Deep Hough Transform for Semantic Line Detection

Kai Zhao*, Qi Han*, Chang-Bin Zhang, Jun Xu, Ming-Ming Cheng†, *Senior Member, IEEE*

Abstract—We focus on a fundamental task of detecting meaningful line structures, *a.k.a.*, semantic line, in natural scenes. Many previous methods regard this problem as a special case of object detection and adjust existing object detectors for semantic line detection. However, these methods neglect the inherent characteristics of lines, leading to sub-optimal performance. Lines enjoy much simpler geometric property than complex objects and thus can be compactly parameterized by a few arguments. To better exploit the property of lines, in this paper, we incorporate the classical Hough transform technique into deeply learned representations and propose a one-shot end-to-end learning framework for line detection. By parameterizing lines with slopes and biases, we perform Hough transform to translate deep representations into the parametric domain, in which we perform line detection. Specifically, 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 into 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 for accurate line detection. In addition to the proposed method, we design an evaluation metric to assess the quality of line detection and construct a large scale dataset for the line detection task. Experimental results on our proposed dataset and another public dataset demonstrate the advantages of our method over previous state-of-the-art alternatives. The dataset and source code is available at https://mmcheng.net/dhtline/.

Index Terms—Semantic line detection, Hough transform, CNN, Deep Learning.

1 Introduction

Detecting line structures from digital images has a long history in computer vision. The organization of line structures is an early yet essential step to transform the visual signal into useful intermediate concepts for visual interpretation [2]. Though many techniques have been proposed to detect salient objects [3], [4], [5], [6], [7] and areas [8], [9], [10], little work has been made for detecting outstanding/structure-revealing line structures. A recent study [11] was proposed to detect outstanding straight line(s), referred to as "semantic line", that outlines the conceptual structure of natural images. Identifying these semantic lines is of crucial importance for computer graphics and vision applications, such as photographic composition [12], [13], structure-preserving image processing [14], [15], image aesthetic [16], [17], [18], [19], lane detection [20], and artistic creation [21], [22], [23], [24]. As demonstrated in Fig. 1, Liu *et al.* [12] proposed to crop images according to the golden ratio by using 'prominent line'. Detecting these 'semantic lines' can help to produce images that are visually pleasing in the photographic composition.

The Hough transform [25], [26] is one representative method for line detection, which was first proposed to detect straight lines in bubble chamber photographs [27]. Since its simplicity and efficiency, HT is employed to detect lines in digital images [25], and further extended by [26] to detect other regular shapes like circles and rectangles. The key idea of the Hough transform is to vote evidence from the image domain to the parametric domain, and then detect shapes in the parametric domain by identifying local-maximal responses. In the case of line detection, a line in the image domain can be represented by its parameters, *e.g.,* slope, and offset in the parametric space. Hough transform collects evidence along with a line in an image and accumulates evidence to a single point in the parameter space. Consequently, line detection in the image domain is converted to the problem of detecting peak responses in the parametric domain. Classical Hough transform based line detectors [28], [29], [30], [31] usually detect continuous straight edges while neglecting the semantics in line structures. Moreover, these methods are sensitive to light changes and occlusion. Therefore, the results are often noisy and contain irrelevant lines [32], as shown in Fig. 1(d).

Convolutional Neural Networks (CNNs) have achieved remarkable success in a wide range of computer vision tasks. Several recent studies [11], [34] have proposed CNN-based methods for line detection. Concretely, they regard line detection as a special case of object detection and employ existing object detectors *e.g.,* faster R-CNN [35] or CornerNet [36], for line detection. Limited by the ROI pooling and non-maximal suppression of lines, both [11] and [34] are less efficient in terms of running time. Moreover, ROI pooling [37] aggregates features along with a single line, while many recent studies reveal that richer context information is critical to many tasks, *e.g.,* video classification [38] and semantic segmentation [39]. This point will be validated in Sec. 6.6, in which we experimentally verify that only aggregating features along a single line will produces sub-

* *The first two students contribute equally to this paper.*
† *M.M. Cheng is the corresponding author (cmm@nankai.edu.cn).*

- *Kai Zhao, Qi Han, Chang-Bin Zhang, and Ming-Ming Cheng are with the TKLNDST, College of Computer Science, Nankai University, Tianjin, China, 300350.*
- *Jun Xu is with the School of Statistics and Data Science, Nankai University, Tianjin, China, 300071.*
- *A preliminary version of this work has been presented in [1].*

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

Fig. 1. Example pictures from [12] reveal that semantic lines may help in the photographic composition. (a): a photo was taken with an arbitrary pose. (b): a photo fits the golden ratio principle [21], [33] which is obtained by the method described in [12] using so-called 'prominent lines' in the image. (c): Our detection results are clean and comprise only a few meaningful lines that are potentially helpful in the photographic composition. (d): Line detection results by the classical line detection algorithms often focus on fine detailed straight edges.

optimal results.

Incorporate powerful CNNs to Hough transform is a promising direction for semantic line detection. A simple way of combining CNN with Hough transform is performing edge detection with a CNN-based edge detector [40], [41] and then apply standard Hough transform to the edge maps. However, the two components have diverse optimization targets, leading to sub-optimal results, as evidenced by our experiments. In this paper, we propose to incorporate CNN with Hough transform into an end-to-end manner so that each component in our proposed method shares the same optimization target. Our method first extracts pixel-wise representations with a CNN-based encoder and then performs Hough transform on the deep representations to convert representations from feature space into parametric space. Then the global line detection problem is converted to simply detecting peak response in the transformed features, making the problem much simpler. For example, the time-consuming non-maximal suppression (NMS) can be simply replaced by 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 about lines nearby. As shown in Fig. 1(c), our method detects clean, meaningful and outstanding lines, that are helpful to photographic composition.

To better evaluate line detection methods, we introduce a principled metric to assess the agreement of a detected line *w.r.t.* its corresponding ground-truth line. Although [11] 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. And at last, we collect a large scale dataset with 6,500 carefully annotated images for semantic line detection. The new dataset, namely NKL (short for **N**an**K**ai **L**ines), contains images of diverse scenes, and the scale is much larger than the existing SEL [11] dataset in both terms of images and annotated lines.

The contributions of this paper are summarized below:

- We proposed 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.
- To facilitate the research of semantic line detection, we construct a new dataset with 6,500 images, which is larger and more diverse than a previous SEL dataset [11].
- We introduce a principled metric that measures the similarity between two lines. Compared with the previous IOU based metric [11], our metric has straightforward interpretation and simplicity in implementation, as detailed in Sec. 4.
- Evaluation results on an open benchmark demonstrate that our method outperforms prior arts with a significant margin.

A preliminary version of this work was presented in [1]. In this extended work, we introduce three major improvements:

- We propose a novel "edge-guided refinement" module to adjust line positions and obtain better detection performance with the help of accurate edge information. This part is detailed in Sec. 3.5.
- We introduce a new large-scale dataset for semantic line detection, as presented in Sec. 5. The new dataset, namely NKL (short for **N**an**K**ai **L**ines), contains 6,500 images in total, and each image is annotated by multiple skilled annotators.
- We employ the maximal bipartite graph matching [42] to match ground-truth and detected lines during evaluation (Sec. 6.1). The matching procedure removes redundant true positives so that each ground-truth line is associated with at most one detected line and vice versa.

The rest of this paper is organized as follows: Sec. 2 summarizes the related works. Sec. 3 elaborates the proposed Deep Hough transform method. Sec. 4 describes the proposed evaluating metric, which is used to assess the similarity between a pair of lines. Sec. 5 introduces our newly constructed dataset. Sec. 6 presents experimental details and report comparison results. Sec. 7 makes a conclusion remark.

2 Related Work

The research of line detection in digital images dates back to the very early stage of computer vision research. Here,

we first brief the evolution of Hough transform [25] (HT), one of the most fundamental tools, for line detection. Then we introduce several recent CNN based methods for line detection. At last, we summarize the methods and datasets for semantic line detection.

