PPGNet: Learning Point-Pair Graph for Line Segment Detection

Ziheng Zhang* Zhengxin Li* Ning Bi Jia Zheng Jinlei Wang Kun Huang
Weixin Luo Yanyu Xu Shenghua Gao†
ShanghaiTech University
{zhangzh, lizhx, bining, zhengjia, wangjinlei, huangkun, luowx, xuyy2, gaoshh}@shanghaitech.edu.cn

Abstract

In this paper, we present a novel framework to detect line segments in man-made environments. Specifically, we propose to describe junctions, line segments and relationships between them with a simple graph, which is more structured and informative than end-point representation used in existing line segment detection methods. In order to extract a line segment graph from an image, we further introduce the PPGNet, a convolutional neural network that directly infers a graph from an image. We evaluate our method on published benchmarks including York Urban and Wireframe datasets. The results demonstrate that our method achieves satisfactory performance and generalizes well on all the benchmarks. The source code of our work is available at `https://github.com/svip-lab/PPGNet`.

1. Introduction

Retrieving 3D information from 2D images has long been a fundamental problem in computer vision. The feasibility of conventional methods based on local feature detection, matching and tracking (e.g., corners, edges, SIFT features, and patches) has been proved. However, modern applications, which involve interaction between autonomous agents and man-made physical environments, have presented more complex challenges. On the one hand, man-made environments often contain abundant homogeneous surfaces and highly repeated patterns, which introduces difficulties for feature matching and tracking. On the other hand, for some applications (e.g., visual odometry), of which the performance highly depends on the geometric primitives presenting in different views, the choice of such primitives (e.g., points, lines segments, or other structures) becomes critical: different primitives provide distinct sets of geometric information.

The prior assumption about a spacial structure like *Manhattan World* [7, 11, 38] or *room topology* [24, 57, 60] could significantly benefit the 3D reconstruction, but they are often violated in real man-made environments. Instead, common junctions and line segments are capable of delivering important geometric information without dependency to any prior assumption. For extensive tasks relevant to 3D vision, such as camera calibration [10,43,56], matching across views [44] and 3D reconstruction [20, 37, 56] , edges have demonstrated more robustness to lighting changes and preserve more information than points. Several recent works [22, 55, 60] show that line segments could largely facilitate 3D modeling of indoor scenes.

Traditional line segment detection algorithms [1,2,27,49] generally start from edge detection, followed by merging procedure and optionally some refinement techniques. However, such approaches are usually sensitive to changes in scale and illumination, since they merely depend on the local features. Additionally, some geometrically informative lines, such as intersections between two homochromatic walls, often have low local edge responses, thus tend to be ignored by such methods. On contrast, a human can easily recognize such visually obscure intersections through global semantic inference.

The recent success of deep learning has shown the desirable capability of image understanding, such as image classification [19, 23, 47, 48], object detection [13, 14, 17, 40], and semantic segmentation [26, 41]. On the other hand, deep architectures are also effective in low-level tasks, such as contour detection [45] and super-resolution [9]. [21] is a pioneer work of extracting *wireframe* in man-made scenes with a deep architecture, for human-level perception of scene geometry. Their proposed network outputs pixel-wise junction confidence and directions together with a line heatmap, followed by a post-processing algorithm merging them to generate a parameterized presentation of line segments. As introduced in the literature, the conception of the *wireframe* is a small subset of common line segments and junctions, which is practically defined by their dataset annotation. Considering that line segments outside the *wireframe* subset also contain strong geometrical information, and line segment detection itself is still a challenging problem in computer

*Contributed equally
†Corresponding author

vision, we focus on robust detection of general line segments.

In this paper, we propose to describe junctions, line segments and the relationship between them with a simple graph. In our graph representation, the nodes stand for vertices and the edges stand for the connectivities between junction pairs, *i.e*. the line segments. The graph is fully capable of describing any complex connections between junctions. Following this, we introduce the *PPGNet*, a novel CNN based architecture which directly infers point-pair graph from given images. Specifically, we firstly use a backbone network for feature extraction, which is utilized to detect junctions. Then, we construct line segment candidate for each junction pair, and reuse the extracted feature to infer the connectivity of the line segment candidate. Consequently, all junctions and their connectivities are formed as a graph, which describes all line segments in the input image. It should be noted that our proposed network can predict a graph directly from a given RGB image .

