Wireframe Parsing With Guidance of Distance Map

KUN HUANG1,2,3, (Student Member, IEEE), AND SHENGHUA GAO2, (Senior Member, IEEE)

1Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
2School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
3University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Kun Huang (huangkun@shanghaitech.edu.cn)

ABSTRACT We propose an end-to-end method for simultaneously detecting local junctions and global wireframe in man-made environment. Our pipeline consists of an anchor-free junction detection module, a distance map learning module, and a line segment proposing and verification module. A set of line segments are proposed from the predicted junctions with guidance of the learned distance map, and further verified by the proposal verification module. Experimental results show that our method outperforms the previous state-of-the-art wireframe parser by a descent margin. In terms of line segments detection, our method shows competitive performance on standard benchmarks. The proposed networks are end-to-end trainable and efficient.

INDEX TERMS Artificial neural networks, computer vision, feature extraction, image edge detection.

I. INTRODUCTION Inferring 3D geometric information of a scene from 2D images has been a fundamental yet difficult problem in computer vision. For a long time, researchers model and reconstruct 3D scenes by extracting and matching local features (e.g., SIFT features, corners and patches). The feasibility of incorporating these local features by matching or tracking has been demonstrated by numerous works. Meanwhile, modern scenarios which often involve complex interactions of autonomous agents (UAV, car, home robot) with cluttered man-made environments (indoor or outdoor) present greater challenge for conventional local features based approaches. Specifically, the challenges lie in: Man-made scenes often consist of large areas of textureless surfaces; there may exist areas of repetitive patterns (facades e.g.) which cause much ambiguity for matching. Therefore, it is crucial for vision systems to capture more global features to analyze the scenes more accurately.

To cope with these challenging environments, prior knowledge about the scene is exploited to relax the problem in many works. For instance, Manhattan world assumption [1]–[5] has proven to benefit the 3D reconstruction tasks. However, the Manhattan world assumption is often violated in cluttered man-made environments. Fortunately, independent of the assumption, we can formulate such environments as the “wireframe” defined in [6]. The wireframe [6] consists of a subset of salient lines and junctions in the scene. Conceptually, such junctions and lines could just be a subset among all the local corner and edge features extracted by traditional methods, but they encode rich information about the large-scale geometry of the scene, which is the key to understand the scenes more globally.

Detection of line segments [7] and junctions [5], [8] in 2D images has been studied in previous works. The detection results are further used to recover the geometric of the scenes in [2], [4], [9]–[11]. Typically, these methods rely on low-level cues and involve various heuristics, including search of appropriate thresholds, RANSAC-based verification techniques etc., which limit its scalability in new scenarios.

With the success of deep learning in computer vision, more and more approaches based on deep learning have been proposed to tackle vision tasks. Specifically, in the literature of line segment and junction detection, several methods have been proposed to detect line segments [12] or estimate the wireframe [6] in the image. In [12], a line attraction filed map (AFM) method is proposed to detect line segment by learning an embedding \( \mathbf{v}_{\text{dir}} \) for each pixel \( p \) in the image. The \( \mathbf{v}_{\text{dir}} \) encodes the direction (with length) to the line segment pixel which is the closest to \( p \). In the post-processing stage, the line segment candidate pixels are found by adding...


$\mathbf{v}_{dir}$ to $\mathbf{p}$. These candidate pixels are grouped into line segments using greedy algorithm. AFM shows better performance than previous state-of-the-art line segment detection methods. However, AFM only detects line segments, ignoring the intersecting relationships between pairs of them, which is important for geometric reasoning.

Previous to AFM, the Wireframe Parser [6] formulates wireframe parsing as finding the salient line segments ($\mathcal{L}_i$) and the intersections between these segments, i.e., junctions ($\mathcal{P}_i$) in the images. The concept of wireframe is more complete than pure set of line segments, and the wireframe obviously encodes richer geometric information. The parser network in [6] consists of two sub-networks for junction and line detection, respectively. The junction sub-network divides the image into $K \times K$ grid, and detect at most one junction in each grid cell by regressing the confidence of the junction cell. Besides, the residual of the junction off the grid cell center is regressed in the meanwhile. The line detection sub-network estimates the image pixels which are on long straight lines. Finally, the detected junctions and estimated line pixels are combined into holistic wireframe with a post-processing procedure. Wireframe Parser captures some important geometric information of the scene. However, there are several drawbacks that limit the performance of the parser:

- The junction decoder predicts fixed number of junction (i.e., 1) in each grid cell. Considering the uneven distribution of junctions in scenes, the assumption that there exists only one junction in a single grid cell does not apply to scenes with dense junctions, such as images of facades, urban street views.
- The process of proposing line segment from junctions suffers from the inaccurate junction predictions. Especially, the inaccurate junction branches make it difficult to pair junctions.
- The proposed line segments are not further verified to reject false connections between junctions.

