Deep Hough Transform for Semantic Line Detection

Qi Han* , Kai Zhao * , Jun Xu , and Ming-Ming Cheng†

TKLNDST, CS, Nankai University
https://mmcheng.net/dhtline/

Abstract. In this paper, we put forward a simple yet effective method to detect meaningful straight lines, a.k.a. semantic lines, in given scenes. Prior methods take line detection as a special case of object detection, while neglect the inherent characteristics of lines, leading to less efficient and suboptimal results. We propose a one-shot end-to-end framework by incorporating the classical Hough transform into deeply learned representations. By parameterizing lines with slopes and biases, we perform Hough transform to translate deep representations to the parametric space and then directly detect lines in the parametric space. More concretely, we aggregate features along candidate lines on the feature map plane and then assign the aggregated features to corresponding locations in the parametric domain. Consequently, the problem of detecting semantic lines in the spatial domain is transformed to spotting individual points in the parametric domain, making the post-processing steps, *i.e.* non-maximal suppression, more efficient. Furthermore, our method makes it easy to extract contextual line features, that are critical to accurate line detection. Experimental results on a public dataset demonstrate the advantages of our method over state-of-the-arts. Codes are available at the project page.

Keywords: Straight line detection, Hough transform, CNN

1 Introduction

We investigate an interesting problem of detecting meaningful straight lines in natural scenes. This kind of line structure which outlines the conceptual structure of images is referred to as 'semantic line' in a recent study [29]. The organization of such line structure is an early yet important step in the transformation of the visual signal into useful intermediate concepts for visual interpretation [5]. As demonstrated in Fig. 1, semantic lines belong to a special kind of line structure, which outlines the global structure of the image. Identifying such semantic lines is of crucial importance for applications such as photographic composition [31] and artistic creation [27].

*Equal contribution
†Corresponding author

(a) (b) (c) (d)

Fig. 1. Example pictures from [31] reveals that semantic lines may help in photographic composition. (a): a photo taken with arbitrary pose. (b): a photo fits the golden ratio principle [7,27] which abtained by the method described in[31] using so-called 'prominent lines' and salient objects [18,14,13] in the image. (c): Our detection result is clean and comprises only few meaningful lines that are potentially helpful in photographic composition. (d): Line detection result by the classical line detection algorithms often focus on fine detailed straight edges.

The research of detecting line structures (straight lines and line segments) dates back to the very early stage of computer vision. Originally described in [22], the Hough transform (HT) is invented to detect straight lines from bubble chamber photographs. The core idea of the Hough transform is translating the problem of pattern detection in point samples to detecting peaks in the parametric space. This idea is quickly extended [11] to the computer vision community for digital image analysis, and generalized by [3] to detect complex shapes. In the case of line detection, Hough transform collects line evidence from a given edge map and then votes the evidence into the parametric space, thus converting the global line detection problem into a peak response detection problem. Classical Hough transform based methods [16,44,35,26] usually detect continuous straight edges while neglecting the semantics in line structures. Moreover, these methods are very sensitive to light changes and occlusion. Consequently, the results are noisy [2] and often contain irrelevant lines, as shown in Fig. 1(d). In this paper, our mission is to only detect clean, meaningful and outstanding lines, as shown in Fig. 1(c), which is helpful to photographic composition.

Since the success of Convolutional Neural Networks (CNNs) in vast computer vision applications, several recent studies take line detection as a special case of object detection, and adopt existing CNN-based object detectors, *e.g.* faster R-CNN [38] and CornerNet [28], for line detection. Lee *et al.* [29] make several modifications to the faster R-CNN [38] framework for semantic line detection. In their method, proposals are in the form of lines instead of bounding boxes, and features are aggregated along straight lines instead of rectangular areas. Zhang *et al.* [46] adopt the idea from CornetNet, which identifies object locations by detecting a pair of key points, *e.g.* the top-left and bottom-right corners. [46] detects line segments by localizing two corresponding endpoints. Limited by the ROI pooling and non-maximal suppression of lines, both [29] and [46] are less efficient in terms of running time. Moreover, ROI pooling [19] aggregates features along a single line, while many recent studies reveal that richer context information is critical to many tasks [17,21], *e.g.* video classification [42], edge

detection [33], salient object detection [4,15], and semantic segmentation [23]. In Tab. 3, we experimentally verify that only aggregating features along a single line leads to suboptimal results.

In this paper, we propose to incorporate CNNs with Hough transform for straight line detection in natural images. We firstly extract pixel-wise representations with a CNN-based encoder, and then perform Hough transform on the deep representations to convert representations from feature space into parametric space. Then the global line detection problem is converted into simply detecting peak response in the transformed features, making the problem simpler. For example, the time-consuming non-maximal suppression (NMS) is simply calculating the centroids of connected areas in the parametric space, making our method very efficient that can detect lines in real-time. Moreover, in the detection stage, we use several convolutional layers on top of the transformed features to aggregate context-aware features of nearby lines. Consequently, the final decision is made upon not only features of a single line, but also information of lines nearby.