2.1 Hough transform

The Hough transform (HT) was firstly proposed in [27] for machine analysis of bubble chamber photographs. It parametrizes straight lines with slope-offset, leading to an unbounded transform space (since the slope can be infinity). [25] extended HT by using angle-radius rather than slope-offset parameters, and is conceptually similar to two-dimensional Radom transform [43]. Then Ballard *et al.* [26] generalized the idea of HT to localize arbitrary shapes, *e.g.,* ellipses and circles, from digital images. For example, by parameterizing with angle and radius, line detection can be performed by voting edge evidence and finding peak response in the finite parametric space. Typically, with the edge detectors such as Canny [44] and Sobel [45], the detected lines are the maximal local response points in the transformed parametric space. The core idea of HT is used in two recent works which parameterize the outputs of CNNs with offsets and orientations to predict surface meshes [46] or convex decomposition [47] of 3D shapes.

Despite the success of HT on line detection, it suffers from high computational costs and unstable performance. To accelerate the voting of HT, Nahum *et al.* [31] proposed the "probabilistic Hough transform" to randomly pick sample points from a line, while [48] using the gradient direction of images to decide the voting points. Meanwhile, the work of [28], [49] employed kernel-based Hough transform to perform hough voting by using the elliptical-Gaussian kernel on collinear pixels to boost the original HT. Besides, John *et al.* [29], [30] partitioned the input image into hierarchical image patches, and then applied HT independently to these patches. Illingworth *et al.* [50] use a coarse-to-fine accumulation and search strategy to identify significant peaks in the Hough parametric spaces. [51] tackled line detection within a regularized framework, to suppress the effect of noise and clutter corresponding to nonlinear image features. The Hough voting scheme is also used in many other tasks such as detecting centroid of 3D shapes in point cloud [52] and finding image correspondence [53].

2.2 Line Segments Detection

Though its robustness and parallelism, Hough transform cannot be directly used for line segments detection, since it cannot determine the endpoints of line segments. Probabilistic Hough transform [31] uses random sampling in the voting scheme, and reconstructs line segments by localizing the sample locations. But this method still prefers long straight lines. In addition to Hough transform, many other studies have been developed to detect line segments. Burns *et al.* [2] used the edge orientation as the guide for line segments extraction. The main advantage is that the orientation of the gradients can help to discover low-contrast lines and endpoints. Etemadi *et al.* [54] established a chain from the given edge map and extracted line segments and orientations by walking over these chains. Chan *et al.* [55] used a quantified edge orientation to search and merge short line segments. Gioi *et al.* [56] proposed a linear-time line segment detector (LSD) without tuning parameters, and is used by many subsequent studies [32], [57], [58]

2.3 CNN based Line Detection.

Recently, CNNs have brought remarkable improvements in computer vision tasks, and also be applied to line detection. These methods either focus on straight line detection, *e.g.,* semantic line detection [11], [59], or line segments detection, *e.g.,* wireframe parsing [34], [60], [61], [62]. Lee *et al.* [11] followed the two-branch pipeline of faster-RCNN [35] and proposed a straight line detection framework to find the meaningful semantic straight line in an image. One branch verifies the existence of a line and the other branch further refines the position of the line by regression. Zhang *et al.* [34] adopted the conception of CornerNet [36] to extract line segments as a pair of key points in indoor scenes. Huang *et al.* [60] proposed a two-head network to predict lines and junction points for wireframe parsing. This is extended in [61] by adding a line proposal sub-network. Zhou *et al.* [61] proposed an end-to-end architecture to perform accurate line segments detection in wireframe parsing.

All these methods extract line-wise feature vectors by LoI pooling that aggregate deep features solely along each line, leading to inadequate context information.

2.4 Semantic Line Detection

The meaningful straight line which helps photographic composition was firstly discussed in [11], and named as "semantic line". [11] regarded semantic line detection as a special case of object detection. It first extracts CNN representations of line proposals using LoI pooling, which bilinearly interpolates the features along the entire straight line. Then the line representations are verified by a classifier and a regressor, similar to Faster-RCNN [37]. The line proposals are all unique lines in an image. The metric of the intersection of union (IoU) of two straight lines is proposed in [11] to evaluate the similarity of two straight lines in an image. This metric may produce ambiguous definitions in some scenarios, as will be mentioned in Sec. 4. Besides, Lee *et al.* [11] collected a semantic line detection dataset which contains about 1,700 outdoor images, and most of them are natural landscape.

3 Approach

In this section, we give the details of the proposed deep Hough transform for semantic line detection. Our proposed method mainly contains four components: 1) a CNN encoder that extracts pixel-wise deep representations; 2) the deep Hough transform (DHT) that converts the deep representations from the spatial domain to the parametric domain; 3) a line detector that is responsible for detecting lines in the parametric space, and 4) a reverse Hough transform (RHT) component that converts the detected lines back to image space. All these components are integrated in an end-to-end framework that performs forward inference and backward training within a single step. The pipeline is illustrated in Fig. 2, and the detailed architecture is shown in the supplementary materials.

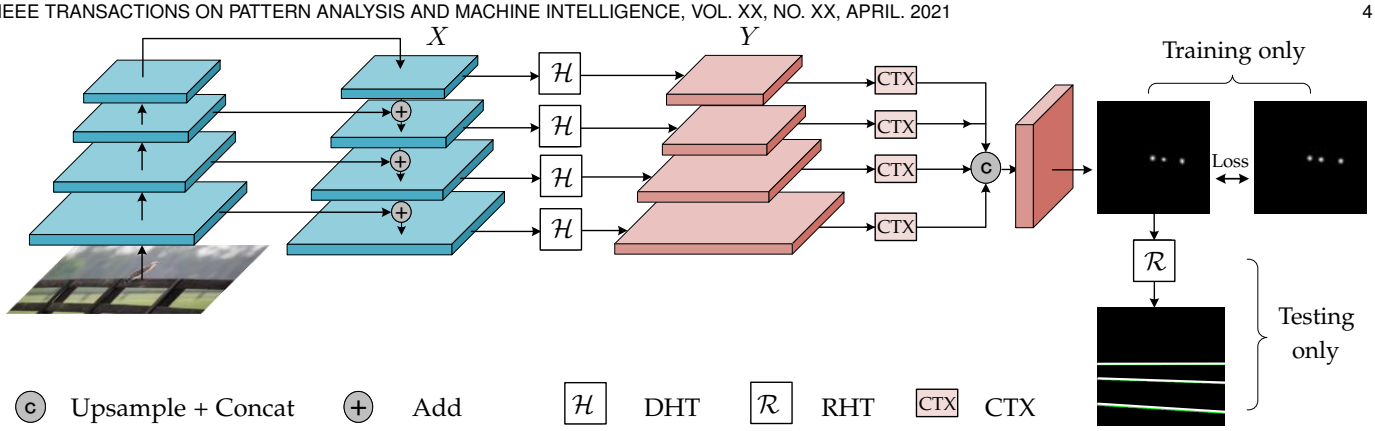

Fig. 2. The pipeline of our proposed method. DHT is short for the proposed Deep Hough Transform, and RHT represents the Reverse Hough Transform. CTX means the context-aware line detector which contains multiple convolutional layers.

3.1 Line Parameterization and Reverse

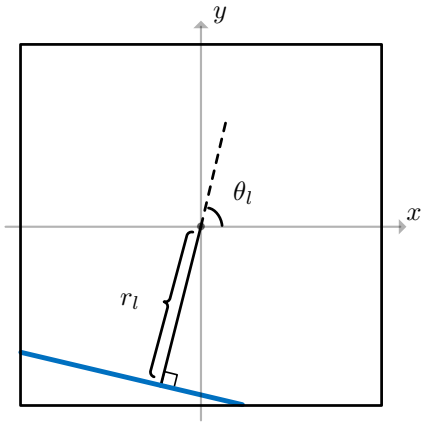

Fig. 3. A line can be parameterized by bias r_l and slope θ_l.

