Computers & Geosciences 85 (2015) 22–35

Contents lists available at ScienceDirect

Computers & Geosciences

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

Case study

Extraction of microcracks in rock images based on heuristic graph searching and application

Zhihua Luo [a,b], Zhende Zhu [a,b], Huaining Ruan [a,b,*], Chong Shi [a,b]

[a] Institute of Geotechnical Research, Hohai University, Nanjing 210098, China
[b] Key Laboratory of Ministry of Education of Geomechanics and Embankment Engineering, Hohai University, Nanjing 210098, China

ARTICLE INFO

Article history:
Received 19 February 2015
Received in revised form
28 August 2015
Accepted 31 August 2015
Available online 2 September 2015

Keywords:
Microcrack detection
Graph searching
Crack separation
Image processing

ABSTRACT

In this paper, we propose a new method, based on a graph searching technique, for microcrack extraction from scanning electron microscopic images of rocks. This method mainly focuses on how to detect the crack and extract it, and then quantify some basic geometrical features. The crack can be detected automatically with the aid of two endpoints of the crack. The algorithm involves the following process: the A* graph searching technique is first used to find a path throughout the crack region, defined by the initial two endpoints; the pixels of the path will be used as the seeds for the region growing method to restore the primary crack area; then, an automatic filling holes' operation is used to remove the possible holes in the region growing result; the medial axis and distance transformation of the crack area are acquired, and then the final crack is rebuilt by painting disks along a medial axis without branches. The crack result is separated without interaction. In the remaining parts, the crack features are quantified, such as the length, width, angle and area, and error analysis shows that the error percentage of the proposed approach reduces to a low level with actual width increases, and results of some example images are illustrated. The algorithm is efficient and can also be used for image detection of other linear structural objects.

1. Introduction

Fractures in rocks are usually associated with the morphogenesis, expansion and coalescence of microcracks (Kranz, 1983). Hence, the quantitative analysis of geometrical characteristics and other microstructural features of such microcracks is of fundamental importance. The traditional manual measuring method is unsuitable because of its low accuracy, efficiency and the micro-size of microcracks. Scanning electron microscope (SEM) technology provides a new alternative research method of image processing.

Image processing is an effective way to perform quantitative analysis in geosciences. Mostly, the quantification process can be generalized into three steps:

(1) *Image preprocessing*: in practice, the original images, captured from the region of interest, are always mixed with some noise. If the noise distribution is known a priori, the effects can be eliminated by a filtering operation, such as Gaussian blur (Gonzales and Woods, 2002). In addition, some features of the target object can be extruded to demonstrate more information; Pu et al. (2008) give an example, who use a fractional differential approach to detect textural features, and more details are revealed in the image.
(2) *Object extraction*: binarization is usually the fundamental operation in this step, and it will generate a binary image of the object. Subsequently, other processes might be used to remove fragments, or to link the disconnect object parts or separate the interacting nodes.
(3) *Feature measurement*: the final step is used to measure the object features such as the area, length, width, direction, and fractal dimension, using the binary image of the object.

A significant amount of research has been carried out in an attempt to achieve a fully automated procedure for the three steps illustrated above. However, there is no completely automatic algorithm (Dare et al., 2002). Liu et al. (2011) introduced an automatic quantification method for pores on clay materials based on SEM images, which uses a global thresholding method to segment the image and then divide the segmented objects, and mainly discusses the analysis of geometrical parameters. Valença et al. (2012) used a digital image processing method to automatically characterize concrete cracking. Sinha and Fieguth (2006)

* Corresponding author at: Institute of Geotechnical Research, Hohai University, Nanjing 210098, China.
E-mail addresses: lzhhhu@163.com (Z. Luo), zzdnj@hhu.edu.cn (Z. Zhu), hnruan@hhu.edu.cn (H. Ruan).

http://dx.doi.org/10.1016/j.cageo.2015.08.013
0098-3004/

presented a two-step crack detection approach for buried concrete pipe images. In the first step, the crack features at a local level are extracted and then detectors are used to fuse these features; the second step is to clean and link the segmented candidates at a global scale. LeRoux et al. (2013) proposed a semi-automatic method to analyze the microscopic crack pattern of thermal fatigue steel. This method uses thresholding and the region growing method to segment the image, followed by water shed segmentation, and analysis of the geometrical and topological parameters. Jahanshahi et al. (2013) proposed a classification method to detect and analyze cracks. This method first segments the image and uses a neural network and support vector machine to classify the crack features and detect cracks, and then quantifies the crack thickness.