In addition to the proposed method, we introduce a principled metric to assess the agreement of a detected line w.r.t its corresponding ground-truth line. Although [29] has proposed an evaluation metric that uses intersection areas to measure the similarity between a pair of lines, this measurement may lead to ambiguous and misleading results. The contributions are summarized below:

- We propose an end-to-end framework for incorporating the feature learning capacity of CNN with Hough transform, resulting in an efficient real-time solution for semantic line detection.
- We introduce a principled metric which measures the similarity between two lines. Compared with previous IoU based metric [29], our metric has straightforward interpretation without ambiguity in implementation, as detailed in Sec. 4.
- Evaluation results on an open benchmark demonstrate that our method outperforms prior arts with a significant margin.

2 Related Work

The research of line detection in digital images dates back to the very early stage of computer vision. Since the majority of line detection methods are based on the Hough transform [11], we first brief the Hough transform, and then summarize several early methods for line detection using Hough transform. Finally, we describe two recently proposed CNN-based methods for line/segments detection from natural images.

Hough based line detectors. Hough transform (HT) is originally devised by Hough [22] to detect straight lines from bubble chamber photographs. The algorithm is then extended [11] and generalized [3] to localize arbitrary shapes, *e.g.* ellipses and circles, from digital images. Traditional line detectors start by edge detection in an image, typically with the Canny [6] and Sobel [40] operators.

Then the next step is to apply the Hough transform and finally detect lines by picking peak response in the transformed space. HT collects edge response alone a line and accumulates them to a single point in the parametric space.

There are many variants of Hough transform (HT) trying to remedy different shortcomings of the original algorithm. The original HT maps each image point to all points in the parameter space, resulting in a many-to-many voting scheme. Consequently, the original HT presents high computational cost, especially when dealing with large-size images. Kiryati *et al.* [26] try to accelerate HT by proposing the 'probabilistic Hough transform' that randomly picks sample points from a line. Princen *et al.* [35] and Yacoub and Jolion [44] partition the input image into hierarchical image patches, and then apply HT independently to these patches. Fernandes *et al.* [16] use an oriented elliptical-Gaussian kernel to cast votes for only a few lines in the parameter space. Illingworth *et al.* [24] use a 'coarse to fine' accumulation and search strategy to identify significant peaks in the Hough parametric spaces. [1] approaches line detection within a regularized framework, to suppress the effect of noise and clutter corresponding to image features which are not linear. It's worth noting that a clean input edge map is critical to these HT-based detectors.

Line segments detection. Despite its robustness and parallelism, Hough transform cannot directly be used for line segments detection because the outputs of HT are infinite long lines. In addition to Hough transform, many other studies have been developed to detect line segments. Burns *et al.* [5] use the edge orientation as the guide for line extraction. The main advantage is that the orientation of the gradients can help to discover low-contrast lines. Etemadi *et al.* [12] establish a chain from the given edge map, and then extract line segments and orientations by walking over these chains. Chan *et al.* [8] use quantized edge orientation to search and merge short line segments.

CNN-based line detectors. There are two CNN-based line (segment) detectors that are closely related to our method. Lee *et al.* [29] regard line detection as a special case of object detection, and adopt the faster R-CNN [38] framework for line detection. Given an input image and predefined line proposals, they first extract spatial feature maps with an encoder network, and then extract line-wise feature vectors by uniformly sampling and pooling along line proposals on the feature maps. A classification network and a regression network are applied to the extracted feature vectors to identify positive lines and adjust the line positions. Zhang *et al.* [46] adopt the CornerNet [28] framework to extract line segments as a pair of key points. Both of the methods as mentioned above extract line-wise feature vectors by aggregating deep features solely along each line, leading to inadequate context information. Besides, there are many works [36,37] using the conception of Hough voting in 3D object detection.

3 Deep Hough Transform for Line Detection

Our method comprises the following four major components: 1) a CNN encoder that extracts pixel-wise deep representations; 2) the deep Hough transform

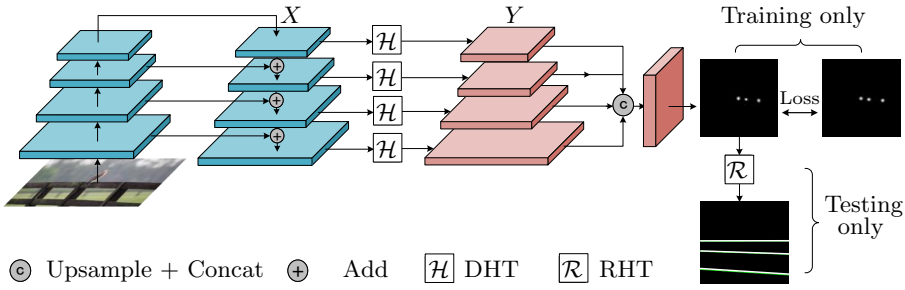

Fig. 2. Pipeline of our proposed method. DHT is short for the proposed Deep Hough Transform, and RHT represents the Reverse Hough Transform.

