Pattern Recognition Letters 128 (2019) 247–254

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

LB-LSD: A length-based line segment detector for real-time applications

Yang Liu*, Zongwu Xie, Hong Liu

State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin, 150000, China

ARTICLE INFO

Article history:
Received 23 January 2019
Revised 19 August 2019
Accepted 14 September 2019
Available online 17 September 2019

Keywords:
Line segment detection
Adaptive threshold
Edge segment
Helmholtz principle
Real-time
Length condition

ABSTRACT

Line segments play an important role in the perception and representation of images by providing the geometry information about the scene. These segments can also be used as low-level features to analyze and detect more elaborated shapes. A length-based line segment detector is proposed in this paper. Based on the edge proportion statistics, an adaptive, robust and effective edge detection method is presented to extract edge segments from the image. Each of the segments is a clean, contiguous, 1-pixel wide chain of pixels. The line segment detector then approximates all the edge segments in a way that the curve satisfies a criterion of length condition. The detection result is a series of piecewise-linear segments with some dominant corners. Experimental results indicate that the proposed detector has a good detection accuracy and outperforms the state-of-the-art methods in terms of execution time.

1. Introduction

Line detection is an important and classical problem in image processing and computer vision. The line segments provide a high-level description of the objects, and thus are widely used in various applications such as compression, calibration, detection, industrial inspection and autonomous driving [1–4].

In this paper, the definition of the line segments has been adjusted to describe more geometric boundaries of the scene. In previous works, line detection results are made up of individual segments, each of which is a single straight line. In contrast, our result is a series of curves described by some dominant corners. Each curve is approximated reasonably well as aggregates of piecewise-linear segments. On the one hand, the improved line segment reduces the amount of data to be processed for the post-processing applications, on the other hand, it can describe more curves such as circles, ellipses, etc.

1.1. Literature review

Among numerous algorithms in the literature of line detection, the Hough Transform (HT) is one of the most commonly used. It is a likelihood-based parameter extraction method that maps collinear points in the image space to the intersection line in the parameter space. Thus, it is tolerant of gaps in the edge boundary as well as the image noise. However, the voting process always produces false positives on complicated regions.

There have been several approaches to improve drawbacks of HT, such as Generalized Hough Transform (GHT) [5], Randomized Hough Transform (RHT) [6], Binary Hough Transform (BHT) [7], Digital Hough Transform (DHT) [8], Progressive Probabilistic Hough Transform (PPHT) [9] and Kernel-based Hough Transform (KHT) [10].

GHT makes use of the template matching to detect predefined shapes. However, it requires a large number of elements for parallel processing, and the scale of the object should be given. RHT takes advantage of the fact that some analytical curves can be fully determined by a certain number of points on the curve. But the Hough space is defined in a heuristic way, which results in a large storage requirement and low speed for real-time applications. BHT minimizes the computation expressions to simple addition and shift operations. Unfortunately, it cannot check the connectedness of the edge elements nor detect the endpoints of the line segments. DHT adopts a pattern of judging whether a given set of edge points form a straight line or not. The detection accuracy has been significantly improved on low-resolution images, but it decreases largely on high-resolution or noisy images. PPHT accelerates the calculation of HT by the selection of random edge points. Nonetheless, the cut-off values are set for a predefined image size, and the guard against false positives is not ensured on a larger image. KHT presents an efficient voting scheme that mod-

* Corresponding author.
E-mail address: liuyanghit@hit.edu.cn (Y. Liu).

https://doi.org/10.1016/j.patrec.2019.09.011
0167-8655/

els the uncertainty associated with the best-fitting line. It produces a much cleaner accumulator and is more robust to the spurious lines. More variations of HT can be found in [11,12].

In most cases, these methods suffer from inherent limitations such as long computation times, large memory requirements [13]. Besides, they always fail to provide the length or the end points of the line segments. By analyzing the spread of votes in the accumulator array, the endpoints can be solved with the simultaneous equations [14]. However, how to get the first and the last nonzero voting cells is a hard problem in the presence of image noise. Graphics Processing Unit (GPU) is introduced in [15] to accelerate the running time of HT with a parallel process. Based on the minimum entropy [16], a line segment can be determined, and the results can be further improved by combining the voting analysis in image space and the voting distribution in Hough space [17]. However, only one segment can be found per line, whereas multiple collinear segments are quite common in real images. Besides, the latest techniques such as neural networks [18] and neuromorphic sensors [19] have also been introduced to new implementations.

Instead of using the binary edge map, some methods fit lines with the gradient information. Based on the gradient orientations, Burns [20] presents a novel method that organizes the spatial extent of a straight line without the meaningfulness of an edge event. According to the Helmholtz principle, Desolneux [21,22] proposes a parameter-less line detector. It succeeds in controlling the number of false positives without any priori information. Line Segment Detector (LSD) [23] produces accurate line segments and controls the number of false detections in a low level by efficiently combining gradient orientations and the line validation. However, the accuracy decreases largely in noisy images with complex background. Edge Drawing Lines (EDLines) [24] walks over each edge segment in sequence and fits lines to the pixels using the least square method. The line segments in CannyLines [25] are extended in both directions to collect more edge pixels and merged with other collinear line segments around, but the effect of its promotion is not extremely evident. Recently, machine learning based approaches [26,27] have been proposed to extract the geometric boundaries directly from the image. As for the man-made environments, Huang [28] presents a learning-based method to detect the wireframe representation of the scene without relying on the low-level features.