The key issue to extract objects from images is how to segment or binarize the image. To the best of our knowledge, most methodologies use a global computed or local dynamic threshold to segment an image, and these are better suited for good condition images where the object is significantly different from the background. For some poor condition images, Dare et al. (2002) delineated the crack in a concrete structure as a polyline and measured the crack width along this polyline. Li et al. (2011) and Zou et al. (2012) detected crack lines on pavement images with grain-like characteristics.

Another key issue to microcrack quantification is to separate the network of interacting cracks. Arena et al. (2014) and Liu et al. (2013) attempted to separate single cracks based on the identification of the nodes of the skeletonized pattern. However, the separation based on nodes is usually not sufficient for some cracks with branches. For instance, in the meso-mechanical analysis of rocks, we sometimes need the main crack information, but the crack always has small branches in certain central places.

A typical microcrack SEM image is shown in Fig. 1a. Fig. 1b is the result of the Otsu (1975) method, where it is very hard to recognize the crack. Due to the poor condition of microcrack SEM images (Fig. 1a) and the linear structural characteristics of microcracks, we propose a novel method for crack extraction, which requires two endpoints of the crack and is based on a graph searching technique. This technique was first used for edge detection in image fields (Martelli, 1972). The elaborate procedure mainly focuses on the extraction step.

In Section 2, the graph searching technique is first introduced and Section 3 is dedicated to the crack extraction process. Section 4 provides methods to quantify the basic geometrical features of a crack, and some discussions and examples are described in Section 5. The conclusion is presented in Section 6.

2. Heuristic graph searching technique

In this section, we first provide a brief introduction of graph searching, and then describe the theory of the A* algorithm, which belongs to the heuristic graph searching technique.

2.1. Graph searching

As shown in Fig. 2, a graph consists of a set of circle dots (named nodes), joined by lines (named edges). The edges may be directed or undirected, which means that the lines are arrowed or not, but this study only employs the latter. When the nodes or edges of a graph are associated with weights, it is termed as a weighted graph.

Fig. 3 shows an image-generated graph; each node is labeled with the corresponding pixel position (i.e., a node can be located by the pixel coordinate in the image). Each node corresponds to a pixel and every pixel is linked with its orthogonal or diagonal neighbors. The node weight can be a function of the pixel value, while the edge weight can be 1 for the orthogonal and $\sqrt{2}$ for the diagonal.

Graph searching in this study aims at finding a least-cost path from a given initial start node to a goal node in an image-generated graph. A least-cost path between two nodes is obtained when the sum of the weights of its constituent nodes and edges is minimized.

2.2. A* algorithm

The A* algorithm (Hart et al., 1968; Nilsson, 1982) is an extension of Dijkstra's algorithm (Dijkstra, 1959) and uses heuristics to speed up the search of the least-cost path.

Let p_s and p_g be the given initial start node and goal node, respectively. The main process of the A* graph searching algorithm can be described as follows (see also Appendix A for details):

(i) Initialize status.
(ii) Repeat the visiting of nodes from p_s until p_g is reached.
(iii) Backtrack the nodes from p_g until p_s is reached to save the path.

The least-cost path is shown in Fig. 3 using a thick red line.

3. Microcrack extraction process

In this section, the complete extraction procedure for the microcrack is described. We start with an overview of its design, followed by the details of each part.

3.1. Overview of extraction

Cracks in rock images generally present a very irregular linear shape characteristics with a dark color. The basic idea of the approach presented in this paper is as follows: once a crack is identified by the user, only its two endpoints are needed, and then the algorithm will solve the crack result automatically. The algorithm first searches a continuous path running through the

