Supplementary Materials for Learning to Parse Wireframes in Images of Man-Made Environments

Kun Huang[1], Yifan Wang[1], Zihan Zhou[2], Tianjiao Ding[1], Shenghua Gao[1], and Yi Ma[3]
[1]ShanghaiTech University {huangkun, wangyf, dingtj, gaoshh}@shanghaitech.edu.cn
[2]The Pennsylvania State University zzhou@ist.psu.edu
[3]University of California, Berkeley yima@eecs.berkeley.edu

1. Wireframe Construction Algorithm Detail

Given an image, our wireframe construction algorithm takes a set of junctions $\{\boldsymbol{p}_i\}_{i=1}^N$, $\boldsymbol{p}_i = \left(\boldsymbol{x}_i, \{\theta_{ik}\}_{k=1}^{K_i}\right)$, and a line heat map h as input. Note that for the junctions and their branches predicted by our network, we only keep those with confidence scores higher than certain thresholds τ_c and τ_b, respectively. As a pre-processing step, we further adopt a strategy similar to non-maximum suppression to remove duplicate detections.

Our wireframe construction algorithm is presented in Alg. 1. In the algorithm, we first apply a threshold ω to convert the line heat map $h(p)$ into a binary map $\mathcal{M}$ (line 2). Note that this threshold ω is varied to obtain the precision-recall curve in our experiments on wireframe construction. The algorithm then proceeds as follows:

First, we connect all pairs of junctions which are aligned with each other's branch directions (lines 3-22). Let $\boldsymbol{r}_{ik}$ represent the ray starting at $\boldsymbol{p}_i$ along its k-th branch. We collect all possible rays as $\mathcal{R} = \{\boldsymbol{r}_{11}, ..., \boldsymbol{r}_{1K_1}, ..., \boldsymbol{r}_{i1}, ..., \boldsymbol{r}_{iK_i}, ...\}$, and use $(i,k) = \pi(t)$ to map the t-th ray in $\mathcal{R}$ to its junction index i and branch index k. Then, for the rays in $\mathcal{R}$, we use $\mathcal{V} \in \mathbb{R}^{N_r \times N_r}$, $N_r = |\mathcal{R}|$, to record the indices of the corresponding ray/branch of the closest opposite junction. Specifically, $\forall t_1 \in \{1, \ldots, N_r\}$, we set $\mathcal{V}(t_1, t_2)$ to 1 if and only if (i) $\boldsymbol{p}_i$ is the on the ray $\boldsymbol{r}_{jk_2}$ and $\boldsymbol{p}_j$ is on the ray $\boldsymbol{r}_{ik_1}$, where $(i, k_1) = \pi(t_1)$, $(j, k_2) = \pi(t_2)$, and (ii) the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is the shortest among all such aligned pairs (lines 5-15). Then, we consider two rays are matched if $\mathcal{V}(t_1, t_2) = \mathcal{V}(t_2, t_1) = 1$ and add the corresponding junctions and line segments to $\boldsymbol{P}$ and $\boldsymbol{L}$, respectively (lines 17-21).

Second, for any ray $\boldsymbol{r}_{ik}$ which fails to find a matching ray using the above procedure, we attempt to recover additional line segments using the line support $\mathcal{M}$ (lines 23-38). We consider the following cases:

(a) If the distance between $\boldsymbol{p}_i$ and $\boldsymbol{q}_b$, the intersection of $\boldsymbol{r}_{ik}$ and the image boundary, is smaller than certain threshold (say $0.05 \times m$ where m is the maximum of image width and height), we add $\{\boldsymbol{p}_i, \boldsymbol{q}_b\}$ and the connecting line segment to $\boldsymbol{P}$ and $\boldsymbol{L}$, respectively (lines 24-26).