In order to train our proposed PPGNet, we need a dataset with annotated junctions as well as connectivity between every possible junction pair. However, annotations in existing datasets often ignore some overlapped line segments, thus can not be directly used to train our network. To fix this, we generate more informative graph-based annotations for existing datasets. Further, we also introduce a new large scale line segment dataset containing fully annotated indoor and outdoor samples, which fills the gap of current datasets that are either small in size for training deep architectures or lacking indoor/outdoor samples.

The contribution of this work can be summarized as follow: First, we introduce the new graph-based representation of line segments against commonly used endpoint representation, which is capable to describe all possible line segments in a more structured and informative way; second, we design a novel deep architecture that directly infers the line segment graph from the input image; third, we build a new dataset which covers both indoor and outdoor scenes with fully annotated line segments; fourth, results demonstrate that our method achieves satisfactory performance and generalizes well on multiple datasets.

2. Related Works

2.1. Line segment detection

The mainstream pipeline of hand-crafted line segment detector generally consists of local feature extraction, pixel grouping, and optional refinement. These methods usually start from detecting pixels with high local gradient and/or edge response and then group them into line segments through iterative growing [35], co-linear clustering [27], Hough domain accumulation [12,30,54] or Markov chain [2], *etc*. The line segments are optionally refined with false detection control based on the Helmholtz principle [1, 49], as well as fragment merging and endpoint relocation [4].

The hand-crafted line segment detectors highly depend on carefully designated parameters. Even though some of them are parameter-free, the results still are highly sensitive to the choice of threshold. In a recent research [21], a CNN including two branches is proposed to parse a junction map and a line heatmap from an image, which are then merged into a set of line segments. This learning-based approach outperforms the hand-crafted methods with a large margin. Nonetheless, there is not a framework that directly outputs a parameterized presentation of line segments by far.

2.2. Junction detection

Although junction detection has been studied for long [15, 42], it remains a challenging problem. A typical work is to compute local *cornerness* based on so-called Harris matrix [16], which is, however, sensitive to scale and localization. Some works focus on contour curvature or continuation for detecting junctions [3, 5, 32]. Other works exploit the consistency between textures and gradient-based [6, 46] or pattern-based [36, 51] templates as an effective cue for junction detection.

According to a psychophysical analysis, it is difficult to recognize junctions without context information in a large enough area, even for humans [31]. In this direction, [28] attains robust edge and junction detection by combining local cues (e.g., lightness, color and gradient) with a *global probability of boundary* (gPb) detector, which is learned from human-annotated data. Benefiting from the large receptive field of the deep neural network, [21] achieves state of the art performance on junction detection.

2.3. CNN-based Graph Inference

A convolutional neural network is capable of inferring graphs from images. In [34], the multi-person pose estimation problem is resolved by considering each person as a graph and grouping the body joints with associative embedding. As a more general work, a CNN is trained to detect all the objects and relationships between them in an image by means of associative embedding [33]. The objects and relationships in a scene graph can be further refined with a gated recurrent unit (GRU) [53]. However, their network only outputs nodes and edges, together with embedding that associates edges to nodes, therefore extra steps are required to construct the final graph. Furthermore, their framework cannot handle arbitrary overlapped edges. In contrast, our network can infer an arbitrary simple graph parameterized by nodes and an adjacency matrix directly from the input.

3. PPGNet for Line Segment Detection

3.1. Junction-line Graph Representation

Here, we consider the problem of detecting line segments directly from an RGB image. We propose to use a simple graph $G^n = \{V^n, E^n\}$ to represent all line segments in a given image X^n (a sample indexed by n in a dataset), of which V^n stands for the set of junctions and E^n for the set of connectivities between any junction pairs. We now transfer the original line segment detection problem into the graph inference problem. In our implementation, V^n and E^n are parameterized using an ordered list of junctions J^n and an adjacency matrix A^n, respectively. Hence, each element in J^n is the coordinate of the junction and the entry A^n_{ij} in the i-th row and j-th column of matrix A^n equals one only if junction pair J^n_i and J^n_j forms a line segment.

The graph representation is more structured than endpoint representation for line segments. Line segments that share the same endpoints only add more ones to the adjacency matrix without introducing extra terms. Besides, graph representation is also much more informative. Connectivities between junctions are fully described in a combinational way (Fig.1) and both longer line segments and any inner shorter ones are depicted by the graph, which benefits the selection of befitting line segments from the graph in accordance with specific applications.

