Pattern Recognition 113 (2021) 107834

Contents lists available at ScienceDirect

Pattern Recognition

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

AG3line: Active grouping and geometry-gradient combined validation for fast line segment extraction

Yongjun Zhang*, Dong Wei, Yansheng Li

School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China

ARTICLE INFO

Article history:
Received 5 June 2019
Revised 17 November 2019
Accepted 16 January 2021
Available online 20 January 2021

Keywords:
Line extraction
Active grouping
Anchor map
Alignment of gradient magnitude

ABSTRACT

Line segment detectors based on local image domain passively fit a line segment from a set of pixels, but no constraint on line geometry is set in the grouping process. Therefore, unstable pixels, such as the pixels in grass, clouds, or weak gradient edges, may cause false positives and fractures. This paper proposes the detector named AG3line, which employs an efficient active grouping strategy. In AG3line, the pixel for the next grouping is calculated actively with the line geometry and it can even be accurate to one pixel. To reduce the fracture caused by unstable pixels, when the adjacent pixel cannot satisfy the grouping rules, the candidate pixels for the next grouping are expanded with the line geometry constraint. To furtherly control false positives, AG3line then validates and refines the line segments by exploiting both the line geometry and the alignment of gradient magnitude. When AG3line was evaluated utilizing the image dataset with the ground truth, it outperformed both the classical and the latest detectors.The implementation of AG3line is available at https://github.com/weidong-whu/AG3line.

1. Introduction

Line segments are essential for a wide range of algorithms in computer vision, such as image registration [1], vanish point detection [2], 3D reconstruction [3], and object detection [4]. The chosen line segments are expected to be complete with no false positives; thus, in some applications, the short line segments are merged [4] or discarded [5]. The primary concern in the line segment extraction research field is finding a way to provide more comprehensive extraction of line segments with fewer false positives.

Hough Transform (HT) [6] extracts straight lines that are appropriate in geometric image parsing by converting line detection into a problem of peak detection in the parameter space [7]. Many improved Hough detectors have been proposed, e.g., probabilistic HT [8]; randomized HT [9] and its extension[10]; progressive probabilistic HT (PPHT) [11], which reduced the voting of pixels in Hough space; the weight method [12]; and Kernel-Based Hough Transform (KHT) [13], which improved peak detection. Since these Hough methods cannot extract accurate endpoints, Du et al. [14] exploited the first and last non-zero voting cells in the extraction process, and Xu et al. [15] extracted the endpoints by minimum entropy analysis in the Hough space. Xu et al. [16] also proposed a closed-form solution for complete line-segment extraction, which was unable to achieve satisfactory results in complex real images. Recently, Almazan et al. [17] proposed a dynamic programming approach (MCMLSD) based on PPHT, which unfortunately produces many false positives in complex real images. Therefore, Hough-based detectors are now considered more appropriate for extracting straight lines than line segments with explicit endpoints.

To accurately obtain endpoints, some algorithms first group the pixels in a local image domain. The grouping is based on the alignment of the gradient orientation or the connection of the gradient peaks. Then, the pixels in the same group are fitted into a line segment.

Burns et al. [18] proposed the first line segment detector based on the alignment of the gradient orientation. To control false positives, Desolneux et al. [19] proposed the Helmholtz principle to validate the line segment. Although the authors' validation was innovative, it runs slowly and extracts line segments with poor accuracy. In a follow-up study, Grompone et al. [20] improved the speed of the Helmholtz principle method; and then several years later, they proposed an algorithm named LSD [21,22], which is a linear-time detector and is currently viewed as the state of the art method for line segment extraction. Recently, Cho et al. [23] proposed a new line segment detector named Linelet, which performs

* Corresponding author.
E-mail address: zhangyj@whu.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.patcog.2021.107834
0031-3203/

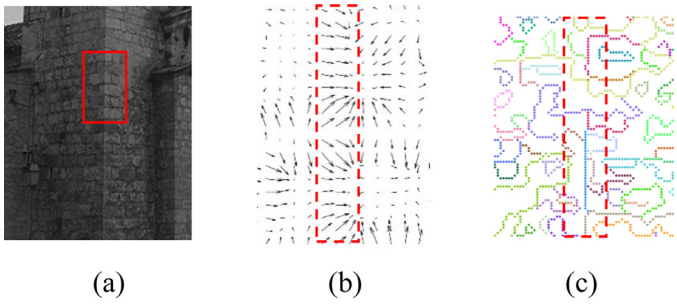


Fig. 1. Illustration of how unstable pixels in the weak gradient edge cause fractures. The red rectangle in (b) shows the gradient orientations of the line region in (a), and some of them do not align with each other. Thus, the pixels grouping methods based on gradient orientation cannot extract the complete line segment. The edges in the red rectangle of (c), which were detected by Edge Drawing, are discontinuous; thus, the edge based detector was used to extract fractures. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

better than LSD in complete extractions. However, our evaluation found that Linelet extracted more false positives than LSD and was unable to overcome the fracture problem in a line region with unstable gradients.

For detectors based on gradient orientation, the orientation aligned-tolerance is set as $\frac{\pi}{8}$, which is an empirical parameter and may cause line segments to fracture because the gradient orientations are unstable in weak gradient regions (see Fig.1(b)). In addition, the pixels in grass and clouds may bring about false positives when their orientations are aligned. Controlling the false positive with Helmholtz Principle also uses the empirical parameter to calculate the probability, which numerous studies [23–25] have pointed out may reject true positives when the Helmholtz Principle is employed in LSD.

Instead of grouping pixels based on gradient orientation, some detectors exploit the gradient magnitude, and the line segment is fitted from the edge segment. In Etemadi's method [26], gradient peaks first are connected to the edge segment, and the segment is then split into line segments based on the line fitting error. Akinlar et al. [25] proposed a line segment detector named EDLines, which is based on Edge Drawing [27] and a validation strategy similar to LSD. Recently, Lu et al. [24] improved the Canny detector [28], making it parameter-free and used a link method after the edge segments were split.

Our analysis of the various detectors based on gradient magnitude found that they also are easy to extract fractures in line regions with unstable gradients. Because they rely on the connection of gradient peaks, while unstable gradients can cause edge detectors, such as Canny and Edge Drawing, to make wrong connections (see Fig. 1(c)). Also, the dense edge pixels in grass, clouds, trees, etc., may bring about false positives.

We found that all the pixels grouping-based detectors extract the line segment passively. In other words, although these detectors are aimed at extracting the line segment, no constraint on line geometry is set in the pixels grouping. Instead, the line segment is fitted from a set of pixels constrained with gradient magnitude or gradient orientation. As has discussed, the passive grouping process may cause both fractures and false positives, and many groupings are meaningless. Previous studies explored the gradient magnitude or gradient orientation to validate the line segment, but the line geometry has yet to be fully considered. To the best of our knowledge, only LSD uses a fixed density tolerance to validate the "line support region", and its refining method may cause fractures. Therefore, both the complete extraction and false positive controlling in line segment extraction still need improvement.

We propose in this paper a novel active grouping and geometry-gradient combined validation method named AG3line. which proceeds as follows. First, the image is simplified by the anchor map (Section 3). Then, the anchors are actively grouped into line segments (Section 4) by taking the line geometry as the primary constraint. To control false positives, the line segments are finally validated and refined by exploiting the line geometry and the distribution of the gradient magnitude (Section 5).