1.2. Introduction to the proposed method

In this paper, a length-based line segment detector (LB-LSD) for real-time applications is proposed. Based on the edge proportion statistics, it first extracts the edge segments from the image. Then, it approximates the edge segments with aggregates of piecewise-linear segments according to the length condition. The detection result is a series of segments with some dominant corners that hold the basic framework of the scene.

The rest of the paper is organized as follows: Section 2 presents an adaptive, robust and effective edge detector. Section 3 details the LB-LSD method which fits lines due to the length condition rather than the least squares errors. It approximates lines under the assumption that the curve satisfies some straightness criterions. Section 4 discusses the length threshold value according to the Helmholtz principle. Section 5 comments on the experimental results, and Section 6 concludes the paper.

2. Adaptive edge detection

Nowadays, most of the line segment detectors adopt an edge detection scheme to get the binary edge map at first. It will significantly reduce the amount of data and preserve the structural properties for the post-processing applications.

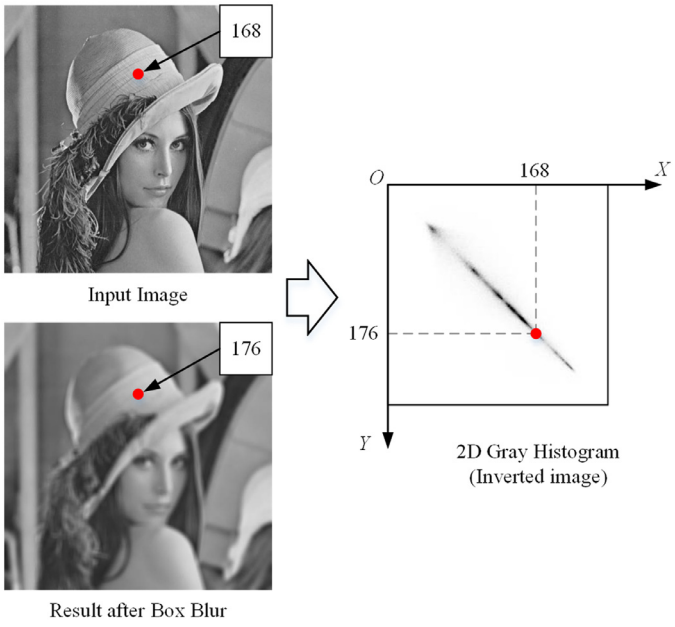

Fig. 1. 2D Gray Histogram of Lena. The horizontal axis is the gray scale and the vertical axis is the average value from the neighborhood.

In this section, an adaptive and robust edge segment detector is proposed. According to the two-dimensional entropy, the images can be classified into three groups, each attached with a reference value based on the edge proportion statistics.

2.1. Two-dimensional entropy

The two-dimensional entropy model not only cares about the gray values of the image but also takes the spatial distribution into account [29]. The higher value the entropy is, the more information the image contains.

Considering a digital image with the size of $M \times N$, let L be its gray level that equals to 256 in this paper. For each point, both the gray intensity I_i and the average value I_j from the neighborhood are calculated. The frequency of each gray pair (I_i, I_j) is then stored in $f_{i,j}$.

As shown in Fig. 1, the gray intensity of the marked point is 168, and a 11 $\times$ 11 box filter is used to get the average value from the neighboring pixels. Then, a new pair (168, 176) is formed and added to the 2D gray histogram.

Moreover, the probability mass function (PMF) $p_{i,j}$ is defined with respect to the total pixels of the image:

$$p_{i,j} = f_{i,j}/MN, i, j = 0, 2, \ldots, 255 \tag{1}$$

Finally, the two-dimensional entropy is referred to (Event with probability zero does not contribute to the entropy result):

$$H = -\sum_{0}^{L-1}\sum_{0}^{L-1} p_{i,j} \log p_{i,j} \tag{2}$$

2.2. Adaptive edge proportion

In general, the edge proportion of one image should be neither too large nor too small. The image becomes disordered with the rise of the proportion. Whereas it contains little information when the proportion is small.

An adaptive threshold method based on the edge proportion statistics is proposed here. To this end, we make use of the Berkeley Segmentation Dataset and Benchmark (BSDS500) [30] and its precision-recall evaluation framework. It consists of 500 images

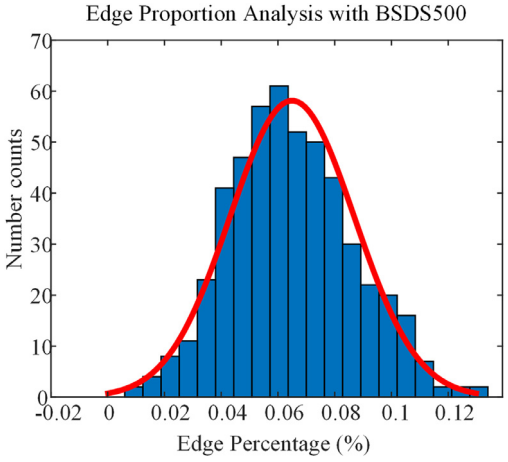

Fig. 2. Edge proportion statistics in BSDS500.