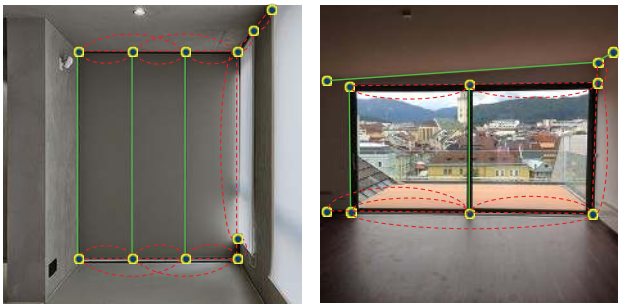


Figure 1. Some cases where junctions are densely connected. The connectivities among junctions, as indicated by red dashed curves, can be more completely identified in graph representation than end-point representation. The green lines are simple line segments without inner junctions.

In this work, we use a deep neural network to learn the mapping from RGB image X to graph G. Due to the fact that G fully describes all line segments in X and can be transferred to endpoint representation with minor efforts, our method is a unified solution, although containing multiple stages, to settle line segment detection problem.

3.2. PPGNet

Motivated by Faster R-CNN [40], we propose a two-staged framework that detects junction points at the first stage and then identifies the connectivities between all point pairs at the second stage. The proposed PPGNet, as illustrated in Fig.2, comprises four parts: (i) a convolutional backbone architecture for feature extraction over the entire input image, (ii) the Junction Detection Module (JDM), (iii) the Line Segment Alignment Module (LSAM) which extracts a feature tensor for the line segment candidate defined by a pair of detected junctions, and (iv) the Adjacency Matrix Inference Module (AMIM) which detects the connectivity between each junction pair. Given an image, our network predicts both junction locations and their connectivities represented by an adjacency matrix.

3.2.1 Backbone Network

We use the semantic segmentation network implemented by CSAIL [52, 58, 59] as our backbone network, which consists of a dilated ResNet-50 encoder and a decoder with pyramid pooling, except for the last convolution layer, of which the number of output channels C is changed to be 256 instead of 1. [1] For an input image of size $H \times W$, the backbone network extracts a 256-channel feature map of size $H/4 \times W/4$.

3.2.2 Junction Detection Module

The JDM extracts junctions over the input image represented by their coordinates. Unlike commonly used anchor based detection methods such as R-CNN [40], YOLO [39] or SSD [25], the JDM first regresses a junction heatmap, then applies Local Maximum Filter (LMF) to get coordinates where junction response is higher than its eight neighbors. Non-maximum Suppression (NMS) is also used to avoid multiple detections of the same junctions. Unlike that in detection methods, the NMS in JDM is implemented by a hierarchical clustering using the single linkage algorithm, where the clusters are formed by the inconsistency method with a cutoff threshold (3 pixels in all our experiments).

In detail, the JDM first regresses junction heatmap from the feature extracted by the backbone network through a convolutional architecture, which comprises two conv3x3-bn-relu blocks followed by a conv1x1 layer with sigmoid activation. Then it identifies all points in the heatmap where junction responses are higher than a threshold τ and are the highest among 8-neighbors. After that, the detected points are clustered into groups, within which the distance between arbitrary two points is no greater than ϵ, and all the points with the highest junction responses in their groups are predicted as junction points. We use $\epsilon = 3$ pixels in all our experiments.

[1] The details about the backbone can be found in the Github page `https://github.com/CSAILVision/semantic-segmentation-pytorch`