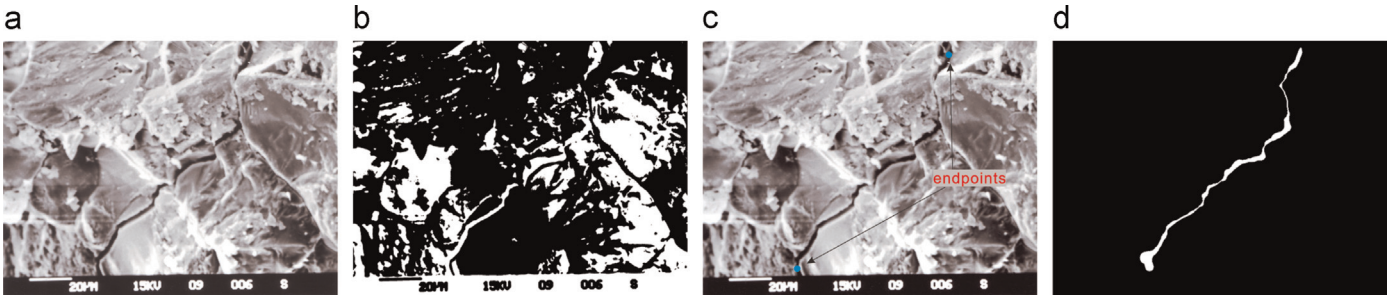

Fig. 1. Crack extraction using different approaches. (a) Crack SEM image, (b) result of segmentation using the classical Otsu threshold method, (c) marked endpoints of a crack, (d) result of the proposed approach.

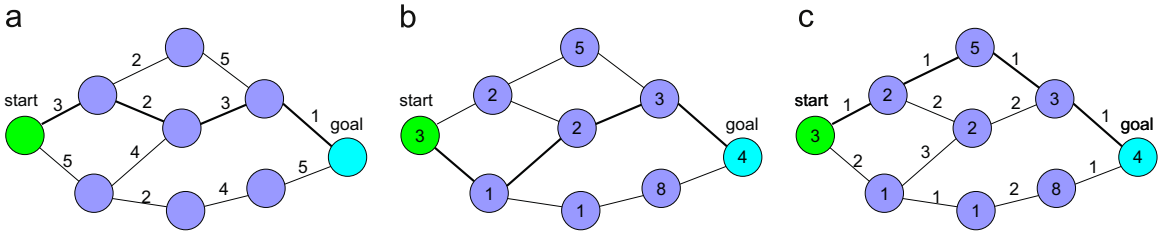

Fig. 2. Graph network. (a) Edge-weighted graph, (b) node-weighted graph, (c) edge- and node-weighted graph.

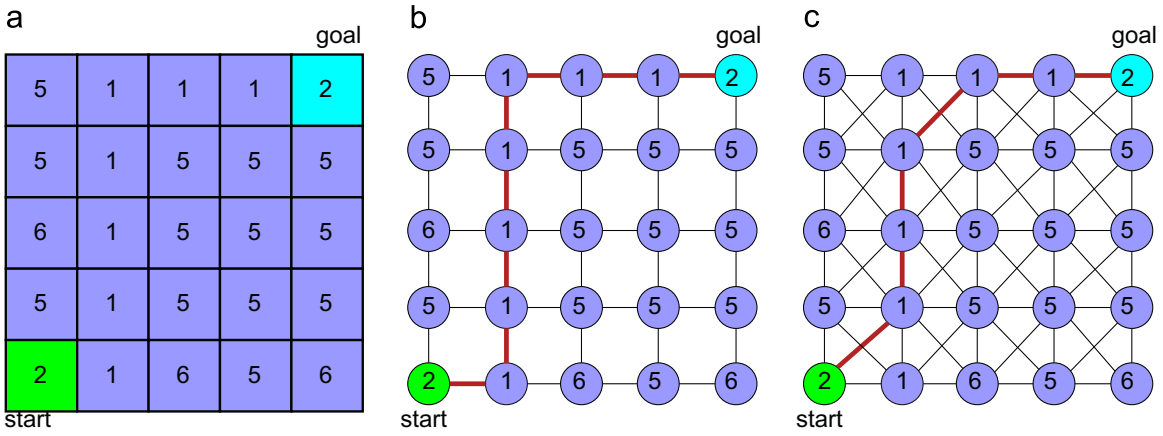

Fig. 3. Image-generated graph. (a) Image pixels matrix, (b) generated graph with orthogonal adjacency, (c) generated graph with orthogonal and diagonal adjacency.