Based on the above discussions of previous works on line segment detection and wireframe parsing, we propose an end-to-end network for parsing the wireframe of man-made environments. Overall, our network contains two stages. In the first stage, our network learns three per-pixel embedding maps, including junction heatmap, junction branch map, and distance map. Then in the second stage, a set of line segment proposals are proposed from the junctions with guidance of the distance map, and further verified through a verification network.

As depicted in Fig. 2, with an image fed into the network, the junction detection module outputs a junction heatmap (heatmap decoder) and a junction branch map (branch decoder). Junctions are obtained by picking the locations of high density over the junction heatmap, thus our network detects junctions in an anchor-free manner, and is able to detect unlimited number of junctions. Meanwhile the distance decoder outputs a distance map. Examples of distance map is shown in Fig. 1. Then we propose a set of line segments using our Least Distance algorithm from the predicted junctions guided by the learned distance map. Finally, the proposals are passed to a verification module and each assigned a confidence score.

Contributions of this work include: (i) We propose an end-to-end learning network for wireframe parsing in the images of man-made environments, consisting of an anchor-free junction detection module, a distance learning module and a line segment proposing and verification module. (ii) Our method outperforms the previous wireframe parser by a descent margin, and is more efficient. (iii) Compared with the previous wireframe parser, our method produces much cleaner wireframe results with guidance of the learned distance map. (iv) In terms of line segment detection, our method is close to the state-of-the-art line segment detection method (AFM [12]).

II. RELATED WORK

A. JUNCTION DETECTION AND KEYPOINT DETECTION

Junctions played an important role in computer vision, however, detecting junctions in images remains a challenging problem. Typical local methods such as Harris operator [14], are based on 2D variation in the intensity signal, and it is weak at handling textured regions. More recent methods find junctions in natural images [15] by studying contour curvature. Some others group line segments [5], [8], [9] to find junctions (i.e., the intersections of segments). As psychophysical analysis suggests that local junction is difficult to detect, even for humans [16].

Keypoint detection is closely related to some reconstruction tasks, such as reconstruction of scenes, faces and human poses. Many recent approaches for keypoint estimation apply deep neural networks (DNNs). Earlier works formulate keypoint estimation as a location (keypoint coordinates) regression problem [17]. In later works, keypoint detection methods mostly rely on regressing a pre-computed keypoint heatmap [18], [19]. The heatmap can be easily converted
to keypoint coordinates by searching the local maximums. Junctions can be naturally considered as geometric keypoints. Huang et al. [6] train a anchor-based junction detector using deep neural networks on a large-scale dataset with junction annotation, and achieve the state-of-the-art results. Recently, Zhou et al. [20] propose to find junctions in an end-to-end manner.

B. PIXEL-LEVEL EDGE DETECTION AND LINE SEGMENT DETECTION

Line segments detection has long been studied. Conventional methods typically rely on grouping low-level cues [7], [21]–[24]. These approaches suffer the pain of searching an appropriate threshold to filter out false detections. Some other work extends Hough transform to line segments detection [25]–[27]. In recent years, machine learning especially deep learning based approaches have been shown to produce the state-of-the-art results in generating pixel-wise edge maps [28]–[31]. Huang et al. propose to parse wireframe in man-made scenes. The wireframe defined in [6] consists of a subset of salient line segments and the intersections between them in scenes. And they build a large-scale dataset for training. Each image in the dataset is annotated with line segments and their intersecting relationship. Following Wireframe Parser, Xue et al. propose a line attraction field network [12] for line segment detection and achieve the state-of-the-art line segment detection results. Unlike the Wireframe Parser, they only focus on detection of line segments, ignoring the relationship between the segments. Zhang et al. [32] formulate the wireframe parsing task as a graph optimization problem.