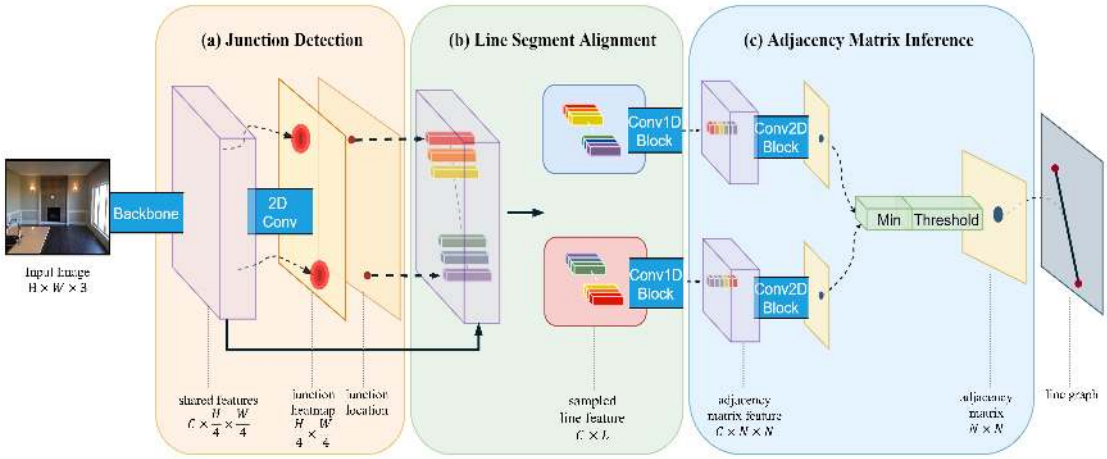


Figure 2. The **PPGNet** architecture. First, the backbone computes shared features of size $C \times \frac{H}{4} \times \frac{W}{4}$ for Junction detection and adjacency matrix inference. Second, the Junction Detection Module output a list of N junctions. Third, each junction pair is formed as two line segment candidates of different directions, over which features are evenly sampled into two feature matrix of size $C \times L$. After that, we apply 1D convolution over each feature matrix, which outputs a feature vector of size C. Fourth, each feature vector is used by the Adjacency Matrix Inference Module to infer the connectivity of the corresponding junction pairs.

3.2.3 Line Segment Alignment Module

Given two junctions and a feature map, the LSAM samples the feature map along the line segment candidate defined by the junction pair, and extracts a fixed-length feature vector from the feature map. LSAM works in a way similar as ROI Align Module [17], except that LSAM aligns feature vectors instead of patches.

For each junction pair and feature map of an image, the LSAM generates a feature tensor of size $C \times L$, where C is the number of channels of feature map and L is the spatial length of line segment feature. Specifically, LSAM first generates L equidistant sampling points from the starting point to the end point of the junction pair, then uses bilinear interpolation to sample pixel value for each point on the feature map. In our main model, L is set to be 64, so that each junction pair yields a feature tensor of size $C \times 64$ for connectivity inference.

3.2.4 Adjacency Matrix Inference Module

The AMIM predicts the connectivity of every combination of junction pairs within an image. It takes the features for all line segment candidates provided by the LSAM, and uses a convolutional structure to predict the connection probability for each candidate.

Given K junctions predicted by JDM, AMIM generates a $K \times K$ adjacency matrix A, by which the line segment detection problem is turned into a binary classification problem of whether two junctions are connected. For every possible junction pair, two feature vectors of a line segment corresponding to different junction orders are extracted by LSAM, which are then fed into three cascaded *conv2d-gn-relu* blocks, where *gn* represents the Group Normalization Layer [50]. The kernel sizes, stride size and padding size of the three convolution layers are $8, 4, 2$, respectively. After that, a single *conv2d-sigmoid* block is used to get the connectivity confidence of the junction pair in different orders, of which the lowest becomes the final confidence for the junction pair. Intuitively, this processing acts as an 'and' logic to ensure that a junction pair is connected regardless of the order of feature concatenation.

In practice, because JDM can detect an arbitrary number of junctions, AMIM predicts one block of matrix A of fixed size 64×64 at a time and runs multiple times to get the whole adjacency matrix. Furthermore, as all possible connectivities between all junction pairs are inspected in AMIM, which causes an $O(n^2)$ complexity, it is impractical to process a too large number of junctions in AMIM. Due to an observation that JDM tends to assign a higher score to the junctions associated with more line segments, we only choose the first 512 junctions with the highest responses on the heatmap when more than 512 junctions are outputted by JDM. In our experiments, it takes about 0.9s to process an image containing 512 junctions with a Tesla P40 GPU.