(b) For a ray exceeding the length threshold in (a), we first find the farthest line pixel $\boldsymbol{q}_{\mathcal{M}}$ along the ray on $\mathcal{M}$. Then, we find all the intersection points $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_S\}$ of line segment $(\boldsymbol{p}_i, \boldsymbol{q}_{\mathcal{M}})$ with existing segments in $\boldsymbol{L}$ (lines 28-29). Let $\boldsymbol{q}_0 = \boldsymbol{p}_i$ and $\boldsymbol{q}_{S+1} = \boldsymbol{q}_{\mathcal{M}}$, we calculate the line support ratio $\kappa(\boldsymbol{q}_{s-1}, \boldsymbol{q}_s), s = \{1, \ldots, S, S+1\}$, for each segment. Here, κ is defined as the ratio of line pixels (pixel p is a line pixel if $\mathcal{M}(p) = 1$) to the total length of the segment. If κ is above a threshold, say 0.6, we add the segment to $\boldsymbol{L}$ and its endpoints to $\boldsymbol{P}$ (lines 30-36).

2. Additional Experiments

2.1. Experiment on Junction Detection Network Parameters

In this section, we examine the choices of two important hyper-parameters in our junction detection network.

Effect of balancing positive and negative samples. In this experiment, we vary the value $r_{\max}$, which controls the maximum ratio between negative and positive samples at each iteration. Note that setting $r_{\max} = \infty$ is equivalent to using all grid cells during training. We can observe in Figure 1(a) that the precision-recall curves largely overlap. But as $r_{\max}$ increases, the curve shifts toward the high-precision-low-recall regime, and vice versa. For example, when $r_{\max} = 1$, the precision and recall at $\tau = 0.5$ are 0.19 and 0.94, respectively. And when $r_{\max} = \infty$, the precision and recall at $\tau = 0.5$ are 0.70 and 0.44, respectively. Note that this has an important implication in practice, as human annotators tend to miss true junctions much more often than labelling wrong junctions. Empirically, we have found that $r_{\max} = 7$ yields more satisfactory results.

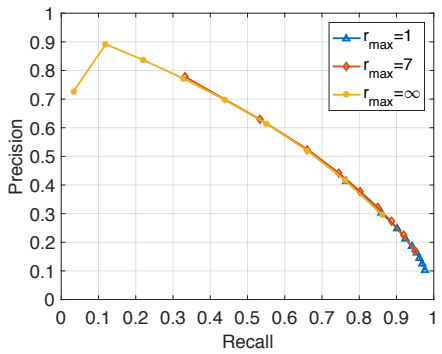
Algorithm 1 Wireframe Construction

Input: Junctions $\{\boldsymbol{p}_i\}_{i=1}^{N}$, $\boldsymbol{p}_i = \left(\boldsymbol{x}_i, \{\theta_{ik}\}_{k=1}^{K_i}\right)$, and a line heat map $h(p)$

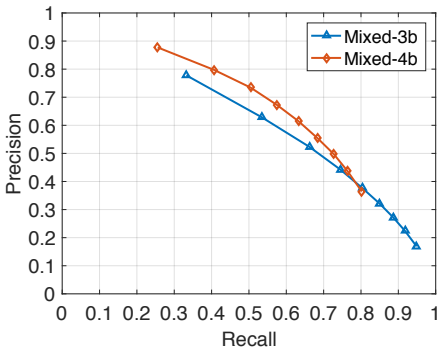
Output: Wireframe $\boldsymbol{W}$ consisting of a set of junction points $\boldsymbol{P}$ connected by a set of line segments $\boldsymbol{L}$