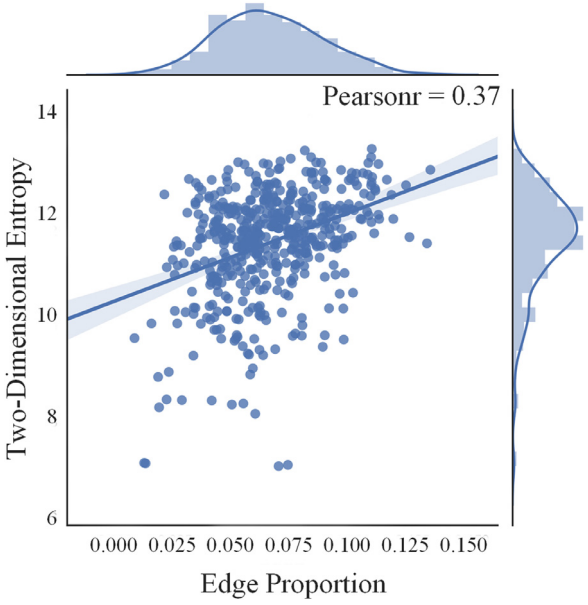

Fig. 3. Linear correlation between edge proportion and 2D entropy.

with human annotated ground truth boundaries. The image size is 481 × 321 or 321 × 481.

As shown in Fig. 2, the edge proportion trends to a normal distribution, which implies that the edge proportion corresponds to a statistical principle despite the diversity of the images. In most cases, the percentage of edge pixels in one image is less than 10%. This assumption will be helpful in the determination of gradient threshold.

The Pearson correlation coefficient is a measure of the linear correlation between two variables. Fig. 3 is a multi-panel figure that shows both the bivariate relationship between the edge proportion and the 2D entropy along with the univariate distribution.

The correlation coefficient indicates a moderate positive linear relationship between the image entropy and the edge proportion. The more information the image contains, the larger the entropy becomes as well as the number of edge pixels. Therefore, the images can be classified into three groups according to the two-dimensional entropy, as listed in Table 1.

Table 1
Edge percentage determination based on the image information entropy.

	Low	Moderate	High
Two-dimensional entropy	[0, 10.8)	[10.8, 12.7]	(12.7, +∞)
Recommended edge percentage	4.3%	6.5%	8.7%

2.3. Edge segment detection

Anchors are the peaks on the gradient map where the edges will probably put over. Compared with the attached points along the gradient direction, one point can be marked as an anchor if its gradient is the largest among these three points.

Then, the segment direction is introduced, which represents the trajectory of one segment. The definition is different from the gradient direction as it is related to the connection of the neighbor pixel. For example, if the next maximum gradient point is on the left of the current one, its direction is left.

Based on the hypothesis that an edge segment always has a continuous curvature until to the sharp corners, a curvature predictive method is proposed. We assume the segment direction of the current point is equal to the former one and check the next point along this direction as well as two linked neighbor points. If the previous segment direction is right, the right, up-right and down-right pixels will be taken into consideration. Pixel with the largest gradient value is regarded as the next edge point, and the segment direction updates as well.

Next, we will connect the anchors together into consecutive edge segments. Each segment is a chain of contiguous pixels. The coordinates as well as the segment directions are collected for the post-processing applications. Details of the proposed edge linking method are as follows:

Algorithm 1 Edge linking method.

Input: Anchor Points $P_i = \{P_1, P_2, ..., P_m\}$, edge map
1: Sort the anchor points by the gradient in descending order
2: **for** $i = 1$ to m **do**
3: Start two edge segments S_p and S_q with opposite directions from P_i
4: **while** *EndFlag* ≠ True **do**
5: Add the next edge point on S_p/S_q based on the Curvature Predictive Method
6: **end while**
7: **if** $length(S_p) + length(S_q) \geq Th_l$, **then**
8: Joint S_p and S_q into one edge segment S_j
9: **if** $EdgePointNum > Th_n$, **then**
10: break
11: **end if**
12: **end if**
13: **end for**
Output: Edge Segments $S = \{S_1, S_2, ..., S_n\}$

In descending order of the gradient, we start two edge segments with opposite directions from each anchor point. Based on the curvature predictive method, each segment stretches until it moves out of the edge areas. Once an edge pixel is detected, it will be removed from the edge map. The end flag *EndFlag* turns to be true when there is no edge pixel in the following candidates. After that, the total length is calculated and compared with the length threshold Th_l. Edge segments whose length are shorter than the given threshold will not be reserved in the final list. Once the total edge number *EdgePointNum* exceeds the recommended threshold Th_n, the edge linking process breaks up. Th_n is the product of the image size and the recommended edge percentage in Table 1. The detection result is a collection of edge segments, each of which is a clean, contiguous, 1-pixel wide chain of pixels.

Table 2
Five short edge segments with directions counted.

Type					
n_h	3	2	1	0	0
n_v	0	0	0	0	3
n_o	0	1	2	3	0

3. Line segment detection

In this section, a length-based line segment detector is introduced based on the length condition rather than least squares errors. On the one hand, it can reduce the number of edge points while keeping the desired accuracy; on the other hand, it avoids the calculation of the linear equations.

3.1. Length condition

For an arbitrary edge segment, the segment direction of each point has been defined in the above section. These edge points can be classified into three groups: horizontal (left and right), vertical (up and down) and oblique (up-right, down-right, down-left and up-left). Three parameters are then defined to count the numbers of edge points in each group: n_h, n_v and n_o. Notice that the segment direction of the last point is indeterminate, which is referred to null in this paper. The physical meaning of these parameters is related to the projection distance onto different directions. n_h represents the width between the start and end points along the x axis, and n_v represents the height information along the y axis. Besides, n_o indicates that the edge projection holds n_o pixels on both axes.