As shown in Fig. 3, given a 2D image $I_{H\times W} \in \mathbb{R}^{H\times W}$, we set the origin to the center of the image. In the 2D plane, a straight line l can be parameterized by two parameters: an orientation parameter $\theta_l \in [0, \pi)$ representing the angle between l and the x-axis and a distance parameter r_l, indicating the distance between l and the origin. Obviously $\forall\, l \in I, r_l \in [-\sqrt{W^2+H^2}/2, \sqrt{W^2+H^2}/2]$.

Given any line l on I, we can parameterize it with the above formulations, and also we can perform a reverse mapping to translate any valid (r, θ) pair to a line instance. We define the line-to-parameters and the inverse mapping as:

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

Obviously, both P and P^{-1} are bijective mappings. 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, denoted 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. 4(a).

3.2 Feature Transformation with Deep Hough Transform

3.2.1 *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, is determined by the quantization intervals, as described in Eq. (3).

As shown in Fig. 4(a), given an arbitrary line l on the image, we aggregate 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).

Given the number of quantization levels Θ and R, we have $\Theta \cdot R$ unique line candidates. Then the DHT is applied to all these candidate lines and their respective features are aggregated to the corresponding position in $\mathbf{Y}$. It is worth noting that DHT is order-agnostic in both the feature space and the parametric space, making it highly parallelizable.

3.2.2 *Multi-scale DHT with FPN.*

Our proposed DHT could be easily applied to arbitrary spatial features. We use the feature pyramid network (FPN) [63] as our encoder. FPN can help to extract multi-scale and rich semantic features.

Specifically, the FPN outputs 4 feature maps X_1, X_2, X_3, X_4 and their respective resolutions are $1/4$, $1/8$, $1/16$, $1/16$ of the input resolution. Then each feature map is transformed by a DHT module independently, as shown in Fig. 2. Since these feature maps are in different

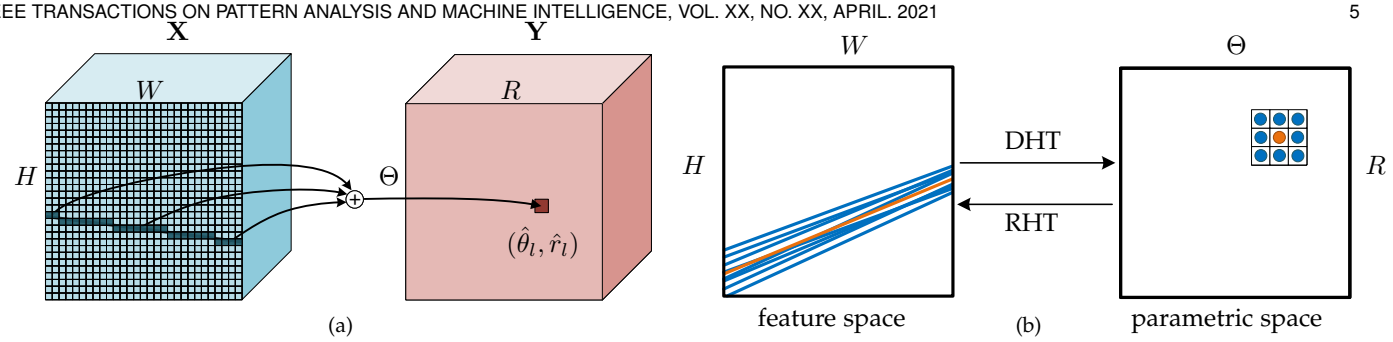

Fig. 4. (a): Features along a line in the feature space (blue, left) are accumulated to a point $(\hat{r}_l, \hat{\theta}_l)$ in the parametric space (red, right). (b): Illustration of the proposed context-aware feature aggregation. Features of nearby lines in the feature space (left) are translated 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.

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

3.3.1 *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 into the parametric space is that the line structures could be more compactly represented. As shown in Fig. 4(b), 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.

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 different stages, as illustrated in Fig. 2, and concatenate the interpolated features together. Finally, a 1×1 convolutional layer is applied to the concatenated feature maps to produce pointwise predictions.

3.3.2 *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 [64], [65] and road segmentation [66]. 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.

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

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

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.

3.5 Edge-guided Line Refinement

Semantic lines are outstanding structures that separate different regions in a scene. Therefore, edges may serve as indicators for semantic lines. We propose to refine the detection results by aligning line positions using edge information. First, we compute an edge map E using HED [41]. Afterward, given a detected line l, the edge density of l is defined as the average edge response along l:

$$\rho(l) = \frac{\sum_{i \in l} E_i}{|l|}, \quad (6)$$

where $|l|$ is the number of pixels on l. For the sake of stability, we widen l by 1 pixel on both sides (totally the width is 3) when dealing with Eq. (6).

Let $\mathcal{L}$ be a set of lines that are close to l. These lines are obtained by moving the end-points of l by δ_r pixels clockwise and anti-clockwise. Since there are two end-points and each one has $\delta_r + 1$ possible locations, the size of the set is $||\mathcal{L}|| = (\delta_r + 1)^2$. Then the refinement can be achieved by finding the optimal line l^* from $\mathcal{L}$ that holds the highest edge density:

$$l^* = \arg\max_{l \in \mathcal{L}} \ \rho(l). \quad (7)$$

The performance of "edge-guided line refinement" with different δ_r is recorded in Sec. 6.6.2.

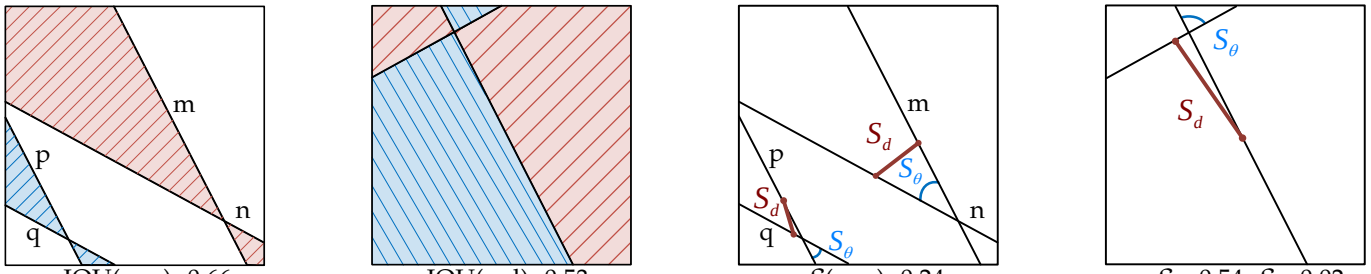

Fig. 5. (a): Two pairs of lines with similar relative position could have very different IOU scores. (b): Even humans cannot determine which area (blue or red) should be considered as the intersection in the IOU-based metric [11]. (c) and (d): Our proposed metric considers both Euclidean distance and angular distance between a pair of lines, resulting in consistent and reasonable scores. Best viewed in color.

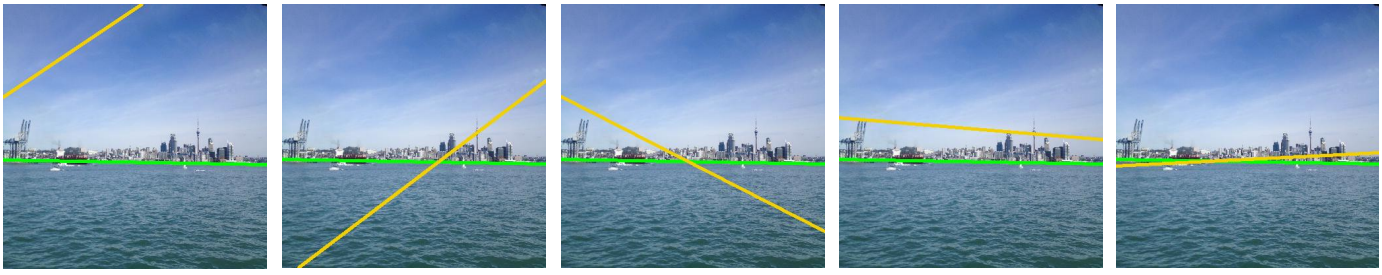

Fig. 6. Example lines with various EA-scores ($\mathcal{S}$ in Eq. (10)). The larger the EA-score is, the more similar the lines are.