Note that some other algorithms also extract the straight line with the line geometry constraint. For example, RANSAC [29] is a well-known method; the random 3-point detector [30] and the random Hough methods [9,10] select N points randomly to fit a straight line, then the straight line is validated in the image domain or the parameter space. But there are several differences between AG3line and these algorithms. First, our grouping process requires no random selection, instead, it starts with the significant pixel and validates the edge pixel in the anchor map. Second, our grouping process is active, which means that the candidate of the next anchor to be grouped is calculated directly based on the line geometry constraint and they can even be constrained to 1 pixel. This active process significantly reduces the computation. Thus, the experiments showed that it even runs faster than EDLines [25], the so called real-time detector. Third, AG3line detector is aimed at extracting the line segment but not the straight line. Though a jump strategy is employed in the active grouping, the jump distance is constrained and the geometry-gradient combined validation is employed to control the over connection (Section 5).

The remainder of this paper is organized as follows. Section 2 describes the overview of AG3line. Section 3 discusses the extraction and properties of the anchor map. Section 4 and Section 5 introduce the details of the anchor grouping and line segment validation process to control false positives. Section 6 presents our quantitative and qualitative evaluation of AG3line by comparing it with the state of the art detection methods. Section 7 concludes the paper.

2. Overview of the proposed method

Fig. 2 shows the AG3line workflow. First, the gradient magnitude and gradient orientation are calculated from an input grayscale image. Second, pixels having a high probability that line segments pass over are extracted as anchors. Third, the anchors are grouped into line segments based on the line geometry and the alignment of gradient orientation. Finally, line segments are validated and refined according to the gradient magnitude distribution and the line geometry. The computation of gradient magnitude and gradient orientation for each pixel is the same as LSD and EDLines, and thus they will not be discussed in the next sections.

3. The anchor map

The anchor map simplifies the image via discarding most of the useless pixels while confirming the significant pixel in line segment extraction. Active grouping and validation via line geometry are employed based on the anchor map.

3.1. Extraction of the anchor map

A good choice to discard the useless pixel is the non-maxima suppression of the Canny algorithm[28], but it extracts many of the edge pixels in grass, trees, clouds, etc. A key difference between an edge pixel and a line segment pixel is that the gradient orientation of the latter should be aligned with its neighbours [18,21,23,25]. Therefore, in AG3line, the anchors obtained via the non-maxima suppression are furtherly validated by the orientation to determine

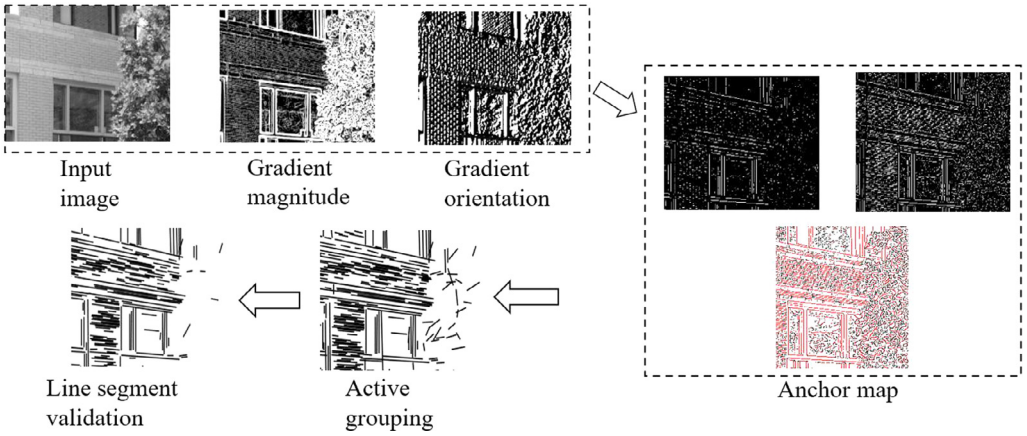


Fig. 2. Workflow of the proposed method.

Fig. 3. Illustration of the anchor map. The anchors in (b) were extracted by non-maxima suppression; the anchors in (c) were the significant edge pixels validated by Algorithm 1; (d) is the anchor map merged from (b) and (c).

Algorithm 1 Test for an anchor with horizontal gradient orientation.

Input a pixel P at (x,y) obtained from the non-maxima suppression;
a gradient orientation matrix G_{ori};
a parameter p of the angle tolerance.
Output a *bool* value marks whether P is a significant anchor.
isSignificantAnchor←*true*
if $G_{ori}(x,y)$ is HORIZONTAL then
  foreach *pixel* $\tilde{\boldsymbol{P}}$ in $\{(\boldsymbol{x},\boldsymbol{y_i})\boldsymbol{y}-1\leq\ \boldsymbol{y_i}\leq\boldsymbol{y}+1,\boldsymbol{y_i}\in\boldsymbol{N}^*\}$
    if **angdiff**($\boldsymbol{Gori}(\boldsymbol{x},\boldsymbol{y}),\boldsymbol{Gori}(\boldsymbol{x},\boldsymbol{y_i})$) $> p$ then
      isSignificantAnchor←false
      return
    end
  end
end

the significant pixels. Algorithm 1 shows the validation process for a horizontal pixel obtained from the non-maxima suppression.

Anchors with two levels (see Fig. 3) are extracted with the non-maxima suppression and Algorithm 1: first, the significant pixel that is aligned with its neighbor edge pixels; second, the general edge pixel obtained from the non-maxima suppression. The active grouping in Section 4 only starts with the significant edge pixel and sets stricter rules for the general edge pixel.

3.2. Properties of anchors along a line segment

Now we introduce the properties of anchors along a line segment, which guide the active grouping and validation.

1. The distance from the anchor to the line segment is smaller than d. This is an inherent law of the line segment, and several studies [24,25] suggest that one pixel is appropriate as an average distance (line segment fitting error).
2. Most anchors are aligned in the gradient orientation, and the aligned tolerance is smaller than p. This property is commonly used in pixels grouping [18] and line segment validation [21,23].
3. Anchors may not be continuous in space. Unstable pixels may cause anchors to be discontinuous. Therefore, a distance tolerance between anchors should be established.
4. Based on property 3, the anchors density (Section 5.1) of a line segment may not be 1. Since a longer line segment may go through more noise, a short line segment should have a higher density threshold.
5. Most pixels, including anchors, are aligned in the gradient magnitude. This property is exploited in Linelet [23] as an intrinsic feature to validate the line segment.

Based on these properties, the anchors are grouped actively into a line segment (details will be discussed in Section 4), then, the line segment is validated or refined for controlling false positives (details will be discussed in Section 5).

4. Grouping anchors actively into the line segment

In AG3line, a line segment is fitted by the anchors. As shown in Fig. 4, in the active grouping, the candidates of the next anchor to be grouped (NA) is directly calculated with the principal axis and the previous anchor that has been grouped (PA). The principal axis is updated after adding a new anchor. With the guidance of the principal axis, the candidate of NA can be expanded, thus unstable pixels can be skipped over (see Fig. 4(b-c)). When a grouping process is finished, the line segment is fitted by the principal axis and the ends of the anchors.