Table 2 illustrates some short edge segments with specific parameters. These segments start from up-left to down-right, with the last points marked in gray. Using the segment directions, the estimated length (distance between the center of the start and end points) is:

$$D_r = \sqrt{(n_h + n_0)^2 + (n_v + n_0)^2} \tag{3}$$

In addition, we can also use the Euler distance to represent the length of the current segment:

$$D_d = \|P_s - P_e\| = \sqrt{(x_s - x_e)^2 + (y_s - y_e)^2} \tag{4}$$

Obviously, D_d should be equal to D_r if the segment is a straight line. Due to the quantization error, there is always a deviation between these two values, which is referred to:

$$E_d = D_d^2 - D_r^2 \tag{5}$$

As illustrated by Fig. 4, the more one segment departs from the straight line, the larger D_r is, while the Euler distance D_d is in the opposite situation. In this case, one segment can be regarded as a straight line if the deviation is shorter than a given threshold Th_d.

Taking one segment with l edge points as an example, it has the following features:

1. D_r has nothing to do with the direction consistency. It is only related to the pixel numbers on different directions.
2. D_r increases with n_o and has a maximum value $\sqrt{2}(l-1)$ when the segment is a straight line along the diagonal. It drops to the minimum $\sqrt{2}l/2$ when n_h equals to n_v. Note that the last point is omitted during length calculation and there is always a diagonal pixel when the segment direction changes (otherwise there will be a multi-pixel wide edge).

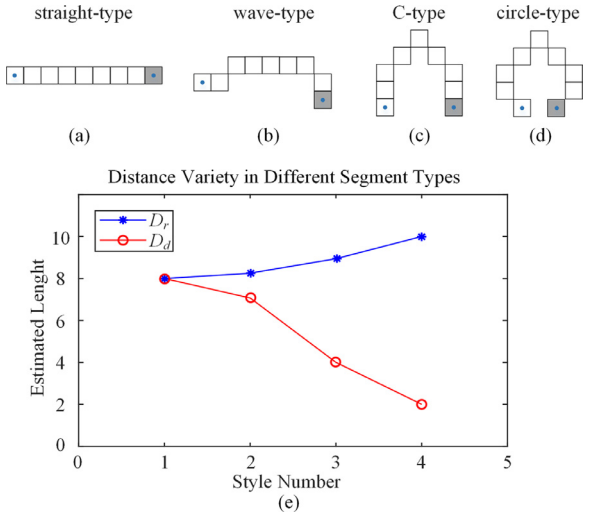

Fig. 4. Deviation tendency over different edge segments.

Table 3
False detection based on the length condition.

Item	n_h	n_v	n_o	D_r	D_d	Result
	4	3	1	5	5	False
	2	1	3	5	5	True

3. D_d measures the Euler distance between the start and end points. Hence the minimum value reaches to 1 pixel when the edge segment forms a circle. Besides, it has the same maximum as D_r.

3.2. False detection

For an arbitrary edge segment, the length deviation is checked pixel by pixel from the start point. When it exceeds the length threshold, the current point is marked as a corner and the past points will be removed from the edge segment. The algorithm then recursively processes the remaining pixels until to the end.

However, this assumption is not always right. Sometimes the detected result is far from a straight line even if the deviation equals to zero. As listed in Table 3, there are two short segments with specific details. The length condition shows that both segments are straight lines. However, the upper one is a false detection which should be split at the corner while the lower is possibly true. To solve this problem, the distribution of the segment directions is also taken into consideration in this paper.

The line segment detection is detailed in Algorithm 2. The detection result is a series of curves made up by some dominant corners rather than continuous points. Each curve is approximated reasonably well as aggregates of piecewise-linear segments. The initial direction proportions are computed by the first l_0 points, and the length deviation as well as the direction distribution is analyzed pixel by pixel. Then, each edge segment can be described by some discrete points, among which are a list of short lines. This line detection method can be regarded as a consecutive linear fitting progress that holds the outline information while cutting down the redundant points. It neither needs to fit lines through the least square method, nor cares about the gradients. Thus, the execution time is much less than the traditional methods.

Algorithm 2 Line segment detection method.

Input: Edge segments $E=\{E_1, E_2,..., E_n\}$
1: **for** $i=1$ to n **do**
2: Generate a new line segment and push the first point in this segment
3: Skip next l_0 consecutive pixels. l_0 is a parameter stands for the initial length
4: Compute the direction proportions of n_h and n_v on the initial length
5: **for** $j=1$ to m **do**
6: Calculate D_d, D_r and E_d to the previous corner point
7: Compute the ideal numbers of n_{h0} and n_{v0} with the initial proportions
8: **if** abs(E_d) > Th_d or abs(n_{h0} - n_h) > Th_s or abs(n_{v0} - n_v) > Th_s **then**
9: Mark the current point as a new corner point
10: Skip next l_0 consecutive pixels and update the direction proportions
11: **end if**
12: **end for**
13: Push the last point in this segment
14: **end for**
Output: Line Segments $S=\{S_1, S_2,..., S_n\}$

Here, l_0 is the initial length of a straight line, and the definition is given in Eq. (11). The details about how to select a suitable value are discussed in Section 4. Th_d and Th_s are two thresholds used to transform the edge segments into short lines. A larger value will decrease the corner points on the line segments as well as the fitting accuracy, whereas a smaller value will result in redundant lines. Th_d defines the length deviation between the two estimated distances with the recommended value of 0.01 × image diagonal. Th_s is a threshold about distribution of segment directions with the default value of 3 pixels.