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.* [11] 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 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 structures 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.

There are other metrics, *e.g.*, the Earth Mover's Distance (EMD) [67] and the Chamfer distance (CD) [68], that can be used to measure line similarities. However, these metrics require to rasterize the lines into pixels and then calculate pixel-wise distances, which is less efficient.

To remedy the deficiencies, we propose a simple yet effective metric that measures the similarity of two lines in the parametric space. Our proposed metric is much more efficient than EMD and CD. Quantitative comparisons in Sec. 6.4 demonstrate that our proposed metric presents very similar results to EMD and CD.

4.2 The Proposed Metric

Our proposed metric, termed **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:

$$\mathcal{S}_\theta = 1 - \frac{\theta(l_i, l_j)}{\pi/2}, \tag{8}$$

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

$$\mathcal{S}_d = 1 - D(l_i, l_j), \tag{9}$$

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:

$$\mathcal{S} = (\mathcal{S}_\theta \cdot \mathcal{S}_d)^2. \tag{10}$$

Eq. (10) is squared to make it more sensitive and discriminative when the values are high.

Several example line pairs and corresponding EA-scores are demonstrated in Fig. 6.

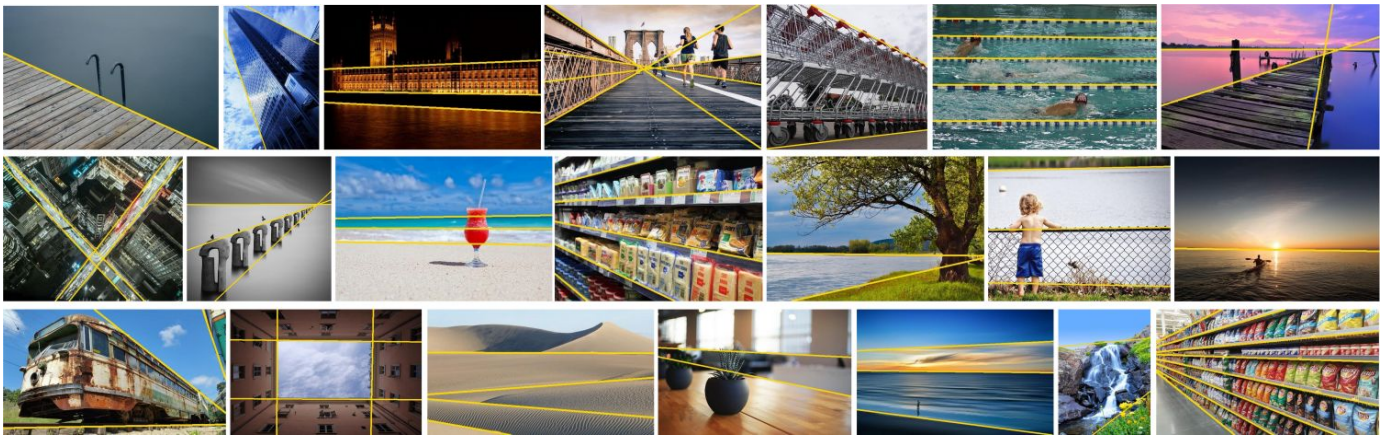

Fig. 7. Example images and annotations (yellow lines) of NKL. Images of NKL present diverse scenes and rich line annotations.

5 NKL: A SEMANTIC LINE DETECTION DATASET

To the best of our knowledge, there is only one dataset, SEL [11], specifically for semantic line detection. SEL contains 1,715 images of which 175 images for testing and others for training. To fulfill the gap between large CNN-based models and the scale of the existing dataset, we collect a new dataset for semantic line detection.

The new dataset, namely NKL (short for **N**an**K**ai **L**ines), contains 6,500 images that present richer diversity in terms of both scenes and the number of lines. Each image of NKL is annotated by multiple skilled human annotators to ensure the annotation quality. The dataset is open available on our project page.

TABLE 1
Number of images and lines in SEL [11] and NKL.

Dataset	Total #images, #lines	Training #images, #lines	Evaluation #images, #lines
SEL [11]	1,715, 2,791	1,541, 2,493	175, 298
NKL (Ours)	6,500, 13,148	5,200, 10,498	1,300, 2,650

5.1 Data Collection and Annotation

All the images of NKL are crawled from the internet using specific keywords such as sea, grassland *et al.* After copyright checking, we carefully filter out images with at least one semantic line. Since the annotation of semantic lines is subjective and depends on annotators, each image is first annotated by 3 knowledgeable human annotators and verified by others. A line is regarded as positive only if all of the 3 annotators are consistent. Then the inconsistent lines are reviewed by two other annotators. In a word, for each line, there are at least 3 and at most 5 annotators, and a line is regarded as positive only if the line is marked as positive by more than 3 annotators.

5.2 Dataset Statistics

5.2.1 Number of images and semantic lines

There are totally 13,148 semantic lines in NKL and 2,791 semantic lines in SEL [11] dataset over all images. Tab. 1 summarizes the number of images and lines of the two datasets, respectively.

Fig. 8 summarizes the histogram of the per-image number of lines in NKL and SEL [11] datasets. More than half (67%, 4,356/6,500) of the images in NKL dataset contain more than 1 semantic line, while the percentage of SEL is only 45.5%.

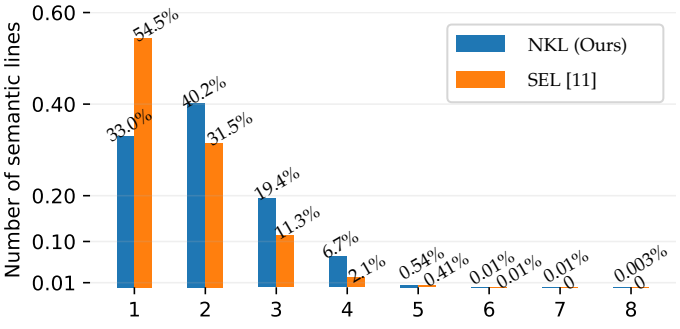

Fig. 8. Histogram chart of number of lines. Lines of our dataset are more fairly distributed compared to SEL.

5.2.2 Diversity Analysis

To analyze the diversity of SEL and NKL datasets, we feed all the images into a ResNet50 [69] network that is pretrained on the Place365 [70], and then collect the outputs as category labels. The results are presented in Fig. 9.

There are totally 365 categories in Place365 [70] dataset, among which we got 167 unique category labels on SEL dataset and 327 on NKL. Besides, as shown in Fig. 9, scene labels on NKL dataset are more fairly distributed compared to SEL dataset. For example, in SEL dataset, top-3 populated categories (sky, field, desert) make up more than a quarter of the total. While in NKL, top-3 makes up less than one-fifth of the total.

6 EXPERIMENTS

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

6.1 Implementation Details

Our system is implemented with the PyTorch [71] framework, and a Jittor [72] implementation is also available.