C. LEARNING THE GEOMETRY

With the success of deep learning, learning based approaches on inferring pixel-level geometric properties of scenes have been developed for several years. Geometric properties, such as the depth [33], [34], and the surface normal [35], [36] have been extensively studied. Recently, more and more work pays attention to the higher-level geometric primitives. For instance, [37] proposes a method to recognize planes in a single image, [38] uses SVM to classify indoor planes, and [39]–[41] train fully convolutional networks to predict room layout edges formed by the pairwise intersections of room faces. Liu et al. propose PlaneNet [42] to detect piece-wise planar regions from a single image. Yang and Zhou [43] recover planar regions with deep networks.

D. EMBEDDING LEARNING FOR DETECTION

Embedding learning are frequently used in instance segmentation. Usually the embedding is learned by regressing the pre-computed embedding map generated from the ground-truth annotations and acts as the cue to cluster pixels or group keypoints into instances. For example, Liang et al. use proposal-free networks (PFN) [44] to handle instance segmentation task. PFN take the coordinates of the center, top-left corner and bottom-right corner of the object instance that a specific pixel belongs to as the target embedding, which is used to cluster segmentation results into object instances. Bai and Urtasun [45] apply deep networks to learn a distance embedding, which measures how close a pixel in an instance object is to the boundary of the instance. The network outputs an embedding map with peaks and basins. Watershed transform is then used in post-processing to find the basins, i.e., the instance boundaries. Papandreou et al. [46] propose to learn pose keypoints heatmap along with the displacement of each keypoint to other keypoints. Recently, Xue et al. [12] propose line attraction field networks to learn an embedding for line segment detection. The line attraction embedding encodes the direction (with length) of each pixel to the closest line segment pixel, and is post-processed with the squeeze module to find line segments.

III. METHOD

For clarity of notation, we let \( p \) and \( x \) denote an image pixel and a junction, respectively. And we denote a line segment by \( l \). In addition, superscript ‘\( g \)’ and ‘\( p \)’ represent ground-truth and predictions, respectively. In terms of distance functions,
heatpmap density value at candidates. The projection of the point on the line segment is (a) inside the line, \( d(x, l) = d^2(x, l) \), the perpendicular distance of \( x \) from \( l \), (b) outside the line, \( d(x, l) = \min(d(x, s), d(x, s')) \), i.e., the smaller euclidean distance between \( x \) from \( s_1 \) and \( s_2 \).

A. ANCHOR-FREE JUNCTION DETECTION

Junction acts as an important role in our line segment proposing algorithm. We follow the definition and representation in [6]. However, we made some modifications to make the representation more convenient.

1) JUNCTION HEATMAP REGRESSION

In [6], the junction sub-network divides the image into \( K \times K \) grid, and detect at most one junction in each grid cell. The cell centers act as a kind of anchor commonly used in object detection methods [47]. And the use of anchor is based on the assumption that there exists at most one junction in each cell. However, the assumption does not apply to some scenes with dense junctions.

As mentioned in Section II, heatmap regression has become a common practice in keypoint detection tasks, such as human pose estimation [18]. By applying heatmap, estimation of the keypoint coordinates is transformed to spatial density regression. Inspired by this, we detect junctions by first regressing a junction heatmap. The ground-truth of junction heatmap is generated by placing a Gaussian probability distribution with a radius of 11 (the image size is 384) centered at each junction \( x \). As is shown in Fig. 2, the junction heatmap decoder outputs a \( H \times W \) junction heatmap. Then locations with local maximum density on the learned heatmap are extracted as junction candidates through non-maximum suppression (NMS). \(^1\) After NMS, we obtain a set of junction candidates \( \{(x_i, c_i), i = 1, 2, \ldots, N_{\text{max}} \} \). \( N_{\text{max}} \) is the maximum number (pre-set value) of junctions in an image. \( c_i \) is heatmap density value at \( x_i \). Overall, the proposed junction detection method shows more flexibility without relying on any anchors or assumptions.

\(^1\) Here, the NMS is a simple max-pooling operation plus a filtering of non-maximum values.

2) JUNCTION BRANCH DEFINITION AND REPRESENTATION

For junction branches, we adopt the same multi-bin representation proposed in [6]. As depicted in Fig. 4, the circle (i.e., from 0 to 360 degrees) is divided into \( K \) equal bins, with each bin spanning \( \frac{360}{K} \) degrees. Let the center of the \( k \)-th bin be \( b_k \), then an angle \( \theta \) is represented as \( (k, \Delta_k) \), if \( \theta \) is located at the \( k \)-th bin, where \( \Delta_k \) is the angle offset from the center \( b_k \) in the clockwise direction. Thus, each valid bin \( (e.g., \text{exists a junction branch}) \) will regress to this local orientation \( \Delta_k \). We normalized \( \Delta_k \) to be in \([-1, 1]\) in all our experiments. For each predicted junction location (from heatmap), the branch decoder in Fig. 2 outputs a \( 2K \) vector, i.e., confidence score and angle offset for \( K \) bins.