4. Length validation

In the line segment detection step, the initial number of points l_0 has a great influence on the detection results. A smaller value will lead to the wrong distribution due to the image noise. Whereas false detection (as displayed in Table 3) appears when there are excessive points in the initial segment.

In this paper, the Helmholtz principle is introduced to measure the initial length of a line segment. For an arbitrary geometric structure, it is perceptually meaningful if the expectation by chance is small in a random situation. Without any prior experience, the segment validations are carried out under a contrario random assumption, which is also known as the "Number of False Alarms" (NFA). And NFA has been popularly applied in the detection of image structures like lines, circles and vanishing points [21,23,31,32].

For each point, X_i denotes that whether the segment direction is aligned with the attached segment. The probability of this event is given by:

$$P(X_i = 1) = p \quad and \quad P(X_i = 0) = 1 - p \tag{6}$$

where p is the probability of a point to have the same direction with the segment, which equals to 1/8 in this paper. Thus, the direction consistency is defined as:

$$S_l = X_1 + X_2 + \cdots + X_l \tag{7}$$

When $S_l = l_0$, all the points seem to have the same segment direction, which means the current segment is a straight line. However, the directions have a random distribution in most cases. Then we take the following event into consideration: there are at least k points whose directions are aligned with the segment. The probability of this event can be computed using the binomial distribution:

$$P(k, l_0) = P(S_l \geq k) = \sum_{i=k}^{l_0} B(l_0, i) p^i (1-p)^{l_0 - i} \tag{8}$$

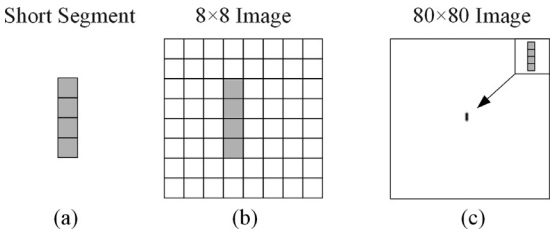

Fig. 5. Evolution of a short line aegment.

Finally, the NFA is formulated as Eq. (9), which is associated with the number of possible configurations for a given event.

$$NFA = N_{conf} \cdot P(k, l_0) \tag{9}$$

where N_{conf} is the theoretical number of lines in the image. In general, a straight line can be described by the start and end points. The start point has WH potential positions in a $W \times H$ image, and the end point can be in any of the other $WH - 1$ points. Therefore, there is a total of $N_{conf} \approx (WH)^2$ line segments.

An event is called ε-meaningful if the expectation under a contrario random assumption is less than ε. From Eqs. (8) and (9), the ε-meaningful segment satisfies the following inequality:

$$p^{l_0} \leq P(S_l \geq k) \leq \frac{\varepsilon}{(WH)^2} \tag{10}$$

And the minimum length of 1-meaningful line segment is

$$l_0 = -2 \log (WH) / \log (p) \tag{11}$$

Here, the minimum length of l_0 is only related to the image size. This is due to the fact that the human visual system focuses on the relative size rather than the physical appearance [33]. Taking one short segment with four points as an example, the meaningful image should include no more than 64 pixels. The resolution of the source image is 8 × 8 if it has a 1:1 aspect ratio, leading to the easily visible pixels.

As shown in Fig. 5(b), the short segment located in the middle of the image seems like number "1". Thus, it is regarded as a meaningful line which possibly comes from a flat boundary. One equal segment is displayed on a white image with decuple size in Fig. 5(c), which looks more like a narrow dot. This segment is less meaningful than the previous one as it is easily affected by the noise. Note that this short segment is still meaningful in applications such as infrared image or space rendezvous and docking system, where there is always a pure background.

5. Experimental results

In this section, we evaluate the performance of our algorithm by contrast with the state-of-the-art methods, i.e., LSD [23], EDLines [24] and NFA + RNFA [32]. All the source codes of the methods are publicly available online. And the following experiments are implemented by the C++ programming language and tested on a PC (I3-3240 at 3.4 GHz, 4 GB of Ram) with Win10. Neither parallel speedup nor GPU acceleration is used for time comparison.

The LSD detector applies a region growing method to obtain a line-support region. Then, the Helmholtz principle is applied to validate the meaningfulness of this region according to the number of aligned orientations. Default control parameters are used for comparison (the maximal possible error in gradient is set to 2; the gradient angle tolerance is 22.5°; the suggested number of bins used in the pseudo-ordering is 1024, etc.). EDLines makes use of the ED detector to extract edge segments and then detect line segments based on the least square method. All the parameters are set to the default values (including the Edge Drawing step). The