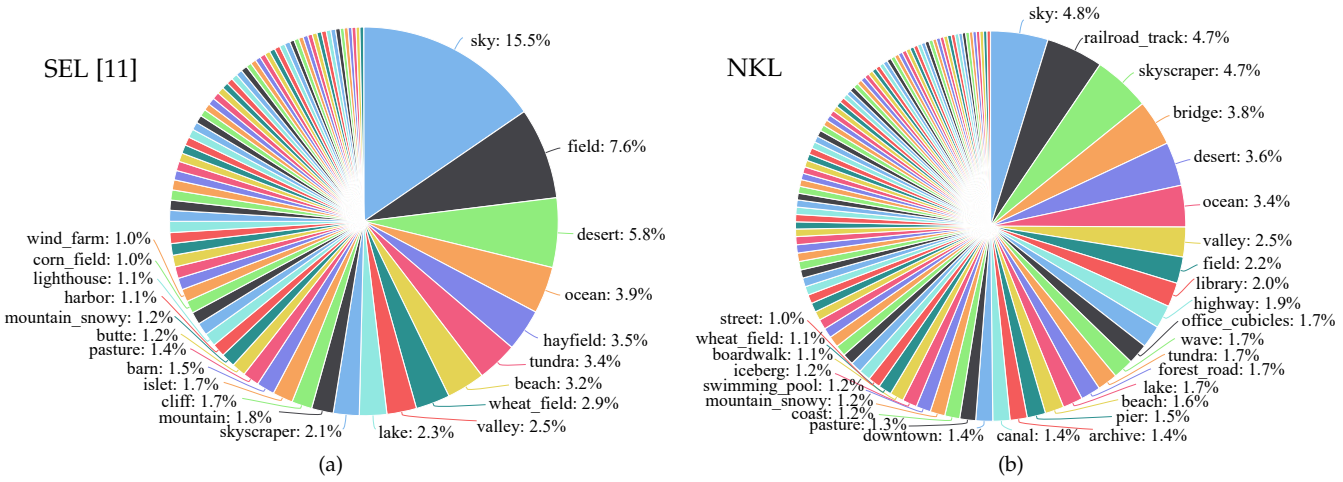

Fig. 9. Category distribution of SEL (a) and NKL (b) datasets. Category labels are obtained through a Places365 pretrained model. There are 327 (totally 365) scene labels presented in NKL dataset, in contrast to 167 in SEL dataset. The labels of NKL are also more fairly distributed compared to that of SEL.

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 framework level Python API. We use a single RTX 2080 Ti GPU for all experiments.

6.1.1 Network architectures.

We use two representative network architectures, ResNet50 [69] and VGGNet16 [73], as our backbone and the FPN [63] to extract multi-scale deep representations. For the ResNet network, following the common practice in previous works [74], [75], [76], the dilated convolution [77] is used in the last layer to increase the resolution of feature maps.

6.1.2 Hyper-parameters.

The size of the Gaussian kernel used in Sec. 3.3.2 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 Adam optimizer [78] 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. 6.3 and Eq. (12).

6.1.3 Datasets and data augmentation.

Our experiments are conducted on the SEL [11] dataset and our Proposed NKL dataset. The Statistics of the two datasets are detailed in Sec. 5. Following the setup in [11], we use only left-right flip data augmentation in all our experiments.

6.2 Evaluation Protocol

We measure the quality of detection lines in terms of *precision*, *recall* and *F-measure*. The first step is to match the detected lines and ground-truth lines.

Let $\mathcal{P}$ and $\mathcal{G}$ be the sets of predicted lines and ground-truth lines, respectively. p_i and g_j are individual predicted and ground-truth line. We first match the lines in $\mathcal{P}$ and $\mathcal{G}$ based on bipartite matching. Suppose $G = \{V, E\}$ be a bipartite graph [1]. The vertice set V can be divided into two disjoint and independent sets, in our case, $\mathcal{P}$ and $\mathcal{G}$:

$$V = \mathcal{P} \cup \mathcal{G}$$
$$\mathcal{P} \cap \mathcal{G} = \emptyset.$$

Each edge in E denotes the similarity between a pair of lines under a certain similarity measure. Apart from the proposed EA-score, we also use two other popular metrics: the earth mover's distance (EMD) [67] and the Chamfer distance (CD) [68], as described in Sec. 4. Note that we normalize both EMD and chamfer distance by their maximal possible value to bound the value within [0, 1] (for both EMD and Chamfer distance, the maximal distance occurs when two lines shrinkage to two points on the opposite diagonals).

Given the graph $G = \{V, E\}$, a matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share a common vertice. In our task, given the set of predicted lines $\mathcal{P}$ and the set of ground-truth lines $\mathcal{G}$, we seek to find a matching so that each ground-truth line g_i corresponds to no more than one detected line p_j and vice versa. This problem, maximum matching of a bipartite graph, can be easily solved using the classical Hungarian method [42] with polynomial time complexity.

After matching $\mathcal{P}$ and $\mathcal{G}$, we can calculate true positive (TP), false positive (FP) and false negative (FN) accordingly. As illustrated in Fig. 10, predicted lines (p_1, p_2) that are paired with ground-truth lines (g_2, g_1) are considered as true positive. Predicted line (p_3) that is not matched with any ground-truth line is a false positive, and ground-truth line (g_3) without a corresponding predicted line is a false negative.

Finally, the **P**recision, **R**ecall, and **F**-measure are:

$$P = \frac{TP}{TP + FP}, \; R = \frac{TP}{TP + FN}, \; F = \frac{2PR}{P + R}. \tag{11}$$

We apply a series thresholds $\tau = 0.01, 0.02, ..., 0.99$ to prediction & ground-truth pairs. Accordingly, we derive a series of precision, recall, and F-measure scores. Finally,

1. https://en.wikipedia.org/wiki/Bipartite_graph

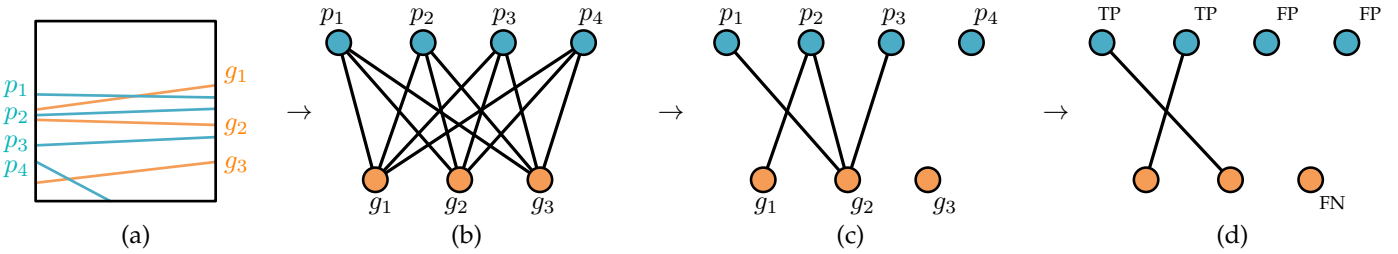

Fig. 10. Illustration of the bipartite graph matching in evaluation. (a) An example image with 3 ground-truth lines (g_1, g_2, g_3) and 4 predictions (p_1, p_2, p_3, p_4). (b) the corresponding bipartite graph. The edge between a pair of nodes represents the similarity ($\mathcal{S}$ in Eq. (10)) between lines. (c) after maximum matching of a bipartite graph, each node in a subgraph is connected with no more than 1 node from the other subgraph. (d) true positive (TP), false positive (FP) and false negative (FN).