However, the junction branch defined in [6] is incomplete for proposing line segments. Geometrically, junction is the intersection of \( n (n \geq 2) \) line segments, that is, each junction should have at least two branches. However, the endpoints of some line segments lie on the image boundary (the line is cut off by the image boundary), or some curve lines. Both cases are not considered in the previous definition. Hence we extend the junction definition to the union of intersections of line segments and isolated endpoints of line segments (e.g., on the image boundary or none-straight lines). Then each junction contains \( n (n \geq 1) \) branches. This definition is more flexible, and more suitable for finding line segments from junctions.

In [6], junctions locations and branches are predicted separately, we similarly use two decoders to predict junction heatmap \( (1 \times H \times W) \) and branch map \( (2K \times H \times W) \), respectively.

3) MATCHING KEYPOINTS WITH GROUND-TRUTH JUNCTIONS

A set of junction candidates \( \{(x_i, c_i), i = 1, 2, \ldots, N_{\text{max}} \} \) are extracted from the junction heatmap, which is the output of heatmap decoder in Fig 2. Those with confidence score above a threshold \( \tau_j \) are kept as junction predictions. For a junction prediction \( x_j^p \), we first find the ground-truth junction \( x_j^g \) which is closest to \( x_j^p \). If \( d(x_j^p, x_j^g) < \tau_M \), where \( \tau_M \) is a threshold, then we assign the branches of \( x_j^p \) to \( x_j^g \) as its branch map regression target. Intuitively, \( \tau_M \) is set to be half size of the Gaussian kernel used in generating the ground-truth of junction map.
B. DISTANCE MAP LEARNING

The shortest distance of a point to a line segment is depicted in Fig. 3. For an image, the distance map $D$ is computed by choosing the smallest among all distances $\{d(p, l_i), i = 1, 2, \ldots, N\}$ for each pixel $p$, where $N$ is the total number of ground-truth line segments in the image. As mentioned previously, the embedding learned in instance segmentation is exploited to group pixels into instances in post-processing.

In our case, the distance map is designed to filter out non-line textures and encodes a value for each pixel. Examples of ground-truth distance maps are presented in Fig. 1. We use the log-normalized distance $d_p^p = \log(d_p + 1)$ as the target for distance map learning, since $d_p$ can be very large. And $d_p^p \in [0, \log(\sqrt{H^2 + W^2} + 1)]$.

C. PROPOSING LINE SEGMENTS FROM JUNCTIONS GUIDED BY DISTANCE MAP

For an image $I$, our line segment proposing algorithm takes the distance map $D$ and the predicted junctions $X = \{(x_i^p, \theta_{i\theta}), i = 1, 2, \ldots, N\}$ (where $\theta_{i\theta}$ is the $\theta$ branch of junction $x_i^p$) as input. Instead of striving to pair junctions according to their branches as in [6], we exploit our Least Distance algorithm to generate a set of line segment proposals $P = \{(x_i^p, x_j^p)\}$.

Let $r_{ik}$ denote the ray formed by junction $x_i$ and branch $\theta_{ik}$, and $\psi(x_i, x_j, r_{ik})$ be the angle difference between vector $x_i - x_j$ and the ray $r_{ik}$. The Least Distance algorithm takes following steps.

- At first, we initialize an empty candidate set $C_i$ and an empty proposal set $P_i$. $C_i$ stores the line segment candidates starting from $x_i^p$.
- For $k_{ih}$ branch $\theta_{ik} \in [0, 360]$ of $x_i^p$, for all other $N - 1$ junctions, if $\psi(x_i, x_j, r_{ik}) < \tau_p$, we add line $(x_i^p, x_j^p)$ to $C_i$.
- For a single candidate in $C_i$, we extract a fixed length of distance value from $D$ using our modified version of Rol-Align Module [13]. Specifically, for candidate $(x_m^p, x_n^p)$, we sample the distance values from $S$ equidistant sampling points starting from $x_m^p$ to $x_n^p$, and sum up the sampled values.
- Among all candidates in $C_i$, we choose the one $x_i^p$ with the least summed distance value (least distance indicates higher confidence of line segment), and add $(x_i, x_n^p)$ to $P_i$.
- $P$ is the set of line proposals.