(DHT) that converts the spatial representations to a parametric space; 3) the line detector that is responsible to detect lines in the parametric space, and 4) a reverse Hough transform (RHT) that converts the detected lines back to image space. All these components are unified in a framework that performs forward inference and backward training in an end-to-end manner.

3.1 Line Parameterization and Reverse

In the 2D case, all straight lines can be parameterized with two parameters: an orientation parameter and a distance parameter. As shown in Fig. 3(a), given a 2D image $I_{W\times H}$ where H and W are the spatial size, we set the origin to the center of the image. Then a line l can be parameterized with r_l and $\theta_l \in [0, \pi)$, representing the distance between l and the origin, and the angle between l and the x-axis, respectively. Obviously $\forall\ l \in I, r_l \in [-\sqrt{W^2+H^2}/2, \sqrt{W^2+H^2}/2]$.

Given any line l from I, we can parameterize it with the above formulations, and also we can perform a reverse mapping to translate any (r, θ) pair to a line instance. Formally, we define the line parameterization and reverse as:

$$\begin{aligned} r_l, \theta_l &= P(l), \\ l &= P^{-1}(r_l, \theta_l). \end{aligned} \tag{1}$$

Obviously, both P and P^{-1} are bijective functions. In practice, r and θ are quantized to discrete bins to be processed by computer programs. Suppose the quantization interval for r and θ are Δr and $\Delta\theta$, respectively. Then the quantization can be formulated as below:

$$\hat{r}_l = \left\lceil \frac{r_l}{\Delta r} \right\rceil,\ \hat{\theta}_l = \left\lceil \frac{\theta_l}{\Delta\theta} \right\rceil, \tag{2}$$

where $\hat{r}_l$ and $\hat{\theta}_l$ are the quantized line parameters. The number of quantization levels, donoted with Θ and R, are:

$$\Theta = \frac{\pi}{\Delta\theta},\ R = \frac{\sqrt{W^2+H^2}}{\Delta r}, \tag{3}$$

as shown in Fig. 3(b).

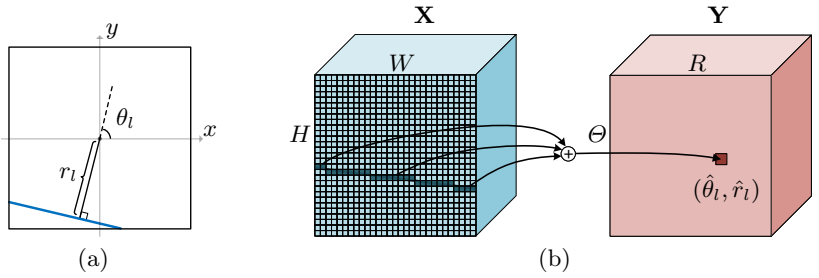

Fig. 3. (a): A line is parameterized by r_l and θ_l; (b): Features along a line in the feature space (left) are accumulated to a point $(\hat{r}_l, \hat{\theta}_l)$ in the parametric space (right).

3.2 Feature Transformation with Deep Hough Transform

Deep Hough transform. Given an input image I, we first extract deep CNN features $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$ with the encoder network, where C indicates the number of channels and H and W are the spatial size. Afterward, the deep Hough transform (DHT) takes $\mathbf{X}$ as input and produces the transformed features, $\mathbf{Y} \in \mathbb{R}^{C \times \Theta \times R}$. The size of transformed features, Θ, R, are determined by the quantization intervals, as described in Eq. (3).

As shown in Fig. 3(b), given a line $l \in \mathbf{X}$ in the feature space, we accumulate features of all pixels along l, to $(\hat{\theta}_l, \hat{r}_l)$ in the parametric space Y:

$$\mathbf{Y}(\hat{\theta}_l, \hat{r}_l) = \sum_{i \in l} \mathbf{X}(i), \tag{4}$$

where i is the positional index. $\hat{\theta}_l$ and $\hat{r}_l$ are determined by the parameters of line l, according to Eq. (1), and then quantized into discrete grids according to Eq. (2).

The DHT is applied to all unique lines in an image. These lines are obtained by connecting an arbitrary pair of pixels on the edges of an image, and then excluding the duplicated lines. It is worth noting that DHT is order-agnostic in both the feature space and the parametric space, making it highly parallelizable.

Multi-scale DHT with FPN. Our proposed DHT could be easily applied to arbitrary spatial features. We use an FPN network [30] as our encoder, which helps to extract multi-scale feature representations. Specifically, the FPN outputs 4 feature maps X_1, X_2, X_3, X_4 and their respective resolutions are (100, 100), (50, 50), (25, 25), (25, 25). Then each feature map is transformed by a DHT module independently, as shown in Fig. 2. Since these feature maps are in different resolutions, the transformed features Y_1, Y_2, Y_3, Y_4 also have different sizes, because we use the same quantization interval in all stages (see Eq. (3) for details). To fuse transformed features together, we interpolate Y_2, Y_3, Y_4 to the size of Y_1, and then fuse them by concatenation.