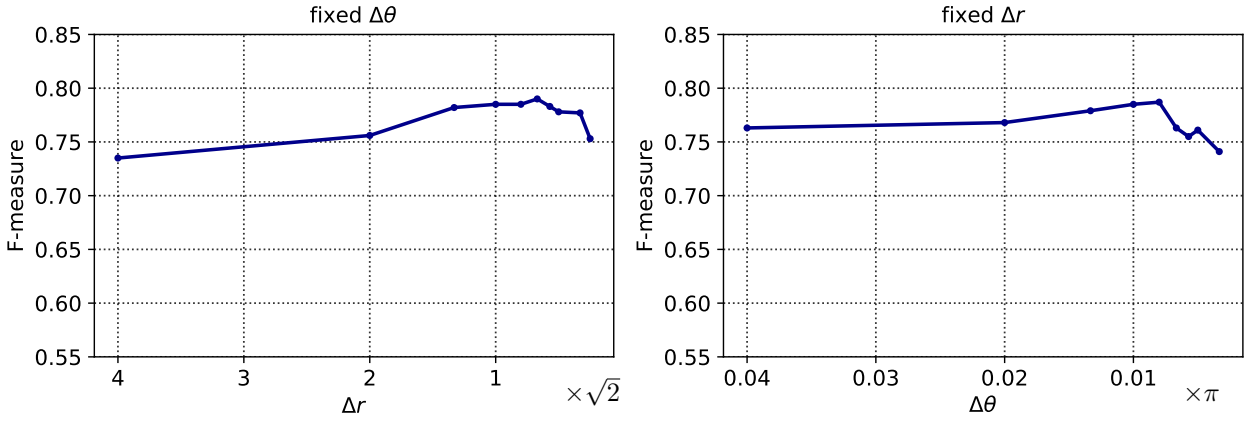

Fig. 11. Left: performance under different distance quantization intervals Δr with a fixed angular quantization interval $\Delta\theta = \pi/100$. Larger Δr indicates less quantization levels R. Right: performance under different angular quantization intervals $\Delta\theta$ with a fixed distance quantization interval $\Delta r = \sqrt{2}$.

we evaluate the performance in terms of average precision, recall, and F-measure. We use EMD [67], CD [68], and our proposed EA metric for quantitative comparisons . In the ablation study, we only use EA metric for simplicity.

6.3 Tuning the Quantization Intervals

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.

We perform a coordinate descent on SEL [11] dataset to find proper intervals that are computationally efficient and functionally effective. Note that we use the EA-score as line similarity measure since its simplicity. In the first round, we fix the angular quantization interval to $\Delta\theta = \pi/100$ and then search for Δr, the results are shown in Fig. 11(a). According to Fig. 11(a), the performance first rises slowly and then drops down with the decrease of Δr, and the turning point is near $\Delta r = \sqrt{2}$. In the second round, we fix $\Delta r = \sqrt{2}$ and train with different $\Delta\theta$. Similar to Fig. 11(a), the results in Fig. 11(b) demonstrate that the performance first increases smoothly with the drop of $\Delta\theta$, and then quickly decreases with vibration. Therefore, the turning point $\Delta\theta = \pi/100$ 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.

6.4 Quantitative Comparisons

We compare our proposed method with the SLNet [11] and the classical Hough line detection [25] with HED [41] as the edge detector. Note that we train the HED edge detector on the SEL [11] training set using the line annotations as edge ground-truth.

6.4.1 Results on SEL dataset

Tab. 2 summarizes the results on the SEL dataset [11]. With either VGG16 or ResNet50 as the backbone, Our proposed method consistently outperforms SLNet and HT+HED with a considerable margin. In addition to Tab. 2, we plot the F-measure *v.s.* threshold and the precision *v.s.* recall curves. Fig. 12 reveals that our method achieves higher F-measure than others under a wide range of thresholds.

6.4.2 Results on the NKL dataset

We report the performance of our newly constructed NKL dataset. Since SLNet [11] did not release the training code, we only compare our method with HED+HT. As shown

TABLE 2
Quantitative comparisons on the SEL [11] and NKL dataset. On SEL [11] dataset, our method (without ER) significantly outperforms other competitors in terms of average F-measure. 'CD,' 'EMD,' and 'EA' are different evaluation metrics described in Sec. 4.

Dataset	Method	CD			EMD			EA		
		Avg. P	Avg. R	Avg. F	Avg. P	Avg. R	Avg. F	Avg. P	Avg. R	Avg. F
SEL [11]	HED [41] + HT [25]	0.491	0.578	0.531	0.461	0.543	0.498	0.356	0.420	0.385
	SLNet-iter1 [11]	0.740	**0.905**	0.812	0.723	**0.888**	0.797	0.654	**0.803**	0.721
	SLNet-iter5 [11]	0.826	0.841	0.834	0.810	0.824	0.817	0.735	0.747	0.741
	SLNet-iter10 [11]	0.858	0.821	0.839	0.840	0.804	0.822	0.762	0.729	0.745
	Ours (VGG16)	0.841	0.835	0.838	0.830	0.824	0.827	0.756	0.774	0.765
	Ours (ResNet50)	**0.886**	0.815	**0.849**	**0.878**	0.807	**0.841**	**0.819**	0.755	**0.786**
NKL	HED [41] + HT [25]	0.301	0.878	0.448	-	-	-	0.213	0.622	0.318
	Ours (VGG16)	0.750	0.864	0.803	0.726	0.837	0.778	0.659	0.759	0.706
	Ours (ResNet50)	0.766	0.864	0.812	0.743	0.839	0.789	0.679	0.766	0.719

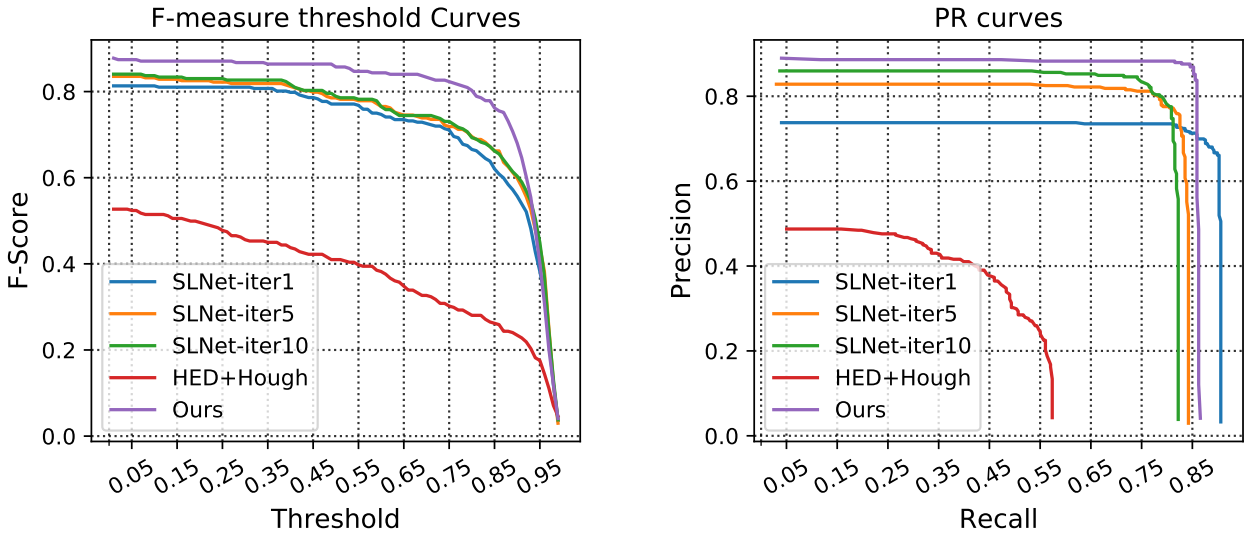

Fig. 12. Left: F-measure under various thresholds. Right: The precision-recall curve. Out method outperforms SLNet [11] and classical Hough transform [25] with a considerable margin. Moreover, even with 10 rounds of location refinement, SLNet still presents inferior performance.

in Tab. 2, our proposed method outperforms the baseline method (HED edge detector + Hough transform) with a clear margin.

6.4.3 *Runtime efficiency.*

In this section, we benchmark the runtime of different methods including SLNet [11] with various iteration steps, classical Hough transform and our proposed method.