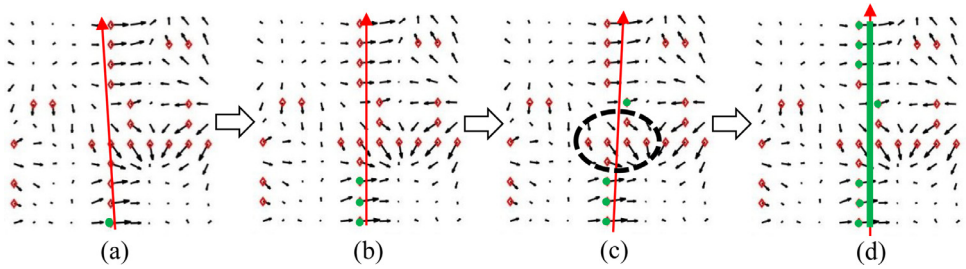


Fig. 4. Anchor grouping in a line region. The arrow's direction and length represent the orientation and magnitude of each pixel. The anchors are shown in the red diamond; anchors already grouped are shown in the green circle; the red line with an arrow is the principal axis estimated by the anchors having been grouped; the pixels inside the black ellipse are unstable and are skipped over with the guidance of the principal axis; and the green line segment is the result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.1. Principal axis of a grouping

The principal axis is formulated via the center $\mathbf{c} = (x_c, y_c)$ and the angle θ_l of the group:

$$l = \{(x, y) | y = \tan(\theta_l)(x - x_c) + y_c\} \tag{1}$$

The center of the group that contains n anchors are calculated as,

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i, y_c = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2}$$

where (x_i, y_i) is the coordinate of each anchor.

For the initial anchor with the gradient orientation θ_1, θ_l is calculated as $\frac{\pi}{2} - \theta_1$ under the assumption that the gradient orientation along l is approximately orthogonal to l. After grouping at least two anchors, θ_l can be determined from the eigenvalue of the inertia matrix $\mathbf{M} \in \mathbb{R}^{2\times 2}$[31], where $\mathbf{M}_{11} = \sum_{i=1}^{n}(x_i - x_c)^2$, $\mathbf{M}_{22} = \sum_{i=1}^{n}(y_i - y_c)^2$ and $\mathbf{M}_{12} = \mathbf{M}_{21} = \sum_{i=1}^{n}(x_i - x_c)(y_i - y_c)$. then, the smallest eigenvalue is calculated by,

$$\lambda = 1/2\left(\mathbf{M}_{11} + \mathbf{M}_{22} - \sqrt{(\mathbf{M}_{11} - \mathbf{M}_{22})^2 + 4\mathbf{M}_{12}\mathbf{M}_{21}}\right) \tag{3}$$

and θ_l is calculated by,

$$\theta_l = \begin{cases} arctan((\lambda - \mathbf{M}_{11})/\mathbf{M}_{12}), & \mathbf{M}_{11} > \mathbf{M}_{22} \\ arctan(\mathbf{M}_{12}/(\lambda - \mathbf{M}_{22})), & \mathbf{M}_{11} \le \mathbf{M}_{22} \end{cases} \tag{4}$$

Calculating $\mathbf{M}$ and $\boldsymbol{c}$ is time-consuming if the algorithm accesses all of the anchors after one anchor is added to the group. Thus, $\mathbf{M}^{(n)}$ and $\mathbf{c}^{(n)}$, which represent the $\mathbf{M}$ and $\mathbf{c}$ with n anchors in the group, respectively, are calculated via a recursive method. Giving the coordinate of NA $\mathbf{x}_{\boldsymbol{NA}} \in \mathbb{R}^{2\times 1}$, $\mathbf{c}$ and $\mathbf{M}$ are calculated as:

$$\mathbf{c}^{(n)} = \left(1 - \frac{1}{n}\right)\mathbf{c}^{(n-1)} + \frac{1}{n}\mathbf{x}_{\boldsymbol{NA}}, \ \mathbf{M}^{(n)} = \left(1 - \frac{1}{\boldsymbol{n}}\right)(\mathbf{M}^{(n-1)} + \frac{1}{\boldsymbol{n}}\left(\mathbf{c}^{(n)} - \mathbf{x}_{\boldsymbol{NA}}\right)\left(\mathbf{c}^{(n)} - \mathbf{x}_{\boldsymbol{NA}}\right)^T) \tag{5}$$

4.2. Active grouping process

When the principal axis is estimated, the candidates of NA are constrained to a small range and their coordinates can be calculated directly. The steps of the active grouping are as follows:

1. Calculate the candidates of NA based on property 1 of the anchor. Giving the direction of the principal axis defined as $dir_{PA} = (\cos(\theta_l), \sin(\theta_l))$ and the coordinate of PA defined as $\mathbf{x}_{PA} = (x_{PA}, y_{PA})$, the nearest candidate of NA is calculated as
$$\mathbf{x}_{NA} = \text{round}(\mathbf{x}_{PA} + dir_{PA}) \tag{6}$$
Because dir_{PA} only represents the last state of the grouping, an interval is set to $\mathbf{x}_{NA}$: if the principal axis is horizontal, i.e., $d_x > d_y$, the up and down anchors of $\mathbf{x}_{NA}$ are added as the candidates (see Fig. 5), or the left and right anchors are added.
2. Validate the candidates based on property 1 and 2 of the anchor. First, the difference between the candidate's orientation and the group's average orientation should be smaller than p. Second, the orthogonal distance between the candidate and the principal axis should be within 1 pixel. If one candidate satisfies the two constraints, the algorithm goes to step 4.
3. Expand the candidate. To reduce the fracture caused by the unstable pixels, when the nearest three candidates cannot satisfy the validation in step 2, the candidates are expanded via a jump under the line geometry constraint, which is achieved via adding a ratio to Eq. (6):
$$\mathbf{x}_{NA} = \text{round}(\mathbf{x}_{PA} + j{*}dir_{PA}), j \in N^*, j \le t \tag{7}$$
j is increased from 2 to t until one candidate satisfies the validation in step 2 (see Fig. 5).
4. Update the principal axis as introduced in Section 4.1 and set the anchor as PA, then, the algorithm goes to step 1.

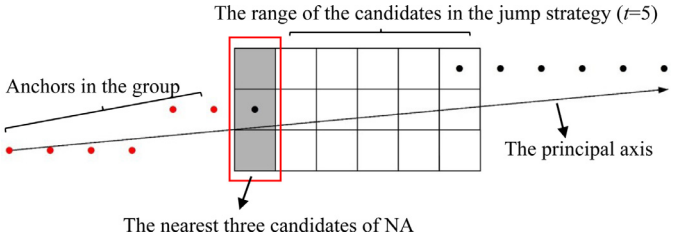


Fig. 5. Illustration of how the active grouping confirm the candidates when the principal axis is horizontal. The dot symbols represent the anchors in the anchor map.

There are two thresholds in the active grouping: the angle tolerance in step 2 and the jump distance t in step 3. We follow the past literature that use $\frac{\pi}{8}$ as the angle tolerance. t determines how far the grouping jumps when meeting unstable pixels. In AG3line, t is the variable based on the level of NA. If NA is a general edge pixel, t is set as 3, which means in AG3line, two collinear line segments with only a two-pixel distance can be merged into one. If NA is a significant edge pixel, t is set as 10, a loose threshold, which can jump over most noise interference. However, this loose threshold likely will cause many false positives. Thus, the line segment validation and refining methods in the next section are employed to control false positives.