Initialize $\boldsymbol{P} \leftarrow \emptyset$, $\boldsymbol{L} \leftarrow \emptyset, \mathcal{V} \leftarrow \boldsymbol{0}$
Binarize $h(p)$ with threshold ω into $\mathcal{M}$
for $t_1 \in \{1, 2, \ldots, N_r\}$ **do**
  $(i, k_1) \leftarrow \pi(t_1)$, $d_{\min} \leftarrow \infty$, $z \leftarrow 0$
  for $t_2 \in \{1, 2, \ldots, N_r\}$ **do**
    $(j, k_2) \leftarrow \pi(t_2)$
    if $j \neq i$ **and** $\boldsymbol{p}_j$ on $\boldsymbol{r}_{ik_1}$ **and** $\boldsymbol{p}_i$ on $\boldsymbol{r}_{jk_2}$ **then**
      if $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| < d_{\min}$ **then**
        $d_{\min} \leftarrow \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, $z \leftarrow t_2$
      end if
    end if
  end for
  if $z \neq 0$ **then**
    $\mathcal{V}(t_1, z) \leftarrow 1$
  end if
end for
for all $t_1, t_2 \in \{1, 2, \ldots, N_r\}, t_1 \neq t_2$ **do**
  if $\mathcal{V}(t_1, t_2) = 1$ **and** $\mathcal{V}(t_2, t_1) = 1$ **then**
    $(i, k_1) \leftarrow \pi(t_1)$, $(j, k_2) \leftarrow \pi(t_2)$
    $\boldsymbol{P} \leftarrow \boldsymbol{P} \bigcup \{\boldsymbol{p}_i, \boldsymbol{p}_j\}$, $\boldsymbol{L} \leftarrow \boldsymbol{L} \bigcup \{(\boldsymbol{p}_i, \boldsymbol{p}_j)\}$
  end if
end for
for all $\boldsymbol{r}_{ik}$ not matched to another ray **do**
  Find the intersection of $\boldsymbol{r}_{ik}$ and image boundary $\boldsymbol{q}_b$
  if $\|x_i - \boldsymbol{q}_b\| \leq 0.05 \times m$ **then**
    $\boldsymbol{P} \leftarrow \boldsymbol{P} \bigcup \{\boldsymbol{p}_i, \boldsymbol{q}_b\}$, $\boldsymbol{L} \leftarrow \boldsymbol{L} \bigcup \{(\boldsymbol{p}_i, \boldsymbol{q}_b)\}$
  else
    Find the farthest point $\boldsymbol{q}_{\mathcal{M}}$ along $\boldsymbol{r}_{ik}$ on $\mathcal{M}$
    Find all intersections $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_S\}$ of $(\boldsymbol{p}_i, \boldsymbol{q}_{\mathcal{M}})$ with segments in $\boldsymbol{L}$
    $\boldsymbol{q}_0 \leftarrow \boldsymbol{p}_i$, $\boldsymbol{q}_{S+1} \leftarrow \boldsymbol{q}_{\mathcal{M}}$
    for $s \in \{1, 2, ..., S, S+1\}$ **do**
      if $\kappa(\boldsymbol{q}_{s-1}, \boldsymbol{q}_s) > 0.6$ **then**
        $\boldsymbol{P} \leftarrow \boldsymbol{P} \bigcup \{\boldsymbol{q}_{s-1}, \boldsymbol{q}_s\}$
        $\boldsymbol{L} \leftarrow \boldsymbol{L} \bigcup \{(\boldsymbol{q}_{s-1}, \boldsymbol{q}_s)\}$
      end if
    end for
  end if
end for

Going deeper. It is also interesting to investigate how the network depth of the encoder affects the performance. In this experiment, we compared two different choices based on Google Inception-v2, namely the first layer to "Mixed_3b", and the first layer to "Mixed_4b". Note that the latter has a larger depth and receptive field, at the cost of spatial resolution (30×30). As one can see in Figure 1(b),



(a) $r_{\max}$



(b) Network depth

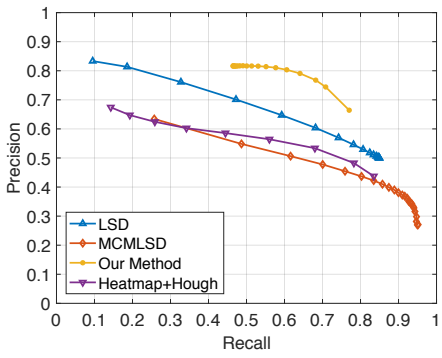
Figure 1. Experiment on junction detection network parameters.

increasing the depth (i.e., predicting at the "coarser" level) results in higher precision but lower recall. This suggests possibilities to further improve the performance of our method using a "skip-net" architecture, that is, combining predictions at multiple levels. We leave this for future work.

