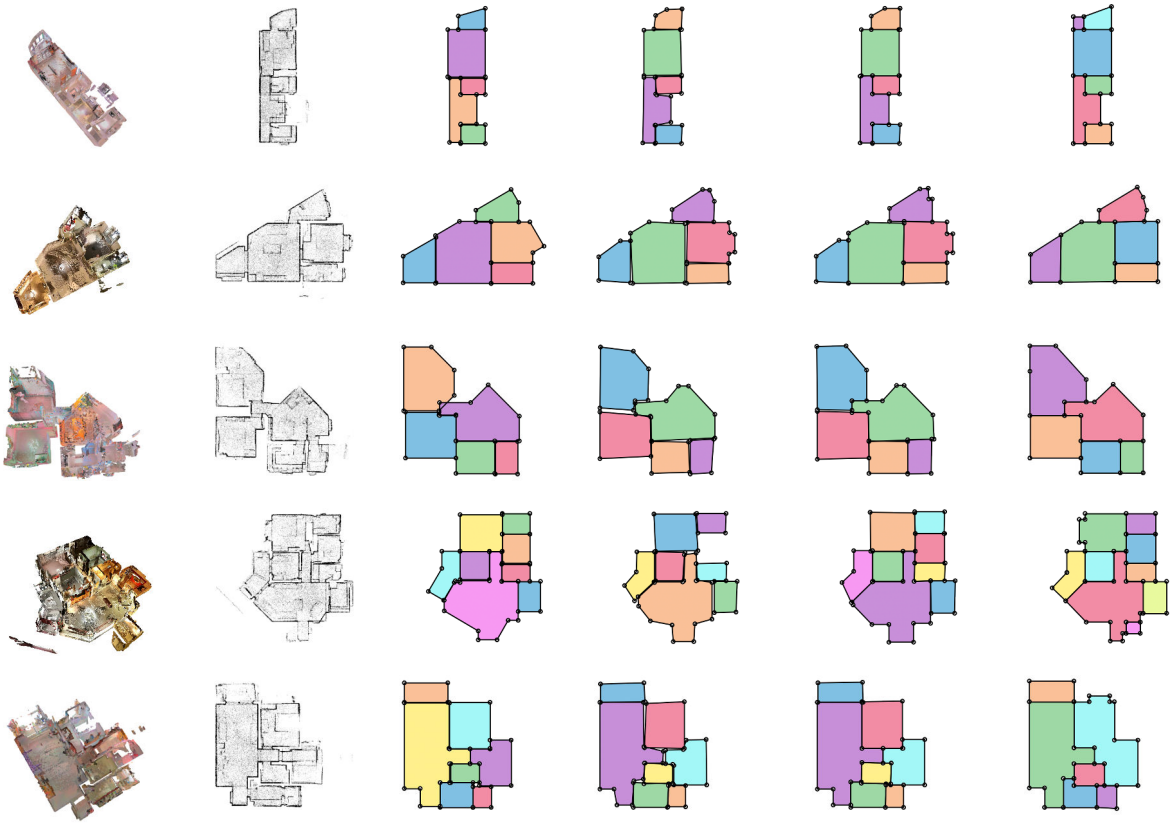

Figure 2: Qualitative results on the test set from [1]. In the first example, we believe that the floor plan retrieved by our method is better than the manual annotation as the purple room appears to be an annotation error and the light blue room is oversimplified. Similarly, in the second example, our reconstructions are much more consistent with the actual input. In the third example, Floor-SP produces self-intersecting rooms but our reconstructed green room is slightly more complex than its corresponding ground truth. In the fourth example, the little pink room on the bottom right of the ground truth is actually an annotation error. In the fifth example, the light blue room in the ground truth are actually two rooms, as estimated by both Floor-SP and our method. In addition, Floor-SP produces a small squared hole between the green, blue, red, and purple rooms.

