A. Supplementary Materials

A.1. Pseudocode for Line Vectorization

Algorithm 1 gives a more detailed description for line vectorization. The algorithm takes the C-junction set V_C and T-junction set V_T as the input and outputs a vectorized wireframe (V, E). In the first stage (Lines 2-3), we find the lines among C-junctions according to the line confidence function. The procedure PRUNE-LINES greedily removes the lines with the lowest confidence that either intersect with other lines (Line 32) or are too close to other lines in term of the polar angle (Line 35). In the second stage (Lines 4-25), we add the T-junctions into the wireframe. From Lines 6-14, we find the T-junctions that are on the existing wireframe. We first adjust the positions of those T-junctions by projecting them onto the line (Line 9) and then add them to the candidate T-junction set V' (Line 10). Because the degree of a T-junction is always one, we try to find the connection with the highest confidence for those candidates T-junctions (Lines 15-25). We repeat the this process until V, V', and E remain the same in the last iteration.

A.2. Line Assignments for Vanishing Points

In Equation (4), we need to find the set of lines $\mathsf{A}_i \subseteq \mathsf{E}$ corresponding to the vanishing point i. Mathematically, we define the objective function

$$\min_{\mathsf{A}} \sum_{i}^{3} \sum_{(\boldsymbol{u},\boldsymbol{v})\in\mathsf{A}_i} \|(\boldsymbol{u} - \boldsymbol{V}_i) \times (\boldsymbol{u} - \boldsymbol{v})\|_2 ,$$

where $\|(\cdot) \times (\cdot)\|_2$ can be understood as the parallelogram area formed by two vectors. Since each line in this equation is mutually independent, we can solve this optimization problem by greedily assigning each line to the best vanishing point i to minimize the objective function.

A.3. Sampled Failure Cases and Discussions

Figure 7 demonstrates some failure cases in our SceneCity dataset. We found that our pipeline might not work well on the scenes in which there are many lines and junctions that are close to each other. This is because the resolution of the output heat map is 128×128, so any detail whose size is below two or three pixels might get lost during the vectorization stage. Therefore, one of our future work is to explore the possibility of using high-resolution input and output images. There are also issues in the 3D depth refinement stage. When the scene is complex, finding the assignment A_i for each line can be hard, due to the error in the junction position and line direction. In addition, the term contributed by erroneous lines in Equation (4) can make the depth of some junctions inaccurate. Such problem might potentially be alleviated by increasing the resolution of the input and output images, using a more data-driven

Algorithm 1 Edge Vectorization Algorithm

Require: Candidate C-junction set V_C, T-junction set V_T.
Require: Hyper-parameters η_c and $\eta_\circ$.
Ensure: Wireframe (V, E).

1: **procedure** VECTORIZE(V_C, V_T)
2: $\mathsf{V} \leftarrow \mathsf{V}_C$
3: $\mathsf{E} \leftarrow$ PRUNE-LINES($\{(\boldsymbol{u},\boldsymbol{v}) | \boldsymbol{u},\boldsymbol{v} \in \mathsf{V}, c(\boldsymbol{u},\boldsymbol{v}) > \eta_c\}$)
4: $\mathsf{V}' \leftarrow \emptyset$
5: **while** V, V', or E change in the last iteration **do**
6: **for all** $\boldsymbol{w} \in \mathsf{V}_T$ **do**
7: **for all** $e = (\boldsymbol{u},\boldsymbol{v}) \in \mathsf{E}$ **do**
8: **if** $\boldsymbol{w}$ is near e **then**
9: project $\boldsymbol{w}$ to the line e
10: $\mathsf{V}' \leftarrow \mathsf{V}' \cup \{\boldsymbol{w}\}$
11: **break**
12: **end if**
13: **end for**
14: **end for**
15: **for all** $\boldsymbol{u} \in \mathsf{V}'$ **do**
16: $\boldsymbol{v} \leftarrow \text{argmax}_{\boldsymbol{v}\in\mathsf{V}\cup\mathsf{V}'} c(\boldsymbol{u},\boldsymbol{v})$
17: **if** $c(\boldsymbol{u},\boldsymbol{v}) \geq \eta_c$ **then**
18: $\mathsf{V}' \leftarrow \mathsf{V}' \backslash \{\boldsymbol{u}\}$
19: $\mathsf{V} \leftarrow \mathsf{V} \cup \{\boldsymbol{u}\}$
20: $\mathsf{E} \leftarrow \mathsf{E} \cup \{(\boldsymbol{u},\boldsymbol{v})\}$
21: **end if**
22: **end for**
23: $\mathsf{V}_T \leftarrow \mathsf{V}_T \backslash (\mathsf{V} \cup \mathsf{V}')$
24: **end while**
25: $\mathsf{E} \leftarrow$ PRUNE-LINES(E)
26: **return** (V, E)
27: **end procedure**
28: **procedure** PRUNE-LINES(E)
29: sort E w.r.t confidence values in descending order
30: $\mathsf{E}' \leftarrow \emptyset$
31: **for all** $e \in \mathsf{E}$ **do**
32: **if** $\exists e' \in \mathsf{E}'$: e intersects with e' **then**
33: **continue**
34: **end if**
35: **if** $\exists e' \in \mathsf{E}'$: $e' \cap e \neq \emptyset$ **and** $\angle(e, e') < \eta_\circ$ **then**
36: **continue**
37: **end if**
38: $\mathsf{E}' \leftarrow \mathsf{E}' \cup \{e\}$
39: **end for**
40: **return** E'
41: **end procedure**

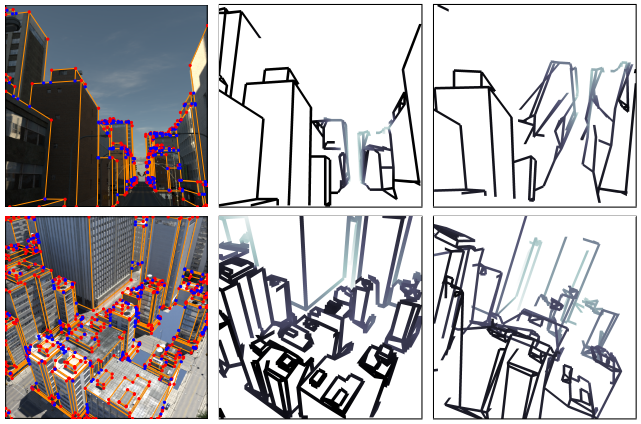

Figure 7: Failure cases on the SceneCity dataset.

method, designing a better objective function, or employing a RANSIC approach in those two stages.

A.4. Network Comparison

In this section, we discuss the difference between our network and the one in [12]. Our network is similar as their line detection network with the difference in the following perspectives:

1. In each hourglass module, they use two consecutive residual modules (RM) at each spatial resolution while we only use one RM, resulting less parameters in each hourglass module. Note that our design is the same as the original hourglass paper [23], which enables us to use more RM in each hourglass module and reduce the computational complexity since more computation is allocated in lower resolution stages. We adopt such design since we find using two RM gives negligible gains to the performance compared with only one.

2. We apply the intermediate supervision to the stacked hourglass network. For each hourglass modules, the loss term associated with the predicted heat maps is added to the final loss. [12] does not such intermediate supervision in their method. We find such intermediate supervision vital in both our synthetic dataset as well as their 2D dataset in terms of both accuracy and robustness.

3. We observe that using 2 stacked hourglass modules gives a similar performance as the 5 stacked hourglass modules in [12]. On the other hand, using 2 stacks consumes less memory, which gives us more design flexibility. For example, we are able to utilize a larger batch size to make the gradients more stable.