3.3 Line Detection in the Parametric Space

Context-aware line detector. After the deep Hough transform (DHT), features are translated to the parametric space where grid location (θ, r) corresponds to features along an entire line $l = P^{-1}(\theta, r)$ in the feature space. An important reason to transform the features to the parametric space is that the line structures could be more compactly represented. As shown in Fig. 4, lines nearby a specific line l are translated to surrounding points near (θ_l, r_l). Consequently, features of nearby lines can be efficiently aggregated using convolutional layers in the parametric space.

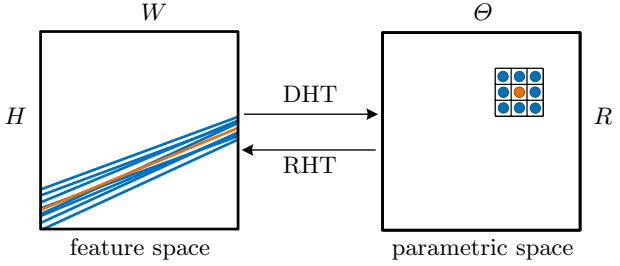

Fig. 4. Illustration of the proposed context-aware feature aggregation. Features of nearby lines in the feature space (left) are translated into neighbor points in the parametric space (right). In the parametric space, a simple 3×3 convolutional operation can easily capture contextual information for the central line (orange). Best viewed in color.

In each stage of the FPN, we use two 3×3 convolutional layers to aggregate contextual line features. Then we interpolate features to match the resolution of features from various stages, and concatenate the interpolated features together. Finally, a 1×1 convolutional layer is applied to the concatenated feature maps to produce the pointwise predictions.

Loss function. Since the prediction is directly produced in the parametric space, we calculate the loss in the same space as well. For a training image I, the ground-truth lines are first converted into the parametric space with the standard Hough transform. Then to help converging faster, we smooth and expand the ground-truth with a Gaussian kernel. Similar tricks have been used in many other tasks like crowed counting [32,10] and road segmentation [41]. Formally, let $\mathbf{G}$ be the binary ground-truth map in the parametric space, $\mathbf{G}_{i,j} = 1$ indicates there is a line located at i, j in the parametric space. The expanded ground-truth map is

$$\hat{\mathbf{G}} = \mathbf{G} \circledast K,$$

where K is a 5×5 Gaussian kernel and $\circledast$ denotes the convolution operation. An example pair of smoothed ground-truth and the predicted map is shown in Fig. 2.

Finally, we compute the cross-entropy between the smoothed ground-truth and the predicted map in the parametric space:

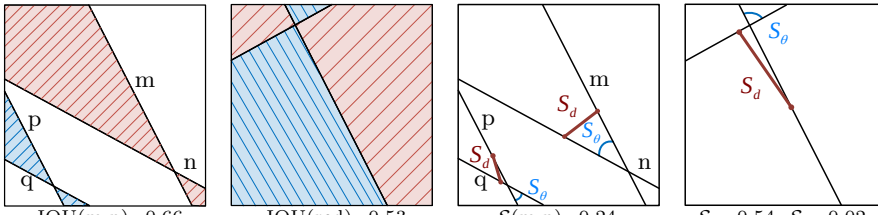

Fig. 5. (a): Two pairs of lines with similar relative position could have very different IOU scores. (b): Even humans cannot determine which area (blue or red) should be considered as the intersection in the IOU-based metric [29]. (c) and (d): Our proposed metric considers both Euclidean distance and angular distance between a pair of lines, resulting in consistent and reasonable scores. Best viewed in color.

$$\mathcal{L} = -\sum_i (\hat{\mathbf{G}}_i \cdot \log(\mathbf{P}_i) + (1 - \hat{\mathbf{G}}_i) \cdot \log(1 - \mathbf{P}_i)) \tag{5}$$

3.4 Reverse Mapping

Our detector produces predictions in the parametric space representing the probability of the existence of lines. The predicted map is then binarized with a threshold (*e.g.* 0.01). Then we find each connected area and calculate respective centroids. These centroids are regarded as the parameters of detected lines. At last, all lines are mapped back to the image space with $P^{-1}(\cdot)$, as formulated in Eq. (1). We refer to the "mapping back" step as "Reverse Mapping of Hough Transform (RHT)", as shown in Fig. 2.

4 The Proposed Evaluation Metric

In this section, we elaborate on the proposed evaluation metric that measures the agreement, or alternatively, the similarity between the two lines in an image. Firstly, we review several widely used metrics in the computer vision community and then explain why these existing metrics are not proper for our task. Finally, we introduce our newly proposed metric, which measures the agreement between two lines considering both Euclidean distance and angular distance.