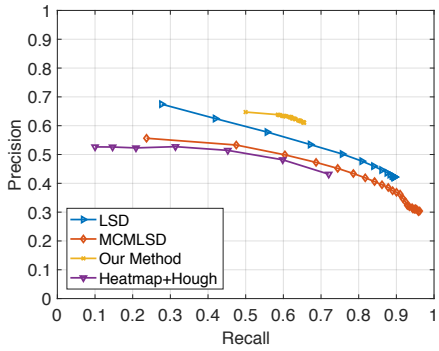
2.2. Experiment on Line Segment Detection

In this experiment, we study the possibility of extracting line segments *directly* from the pixel-wise line heat map predicted by our network (i.e., without using junctions). To this end, we simply perform a probabilistic hough transform [1] on the line heat map to generate line segments. We compare the results with LSD, MCMLSD, and our full wireframe construction method.

Figure 2 shows the precision-recall curves of all methods. We make the following observations on the results: *First*, the performance of our "Heatmap + Hough" approach is comparable to that of the state-of-the-art line segment detection method MCMLSD, verifying the effectiveness of the our line detection network. *Second*, by combining the predicted junctions with the line heat map, our full wireframe construction method performs significantly better than using the line heat map alone. This further illustrates the importance of junction detection in parsing the wireframe: By detecting the "endpoints" of the line segments, we effectively overcome the difficulties faced by tra-



(a) Our test dataset



(b) York Urban dataset

Figure 2. Experiment on line segment detection.

ditional line segment detection methods, including the false detection problem and the inaccurate endpoint problem.

2.3. Additional Results on Junction Detection

In Figure 4, we show additional junction detection results obtained by all methods. One can see that our method is able to detect most junctions and their branches in the image, achieving superior performance over existing methods.

From Figure 4 we can also observe some limitations of our method. Specifically, there are occasionally repeated detections in our result. This may be caused by junctions located at the boundary of two adjacent grid cells used in our junction detection network. Similarly, the use of grid could also lead to missed detection if two junctions are very close to each other. But we note that such cases are rather uncommon in practice and have very small effect on the overall scene structure estimation.

2.4. Additional Results on Wireframe Construction

In Figure 5, we show additional wireframe detection results obtained by all methods. Our method outperforms other two in most areas and produces much cleaner results as we focus on long line segments and exploit their relations (junctions). Therefore, the resulted wireframes are potentially more suitable for 3D reconstruction tasks.

In Figure 3, we further show some failure cases of our method. One challenging case corresponds to structures with relatively small scale and weak image gradients (e.g., the stairs in the first image). Also, our method sometimes has difficulty in image region of repetitive patterns (e.g., the handrails in the second image and the brick wall in the third image), generating fragment, incomplete results. This suggests opportunities for further improvement by explicitly harnessing such geometric structure in our wireframe construction.

Figure 3. Failure cases on our test dataset. **First row:** Our method. **Second row:** Ground truth.

References

[1] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000. 2

Figure 4. Junction detection results. **First row:** MJ ($d_{\max} = 20$). **Second row:** ACJ ($\epsilon = 1$). **Third row:** Our method ($\tau = 0.5$).