3.2.5 Loss Function

Both junction heatmap and adjacency matrix are supervised using binary cross entropy loss, and the final loss is the

Table 1. Dataset statistics

	# images	resolution	# avg. junc.	# avg. lines	scenes\line types
Wireframe	5462	$480 * 405(avg.)$	150	75	indoor*wireframe*
York Urban	102	$640 * 480$	209	119	both\Manhattan
Ours-indoor	1378	$900 * 1200$	67	41	indoor\general
Ours-outdoor	2534	$2048 * 1080$	537	311	outdoor\general

weighted sum of two losses, *i.e.*

$$\mathcal{L} = \lambda_{junc}\mathcal{L}_{junc} + \lambda_{adj}\mathcal{L}_{adj}$$
$$\mathcal{L}_{junc} = -\sum\nolimits_i \tilde{H}_i \log H_i + (1 - \tilde{H}_i) \log (1 - H_i)$$
$$\mathcal{L}_{adj} = -\sum\nolimits_i \tilde{A}_i \log A_i + (1 - \tilde{A}_i) \log (1 - A_i)$$

where $\tilde{H}_i$ and H_i are the elements of prediction and ground truth of junctions, respectively, and $\tilde{A}_i$ and A_i are the elements of prediction and ground truth of the adjacency matrix, respectively. BCE stands for the cross en We set $\lambda_{junc} = \lambda_{adj} = 1$ in all our experiments.

3.3. Training and Evaluating Details

All modules are jointly optimized using Stochastic Gradient Decent (SGD), with $lr = 0.2$, $weight\_decay = 5 \times 10^{-4}$, and $momentum = 0.9$, except for all normalization layers, of which $weight\_decay$ is set to zero. The backbone network is initialized with parameters pretrained for segmentation task on the MIT ADE20K dataset, and other modules are initialized with *kaiming initialization* [18], as the common practice. During the training phase, AMIM infers adjacency matrix for ground truth junctions instead of junctions predicted by JDM because we do not have corresponding ground truth adjacency matrix for supervision. During evaluating phase, junctions and adjacency matrix are jointly estimated by our PPGNet.

4. Experiments and Results

We conduct experiments to evaluate the performance of our proposed approach and compare it with several SOTA methods. Our model is implemented with the Pytorch framework trained with four Tesla M40 GPUs.

4.1. Datasets and Evaluation Metrics

Experimental Datasets So far as we know, there exist two line segment datasets namely Wireframe [21] and York Urban [8]. However, the former only has *wireframes* line segments in mostly indoor scenes annotated, while the later, though containing both indoor and outdoor scenes, is small in size (102 samples), and only has Manhattan lines labeled. In order to validate the capability of our framework to detect general line segments for new indoor and outdoor scenes, we build a new line segment dataset, which consists of 1,378 indoor images and 2,534 outdoor images, together with carefully labeled line segments.

For the indoor part, we capture the images with resolution 900×1200 using a camera array that comprises seven GoPro cameras. For the outdoor part, we take aerial videos of our campus with a 4K camera equipped on a DJI UAV, and extract frames with high quality at the interval of at least two seconds. Since the resolution of original videos is too large for labeling and training neural networks, we further crop each frame into four $2048 * 1080$ images.

All the line segments in both indoor and outdoor part are annotated following the protocols that any visible line segments that are longer than 10% of the image diagonal and occluded by less than 10% of their length is labeled. Every sample is annotated by one volunteer and then double checked by another. Different from the conception of *wireframe* [21] or Manhattan lines, the labeled line segments in our dataset only need to be visible and geometrically informative. Table 1 summarizes the statistics of existing datasets and ours.

Data Preprocessing In order to learn the mapping from image to line segment graph, the complete description of connectivities among all junctions is required. However, all existing datasets use endpoints to represent line segments, where both junctions and connectivities can be missing in some cases. Therefore, we introduce the data preparation scheme to convert each annotation in the original datasets into their graph representation version, which can be outlined as follows.

1. Remove isolate junctions associated with no line segments.
2. Find the longest line segments by searching the furthest endpoint junctions for every line segment, then mark all junctions in a longest line segment as connected. Note that the searching is not exactly along the line, but within a belt around the line, due to possible annotation error.
3. Remove a line segment if all its inner junctions is a subset of those of another line segment. The inner junctions of a line segment are determined by their distance to the line segment.

Figure 3. Qualitative evaluation of our line segment detection method. 1st row: ground truth (Wireframe); 2nd row: prediction (Wireframe); 3rd row: ground truth (York Urban); 4th row: prediction (York Urban); 5th row: ground truth (Our dataset); 6th row: prediction (Our dataset).