4.1 Review of Existing Metrics

The intersection over union (IOU) is widely used in object detection, semantic segmentation and many other tasks to measure the agreement between detected bounding boxes (segments) w.r.t the ground-truth. Lee *et al.* [29] adopt the original IOU into line detection, and propose the line-based IOU to evaluate the quality of detected lines. Concretely, the similarity between the two lines is measured

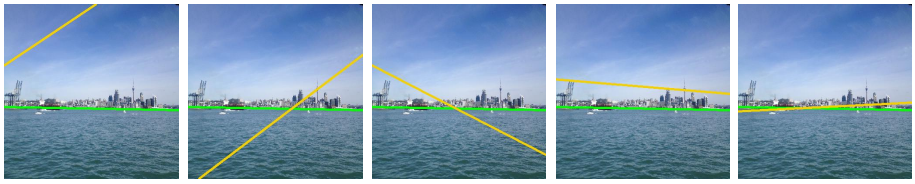

Fig. 6. Example lines with various EA-scores.

by the intersection areas of lines divided by the image area. Take Fig. 5(a) as an example, the similarity between line m and n is $\text{IOU}(m, n) = area(red)/area(I)$.

However, we argue that this IOU-based metric is improper and may lead to unreasonable or ambiguous results under specific circumstances. As illustrated in Fig. 5(a), two pairs of lines (m, n, and p, q) with similar structure could have very different IOU scores. In Fig. 5(b), even humans cannot determine which areas (red or blue) should be used as intersection areas in line based IOU. To remedy the aforementioned deficiencies, we elaborately design a new metric that measures the similarity of two lines.

4.2 The Proposed Metric

We propose a simple yet reasonable metric to assess the similarity between a pair of lines. Our metric $\mathcal{S}$, named **EA-score**, considers both **E**uclidean distance and **A**ngular distance between a pair of lines. Let l_i, l_j be a pair of lines to be measured, the angular distance $\mathcal{S}_\theta$ is defined according to the angle between two lines:

$$\mathcal{S}_\theta = 1 - \frac{\theta(l_i, l_j)}{\pi/2}, \tag{6}$$

where $\theta(l_i, l_j)$ is the angle between l_i and l_j. The Euclidean distance is defined as:

$$\mathcal{S}_d = 1 - D(l_i, l_j), \tag{7}$$

where $D(l_i, l_j)$ is the Euclidean distance between midpoints of l_i and l_j. Note that we normalize the image into a unit square before calculating $D(l_i, l_j)$. Examples of $\mathcal{S}_d$ and $\mathcal{S}_\theta$ can be found in Fig. 5(c) and Fig. 5(d). Finally, our proposed EA-score is:

$$\mathcal{S} = (\mathcal{S}_\theta \cdot \mathcal{S}_d)^2. \tag{8}$$

Note that the Eq. (8) is squared to make it more discriminative when the values are high. Several example line pairs and corresponding EA-scores are demonstrated in Fig. 6.

5 Experiments

In this section, we introduce the implementation details of our system, and report experimental results compared with existing methods.

5.1 Implementation Details

Our system is implemented with the PyTorch [34] framework. Since the proposed deep Hough transform (DHT) is highly parallelizable, we implement DHT with native CUDA programming, and all other parts are implemented based on PyTorch's Python API. We use a single RTX 2080 Ti GPU for all experiments.

Network architectures. We use two representative network architectures, ResNet50 [20] and VGGNet16 [39], as our backbone and the FPN [30] to extract multi-scale deep representations. For the ResNet network, following the common practice in previous works [47,9], the dilated convolution [45] is used in the last layer to increase the resolution of feature maps.

Hyper-parameters. The size of the Gaussian kernel used in Sec. 3.3 is 5×5. All images are resized to (400, 400) and then wrapped into a mini-batch of 8. We train all models for 30 epochs using the Adam optimizer [25] without weight decay. The learning rate and momentum are set to 2×10^{-4} and 0.9, respectively. The quantization intervals $\Delta\theta, \Delta r$ will be detailed in Sec. 5.3 and Eq. (12).

Datasets and data augmentation. Lee *et al.* [29] construct a dataset named SEL, which is, to the best of our knowledge, the only dataset for semantic line detection. The SEL dataset is composed of 1715 images, 1541 images for training and 174 for testing. There are 1.63 lines per image on average, and each image contains 1 line at least, and 6 lines at most. Following the setup in [29], we use only left-right flip data augmentation in all our experiments.

5.2 Evaluation Protocol

Given the metric in Eq. (8), we evaluate the detection results in terms of precision, recall, and F-measure.

For a pair of predicted and ground-truth line $(\hat{l}, l)$, we first calculate the similarity $\mathcal{S}(\hat{l}, l)$ as depicted in Eq. (8). $\hat{l}$ is identified as positive only if $\mathcal{S}(\hat{l}, l) > \epsilon$, where ϵ is a threshold. We calculate the precision and recall as:

$$Precision = \frac{\sum_{\hat{l} \in \mathcal{P}} \mathbb{1}(\mathcal{S}(\hat{l}, l) \geq \epsilon)}{||\mathcal{P}||}, \tag{9}$$

$$Recall = \frac{\sum_{l \in \mathcal{G}} \mathbb{1}(\mathcal{S}(l, \hat{l}) \geq \epsilon)}{||\mathcal{G}||}. \tag{10}$$

$\mathcal{P}$ and $\mathcal{G}$ are sets of predicted and ground-truth lines, respectively, and $||\cdot||$ denotes the number of elements in a set. $\mathbb{1}(\cdot)$ is the indicator function evaluating to 1 only if the condition is true. In Eq. (9), given a predicted line $\hat{l}$, l is the nearest ground-truth in the same image. Whereas in Eq. (10), $\hat{l}$ is the nearest prediction given a ground-truth line l in the same image. Accordingly, the F-measure is:

$$F\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{11}$$

We apply a series thresholds, *i.e.* $\epsilon = 0.01, 0.02, ..., 0.99$, to predictions. Accordingly, we derive a series of precision, recall and F-measure scores. Finally, we evaluate the performance in terms of average precision, recall and F-measure.