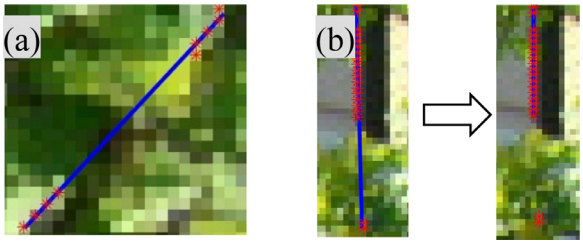


Fig. 6. False positives that should be rejected or refined according to the anchors density. The anchors are indicated by red stars and the line segments in blue. (a) shows a false positive in a noise background, which should be discarded and (b) shows the refined process of an over-grouped line segment. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. False positive control

The fixed jumping threshold in the pixels grouping may cause false positives, which arise from the noise background or when the distance of two collinear line segments is smaller than the jumping threshold. Fig. 6 shows two types of false positives. In Fig. 6(a), the whole group should be discarded since there is no line segment in this area. In Fig. 6(b), a part of the anchors should be discarded since they are wrongly connected.

Our algorithm controls the false positive from a rough to fine process. The line geometry, which is represented as the anchors density along the line segment, is first exploited to control the obvious false positives, then the distribution of the gradient magnitudes along the line segment is utilized for further validation.

5.1. The anchors density

The anchors density is the ratio between the number of the anchors along the line segment (we denote it as k) and the length of the line segment,

$$d = k/\text{length}(l) \tag{8}$$

When the anchors are well grouped into a line segment, the anchors density is high. On the other hand, when the jumping threshold causes a false positive, the density is low.

The anchors density should be no less than the threshold D, and Grompone et al. [22] set 0.7 as the minimum density of the aligned points in a rectangle. Generally, long line segments have a higher probability of covering noise than short ones while short line segments have a higher probability of being grouped in noise areas such as grasses, trees, or clouds. Therefore, in our method the density threshold is a variable based on the line segment, and a power function can be used to formulate this strategy:

$$D = a \cdot \text{length}(l)^b + c \tag{9}$$

where a, b, and c are coefficients. With the increase in the line segment length, D declines sharply and finally approaches to a fixed value.

We set 0.95 as the density threshold of the shortest line segment. When the line segment is ten times longer than the shortest, we think it is long enough and set 0.7 as the density threshold, and any line segment longer than it has the same threshold. Therefore, we can list at least three equations to solve the coefficients in formula (9).

For a line segment, of which d is smaller than D, the anchors are first split into two groups by the anchor having the maximum distance with its neighbor; then, the smaller group is discarded. This process is iterative and is terminated when $d \geq D$ or the length of the line segment is smaller than the minimum.

5.2. Alignment of the gradient magnitude

As shown in Fig. 7(a), when the false positive is long enough and the anchors density is large enough, the line geometry validation introduced in Section 5.1 will fail. Therefore, a further validation is employed by exploiting the alignment in gradient magnitude.

The validation via the gradient magnitude is based on the hypothesis that the gradient magnitudes along the line segment are aligned with each other. Cho et al. [23] showed that these gradient magnitudes can be modeled as a normal distribution given the distribution $N(\mu, \sigma^2)$, where μ is the mean gradient magnitude of a line segment and σ^2 is the variance. According to the empirical rule [32], the gradient magnitude along a line segment has a great probability of about 0.997 to fall in the interval between $\mu - 3\sigma$ and $\mu + 3\sigma$. In the gradient magnitude space of an image with the range of R, this interval covers a proportion of $6\sigma/R$. A higher proportion implies a less stable distribution of gradient magnitudes. In other words, the stability of these gradient magnitudes in the gradient magnitude space is compressed to

$$P = \frac{R - 6\sigma}{R} \tag{10}$$

P is the threshold that constrains the gradient magnitude given the stability of each gradient magnitude, which is presented by the coefficient of the variance form:

$$CV = \frac{|\mu - G_{mag}|}{u} \tag{11}$$

if $CV<P$, we say that the pixel is aligned with the whole in the gradient magnitude.

Fig. 7(b) shows the validation and refining process. First, the value of CV for each pixel and the value of P for the whole group are calculated. Then, for each pixel, if $CV<P$ and the pixel is not an anchor, then the group is divided into two parts at the pixel's nearest anchor and the smaller group is set free for regrouping. This process is iterative until all the pixels are aligned in gradient magnitude or the line segment length is smaller than the minimum.

6. Experiments

In this study, both quantitative and qualitative experiments were employed for evaluation of the AG3line detector approach. The performance of AG3line also was compared to PPHT, LSD, EDLines, MCMLSD, and Linelet. PPHT is a classical Hough transform-based detector that can be employed in OpenCV; and we followed Linelet to set the connection gap as 5 and set both the threshold minimum gain and the number of pixels contributed as 30. LSD and EDLines are widely used detectors, and MCMLSD and Linelet are the newest detectors. Their implementations are available on the authors' websites [33–36]. We made no changes on their code or internal parameters in the experiment. Our implementation is available on the website [37], and the internal parameters were fixed in the experiment. The parameters of AG3line are as follows.

1. Suppressing image noise: following EDLines, the image was filtered by a 5×5 Gaussian kernel filter with $\sigma = 1$.
2. Minimum gradient magnitude: following LSD and EDLines, a gradient magnitude smaller than 5.2 was not used.
3. Gradient orientation tolerance: in Sections 3.1 and 4.2, two pixels' gradient orientations were aligned when the difference between their angles is smaller than $\frac{\pi}{8}$.
4. Jumping distance: in Section 4.2, the general anchor can jump two pixels and the anchor in Algorithm 1 can jump ten pixels.
5. The anchors density: in Section 5.1, the anchors density varied from 0.95 to 0.7 based on the length of the line segment.

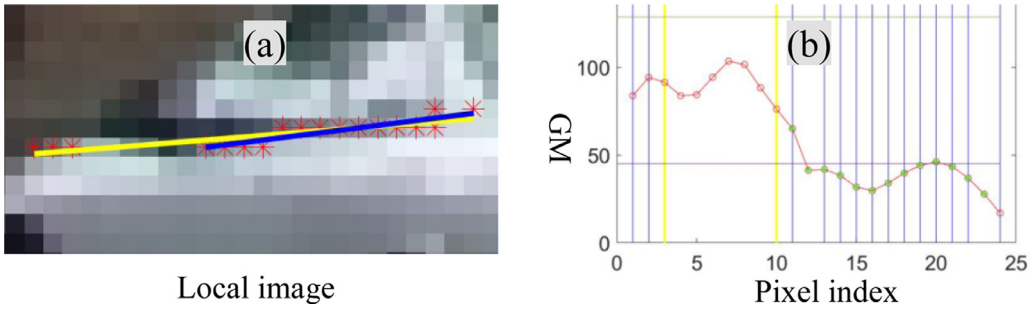


Fig. 7. Line segment validation based on the gradient magnitude (GM). In (a), the anchors are red; the yellow line segment is a false positive; and the blue line segment is the refined result. In (b), the anchors are crossed by vertical lines; the gradient magnitudes along the false positive in (a) are plotted by circles; the circles in green are aligned with the whole; and the circles in red are discarded since they are not aligned. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6.1. Evaluation details

Based on the ground truth, three indicators were used to quantitatively evaluate the performance of the detectors on an image: 1) Recall: the length ratio between the true positives and the ground truth, 2) Precision: the length ratio between the true positives and the extracted line segments, and 3) F-score: the combination of precision and recall, which implies the overall performance and is calculated by,

$$F-\text{score} = \frac{2 \times precision \times recall}{precision + recall} \tag{12}$$

In the evaluation of Linelet, the true positive L_{tp} and its ground truth L_{gt} satisfied the following constraints: their angle difference was smaller than $\frac{\pi}{36}$; the midpoint of L_{tp} was within 1 pixel to L_{gt}; and the intersection over L_{gt} was larger than the threshold.

$$\frac{L_{gt} \cap L_{tp}}{L_{gt}} \geq \lambda_{area} \tag{13}$$

To avoid setting the over-connection as a true positive, we added an additional constraint on the intersection ratio:

$$\frac{L_{gt} \cap L_{tp}}{L_{tp}} \geq \lambda_{area} \tag{14}$$

Note that the true positive's length is the intersection of L_{tp} and L_{gt}, but not the length of L_{tp}.