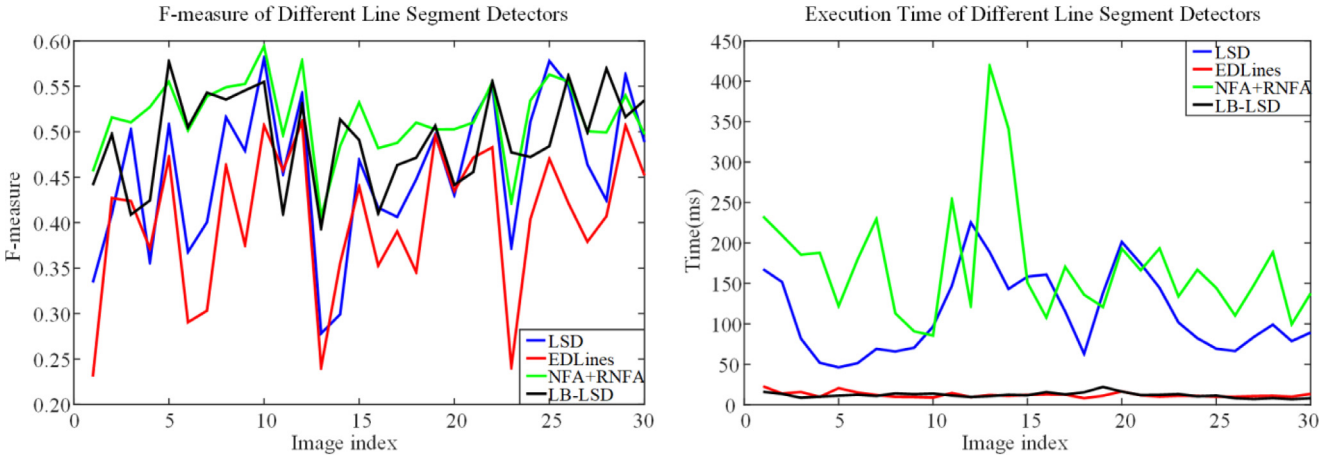

Fig. 6. Performance analysis of different line segment detectors on YorkUrbanDB.

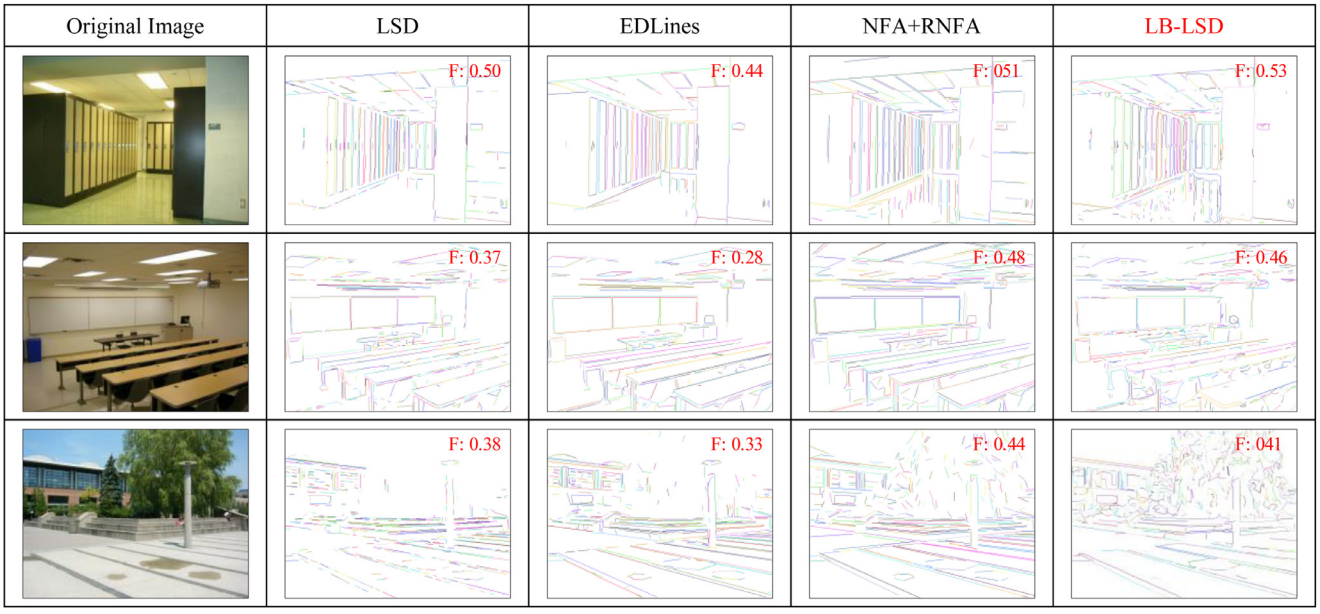

Fig. 7. Line Segment detection results on improved YorkUrbanDB.

NFA+RNFA method combines NFA and RNFA to double check the line segments, which results in fewer false alarms in noisy images. Besides, the recommended values are used for fair comparison.

In order to evaluate the performance of the proposed method, we make use of the improved YorkUrbanDB dataset [32], which is a subset from the YorkUrbanDB dataset [34]. It consists of 30 colored images with the same size 640 × 480 and covers a range of indoor and outdoor scenes.

In this paper, the performance of the presented method is evaluated in terms of the detection efficiency and execution time. The detection efficiency is assessed by the F-measure. A higher value indicates a better result of the algorithm.

Let LS be the set of line segments detected by a certain method, GT denote that of the ground truth data, the precision (P) and recall ratio (R) are defined as follows:

$$P = \frac{LS \cap GT}{GT} \quad \text{and} \quad R = \frac{LS \cap GT}{LS} \tag{12}$$

Then, F-Measure is defined as $F = 2PR/(P+R)$, which can be used for quantitative comparisons.

Table 4
Performance of different line segment detection methods on YorkUrbanDB.