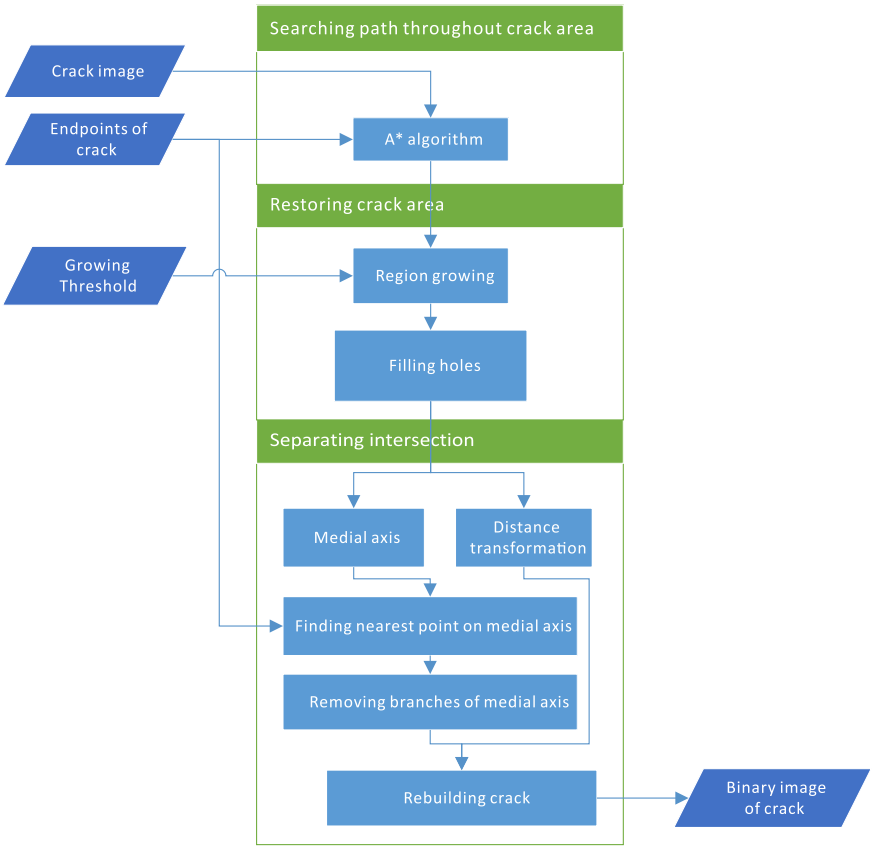

Fig. 4. Flowchart of microcrack extraction.

designated crack area and then restores the crack area using that path by a region growing algorithm. Because of the influence of noise in the original image, holes may exist in the restored crack area, and an automatic filling-holes operation, which is introduced in Section 3.3, is executed next. When the crack intersects with others, the above result will be jagged, and a rebuilding step that draws disks along a central line is used to separate it. The central line is acquired from the medial axis of the restored area after

filling the holes, which retains the medial axis of the needed crack area and removes the others. The radius of the rebuilding disks depends on the location of the central point and is set to the distance transformation value of the central point, which is the nearest distance to the background. The A* graph searching technique is employed in the finding path step and the removing intersected medial axis step. The entire flowchart is shown in Fig. 4.

For clearness and conciseness, it is assumed that the input image is in gray scale, and cracks in the image are dark (i.e., with a lower pixel value).

3.2. Searching the path throughout the crack

After the rock digital image input, the two endpoints of a crack need to be chosen by the user, and then the A* graph searching algorithm will be used in the graph (which is generated by the input image) to find a path throughout the crack.

The graph is constructed with orthogonal and diagonal adjacency. As nodes are labeled by pixel position, the eight neighbors of a node p can be expressed by eight vectors, namely $p + \vec{u}_i$, ($i = 0, 1, 2, \ldots, 7$), as illustrated in Fig. 5.

Since the A* algorithm is designed for the least-cost path here, the searching will be faster and the path result will be more accurate if the crack node weight is far less than the background node. To enlarge the differences between the crack and non-crack image area, i.e., to enhance the contrast of the pixel value, a nonlinear transformation of the pixel value is used. A simple and efficient function is $f = x^n$ (here x represents the pixel value), because the derivative of f is greater than 1 when $x > 1$. Although the contrast may be higher with a larger n value, $f = x^3$ is adequate and is set as the node weight. Fig. 6 shows the $f = x^3$ transformation function.

The edge weight of the graph is adopted as integers instead of 1 and $\sqrt{2}$, since integer calculation will be faster. Let t_i be the edge weight (the location of t_i corresponds to $\vec{u}_i$, as illustrated in Fig. 5), then

$$t_i = \begin{cases} 10 & \text{if i mod 2 = 1} \\ 14 & \text{otherwise} \end{cases}, \; i = 0, 1, \ldots, 7. \tag{1}$$

After the A* search, a path throughout the crack is obtained, as shown in Fig. 7b. Note that the object of this step is a path running through the crack, regardless of the cost.