6.2. Dataset description

The York Urban image dataset [38] containing 102 images (45 indoor and 57 outdoor) was used for the evaluation. These images are 640 × 480 in size and contain urban environments consisting mostly of scenes from the campus of York University and downtown Toronto.

We used two sets of hand-labelled ground truth of this dataset to evaluate the performance. The first set was York Urban ground truth (YUGT) [38]. Each image in the dataset was hand-labelled to identify the line segments satisfying the "Manhattan-world assumption" [39], but many short line segments were not labelled because they were mainly used to detect the vanish point. The second set was the Linelet ground truth (LtGT) [17], which was hand-labelled by the author of Linelet to validate the performance of Linelet, thus more line segments were labelled in this set than for the first set.

Since YUGT was known to miss many short line segments, we used the short line segments in LtGT to complete YUGT. The line segment of LtGT, which did not find the intersection in YUGT, was added to YUGT. Finding an intersection line segment is the same as finding a true positive in Section 6.1.

6.3. Quantitative evaluation results

Fig. 8 shows the average evaluation results of YUGT with different λ_{area}. For the F-score, AG3line performed the best under all conditions; MCMLSD performed better than Linelet and both achieved performance comparable to AG3line; LSD had a certain gap with the three detectors in front; and EDLines and PPHT had an obvious gap with the other detectors. With an increase of λ_{area}, the F-scores of all the detectors declined; and the F-scores of AG3line, MCMLSD, and Linelet were similar because when λ_{area} was larger than 0.5, many true positives were counted as false positives. Note that these conditions were tight when true positives should have a tiny center (one pixel) distance, a small angle difference (five degrees), and at least a half overlapping.

Fig. 8(b) shows the average precisions and recalls of the six detectors, both of which decreased with an increase of λ_{area} but the rankings were fixed. AG3line achieved first place in precision and second place in recall, which means that AG3line achieved a good balance in controlling both false positives and false negatives. MCMLSD outperformed the others in recall because it improved PPHT and was able to extract more long line segments than the others but it cannot extract accurate endpoints in many cases, which will be illustrated in Section 6.4. Also, the precision of MCMLSD ranked third, which implied that it cannot control false positives well. Linelet performed better than LSD in recall, but its precision was worse than LSD, which indicates that Linelet detected more ground truths as well as more false positives than LSD. EDLines did not perform as well as LSD in both recall and precision and PPHT obtained the lowest score.

Fig. 9 shows the average evaluation results of LtGT with different λ_{area}. For the F-score (Fig. 9(a)), AG3line outperformed the others with various λ_{area}, but the rankings of MCMLSD, Linelet, and LSD changed. Linelet rose from third to second place and LSD outperformed MCMLSD when λ_{area} was larger than 0.4. In addition, EDLines performed better when evaluated by YUGT. This result was expected because the long line segment annotated in LtGT was shorter than YUGT. MCMLSD extracted many straight lines, and as a result, they failed to satisfy the formula (14). On the contrary, more of the line segments extracted by AG3line, Linelet, LSD, and EDLines satisfied the formula (13–14). Fig. 9(b) shows that both the precisions and recalls of AG3line, Linelet, LSD, and EDLines increased.Table 1 shows the ranking statistics of the F-scores for all the images when λ_{area} was 0.5. When evaluated by LtGT, 50% of AG3line's F-scores ranked first while Linelet and MCMLSD obtained a proportion of 25% and 17% that ranked first, respectively. When evaluated by the YUGT, the proportions of ranking first of AG3line and Linelet decreased to 44% and 19%, respectively, while MCMLSD increased to 34%. Note that compared with Linelet and MCMLSD,

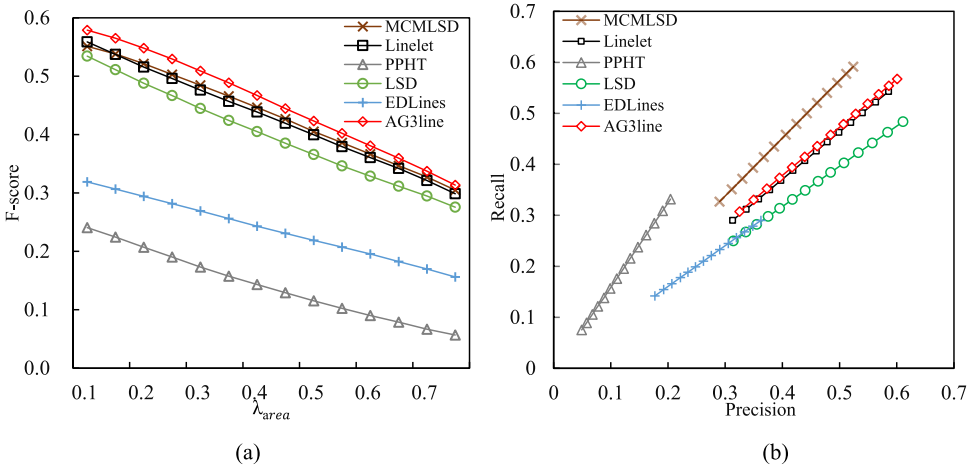


Fig. 8. Quantitative evaluation results based on YUGT.

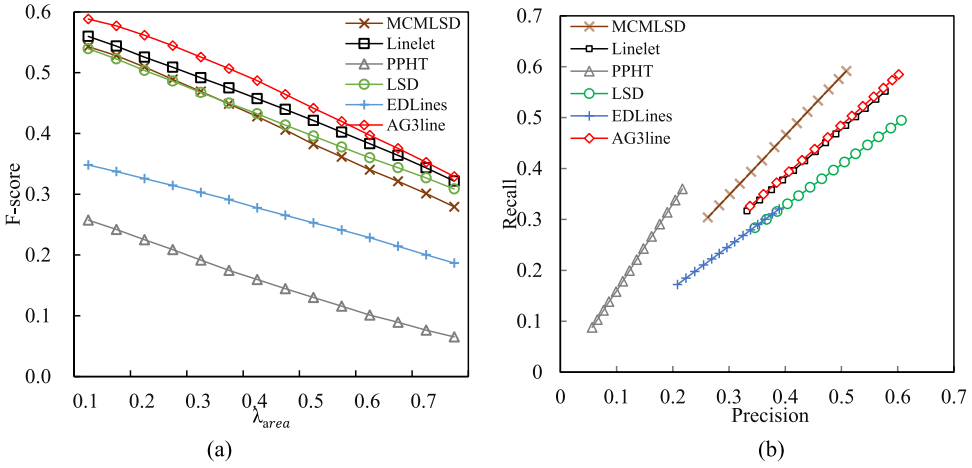


Fig. 9. Quantitative evaluation results based on LtGT.