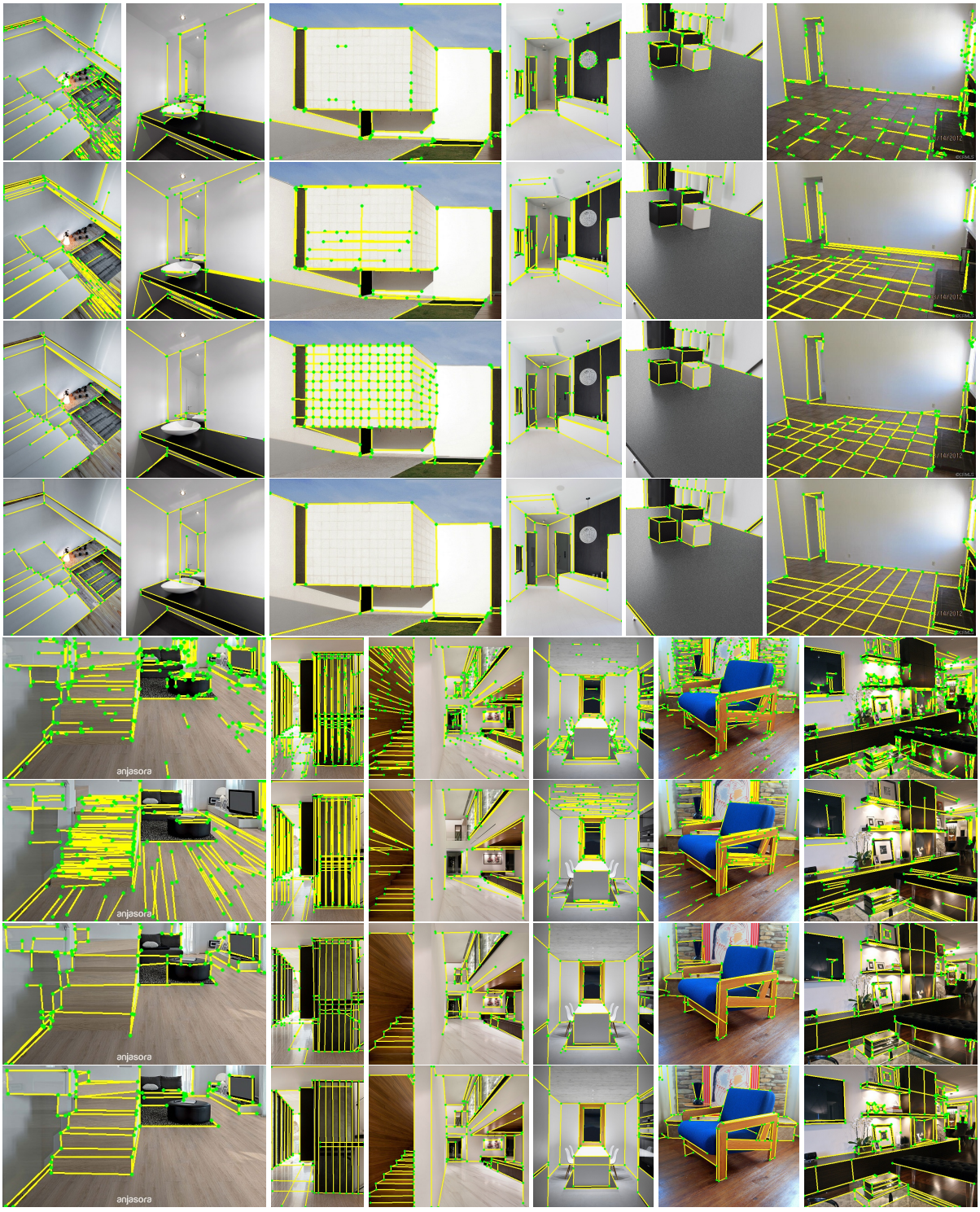


Figure 5. Line/wireframe detection results. **First row:** LSD (-log(NFA) $> 0.01 \times 1.75^8$). **Second row:** MCMLSD (confidence top 100). **Third row:** Our method (line heat map $h(p) > 10$). **Fourth row:** Ground truth.