4. Refine each line segment by re-fitting it to its inner junctions.
5. Refine all of the junctions that are the intersections of two or more line segments by solving the linear equations imposed by the corresponding line segments.
6. Retrieve possibly missing junctions by finding all intersections of all line segment pairs.
7. Construct the final line segment graph, which is parameterized by an ordered list of junctions and an adjacency matrix.

Generally, the data preparation scheme tries to correct some bad annotations and to supplement possibly missing junctions and connectivities. We refer readers to our released code for the details of the scheme.

Evaluation Metric We quantitatively evaluate the methods using *recall* and *precision* as described in [21, 28, 29, 51]. The *recall* is the fraction of true line segment pixels that are detected, whereas the *precision* is the fraction of detected line segment pixels that are indeed true positive. Specifically, they are calculated as follows:

$$\text{Recall} \doteq |G \cap Q|/|G|, \quad \text{Precision} \doteq |G \cap Q|/|Q|,$$

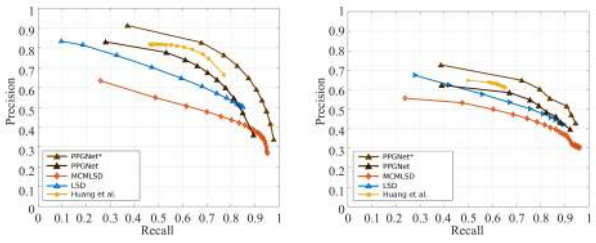
where G denotes the ground truth and Q denotes the prediction. Note that, following the protocols of previous works [28, 29, 51], the particular measures of recall and precision allow for some small tolerance in the localization of line segment pixels. The tolerance in our experiments is set to be 0.01 of the image diagonal, which is the same as [21].

4.2. Performance evaluation

In order to evaluate the performance of our framework, we compared the performance of our method with the state of the art (LSD [49], MCMLSD [2] and *wireframe* parser of Huang *et al*.. [21]) upon three experiment settings: (a) training and testing on Wireframe dataset using the standard splits; (b) training on Wireframe dataset and testing on the York-Urban dataset; (c) training on Wireframe dataset and

testing on our dataset. Since the Wireframe and York Urban datasets are released benchmarks, we do quantitative comparison under the setting (a) and (b). Qualitative evaluation is done under the third setting to observe the generalizing capability between different data distribution.

Quantitative Comparison For our PPGNet, we conduct two experiments under setting (a) and (b) where (i) AMIM uses junctions predicted by JDM (marked as *PPGNet*) and where (ii) AMIM uses ground truth junctions to predict line segments (marked as *PPGNet**). We include the second experiment for the reason that both the two benchmarks only have a subset of line segments annotated, *i.e. wireframes* and Manhattan lines, but our framework is for general line segment detection. Only with ground truth junctions can our framework understand which type of junctions should be considered.



(a) (b)

Figure 4. Precision-Recall curves of our PPGNet and state of the art methods evaluated on (a) Wireframe dataset and (b) York-Urban dataset. For *PPGNet* and *PPGNet**, we set the threshold for JDM to 0.25, and vary the threshold for AMIM in $[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7]$.

As one can see, though our PPGNet shows worse performance compared to [21] in experiment(i), it achieves superior performance in experiment(ii). In a comprehensive view, our method achieves satisfactory performance.

Qualitative Analysis Fig. 3 illustrates visualized results of line segment detection of our method on several (random) samples. It can be seen that our method is capable to robustly detect the line segments in complicated environments, and generalize well on datasets on which it has not been trained.

In order to qualitatively compare the performance of our method with that of *wireframe* parser [21] on general line segments, we refine the annotations of the test split in Wireframe dataset by adding the missing general line segments apart from wireframes. Fig.5 shows the visualized results on some randomly picked samples for PPGNet and the method in [21]. It can be observed that our method is capable of retrieving more abundant line segment information than [21].

We also noticed that out model fails in some cases, as shown in Fig. 6. There are two typical cases: (1) for small boxes in an image, our model tends to predict the diagonal junctions as connected and (2) for close co-linear line segments, our model tends to ignore the gaps and predict all junctions as connected. These cases may be caused by the sampling procedure in AMIM. For case (1), the bilinear sampling may introduce nearby features around sampled locations. Hence the nearby junctions may interfere with connectivity prediction of current junction. On the other hand, case (2) may happen when few or no sampling points fall in the gaps between those co-linear line segments, which prevents AMIM from recognizing the discontinuity.