Table 1
Ranking statistics of F-scores in two sets of ground truth when the λ_{area} is 0.5.

method ranking	AG3line		PPHT		LSD		EDLines		Linelet		MCMLSD	
	LtGT	YUGT	LtGT	YUGT	LtGT	YUGT	LtGT	YUGT	LtGT	YUGT	LtGT	YUGT
1st	51	44	0	0	8	5	0	0	26	19	17	34
2st	37	37	0	0	15	16	3	1	37	28	10	20
3st	11	19	0	0	33	18	8	2	29	43	21	20
4st	3	2	0	0	41	59	8	7	8	10	42	24
5st	0	0	6	6	5	4	77	86	2	2	12	4
6st	0	0	96	96	0	0	6	6	0	0	0	0

AG3line achieved robust performance in the two sets of ground truth. Also, we concluded from Table 1 that AG3line performed well in most images because 81% of AG3line's F-scores ranked in the top two in the worst condition, which was higher than Linelet and MCMLSD by 44% and 54%, respectively.

6.4. Computation time

The computation time of AG3line was compared with LSD and EDLines using the 102 images described in Section 6.2. LSD and EDLines were reported as the linear-time detector and real-time detector, respectively, and their experiments showed they were faster than the Hough based detectors. Since Linelet and MCMLSD are only available for the MATLAB vision, it is unfair to use the *C++* vision of AG3line to compare with them. All of the three algorithms were compiled by the Visual *C++* Compiler, and the tests were employed on the same laptop with Intel Core i7–7700HQ CPU and Windows 10 system. To reduce the bias, we employed 50 tests for each image and took the average computation time as the final result.

Fig. 10 shows the computation time of the three algorithms. All of them processed each image within about 120 milliseconds (ms). AG3line achieved the first in most images, which took at most 33 milliseconds to process each image. EDLines ran faster than AG3line in some images, but its time distribution was not so stable as AG3line. LSD took more time than other two detectors and its time distribution was the most unstable. The statistic of the computation time in table 2 is corresponding to Fig. 10. AG3line took the least time to process all of the images and its standard deviation was the smallest, which indicated that AG3line was more stable in computation time than LSD and EDLines.

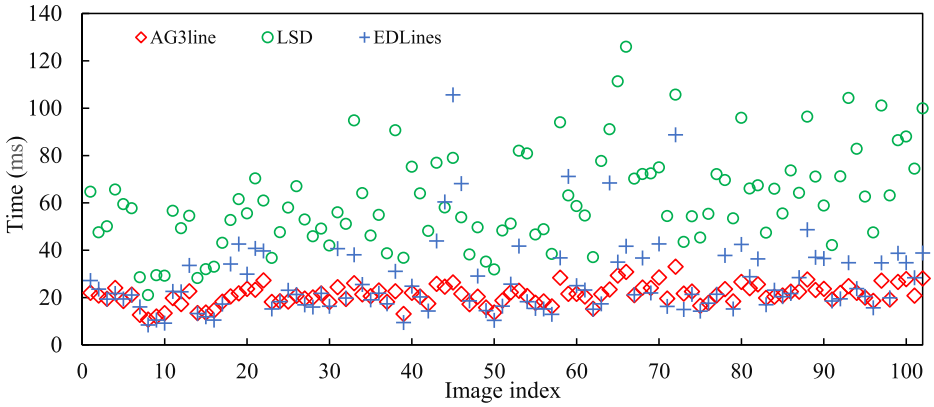


Fig. 10. The computation time of the 102 images. The horizontal axis represents the index of each image in the York Urban dataset.

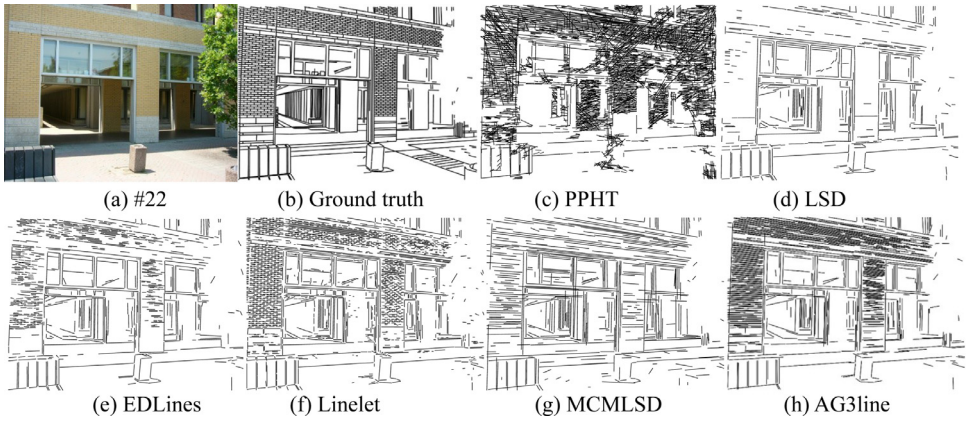


Fig. 11. Line segment detection result on the image #22 of the dataset.

Table 2
The statistic of the computation time (ms) for the York Urban image dataset.

	Mean	Standard deviation	Minimum	Maximum
AG3line	21.12	4.34	9.00	38.00
EDLines	27.40	16.21	8.00	86.00
LSD	61.11	22.14	20.00	192.00

AG3line achieved a good performance in computation time for two reasons. First, the active grouping method only starts with the significant edge pixel, which reduces the validation of the false positive. Second, the candidate anchor for the grouping is constrained to a small range (Section 4.2). Thus, less computation and validation are employed than the passive grouping.

6.5. Qualitative comparison

The image in Fig. 11(a) is a challenging situation that contains many weak gradient line segments on the facade and contains trees that may cause false positives. LSD and EDLines performed well in false positive control but nearly failed to extract the line segments on the façade. Linelet was found to produce many fractures and more false positives than LSD, EDLines, and AG3line. MCMLSD extracted more long line segments but failed to extract accurate endpoints for many line segments. PPHT generated many false positives and false negatives. Note that only AG3line extracted the complete vertical long line segment at the far left. The image in Fig. 12(a) contains a scene where there are many clouds and dense line segments (the dense line segments are not labelled in the two sets of ground truth). For performance in rejecting false positives, AG3line outperformed the others since it rejected many false positives in the cloud regions; LSD performed better than the others except AG3line; EDLines generated many false positives that were generally longer than the others; MCMLSD and PPHT generated many false positives in the area containing dense line segments. For the performance of complete extraction, AG3line, LSD, and Linelet detected the dense line segments while the others did not detect them or generated false endpoints.

The quantitative evaluation in Table 3 is corresponding to the qualitative results. All of the scores were lower than the aver-

Table 3
Quantitative evaluation of the performance ($\lambda_{area} = 0.5$) on the image #22 and #92 of the dataset.