Method	LSD	EDLines	NFA+RNFA	LB-LSD
F-measure	0.457^{-}	0.404^{-}	0.515^{+}	0.493
Time (ms)	112.82	12.27	171.36	11.97

Fig. 6 shows the performance of all the above-mentioned methods on the improved YorkUrbanDB dataset. All the line segment detectors have been tested for ten times to get the average time, and the specific values are reported in Table 4. As one can see, our method performs the best in running time. EDLines takes a similar framework to detect lines: it first extract edge segments from the image and then fit lines according to least squares errors. Thus, it has a similar performance in terms of running time. In contrast, both the LSD and NFA+RNFA methods take the grayscale images as input directly, leading to a large amount of time.

The NFA+RNFA method performs the best detection accuracy, but it is hardly to implement in real-time. The proposed method is in the second place, mainly due to the false detections from short edge contours. In addition, statistical significance tests have also been performed to compare these methods by a one sample t-test with significance level 0.05. Symbols "$^{-/+}$" in the first row means that the current line detector is statistically significantly worse/better than our proposed method.

Several detection results are displayed in Fig. 7. The detection accuracy decreases in outdoor images with noisy background. Taking the last image as an example, the trees lead to a lot of false positives among different methods. LSD has a good performance since it takes a line-support region to extract straight lines. EDLines tends to cut a segment into small fragments in the low contrast area. The NFA+RNFA method outperforms other detectors in most cases with little false alarms. Our result contains some jagged edges and makes a few false positives from the background.

Unlike the other methods, LB-LSD pays more attention on the pixel's connection. The detection result is a series of curves described by some dominant corners, among which is a collection of straight lines. On the one hand, our method reduces the amount of data for further processing, on the other hand, it keeps the basic framework of the image. As a result, the LD-LSD method is more than a fast and effective line segment detector. It also preserves the shape information from the image, which is of great importance in the post-processing applications such as face detection, ellipse fitting, etc.

6. Conclusion

In this paper, a length-based line segment detector for real-time applications is proposed. Based on the edge proportion statistics, an adaptive and robust edge detector is introduced to extract edge segments, each of which is a clean, contiguous, 1-pixel wide chain of pixels. Then, the length condition is taken into consideration to approximate the corners in a way that the curve satisfies some straightness criterions. Finally, the initial length control parameter l_0 is analyzed due to the Helmholtz principle. The detection result is a series of segments with discrete corners, among which is a collection of straight lines.

To evaluate the accuracy and efficiency of the proposed method, we have compared our detector with the state-of-the-art methods on the improved YorkUrbanDB dataset. Experimental results indicate that our method outperforms the others in terms of the execution time. Besides, it approximates the objects with some piecewise-linear segments, which preserves the shape information for post-processing applications. On the one hand, LB-LSD reduces the number of edge points while keeping the desired accuracy; on the other hand, it avoids the calculation of the linear equations. As further improvements, these edge segments can be used in many high-level applications such as ellipse detection, image matching and object detection.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This research was supported by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Nos.51521003 and 61690210) and the Self-Planned Task (No.SKLRS201805B) of State Key Laboratory of Robotics and System (HIT).