Overall, our Least Distance algorithm proposes $O(N)$ line segments for an image with $N$ junctions. Note that the sampling of distance value has efficient GPU implementations.

D. PROPOSAL VERIFICATION NETWORK

1) ASSIGN LABEL TO PROPOSALS

For an image $I$, we obtain a set of proposals $P = \{(x_i^p, x_j^p)\}$ with the Least Distance algorithm. To further verify these proposals, we match the proposed line segments with ground-truth line segments and assign each proposal a label 0 (not matched) or 1 (matched).

Let $\varphi$ denote the angle difference between two lines, $\forall l_i, l_j, \varphi(l_i, l_j) \in [0, 180]$. We match a proposal $P^p = (x_i^p, x_j^p)$ in $P$ with a ground-truth line segment $l^g = (x_{k1}^g, x_{k2}^g)$ if either of the two conditions below is satisfied.

1. Condition 1: $d(x_i^p, l^g) < \tau_d, d(x_j^p, l^g) < \tau_d$.
2. Condition 2: $\varphi(P^p, l^g) < \tau_g$, $d(x_k^p, l^g) < \tau_d, d(x_j^p, l^g) < \tau_d$.

where $\tau_d$ and $\tau_g$ are pre-set thresholds. As mentioned earlier, $\varphi(x_i^p, l^g)$ is the projection of junction $x_i^p$ onto segment line $l^g$. For a proposal $P = (x_i^p, x_j^p)$, if there exists a ground-truth line segment $l^g$ satisfying condition 1 or condition 2, then $P^p$ is assigned the label 1, otherwise 0.

The condition 1 can cover most cases, and condition 2 is to handle the situation where a ground-truth line segment was not annotated with accurate length (a little shorter or longer). The tolerance threshold $\tau_g$ is set to 0.2 in our experiments.

2) PROPOSAL VERIFICATION

In the proposal verification module, features for all the line segment proposals are extracted by performing the modified Rol-Align on $F_n$ (feature layer after the $n_{th}$ hourglass module) and fed into a proposal classification network, which outputs a confidence score for each proposal. As to Rol-Align, we first sample $S$ equidistant sampling points from the starting point to the ending point of a line segment, and sample values on the feature map $F_n$ (shown in Fig. 2) at those sample points. Note that the original Rol-Align Module [13] is intended for extracting features of bounding box areas, hence we modified the module to extract features for line segments.

To align the features with the line segment, our modified Rol-Align module samples feature values by bilinear interpolation to get sub-pixel features. Assume the number of channels of $F_n$ is $C$, the extracted feature map for each line segment proposal is of shape $S \times C$.

E. NETWORK ARCHITECTURE

Our network is typical encoder-decoder style, as shown in Fig. 2. The first few layers are convolution and pooling layers, which downsample the input to 1/8 of original resolution. Then $n (n = 3$ in our experiment) hourglass module [18] are stacked to get the feature maps $\{F_i, i = 1, 2, \ldots, n\}$. The following decoders can take any $F_i$ as input, and we used $F_3$ as input for decoders as shown in Fig. 2. Each decoder consists of a set of convolution and upsampling layers, followed by a $1 \times 1$ convolution layer to get the desired embedding maps. For simplicity, all the decoders share the same layer configuration. To classify the proposals, the proposal verification module consists of two convolution layers and one fully connected layer. Note that all convolution layers are followed by batch normalization layer.
F. LOSS FUNCTION

Our loss function consists of the regression loss of junction heatmap \( L_h \), distance map \( L_D \), the junction branch confidence \( L_{bc} \), junction branch residual (offset) \( L_{br} \), and the line segment proposal confidence \( L_{prop} \). We use the binary cross entropy (BCE) loss for \( L_h \), i.e.

\[
L_h^f = \text{BCE}(J^p, J^g)
\]

where \( J^p \) and \( J^g \) refers to the prediction and ground-truth junction heatmap, respectively. Note that we adopt the weighting scheme of focal loss \([48]\) to combat the imbalance of zero and non-zero value on the heatmap. And

\[
L_D = \sum_{p \in I} \frac{1}{w_D} ||d_p^p - d_p^g||^2
\]

where we weight the loss of each pixel \( p \) with \( d_p^g \), since pixels far from lines result in much larger loss if not weighted.