4.3. Junction Threshold of JDM

In our framework, the AMIM predict connectivities of junctions detected by JDM, for which the threshold τ has a fundamental effect on the performance. We comprehensively compare the performance under different choices of the value for τ. It can be seen in Fig.7 that $\tau \in [0.2, 0.3]$ lead to a better precision-recall curves. According to the quantitative evaluation in AUC, $\tau = 0.25$ is slightly better than $\tau = 0.2$ and $\tau = 0.3$.

4.4. Sampling Rate of LSAM

LSAM predicts the connectivity of junction pairs from the spatially sampled features between the two junctions. Therefore the sampling rate of LSAM has a fundamental effect on the performance of our model. In order to analyze the effect of sampling rate on performance, we conduct experiments in which different sampling rate is used in LSAM. The results are shown in Fig. 8.(a), we can see that LSAM benefits from higher sampling rates. However, higher sampling rates also introduce extra memory usage and computational cost. Hence one should consider both performance and efficiency requirements for different applications when choosing the sampling rate.

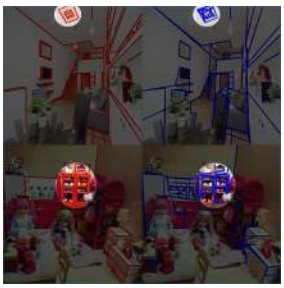
As an extreme case, the sampling rate equals 2 means that only features at the location of junction pairs are sampled. In this case, LSAM suffers from insufficient information to determine the connectivities of junction pairs. Fig.8.(a) shows the typical results when only two points are sampled. As one can see, LSAM fails to recognize gaps between two co-linear line segments and directions of line segments starting from the same junctions.

5. Conclusion and Disscusion

In this paper, we propose to use graphs to represent all line segments in a given image and introduce the PPGNet, an multi-staged deep architecture that directly infers a graph directly from an image. Our method achieves satisfactory performance on multiple public benchmarks and shows remarkable generalization ability.

There is still room for improvement in our framework. For example, currently the LSAM predicts connectivities for all

Figure 5. Qualitative results on the refined *Wireframe* dataset. 1st row: the ground truth; 2nd row: results of the method proposed in [21]; 3rd row:results of *PPGNet*.



(a)

(b)

Figure 6. Failure cases: (a) image samples that contain small rectangulars; (b) image samples that contain very close co-linear line segments.

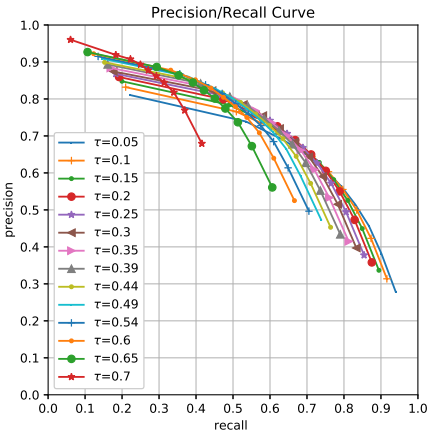
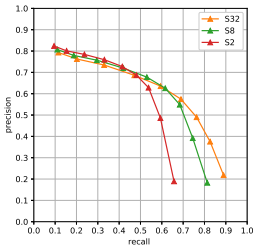


Figure 7. Precision-recall curves for different choices of JDM threshold τ.

possible line segments, which yields the time complexity of $O(n^2)$. Maybe one could filter some line segment candidates according to the specific applications, but there may exists a better way to further reduce the computational cost.



(a)

(b)

Figure 8. Illustrations of (a) precision-recall curves with different sampling rate of LSAM; (b) example prediction when only junction features are pooled.

On the other hand, PPGNet itself is a general framework to infer a graph from an image. In principle, PPGNet could also be used to solve other problems that need to detect visual parts and their spatial connections. Human pose estimation is a typical example of such problems, and we are interested in exploiting possible applications of PPGNet for such problems in future work.

References

[1] Cuneyt Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.

[2] Emilio J Almazan, Ron Tal, Yiming Qian, and James H Elder. Mcmlsd: A dynamic programming approach to line segment detection. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5854–5862. IEEE, 2017.

[3] Mohammad Awrangjeb and Guojun Lu. An improved curvature scale-space corner detector and a robust corner matching approach for transformed image identification. *IEEE Transactions on Image Processing*, 17(12):2425, 2008.