Algorithm	AG3line		PPHT		LSD		EDLines		Linelet		MCMLSD	
Image	#22	#92	#22	#92	#22	#92	#22	#92	#22	#92	#22	#92
F-score	0.42	0.49	0.14	0.26	0.27	0.35	0.24	0.31	0.22	0.40	0.39	0.40
Recall	0.41	0.55	0.23	0.39	0.19	0.36	0.19	0.31	0.20	0.47	0.33	0.49
Precision	0.43	0.44	0.10	0.19	0.46	0.33	0.34	0.31	0.24	0.35	0.47	0.35
Time(ms)	27.25	21.95	63.00	54.13	60.95	47.67	39.70	19.45	–	–	–	–

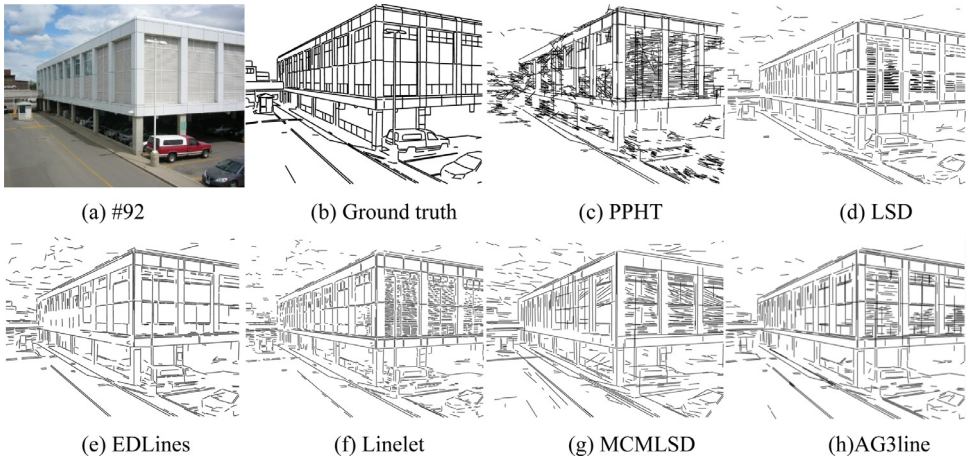


Fig. 12. Line segment detection result on the image #92 of the dataset.

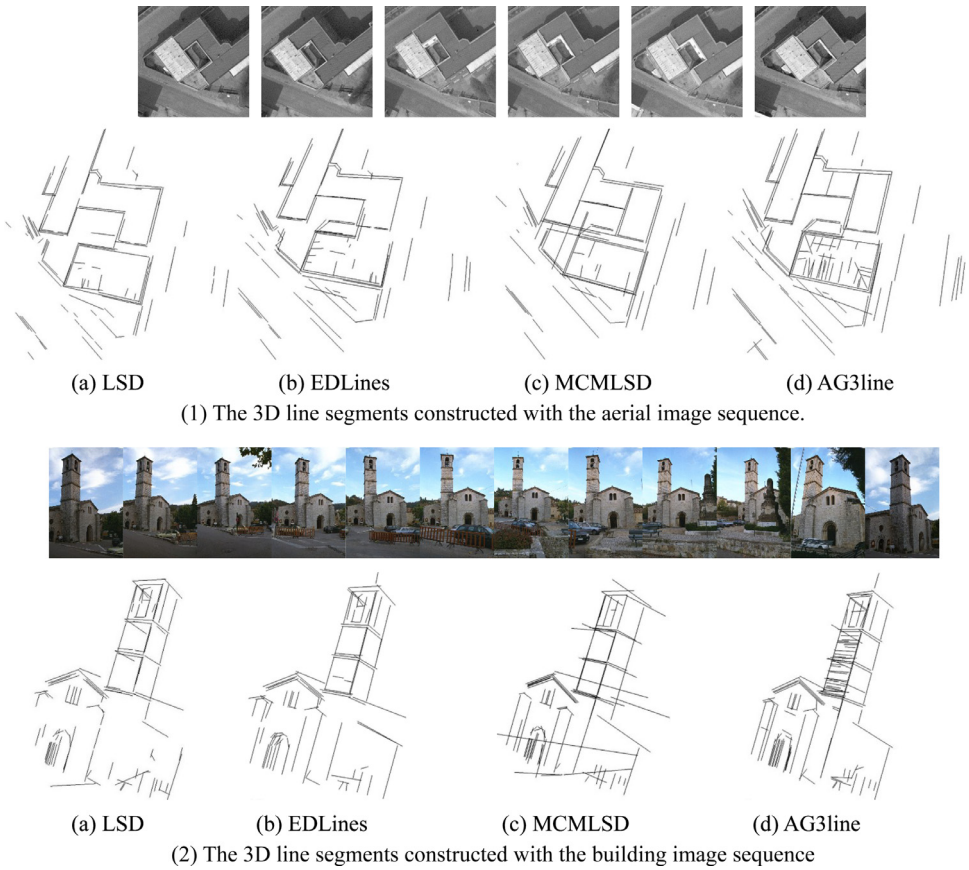


Fig. 13. The two image sequences and the 3D line segments constructed by the line segment matching algorithm [40] with the four line segment detectors.

age scores presented in Section 6.2 for the challenging situations. AG3line achieved the first score in recall on image #22, since many line segment on the facade were extracted. For image #92, AG3line achieved the first score for both the recall and the precision. Note that the computation time of AG3line was as fast as EDLines.

6.6. Practical application in 3D line reconstruction

To evaluate the performance of AG3line, LSD, EDLines, and MCMLSD in practical application, we employed the classical line segment matching algorithm [41], which requires accurate and robust line segments from multiple views to generate the 3D line segment. The codes, the image sequences and their camera matrices were available from the author's website[40]. The parameters were the same when employed with the four detectors.

Fig. 13 shows the 3D line segments generated with the four detectors. The 3D line segments constructed with AG3line were more complete than that constructed with the other three detectors: first, the 3D line segments were with less fractures; second, more 3D line segments with weak gradient were constructed. Note that although the 3D line segments constructed with MCMLSD were with few fractures, many of them were over-connected and not accurate. The results were respected. As shown in Fig. 14, the 2D line segments extracted by AG3line were with less fractures and false positives than others. Also, more line segments with the weak gradient were detected by AG3line.

Fig. 14. The line segment extraction results for one of the image in the second image sequence.

6.7. Discussion

Our experimental results demonstrated that exploiting both the line geometry and the gradient magnitude would be beneficial in line segment extraction. The line extraction steps of the AG3line method resemble the human visual process. When a human is drawing the line segment of an object, the line geometry (principal axis) and sharp area (the anchor map) are primarily considered, and the line segment is drawn directly from one point to the ends (active grouping).

AG3line detector achieved a good balance in extracting complete line segments and controlling false positives. Moreover, it performed well in the computation time. However, other current detectors have their own advantages: 1) PPHT is a fast detector for extracting global lines; 2) EDLines can run in real-time for images smaller than 1000×1000; 3) LSD can locate the endpoints to the subpixels; 4) Linelet can extract more local line segments than the others although its false control strategy generates more false positives than the Helmholtz principle and our method; and 5) MCMLSD is more robust than PPHT because it is an improved version of PPHT.

In the two sets of ground truth, the detectors performed differently, and even the rankings changed. These results show that there were different cognitive perceptions of line segments. In YUGT, the ground truth was more global and the line segment was not separated if it was visually continuous. In LtGT, the line segment was split for the discontinuity in one or two pixels but was labelled as whole in some cases. This distinction is why the rankings of MCMLSD, LSD, and Linelet changed in the two sets of ground truth. On the other hand, labeling the line segments by hand for the 102 images was difficult and time consuming because illumination changes, texture noises, or blurry effects must be considered. Although it was time consuming work, it not only made the evaluation more objective but also helped to improve the line segment detector's performance.

7. Conclusion

In this paper we introduced a new detector called AG3line to solve the problem of line segments being extracted as fractures because of unstable pixels, which also may cause false positives.

The proposed detector takes advantage of the inherent geometry constraint and the alignment in the gradient magnitude of the line segment. Our quantitative and qualitative evaluation showed that both the false negative and the false positive were controlled efficiently, and AG3line ran faster than both LSD and EDLines.

We suggest two extensions to the work presented in this paper: 1) improve AG3line's speed for real-time application with GPU and 2) extend AG3line's algorithm to detect curves and circles, which occur frequently in artificial scenes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by The National Key Research and Development Program of China under Grant 2018YFB0505003, and in part by the National Natural Science Foundation of China under Grant 41871368.