References

[1] H. Bay, V. Ferrari, L. Van Gool, Wide-Baseline Stereo Matching with Line Segments, in: 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., IEEE, n.d.: pp. 329–336. doi:10.1109/CVPR.2005.375.
[2] H. Kim, S. Lee, Simultaneous line matching and epipolar geometry estimation based on the intersection context of coplanar line pairs, Pattern Recognit. Lett. 33 (2012) 1349–1363, doi:10.1016/J.PATREC.2012.03.014.
[3] R. Stoica, X. Descombes, J. Zerubia, A gibbs point process for road extraction from remotely sensed images, Int. J. Comput. Vis. 57 (2004) 121–136, doi:10.1023/B:VISI.0000013086.45688.5d.
[4] Yefeng. Zheng, Huiping. Li, D Doermann, A parallel-line detection algorithm based on HMM decoding, IEEE Trans. Pattern Anal. Mach. Intell 27 (2005) 777–792, doi:10.1109/TPAMI.2005.89.
[5] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (1981) 111–122, doi:10.1016/0031-3203(81)90009-1.
[6] L. Xu, E. Oja, P Kultanen, A new curve detection method: Randomized Hough transform (RHT), Pattern Recognit. Lett. 11 (1990) 331–338, doi:10.1016/0167-8655(90)90042-Z.
[7] L. da Fontoura Costa, M.B. Sandler, A complete and efficient real time system for line segment detection based on the binary Hough transform, in: Proceedings. EUROMICRO '90 Work. Real Time, IEEE Comput. Soc. Press, n.d.: pp. 205–213. doi:10.1109/EMWRT.1990.128252.
[8] N. Kiryati, M. Lindenbaum, A.M. Bruckstein, Digital or analog Hough transform? Pattern Recognit. Lett. 12 (1991) 291–297, doi:10.1016/0167-8655(91)90412-F.
[9] C. Galamhos, J. Matas, J. Kittler, Progressive probabilistic Hough transform for line detection, in: Proceedings. 1999 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (Cat. No PR00149), IEEE Comput. Soc, n.d.: pp. 554–560. doi:10.1109/CVPR.1999.786993.
[10] L.A.F. Fernandes, M.M. Oliveira, Real-time line detection through an improved Hough transform voting scheme, Pattern Recognit. 41 (2008) 299–314, doi:10.1016/J.PATCOG.2007.04.003.
[11] A.S. Hassanein, S. Mohammad, M. Sameer, M.E. Ragab, A Survey on Hough Transform, Theory, Techniques and Applications, 2015 http://arxiv.org/abs/1502.02160 accessed May 22, 2019.
[12] P. Mukhopadhyay, B.B. Chaudhuri, A survey of Hough transform, Pattern Recognit. 48 (2015) 993–1010, doi:10.1016/j.patcog.2014.08.027.
[13] C. Shao, Q. Ding, H. Luo, Z. Chang, C. Zhang, T. Zheng, Step-by-step pipeline processing approach for line segment detection, IET Image Process 11 (2017) 416–424, doi:10.1049/iet-ipr.2016.0493.
[14] M. Atiquzzaman, M.W. Akhtar, A Robust Hough transform technique for complete line segment description, Real-Time Imaging 1 (1995) 419–426, doi:10.1006/RTIM.1995.1043.
[15] T. Zhang, Z. Liu, N. Ouyang, Real-time line segments detection based on graphic processor: {Real}-time line segments detection based on graphic processor, J. Comput. Appl. 29 (2009) 1359–1361, doi:10.3724/SP.J.1087.2009.01359.
[16] Zezhong. Xu, Bok-Suk. Shin, R Klette, Accurate and robust line segment extraction using minimum entropy with hough transform, IEEE Trans. Image Process. 24 (2015) 813–822, doi:10.1109/TIP.2014.2387020.
[17] Z. Xu, B.-S. Shin, R. Klette, Closed form line-segment extraction using the Hough transform, Pattern Recognit. 48 (2015) 4012–4023, doi:10.1016/j.patcog.2015.06.008.
[18] M.W. Spratling, A neural implementation of the Hough transform and the advantages of explaining away, Image Vis. Comput. 52 (2016) 15–24, doi:10.1016/j.imavis.2016.05.001.
[19] D. Reverter Valeiras, X. Clady, S.-H. Ieng, R. Benosman, Event-based line fitting and segment detection using a neuromorphic visual sensor, IEEE Trans. Neural Networks Learn. Syst. 30 (2019) 1218–1230, doi:10.1109/TNNLS.2018.2807983.
[20] J.B. Burns, A.R. Hanson, E.M. Riseman, Extracting straight lines, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (1986) 425–455, doi:10.1109/TPAMI.1986.4767808.
[21] A. Desolneux, L. Moisan, J.-M. Morel, Meaningful alignments, Int. J. Comput. Vis 40 (2000) 7–23, doi:10.1023/A:1026593302236.
[22] A. Desolneux, L. Moisan, J.-M. Morel, From Gestalt Theory to Image Analysis, Springer New York, New York, NY, 2008, doi:10.1007/978-0-387-74378-3.
[23] R.G. von Gioi, J Jakubowicz, J.-M Morel, G. Randall, LSD: a fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell 32 (2010) 722–732, doi:10.1109/TPAMI.2008.300.
[24] C. Akinlar, C. Topal, EDLines: A real-time line segment detector with a false detection control, Pattern Recognit. Lett 32 (2011) 1633–1642, doi:10.1016/J.PATREC.2011.06.001.
[25] X. Lu, J. Yao, K. Li, L. Li, CannyLines: a parameter-free line segment detector, in: 2015 IEEE Int. Conf. Image Process, IEEE, 2015, pp. 507–511, doi:10.1109/ICIP.2015.7350850.
[26] S. Xie, Z. Tu, Holistically-nested edge detection, in: 2015 IEEE Int. Conf. Comput. Vis, IEEE, 2015, pp. 1395–1403, doi:10.1109/ICCV.2015.164.
[27] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, L. Van Gool, Convolutional oriented boundaries, (2016). doi:10.1007/978-3-319-46448-0_35.
[28] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, Y. Ma, Learning to parse wireframes in images of man-made environments, in: 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit, IEEE, 2018, pp. 626–635, doi:10.1109/CVPR.2018.00072.

[29] S. Zhu, Y. Liu, Two-dimensional entropy model for video shot partitioning, Sci. China Ser. F Inf. Sci 52 (2009) 183–194, doi:10.1007/s11432-009-0057-1.
[30] P. Arbeláez, M. Maire, C. Fowlkes, J Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 898–916, doi:10.1109/TPAMI.2010.161.
[31] C. Akinlar, C. Topal, EDLines: a real-time line segment detector with a false detection control, Pattern Recognit. Lett 32 (2011) 1633–1642, doi:10.1016/J.PATREC.2011.06.001.
[32] X. Lu, J. Yao, L. Li, Yahui. Liu, Wei. Zhang, Edge chain detection by applying Helmholtz principle on gradient magnitude map, in: 2016 23rd Int. Conf. Pattern Recognit, IEEE, 2016, pp. 1364–1369, doi:10.1109/ICPR.2016.7899827.
[33] S. Tong, Y.P. Loh, X. Liang, T. Kumada, Wide or narrow? A visual attention inspired model for view-type classification, IEEE Access 7 (2019) 48725–48738, doi:10.1109/ACCESS.2019.2908856.
[34] P. Denis, J.H. Elder, F.J. Estrada, in: Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery, Springer, Berlin, Heidelberg, 2008, pp. 197–210, doi:10.1007/978-3-540-88688-4_15.