3.3. Restoring the crack area

In this step, the region growing method (Gonzales and Woods, 2002) is used to restore the crack area. Pixels on the path obtained following the procedure illustrated in Section 3.2 are used as the initial image growing seeds, and the process follows the steps below.

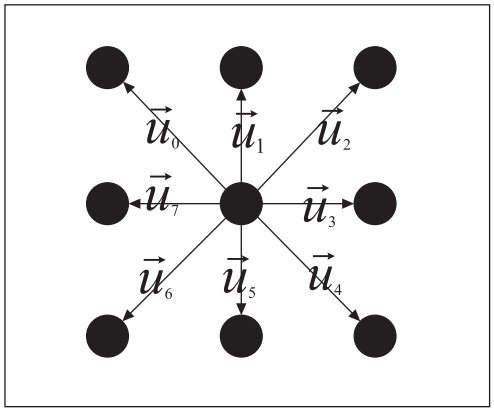

Fig. 5. Eight-neighboring nodes.

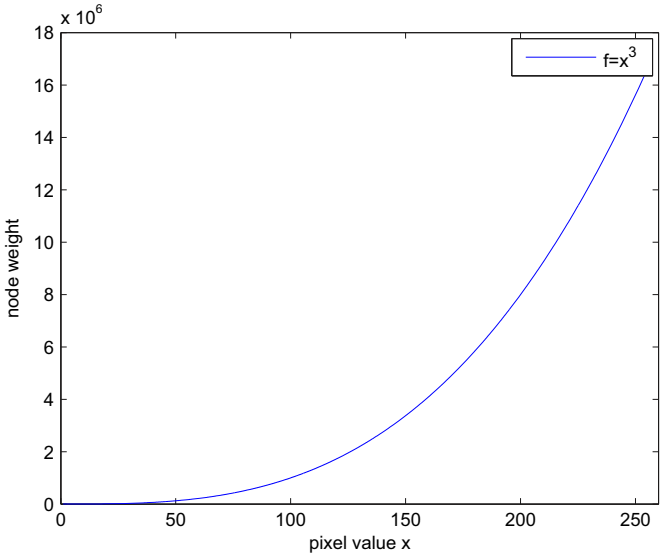

Fig. 6. Node weight value.

(1) Calculate the mean value (denoted by m) of all seed pixel values.
(2) Judge the value of the pixel (denoted by I_p) that is not a seed but that is orthogonally or diagonally adjacent to a seed.
(3) If $|I_p - m| \le \varepsilon_e$, add this pixel to the seeds, and recalculate the mean value m.
(4) Repeat the above process until no pixel can be added to the seeds.
(5) All the seed pixels represent the crack area.

Here, ε_e is a threshold to control the region growing, and different images may have different optimal ε_e values. However, whether the result fits the reality relies on subjective judgment, and it is difficult to determine the optimal value; Section 5.1 provides a method to choose this threshold. Fig. 7c shows the result of an example rock image after region growing based on a threshold value of 30. Note that the result of the crack area is definitely connected.

As is evident in Fig. 7, the region growing area contains holes because some tiny masses invade the crack areas and ε_e values that are too small may also lead to holes in the region growing result because of the noise in the original image. To improve this, an automatic filling holes operation (Gonzales and Woods, 2002) will be executed on the region growing result. This automatic algorithm is applied to the binary image and the following steps below outline the general procedure (provided that the object pixel value is 1, and the other is 0).

(1) Create a new binary image that is the same size as the original image, and set the boundary pixels except for the position with value 1 in the original image to value 1 and the other to 0.
(2) Set all the boundary pixels in the new binary image with value 1 as the initial seeds.
(3) Judge the value of the pixel that is orthogonally neighbored by a seed, and if the pixel is not a seed and the value of the pixel in the original image is 0, add this pixel to the seeds (also set the value of this pixel in the new image to 1).
(4) Repeat step (3) until no pixel can be added to the seeds.
(5) The region with value 0 in the new binary image is the final result after filling the holes.