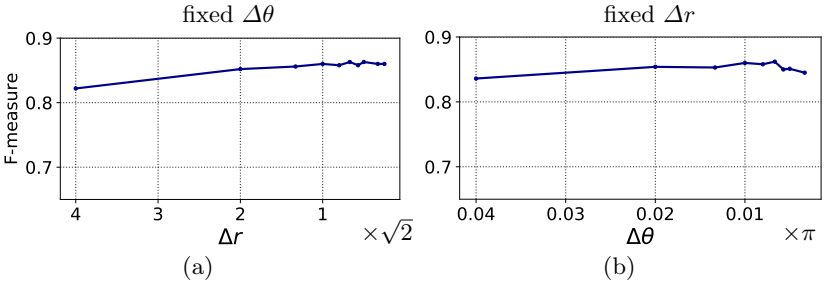

Fig. 7. (a): Performance under different distance quantization intervals Δr with a fixed angular quantization interval $\Delta\theta = \pi/100$. Larger Δr indicates smaller quantization levels R. (b): Performance under different angular quantization intervals $\Delta\theta$ with a fixed distance quantization interval $\Delta r = \sqrt{2}$.

5.3 Grid Search for Quantization Interval

The quantization intervals $\Delta\theta$ and Δr in Eq. (2) are important factors to the performance and running efficiency. Larger intervals lead to fewer quantization levels, *i.e.* Θ and R, and the model will be faster. With smaller intervals, there will be more quantization levels, and the computational overhead is heavier. To achieve a balance between performance and efficiency, we perform a grid search to find proper intervals that are computationally efficient and functionally effective.

We first fix the angular quantization interval to $\Delta\theta = \pi/100$ and then search for different distance quantization intervals Δr. According to the results in Fig. 7(a), with fixed angular interval $\Delta\theta$, the performance first increases with the decrease of Δr, and then gets saturated nearly after $\Delta r = \sqrt{2}$.

Afterward, we fix $\Delta r = \sqrt{2}$ and try different $\Delta\theta$. The results in Fig. 7(b) demonstrate that, with the decrease of $\Delta\theta$, the performance first increases until reaching the peak, and then slightly fall down. Hence, the peak value $\Delta\theta = \pi/100$ is a proper choice for angular quantization.

In summary, we use $\Delta\theta = \pi/100$ and $\Delta r = \sqrt{2}$ in quantization, and corresponding quantization levels are:

$$\Theta = 100, \ R = \sqrt{\frac{W^2 + H^2}{2}}, \tag{12}$$

where H, W are the size of feature maps to be transformed in DHT.

5.4 Comparisons

Quantitative comparison with previous arts. We compare our proposed method with the SLNet [29] and the classical Hough line detection [11] with HED [43] as the edge detector. Note that we train the HED edge detector on the SEL [29] training set using the line annotations as edge ground-truth.

The results in Tab. 1 illustrates that our method, with either VGG16 or ResNet50 as backbone, consistently outperforms SLNet and HT+HED with a

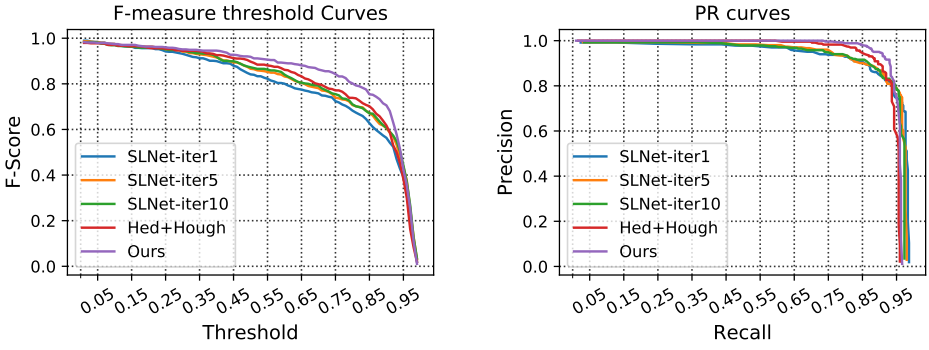

Fig. 8. Left: F-measure under various thresholds. Right: The precision-recall curve. Out method outperforms SLNet [29] and classical Hough transform [11] with a considerable margin. Moreover, even with 10 rounds of location refinement, SLNet still presents inferior performance.

considerable margin. In addition to Tab. 1, we plot the F-measure *v.s.* threshold and the precision *v.s.* recall curves. Fig. 8 reveals that our method achieves higher F-measure than others under a wide range of thresholds.