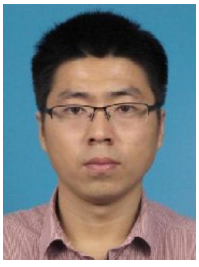
References

[1] Y. Li, R.L. Stevenson, Multimodal image registration with line segments by selective search, IEEE Trans. Cybern. 47 (5) (2017) 1285–1298.
[2] L. Zhang, H. Lu, X. Hu, et al., Vanishing Point Estimation and Line Classification in a Manhattan World with a Unifying Camera Model, Int. J. Comput. Vis. 117 (2) (2016) 111–130.
[3] M. Hofer, M. Maurer, H. Bischof, Efficient 3D scene abstraction using line segments, Comput. Vision Image Underst. 157 (2016) 167–178.
[4] M. Awrangjeb, M. Ravanbakhsh, C.S. Fraser, Automatic detection of residential buildings using LIDAR data and multispectral imagery, Isprs J. Photogramm. Remote Sens. 65 (5) (2010) 457–467.
[5] L. Zhang, R. Koch, An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency, J. Visual Commun. Image Represent. 24 (7) (2013) 794–805.
[6] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1981) 111–122.
[7] O. Barinova, V. Lempitsky, E. Tretiak, et al., Geometric image parsing in man–made environments, Eur. Conf. Comput. Vision (2010) 57–70.
[8] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic Hough transform, Pattern Recognit. 24 (4) (1991) 303–316.
[9] L. Xu, E. Oja, P. Kultanen, A new curve detection method: Randomized Hough transform (RHT), Pattern Recognit. Lett. 11 (5) (1990) 331–338.
[10] Y. Mochizuki, A. Torii, A. Imiya, N-Point Hough transform for line detection, J. Vis. Commun. Image Represent. 20 (4) (2009) 242–253.
[11] C. Galamhos, J. Matas, J. Kittler, Progressive probabilistic Hough transform for line detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999.
[12] S. Guo, T. Pridmore, Y. Kong, et al., An improved Hough transform voting scheme utilizing surround suppression, Pattern Recognit. Lett. 30 (13) (2009) 1241–1252.
[13] L.A.F. Fernandes, M.M. Oliveira, Real-time line detection through an improved Hough transform voting scheme, Pattern Recognit. 41 (1) (2008) 299–314.
[14] S. Du, C. Tu, B.J. Van Wyk, et al., Collinear segment detection using HT neighborhoods, IEEE Trans. Image Process. 20 (12) (2011) 3612–3620.
[15] Z. Xu, B.S. Shin, R. Klette, Accurate and robust line segment extraction using minimum entropy with Hough transform, IEEE Trans. Image Process. 24 (3) (2015) 813–822.
[16] Z. Xu, B.S. Shin, R. Klette, Closed form line-segment extraction using the Hough transform, Pattern Recognit. 48 (12) (2015) 4012–4023.
[17] E.J. Almazan, R. Tal, Y. Qian, et al., MCMLSD: a dynamic programming approach to line segment detection, IEEE Conf. Comput. Vis. Pattern Recognit. (2017) 5854–5862.
[18] J.B. Burns, A.R. Hanson, E.M. Riseman, Extracting Straight Lines, IEEE Trans. Pattern Anal. Mach. Intell. 8 (4) (1986) 425–455.
[19] A. Desolneux, L. Moisan, J.M. Morel, Meaningful alignments, Int. J. Comput. Vision 40 (1) (2000) 7–23.
[20] R.G.V. Gioi, J. Jakubowicz, J.M. Morel, et al., On straight line segment detection, J. Math. Imag. Vision 32 (3) (2008) 313–347.

[21] R.G.V. Gioi, J. Jakubowicz, J.M. Morel, et al., LSD: a fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 722–732.
[22] R.G.V. Gioi, J. Jakubowicz, J.M. Morel, et al., LSD: a line segment detector, Image Process. Line 2 (4) (2012) 35–55.
[23] N.G. Cho, A. Yuille, S.W. Lee, A novel linelet-based representation for line segment detection, IEEE Trans. Pattern Anal. Mach. Intell. 99 (2017) 1195–1208.
[24] X. Lu, J. Yao, K. Li, et al., CannyLines: a parameter-free line segment detector, IEEE Int. Conf. Image Process. (2015) 507–511.
[25] C. Akinlar, C. Topal, EDLines: a real-time line segment detector with a false detection control, Pattern Recognit. Lett. 32 (13) (2011) 1633–1642.
[26] A. Etemadi, Robust segmentation of edge data, in: International conference on image processing, 1992, pp. 311–314.
[27] C. Topal, C. Akinlar, Edge Drawing: a combined real-time edge and segment detector, J. Visual Commun. Image Represent. 23 (6) (2012) 862–872.
[28] J.F. Canny, A Computational Approach to Edge Detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.
[29] Fischler, M.A. and R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM. 24(6): 381–395.
[30] T.-.C. Chen, K.-.L. Chung, A new randomized algorithm for detecting lines, Real-Time Imag. 7 (6) (2001) 473–481.
[31] P. Kahn, L. Kitchen, E.M. Riseman, A fast line finder for vision-guided robot navigation, IEEE Trans. Pattern Anal. Mach. Intell. 12 (11) (1990) 1098–1102.
[32] H. Kanno, A simple derivation of the empirical rule ja:math, J. Non Cryst. Solids 44 (2–3) (1981) 409–413.
[33] EDLines Web Site. Available from: http://ceng.anadolu.edu.tr/cv/EDLines/.
[34] Linelet Web Site. Available from: https://github.com/NamgyuCho.
[35] LSD Web Site. Available from: http://www.ipol.im/pub/art/2012/gjmr-lsd/.
[36] MCMLSD Web Site. Available from: http://www.elderlab.yorku.ca/resources/.
[37] AG3line Web Site. Available from: https://github.com/weidong-whu/AG3line.
[38] P. Denis, J.H. Elder, F.J. Estrada, in: Efficient Edge-Based Methods For Estimating Manhattan Frames in Urban Imagery, DBLP, 2008, pp. 197–210.
[39] J.M. Coughlan, A.L. Yuille, Manhattan World: compass direction from a single image by Bayesian inference, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, 2002.
[40] Werner, T. and A. Zisserman. LMATCH: Matlab Toolbox For Matching Line Segments accross Multiple Calibrated Images. Available from: http://cmp.felk.cvut.cz/~werner/software/lmatch/.
[41] T. Werner, A. Zisserman, New Techniques for Automated Architectural Reconstruction from Photographs, in: Proceedings of the 7th European Conference on Computer Vision-Part II, 2002.

Yongjun Zhang was born in 1975. He received the B.S., M.S., and Ph.D. degrees from Wuhan University (WHU), Wuhan, China, in 1997, 2000, and 2002, respectively. He is currently a Professor of photogrammetry and remote sensing with the School of Remote Sensing and Information Engineering, WHU.

Dong Wei was born in 1992. He received the B.S. and M.S. degrees from Central China Normal University, Wuhan, China, in 2015 and 2018, respectively. He is now studying for the doctoral degree of photogrammetry and remote sensing in the School of Remote Sensing and Information Engineering, WHU.



Yansheng Li received the B.S. degree from the School of Mathematics and Statistics, Shandong University in 2010, and the Ph.D. degree from the School of Automation, Huazhong University of Science and Technology. Since 2015, he has been an Assistant Professor with the School of Remote Sensing and Information Engineering, WHU