Both SLNet [11] and HT require HED [41] edge detector 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 output results with a single forward pass, and the NMS is as simple as computing the centroids of each connected area in the parametric space.

Results in Tab. 3 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.

TABLE 3
Quantitative speed comparisons. Our method (without ER) is much faster than the other two competitors in network forward. Furthermore, our method doesn't require any extra-process *e.g.*, edge detection. As a result, our method can run at 49 FPS, which is remarkably higher than the other two methods.

Method	Network forward	NMS	Edge	Total
SLNet-iter1 [11]	0.354 s	0.079 s	0.014 s	0.447 s
SLNet-iter3 [11]	0.437 s	0.071 s	0.014 s	0.522 s
SLNet-iter10 [11]	0.827 s	0.068 s	0.014 s	0.909 s
HED [41] + HT [25]	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**

6.5 Qualitative Comparisons

Here we give several example results of our proposed method along with SNLet and HED+HT. As shown in Fig. 13, compared with other methods, our results are more compatible with the ground-truth as well as human cognition. In addition to the results in Fig. 13, we provide

Fig. 13. Example detection results of different methods on the SEL dataset. Compared to SLNet [11] and classical Hough transform [25], our results are more consistent with the ground-truth.

Fig. 14. Detection results of our method on the NKL dataset. Our method produces results that are visually compatible with human perception.

all the detection results of our method and SLNet in the supplementary materials.

6.6 Ablation Study

In this section, we ablate each of the components in our method.

6.6.1 Components in DHT

We first ablate components of "deep Hough transform". 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. 3.2.2; (c) the context-aware (CTX) line detector proposed in Sec. 3.3.1. Experimental results are shown in Tab. 4.

We first construct a baseline model with plain ResNet50 and DHT module. 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. 4 indicate that both MS and CTX

can improve the performance of the baseline model.

At last, we combine all the components 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.

TABLE 4
Ablation study for each component. MS indicates DHTs with multi-scale features as described in Sec. 3.2.2, and CTX means context-aware aggregation as described in Sec. 3.3.1.

DHT	MS	CTX	F-measure
✓			0.664
✓	✓		0.758
✓		✓	0.771
✓	✓	✓	0.786

6.6.2 Edge-guided Refinement

Here we ablate the "Edge-guided Refinement" module (abbreviated as ER). First, we test the performance of DHT+ER using different δ_r. The δ_r parameter controls the size of the searching space in ER ($\mathcal{L}$ in Eq. (7)). This experiment is conducted on the SEL dataset using the ResNet50 backbone. Results in Tab. 5 tells that the performance first increases

TABLE 5
Performance DHT+ER with different δ_r. Models are trained/tested on the SEL dataset using the Resnet50 backbone. $\delta_r = 0$ represents with vanilla DHT method without ER.

δ_r	Precision	Recall	F-measure
0	0.8190	0.7530	0.7861
1	0.8199	0.7561	0.7866
3	0.8208	0.7569	0.7874
5	0.8214	0.7574	0.7880
7	0.8213	0.7573	0.7878
9	0.8212	0.7571	0.7877

and then gets saturated with the growth of δ_r. Since the peak performance occurs when $\delta_r = 5$, we set $\delta_r = 5$ for better performance. After setting δ_r to 5, we compare the performance of our method with and without ER, using different backbones and datasets.

TABLE 6
Performance with and without ER ($\delta_r = 5$) using different backbones and datasets.

Dataset	Arch	Edge	P	R	F	F@0.95
SEL [11]	VGG16		0.756	0.774	0.765	0.380
	VGG16	✓	0.758	0.777	0.770	0.439
	Resnet50		0.819	0.753	0.786	0.420
	Resnet50	✓	0.821	0.757	0.788	0.461
NKL	VGG16		0.659	0.759	0.706	0.434
	VGG16	✓	0.664	0.765	0.711	0.472
	Resnet50		0.679	0.766	0.719	0.459
	Resnet50	✓	0.684	0.771	0.725	0.486

Results in Tab. 6 clearly demonstrate that edge-guided refinement can effectively improve detection results regardless of backbone architectures and datasets.

7 Conclusions

In this paper, we proposed a simple yet effective method for semantic line detection in natural scenes. 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. To better assess the similarity between a pair of lines, we designed a new evaluation metric considering both Euclidean distance and angular distance between lines. Besides, a new dataset for semantic line detection was constructed to fulfill the gap between the scale of existing datasets and the complexity of modern CNN models. Both quantitative and qualitative results revealed that our method significantly outperforms previous arts in terms of both detection quality and speed.

Acknowledgment

This research was supported by the National Key Research and Development Program of China (2018AAA0100400), NSFC (61922046,61620106008,62002176), S&T innovation project from Chinese Ministry of Education, and Tianjin Natural Science Foundation (17JCJQJC43700).

References

[1] Q. Han, K. Zhao, J. Xu, and M.-M. Cheng, "Deep hough transform for semantic line detection," in *Eur. Conf. Comput. Vis.*, 2020, pp. 750–766.
[2] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 4, pp. 425–455, 1986.
[3] K. Zhao, S. Gao, W. Wang, and M.-M. Cheng, "Optimizing the F-measure for threshold-free salient object detection," in *Int. Conf. Comput. Vis.*, Oct 2019, pp. 8849–8857.
[4] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 815–828, 2019.
[5] S.-H. Gao, Y.-Q. Tan, M.-M. Cheng, C. Lu, Y. Chen, and S. Yan, "Highly efficient salient object detection with 100k parameters," in *Eur. Conf. Comput. Vis.*, 2020.
[6] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Computational Visual Media*, vol. 5, no. 2, pp. 117–150, 2019.
[7] W. Wang, J. Shen, J. Xie, M.-M. Cheng, H. Ling, and A. Borji, "Revisiting video saliency prediction in the deep learning era," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 220–237, 2021.
[8] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 569–582, 2015.
[9] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 2814–2821.
[10] R. Fan, M.-M. Cheng, Q. Hou, T.-J. Mu, J. Wang, and S.-M. Hu, "S4net: Single stage salient-instance segmentation," *Computational Visual Media*, vol. 6, no. 2, pp. 191–204, June 2020.
[11] J.-T. Lee, H.-U. Kim, C. Lee, and C.-S. Kim, "Semantic line detection and its applications," in *Int. Conf. Comput. Vis.*, 2017, pp. 3229–3237.
[12] L. Liu, R. Chen, L. Wolf, and D. Cohen-Or, "Optimizing photo composition," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 469–478, 2010.
[13] M. Freeman, *The photographer's eye: composition and design for better digital photos*. CRC Press, 2007.
[14] M.-M. Cheng, X.-C. Liu, J. Wang, S.-P. Lu, Y.-K. Lai, and P. L. Rosin, "Structure-preserving neural style transfer," *IEEE Trans. Image Process.*, vol. 29, pp. 909–920, 2020.
[15] S.-M. Hu, F.-L. Zhang, M. Wang, R. R. Martin, and J. Wang, "Patchnet: a patch-based image representation for interactive library-driven image editing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–12, 2013.

[16] K. Ko, J.-T. Lee, and C.-S. Kim, "Pac-net: pairwise aesthetic comparison network for image aesthetic assessment," in *IEEE Int. Conf. Image Process.* IEEE, 2018, pp. 2491–2495.

[17] J.-T. Lee, C. Lee, and C.-S. Kim, "Property-specific aesthetic assessment with unsupervised aesthetic property discovery," *IEEE Access*, vol. 7, pp. 114 349–114 362, 2019.

[18] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes, "Photo aesthetics ranking network with attributes and content adaptation," in *Eur. Conf. Comput. Vis.* Springer, 2016, pp. 662–679.

[19] L. Mai, H. Jin, and F. Liu, "Composition-preserving deep photo aesthetics assessment," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 497–506.