[4] Mark Brown, David Windridge, and Jean-Yves Guillemaut. A generalisable framework for saliency-based line segment detection. *Pattern Recognition*, 48(12):3993–4011, 2015.

[5] Frédéric Cao. Good continuations in digital image level lines. In *ICCV*, volume 1, pages 440–448, 2003.

[6] Miguel A Cazorla and Francisco Escolano. Two bayesian methods for junction classification. *IEEE Transactions on Image Processing*, 12(3):317–327, 2003.

[7] Erick Delage, Honglak Lee, and Andrew Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. *Isrr*, 28:305–321, 2007.

[8] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European conference on computer vision*, pages 197–210. Springer, 2008.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[10] Ali Elqursh and Ahmed Elgammal. Line-based relative pose estimation. In *CVPR 2011*, pages 3049–3056. IEEE, 2011.

[11] Alex Flint, David Murray, and Ian Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *IEEE International Conference on Computer Vision*, pages 2228–2235, 2011.

[12] Yasutaka Furukawa and Yoshihisa Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in hough space. *Computer Vision and Image Understanding*, 92(1):1–25, 2003.

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[15] Marsha J Hannah. Computer matching of areas in stereo images. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1974.

[16] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1026–1034. IEEE, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017.

[21] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 626–635, 2018.

[22] Pyojin Kim, Brian Coltin, and H Jin Kim. Indoor rgb-d compass from a single line and plane. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4673–4680, 2018.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 4875–4884. IEEE, 2017.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[27] Xiaohu Lu, Jian Yao, Kai Li, and Li Li. Cannylines: A parameter-free line segment detector. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 507–511. IEEE, 2015.

[28] Michael Maire, Pablo Arbelaez, Charless Fowlkes, and Jitendra Malik. Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[29] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

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

[31] Josh McDermott. Psychophysics with junctions in real images. *Perception*, 33(9):1101–1127, 2004.

[32] Farzin Mokhtarian and Riku Suomela. Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1376–1381, 1998.

[33] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, pages 2171–2180, 2017.

[34] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[35] Marcos Nieto, Carlos Cuevas, Luis Salgado, and Narciso García. Line segment detection using weighted mean shift procedures on a 2d slice sampling strategy. *Pattern Analysis and Applications*, 14(2):149–163, 2011.

[36] Laxmi Parida, Davi Geiger, and Robert Hummel. Junctions: Detection, classification, and reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):687–698, 1998.

[37] Pietro Parodi and Giulia Piccioli. 3d shape reconstruction by using vanishing points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):211–217, 1996.

[38] Srikumar Ramalingam, Jaishanker K Pillai, Arpit Jain, and Yuichi Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3065–3072, 2013.

[39] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[42] Azriel Rosenfeld. Angle detection on digital curves. *IEEE Transactions on Computer*, 22:243–258, 1973.

[43] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Robust and accurate line-and/or point-based pose estimation without manhattan assumptions. In *European Conference on Computer Vision*, pages 801–818. Springer, 2016.

[44] Cordelia Schmid and Andrew Zisserman. Automatic line matching across views. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 666–671. IEEE, 1997.

[45] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991, 2015.

[46] Eric D Sinzinger. A model-based approach to junction detection using radial energy. *Pattern Recognition*, 41(2):494–505, 2008.

[47] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2017.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[49] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2010.

[50] Yuxin Wu and Kaiming He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[51] Gui-Song Xia, Julie Delon, and Yann Gousseau. Accurate junction detection and characterization in natural images. *International journal of computer vision*, 106(1):31–56, 2014.

[52] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision*. Springer, 2018.

[53] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5419, 2017.

[54] Zezhong Xu, Bok-Suk Shin, and Reinhard Klette. Accurate and robust line segment extraction using minimum entropy with hough transform. *IEEE Transactions on Image Processing*, 24(3):813–822, 2015.

[55] Yang Yang, Shi Jin, Ruiyang Liu, Sing Bing Kang, and Jingyi Yu. Automatic 3d indoor scene modeling from single panorama. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3926–3934, 2018.

[56] Lilian Zhang and Reinhard Koch. Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation*, 25(5):904–915, 2014.

[57] Hao Zhao, Ming Lu, Anbang Yao, Yiwen Guo, Yurong Chen, and Li Zhang. Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 870–878, 2017.

[58] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.

[59] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[60] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018.