For junction branches, as mentioned above, we first matched every predicted junctions \( x_i^p \in X \) to the ground-truth junctions \( G = \{x_i^g\} \). We define a mapping function \( f(i) \) to represent the index of the ground-truth junction which is matched with the predicted junction \( i \). And \( c_i \) and \( r_i \) are both \( K \)-dimension (\( K \) is the number of bins) vector representing the branch confidence and residual of junction \( x_i \), respectively. Assume there are \( N \) junction predictions in \( X \), and

\[
L_{bc} = \sum_{i=1}^{N} \text{BCE}(c_i^p, c_{f(i)}^g)
\]

\[
L_{br} = \sum_{i=1}^{N} ||r_i^p - r_{f(i)}^g||_2^2
\]

As mentioned in Section III-D, each line segment proposal \( l_k^p \) is assigned a label \( g_k \). Let \( \text{clst}(l_k^p) \) denote the confidence score output by the proposal classification network, then

\[
L_{prop} = \sum_{k=1}^{N_{prop}} \text{BCE}(g_k, \text{clst}(l_k^p))
\]

\( L_h^f \) and \( L_D \) are normalized by the number of pixels in \( I \). And \( L_{bc} \) and \( L_{br} \) are normalized by the number of matched junction predictions. And \( L_{prop} \) is normalized by the total number of line segment proposals. Then our loss function is

\[
\mathcal{L} = w_J L_h^f + w_D L_D + w_{bc} L_{bc}^f + w_{br} L_{br}^f + w_{prop} L_{prop}^f
\]

where \( w_J, w_D, w_{bc}, w_{br}, w_{prop} \) are weights to balance the sub-losses.

G. TRAINING

In the training and testing phase of our network, images are resized to \( 384 \times 384 \) before input. And we follow the data augmentation strategy in \([6]\), including mirroring and flipping images upside-down. The training of our network consists of two stages. In the first stage, we set \( w_{prop} \) to zero in the first 25 epochs, since there are not enough proposals for classification in the early stage of training. At the second stage, we resume the training of \( L_{prop} \) and train 5 more epochs.\(^2\) We train our network with the stochastic gradient descent (SGD) optimizer with momentum 0.9, initial learning rates 0.25\(^3\) and a batch size of 4.

IV. EXPERIMENTS

We evaluate the proposed method and make a comparison with the previous state-of-the-art wireframe parser \([6]\).\(^4\) Also we compare our line detection results with the

\(^2\)Weights of other losses are not changed.

\(^3\)Note that our learning rate is large due to the scale of our loss is small

\(^4\)https://github.com/huangkuns/wireframe

![FIGURE 5. The precision-recall curves of different line segment detection methods. Left: Wireframe dataset \([6]\) Right: The York Urban dataset \([49]\).](image-url)
state-of-the-art line detectors, namely the Line Segment Detector (LSD) [7] and the Markov Chain Marginal Line Segment Detector (MCMLSD) [50], and the recent Line Attraction Field (AFM) [12]. We test and compare with these methods on Wireframe dataset [6] and the York Urban dataset [49].

http://www.ipol.im/pub/art/2012/gjmr-lsd/
http://www.elderlab.yorku.ca/resources/

We use the results pre-computed by the author here.
A. DATASETS
The Wireframe dataset [6] contains 5462 images of indoor and outdoor man-made environments. Each image is annotated with a set of junctions and lines. We follow the split in [6], i.e., 5000 images for training and validation, 462 images for testing. On York Urban dataset, we simply use the model trained on Wireframe dataset to evaluate the performance.

B. EVALUATION METRICS
In all our experiments, the precision and recall as described in [6], [51] are reported. Specifically, precision and recall are computed by comparing the line segment pixels map of the prediction and ground-truth. The precision measures the proportion of true detections among all the detected line segments, whereas recall indicates the percentage of true detections among all the ground-truth line segments in the image.

C. COMPARISONS WITH OTHER METHODS
The proposed method is compared with the previous Wireframe Parser [6], the Line Segment Detector (LSD) [7] and the Markov Chain Marginal Line Segment Detector (MCMLSD) [50], and the Line Attraction Field (AFM) [12].