[20] R. Fan, X. Wang, Q. Hou, H. Liu, and T.-J. Mu, "Spinnet: Spinning convolutional network for lane boundary detection," *Computational Visual Media*, vol. 5, no. 4, pp. 417–428, 2019.

[21] B. Krages, *Photography: the art of composition*. Simon and Schuster, 2012.

[22] S.-M. Hu, K. Xu, L.-Q. Ma, B. Liu, B.-Y. Jiang, and J. Wang, "Inverse image editing: Recovering a semantic editing history from a before-and-after image pair," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

[23] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2photo: Internet image montage," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, 2009.

[24] S.-H. Zhang, Z.-P. Zhou, B. Liu, X. Dong, and P. Hall, "What and where: A context-based recommendation system for object insertion," *Computational Visual Media*, vol. 6, pp. 79–93, 2020.

[25] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Sri International Menlo Park Ca Artificial Intelligence Center, Tech. Rep., 1971.

[26] D. Ballard, "Generating the hough transform to detect arbitary shapes," *Pattern Recogn.*, vol. 13, no. 2, 1981.

[27] P. V. Hough, "Method and means for recognizing complex patterns," 1962, uS Patent 3,069,654.

[28] L. A. Fernandes and M. M. Oliveira, "Real-time line detection through an improved hough transform voting scheme," *Pattern Recogn.*, vol. 41, no. 1, pp. 299–314, 2008.

[29] S. B. Yacoub and J.-M. Jolion, "Hierarchical line extraction," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 142, no. 1, pp. 7–14, 1995.

[30] J. Princen, J. Illingworth, and J. Kittler, "A hierarchical approach to line extraction based on the hough transform," *Computer vision, graphics, and image processing*, vol. 52, no. 1, pp. 57–77, 1990.

[31] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic hough transform," *Pattern Recogn.*, vol. 24, no. 4, pp. 303–316, 1991.

[32] C. Akinlar and C. Topal, "Edlines: A real-time line segment detector with a false detection control," *Pattern Recogn.*, vol. 32, no. 13, pp. 1633–1642, 2011.

[33] S. Caplin, *Art and Design in Photoshop*. Elsevier/Focal, 2008.

[34] Z. Zhang, Z. Li, N. Bi, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, "PPGnet: Learning point-pair graph for line segment detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Adv. Neural Inform. Process. Syst.*, 2015, pp. 91–99.

[36] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Eur. Conf. Comput. Vis.*, 2018, pp. 734–750.

[37] R. Girshick, "Fast r-cnn," in *Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[38] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7794–7803.

[39] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Int. Conf. Comput. Vis.*, 2019, pp. 603–612.

[40] 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, 2019.

[41] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Int. Conf. Comput. Vis.*, 2015, pp. 1395–1403.

[42] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[43] J. Radon, "über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten," *Classic papers in modern diagnostic radiology*, vol. 5, p. 21, 2005.

[44] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[45] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project*, 2014.

[46] Z. Chen, A. Tagliasacchi, and H. Zhang, "Bsp-net: Generating compact meshes via binary space partitioning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 45–54.

[47] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi, "Cvxnet: Learnable convex decomposition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 31–44.

[48] F. O'gorman and M. Clowes, "Finding picture edges through collinearity of feature points," *IEEE Trans. Computers*, vol. 25, no. 4, pp. 449–456, 1976.

[49] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," *Pattern Recogn.*, vol. 48, no. 6, pp. 2043–2053, 2015.

[50] J. Illingworth and J. Kittler, "The adaptive hough transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 690–698, 1987.

[51] N. Aggarwal and W. C. Karl, "Line detection in images through regularized hough transform," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 582–591, 2006.

[52] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Int. Conf. Comput. Vis.*, 2019, pp. 9277–9286.

[53] J. Min, J. Lee, J. Ponce, and M. Cho, "Hyperpixel flow: Semantic correspondence with multi-layer neural features," in *Int. Conf. Comput. Vis.*, 2019, pp. 3395–3404.

[54] A. Etemadi, "Robust segmentation of edge data," in *International Conference on Image Processing and its Applications*. IET, 1992, pp. 311–314.

[55] T. Chan and R. K. Yip, "Line detection algorithm," in *Int. Conf. Pattern Recog.*, vol. 2. IEEE, 1996, pp. 126–130.

[56] R. G. Von Gioi, 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, 2008.

[57] C. Akinlar and C. Topal, "Edcircles: A real-time circle detector with a false detection control," *Pattern Recogn. Let.*, vol. 46, no. 3, pp. 725–740, 2013.

[58] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi, and L. Chen, "Automatic fastener classification and defect detection in vision-based railway inspection systems," *IEEE T Instrum. Meas.*, vol. 63, no. 4, pp. 877–888, 2013.

[59] T. Ahmad, P. Campr, M. Čadik, and G. Bebis, "Comparison of semantic segmentation approaches for horizon/sky line detection," in *International joint conference on neural networks*. IEEE, 2017, pp. 4436–4443.

[60] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 626–635.

[61] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Int. Conf. Comput. Vis.*, 2019, pp. 962–971.

[62] N. Xue, T. Wu, S. Bai, F. Wang, G.-S. Xia, L. Zhang, and P. H. Torr, "Holistically-attracted wireframe parsing," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 2788–2797.

[63] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2117–2125.

[64] W. Liu, M. Salzmann, and P. Fua, "Context-aware crowd counting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 5099–5108.

[65] Z.-Q. Cheng, J.-X. Li, Q. Dai, X. Wu, and A. G. Hauptmann, "Learning spatial awareness to improve crowd counting," in *Int. Conf. Comput. Vis.*, 2019, pp. 6152–6161.

[66] Y.-Q. Tan, S. Gao, X.-Y. Li, M.-M. Cheng, and B. Ren, "Vecroad: Point-based iterative graph exploration for road graphs extraction," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[67] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, 2000.

[68] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, vol. 34, no. 3, pp. 344–371, 1986.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[70] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[71] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inform. Process. Syst.*, 2019, pp. 8024–8035.
[72] S.-M. Hu, D. Liang, G.-Y. Yang, G.-W. Yang, and W.-Y. Zhou, "Jittor: a novel deep learning framework with meta-operators and unified graph execution," *Science China Information Science*, vol. 63, no. 222103, pp. 1–21, 2020.
[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Represent.*, 2015.
[74] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2881–2890.
[75] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.
[76] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, 2021.
[77] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Int. Conf. Learn. Represent.*, 2016.
[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learn. Represent.*, 2015.

Kai Zhao received his B.S. and M.S. from Shanghai University. He is currently a Ph.D. Candidate with the College of Computer Science, Nankai University, under the supervision of Prof. Ming-Ming Cheng. His research interests include statistical learning and computer vision.

Qi Han is a master student from the College of Computer Science, Nankai University, under the supervision of Prof. Ming-Ming Cheng. He received his bachelor degree from Xidian University in 2019. His research interests include deep learning and computer vision.

Chang-Bin Zhang is a master student from the College of Computer Science at Nankai University, under the supervision of Prof. Ming-Ming Cheng. Before that, he received his bachelor degree from China University of Mining and Technology in 2019. His research interests include deep learning and computer vision.

Jun Xu received his B.Sc. and M.Sc. degrees from School of Mathematics Science, Nankai University in 2011 and 2014, and his Ph.D. degree from Department of Computing, The Hong Kong Polytechnic University, in 2018. He is a Lecturer with the School of Statistics and Data Science, Nankai University. His homepage is https://csjunxu.github.io/.

Ming-Ming Cheng received his PhD degree from Tsinghua University in 2012. Then he did 2 years research fellow, with Prof. Philip Torr in Oxford. He is now a professor at Nankai University, leading the Media Computing Lab. His research interests includes computer graphics, computer vision, and image processing. He received research awards including ACM China Rising Star Award, IBM Global SUR Award, CCF-Intel Young Faculty Researcher Program, *et al.* .