Method	Precision	Recall	F-measure	FPS
SLNet-iter1 [29]	0.747	**0.862**	0.799	2.67
SLNet-iter3 [29]	0.793	0.845	0.817	1.92
SLNet-iter5 [29]	0.798	0.842	0.819	-
SLNet-iter10 [29]	0.814	0.831	0.822	1.10
HED [43] + HT [11]	0.839	0.812	0.825	6.46
Ours(VGG16)	0.844	0.834	0.839	30.01
Ours(ResNet50)	**0.899**	0.824	**0.860**	**49.99**

Table 1. Quantitative comparisions across different methods. Our method significantly outperforms other competitors in terms of average F-measure.

Runtime efficiency. In this section, we benchmark the runtime of different methods including SLNet [29] with various iteration steps, classical Hough transform and our proposed method.

Both SLNet [29] and HT require edge detection, *e.g.* HED [43], as a preprocessing step. The non-maximal suppression (NMS) in SLNet requires edge maps as guidance, and the classical Hough transform takes an edge map as input. Moreover, SLNet uses a refining network to enhance the results iteratively, therefore, the inference speed is related to the iteration steps. In contrast, our method produces results with a single forward pass, and the NMS is as simple as computing the centroids of each connected area in the parametric space.

Method	Network forward	NMS	Edge	Total
SLNet-iter1 [29]	0.354 s	0.079 s	0.014 s	0.447 s
SLNet-iter3 [29]	0.437 s	0.071 s	0.014 s	0.522 s
SLNet-iter10 [29]	0.827 s	0.068 s	0.014 s	0.909 s
HED [43] + HT [11]	0.014 s	0.117 s	0.024 s	0.155 s
Ours(VGG16)	0.03 s	0.003 s	0	0.033 s
Ours(ResNet50)	0.017 s	0.003 s	0	**0.020 s**

Table 2. Quantitative speed comparisons. Our method is much faster than the other two competitors in forward pass and post-processing, and our method doesn't require any extra-process *e.g.* edge detection. Consequently, our method can run at 49 frames per second, which is remarkably higher than the other two methods.

Results in Tab. 2 illustrate that our method is significantly faster than all other competitors with a very considerable margin. Even with only 1 iteration step, SLNet is still slower than our method.

Qualitative Comparisions Here we give several example results of our proposed method along with SNLet and HED+HT. As shown in Fig. 9, compared with other methods, our results are more compatitable with the ground-truth as well as the human cognition. In addition to the results in Fig. 9, we provide all the detection results of our method and SLNet in the supplementary material.

Fig. 9. Example detection results by different methods. Compared to SLNet [29] and classical Hough transform [11], our results are more consistent with the ground-truth.

DHT	MS	CTX	F-measure
✓			0.845
✓	✓		0.852
✓		✓	0.847
✓	✓	✓	0.860

Table 3. Ablation study for each component. MS indicates DHTs with multi-scale featuresand CTX means context-aware aggregation as described in Sec. 3.2 and 3.3.

5.5 Ablation Study

In this section, we ablate each of the components in our method. Specifically, they are: (a) the Deep Hough transform (DHT) module detailed in Sec. 3.2; (b) the multi-scale (MS) DHT architecture described in Sec. 3.2; (c) the context-aware (CTX) line detector proposed in Sec. 3.3. Experimental results are shown in Tab. 3.

We first construct a baseline model with plain ResNet50 and DHT module. Note that the baseline model achieves 0.845 average F-measure, which has already surpassed the SLNet competitor.

Then we verify the effectiveness of the multi-scale (MS) strategy and context-aware line detector (CTX), individually. We separately append MS and CTX to the baseline model and then evaluate their performance, respectively. Results in Tab. 3 indicate that both MS and CTX can improve the performance of the baseline model.

At last, we combine all the components together to form our final full method, which achieves the best performance among all other combinations. Experimental results in this section clearly demonstrate that each component of our proposed method contributes to the success of our method.

6 Conclusions

In this paper, we proposed a simple yet effective method for semantic line detection in natural images. By incorporating the strong learning ability of CNNs into classical Hough transform, our method is able to capture complex textures and rich contextual semantics of lines. A new evaluation metric was proposed for line structures, considering both Euclidean distance and angular distance. Both quantitative and qualitative results revealed that our method significantly outperforms previous arts in terms of both detection quality and speed.

Acknowledgements. This research was supported by Major Project for New Generation of AI under Grant No. 2018AAA0100400, NSFC (61922046), Tianjin Natural Science Foundation (18ZXZNGX00110), and the Fundamental Research Funds for the Central Universities (Nankai University: 63201169).