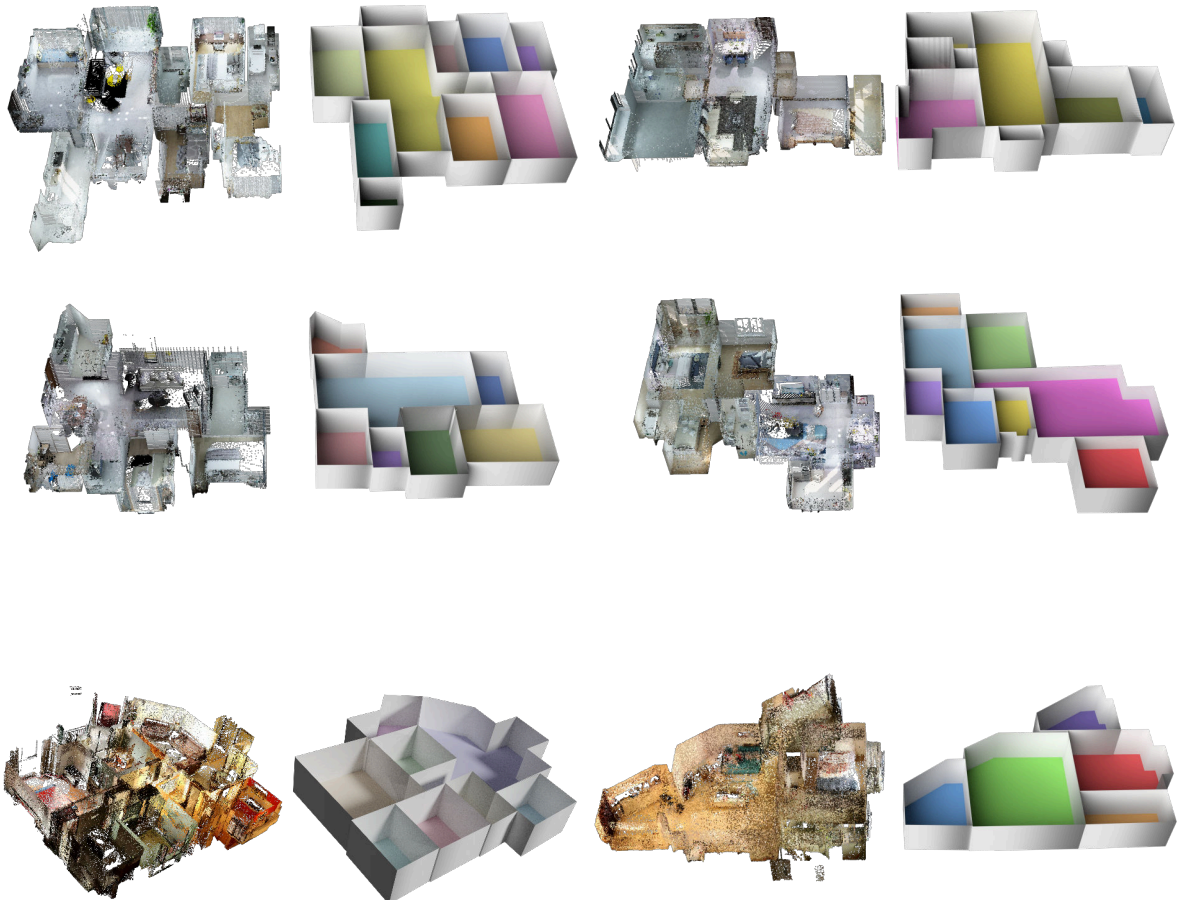

Figure 3: We can generate attractive 3D floor plans from our 2D reconstructions under a constant room height assumption. The examples demonstrate that the floor plans reconstructed by our MonteFloor method are indeed consistent with corresponding 3D scenes.

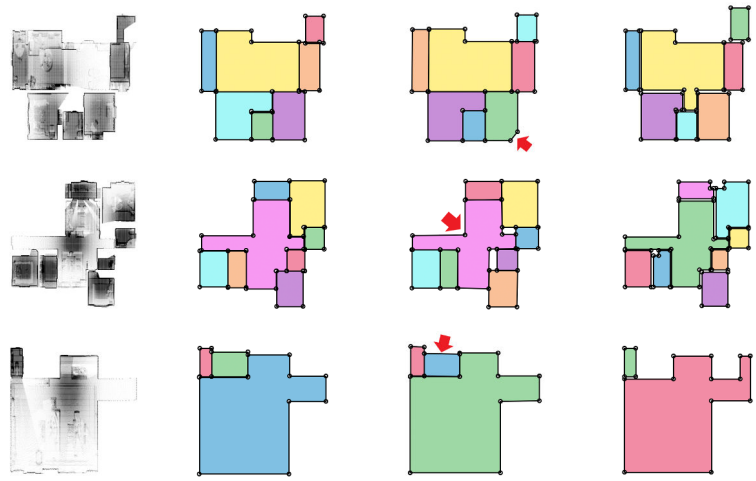

Figure 4: Failure Cases. First row, red arrow: Our reconstruction of the green room is incorrect as there are some ambiguities in the density map for this region. In contrast, Floor-SP is more likely to produce Manhattan layouts in such situations. Second row, red arrow: None of the generated polygonal proposals for the pink room are adequate to represent the actual shape. Floor-SP performs graph-based search on pixel-locations that lie on the polygonal curve. As the final polygon is generated after the search, they obtain better reconstruction in this example. Third row, red arrow: Similarly to Floor-SP, we were not able to remove false positive proposal. This implies that the metric network could be further improved.