These methods filter out false detections with different thresholds. We set an array of thresholds for $-\log$(NFA) (NFA is the number of false alarms) in LSD implementation, i.e., $0.01 \times \{1.75^{0}, \ldots, 1.75^{19}\}$. And for MCMLSD, we select top $K$ (in terms of confidence) line segment detections for comparison. For AFM, aspect ratio is used to filter out false detections, and it varies in the range $(0, 1]$ with a step size of 0.1. The Wireframe Parser rejects false detections by applying an array of thresholds $\{2, 6, 10, 20, 30, 50, 80, 100, 150, 200, 250, 255\}$ to binarize the line heatmap, while keeping the threshold of junction confidence and junction branch confidence fixed. In the proposed method, we simply vary the threshold of the confidence of line segment detections in the range $(0, 1]$ with a step size of 0.1 to pick true detections.

The precision/recall curves are presented in Fig. 5. As is shown, our method outperforms the previous Wireframe Parser by a descent margin on Wireframe dataset, and is close to the Wireframe Parser on York Urban dataset. It is worth noting that the York Urban dataset is a small dataset aimed for Manhattan lines estimation, and some line segments in the images are not annotated. Hence the performance of almost all approaches significantly drops on this dataset. Comparing with the state-of-the-art line segment detectors, our method outperforms LSD and MCMLSD, and is close to the recent AFM on Wireframe dataset.

All our experiments are conducted on one NVIDIA Titan X GPU device. We compare the inference speed of all the methods mentioned above in Table. 1. As is shown, our method is much faster than the previous Wireframe Parser [6], since it does not require any post-processing step. And our method runs as quickly as the AFM-unet [12].

D. QUALITATIVE EVALUATION OF RESULTS
The results of all aforementioned methods are visualized in Fig. 6. For the proposed method and the Wireframe Parser, the junctions (yellow) and line segments (green) are displayed together. Since the line segment detection methods do not output junctions, we simply treat the endpoints of the line segments as junctions and present the “junctions” and the line segments.

As is shown, LSD [7] and MCMLSD [50] are sensitive to local textures, and produces many short line segments. AFM [12] greatly improves this limitation by suppressing the interference of local textures. However, AFM still yields some short line segments. The Wireframe Parser further reduces the effect of local textures by defining junctions explicitly and then establishing connections between junctions. Unfortunately, some wrong connections are found due to the lack of means to reject wrong connections. Compared with the Wireframe Parser [6], our method produces much cleaner wireframe. Specifically, our method works better at suppressing wrong connection between junctions on two aspects. On one hand, the learned distance map guides the establishment of connections between the detected junctions, according to the Least Distance algorithm. On the other hand, the line proposal verification module explicitly suppresses the wrong line proposals.

In terms of line segment detection, our method is close to AFM [12], the state-of-the-art line segment detector. Different from AFM, our network yields both the line segments and the intersections between pairs of them.

V. CONCLUSION
In this paper, we propose an end-to-end wireframe parsing method based on the previous state-of-the-art wireframe parser. The proposed parser consists of an anchor-free junction detection module, a distance map learning module and a line segment proposing and verification module. Line segments are proposed from junctions with guidance of distance map, and further verified by the verification module. The network is end-to-end trainable and efficient. Experimental results show that our method outperforms the previous wireframe parser, and is close to the recent AFM [12] in terms of line segments detection. There is still plenty of room for improvement, and the end-to-end pipeline still can be further optimized. Considering the difficulty of the task and the great variety in the dataset, one future direction is to generate large-scale of synthetic wireframe data to help training better model.

REFERENCES
[1] E. Delage, H. Lee, and A. Y. Ng, “Automatic single-image 3D reconstructions of indoor manhattan world scenes,” in Proc. Int. Symp. Robot. Res., 2005, pp. 305–321.
[2] D. C. Lee, M. Hebert, and T. Kanade, “Geometric reasoning for single image structure recovery,” in Proc. CVPR, Jun. 2009, pp. 2136–2143.
[3] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Manhattan-world stereo,” in Proc. CVPR, Jun. 2009, pp. 1422–1429.
[4] A. Flint, D. Murray, and I. Reid, “Manhattan scene understanding using monocular, stereo, and 3D features,” in Proc. ICCV, Nov. 2011, pp. 2238–2245.
[5] S. Ramalingam, J. K. Pillai, A. Jain, and Y. Taguchi, “Manhattan junction catalogue for spatial reasoning of indoor scenes,” in Proc. CVPR, Jun. 2013, pp. 3056–3072.