References

1. Aggarwal, N., Karl, W.C.: Line detection in images through regularized hough transform. IEEE transactions on image processing **15**(3), 582–591 (2006)
2. Akinlar, C., Topal, C.: Edlines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011)
3. Ballard, D.: Generating the hough transform to detect arbitary shapes. Pattern Recognition **13**(2) (1981)
4. Borji, A., Cheng, M.M., Hou, Q., Jiang, H., Li, J.: Salient object detection: A survey. Computational Visual Media **5**(2), 117–150 (2019). https://doi.org/10.1007/s41095-019-0149-9
5. Burns, J.B., Hanson, A.R., Riseman, E.M.: Extracting straight lines. IEEE transactions on pattern analysis and machine intelligence **PAMI-8**(4), 425–455 (1986)
6. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence **PAMI-8**(6), 679–698 (1986)
7. Caplin, S.: Art and Design in Photoshop. Elsevier/Focal (2008)
8. Chan, T., Yip, R.K.: Line detection algorithm. In: Proceedings of 13th International Conference on Pattern Recognition. vol. 2, pp. 126–130. IEEE (1996)
9. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
10. Cheng, Z.Q., Li, J.X., Dai, Q., Wu, X., Hauptmann, A.G.: Learning spatial awareness to improve crowd counting. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6152–6161 (2019)
11. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Tech. rep., Sri International Menlo Park Ca Artificial Intelligence Center (1971)
12. Etemadi, A.: Robust segmentation of edge data. In: 1992 International Conference on Image Processing and its Applications. pp. 311–314. IET (1992)
13. Fan, D.P., Lin, Z., Zhang, Z., Zhu, M., Cheng, M.M.: Rethinking RGB-D salient object detection: Models, datasets, and large-scale benchmarks. IEEE TNNLS (2020)
14. Fan, D.P., Zhai, Y., Borji, A., Yang, J., Shao, L.: Bbs-net: Rgb-d salient object detection with a bifurcated backbone strategy network. In: European Conference on Computer Vision (ECCV) (2020)
15. Fan, R., Cheng, M.M., Hou, Q., Mu, T.J., Wang, J., Hu, S.M.: S4net: Single stage salient-instance segmentation. Computational Visual Media **6**(2), 191–204 (June 2020). https://doi.org/10.1007/s41095-020-0173-9
16. Fernandes, L.A., Oliveira, M.M.: Real-time line detection through an improved hough transform voting scheme. Pattern recognition **41**(1), 299–314 (2008)
17. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2020)
18. Gao, S.H., Tan, Y.Q., Cheng, M.M., Lu, C., Chen, Y., Yan, S.: Highly efficient salient object detection with 100k parameters. In: European Conference on Computer Vision (ECCV) (2020)
19. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

21. Hou, Q., Cheng, M.M., Hu, X., Borji, A., Tu, Z., Torr, P.: Deeply supervised salient object detection with short connections. IEEE TPAMI **41**(4), 815–828 (2019). https://doi.org/10.1109/TPAMI.2018.2815688
22. Hough, P.V.: Method and means for recognizing complex patterns (1962), uS Patent 3,069,654
23. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: Ccnet: Criss-cross attention for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 603–612 (2019)
24. Illingworth, J., Kittler, J.: The adaptive hough transform. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-9**(5), 690–698 (1987)
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
26. Kiryati, N., Eldar, Y., Bruckstein, A.M.: A probabilistic hough transform. Pattern recognition **24**(4), 303–316 (1991)
27. Krages, B.: Photography: the art of composition. Simon and Schuster (2012)
28. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 734–750 (2018)
29. Lee, J.T., Kim, H.U., Lee, C., Kim, C.S.: Semantic line detection and its applications. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3229–3237 (2017)
30. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
31. Liu, L., Chen, R., Wolf, L., Cohen-Or, D.: Optimizing photo composition. Comput. Graph. Forum **29**(2), 469–478 (2010)
32. Liu, W., Salzmann, M., Fua, P.: Context-aware crowd counting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5099–5108 (2019)
33. Liu, Y., Cheng, M.M., Hu, X., Bian, J.W., Zhang, L., Bai, X., Tang, J.: Richer convolutional features for edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1939 – 1946 (2019). https://doi.org/10.1109/TPAMI.2018.2878849
34. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
35. Princen, J., Illingworth, J., Kittler, J.: A hierarchical approach to line extraction based on the hough transform. Computer vision, graphics, and image processing **52**(1), 57–77 (1990)
36. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3d object detection in point clouds with image votes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4404–4413 (2020)
37. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9277–9286 (2019)
38. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
40. Sobel, I.: An isotropic 3x3 image gradient operator. Presentation at Stanford A.I. Project 1968 (02 2014)

41. Tan, Y.Q., Gao, S., Li, X.Y., Cheng, M.M., Ren, B.: Vecroad: Point-based iterative graph exploration for road graphs extraction. In: IEEE CVPR (2020)
42. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7794–7803 (2018)
43. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE international conference on computer vision. pp. 1395–1403 (2015)
44. Yacoub, S.B., Jolion, J.M.: Hierarchical line extraction. IEE Proceedings-Vision, Image and Signal Processing **142**(1), 7–14 (1995)
45. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
46. Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., Gao, S.: PPGnet: Learning point-pair graph for line segment detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019 (2019)
47. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)