[6] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, “Learning to parse wireframes in images of man-made environments,” in Proc. CVPR, Jun. 2018, pp. 626–635.

[7] R. G. von Gioj, J. Jakubowicz, J.-M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 4, pp. 722–732, Apr. 2010.

[8] G.-S. Xia, J. Delon, and Y. Gousseau, “Accurate junction detection and characterization in natural images,” Int. J. Comput. Vis., vol. 106, no. 1, pp. 31–56, Jan. 2014.

[9] A. Elquish and A. Elgammal, “Line-based relative pose estimation,” in Proc. CVPR, Jun. 2011, pp. 3049–3056.

[10] S. Ramalingam and M. Brand, “Lifting 3D manhattan lines from a single panorama,” in Proc. CVPR, Jun. 2016, pp. 5422–5430.

[11] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, “Learning attraction field representation for robust line segment detection,” in Proc. CVPR, Jun. 2019, pp. 1595–1603.

[12] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region-based fully convolutional networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 379–387.

[13] C. G. Harris and M. Stephens, “A combined corner and edge detector,” in Proc. Alvey Vis. Conf., 1988, pp. 1–6.

[14] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, “Using contours to detect and localize junctions in natural images,” in Proc. CVPR, Jun. 2008, pp. 1–8.

[15] J. McDermott, “Psychophysics with junctions in real images,” Perception, vol. 33, no. 9, pp. 1101–1127, 2004.

[16] A. Toshev and C. Szegedy, “DeepPose: Human pose estimation via deep neural networks,” in Proc. CVPR, Jun. 2014, pp. 1653–1660.

[17] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” in Proc. ECCV, 2016, pp. 206–222.

[18] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” in Proc. CVPR, Jun. 2016, pp. 4724–4732.

[19] Y. Zhou, H. Qi, and Y. Ma, “End-to-end wireframe parsing,” 2019, arXiv:1905.03246. [Online]. Available: https://arxiv.org/abs/1905.03246.

[20] M. Brown, D. Windridge, and J.-Y. Guillemaut, “A generalisable framework for saliency-based line segment detection,” Pattern Recognit., vol. 48, no. 12, pp. 3993–4011, Dec. 2015.

[21] I. Toto, C. Cueva, L. Salgado, and N. García, “Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy,” Pattern Anal. Appl., vol. 14, no. 2, pp. 149–163, 2011.

[22] X. Lu, J. Yao, K. Li, and L. Li, “CannyLines: A parameter-free line segment detector,” in Proc. ICIP, Sep. 2015, pp. 507–511.

[23] X. Liu, Z. Cao, N. Gu, S. Nahavandi, C. Zhou, and M. Tan, “Intelligent line segment perception with cortex-like mechanisms,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 45, no. 12, pp. 1522–1534, Dec. 2015.

[24] J. Matas, C. Galambos, and J. Kittler, “Robust detection of lines using the progressive probabilistic Hough transform,” Comput. Vis. Image Understand., vol. 78, no. 1, pp. 119–137, Apr. 2012.

[25] Y. Furukawa and Y. Shinagawa, “Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space,” Comput. Vis. Image Understand., vol. 92, no. 1, pp. 1–25, 2003.

[26] Z. Xu, B.-S. Shin, and R. Klette, “Accurate and robust line segment extraction using minimum entropy with Hough transform,” IEEE Trans. Image Process., vol. 24, no. 3, pp. 813–822, Mar. 2015.

[27] P. Dollár and C. L. Zitnick, “Structured forests for fast edge detection,” in Proc. ICCV, Dec. 2013, pp. 1841–1848.

[28] S. Xie and Z. Tu, “Holistically-nested edge detection,” in Proc. ICCV, Dec. 2015, pp. 1395–1403.

[29] K.-K. Manninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, “Convolutional oriented boundaries,” in Proc. ECCV, 2016, pp. 580–596.

[30] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, “Richer convolutional features for edge detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1939–1946, Aug. 2019.

[31] Z. Zhang, Z. Li, N. Ji, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, “P3GNet: Learning point-pair graph for line segment detection,” in Proc. CVPR, Jun. 2019, pp. 7105–7114.

[32] A. Saxena, S. H. Chang, and A. Y. Ng, “3-D depth reconstruction from a single still image,” Int. J. Comput. Vis., vol. 76, no. 1, pp. 53–69, 2008.