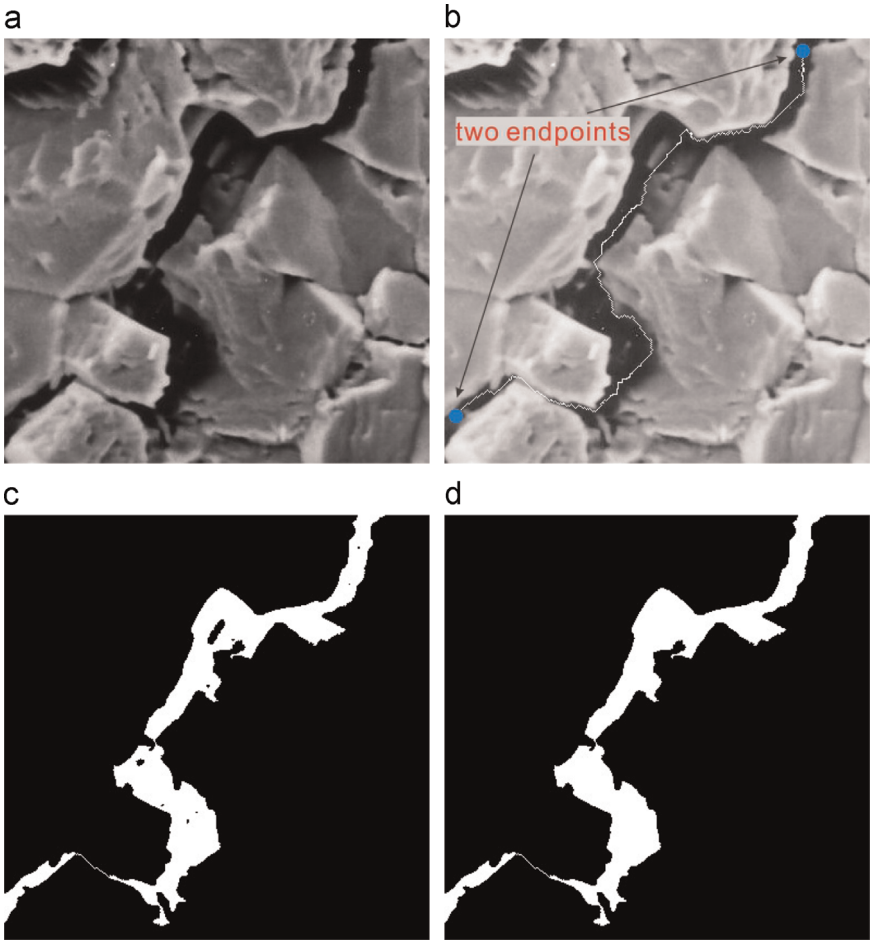

Fig. 7. Restoring the crack area. (a) Crack SEM image, (b) path throughout the crack area with endpoints marked, (c) region growing result with a threshold value of 30, (d) result after filling the holes.

The result after filling the holes is presented in Fig. 7d.

3.4. Separating intersection

Cracks may intersect with each other. As illustrated in Fig. 8d, the crack after the restoring process is connected to the other crack, and it is separated to an isolated one without branches in this step.

The main idea is to paint disks along the medial axis. The radius of the disk will follow the axis line to the change, and the disk center is on the medial axis. A distance transformation is employed to obtain the radius, and the thinning method is used to obtain the medial axis here.

3.4.1. Distance transformation

Distance transformation is an operation that converts a binary image into a gray-level image where all pixels have a value that corresponds to the distance to the nearest background pixel (the non-crack area pixel), as illustrated in Fig. 9. Euclidean distance transformation (Danielsson, 1980; Cuisenaire and Macq, 1999) is accurate but complex; therefore, we use a "chamfer 5-7-11" distance transformation, which is discussed in Borgefors (1986). This is an approximation to the Euclidean distance, but is simple and fast. It has a maximal difference from the Euclidean distance of approximately 2% (Borgefors, 1986).

The binary image, i.e., the result after filling the holes in Section 3.3, will be applied to the transformation. It should first be converted to ensure that the pixel in the crack area has an initial value of infinity (a suitable large number), and the other has an initial value of zero. Then, the "chamfer 5-7-11" uses two masks, as illustrated in Fig. 10, where $a = 5$, $b = 7$, $c = 11$ to sequentially pass over the image twice. For the first passing, it uses the mask in Fig. 10a to convolve the image from left to right and from top to bottom. After that, the second passing uses the mask in Fig. 10b to convolve the image from right to left, and from bottom to top. The distance transformation result is shown in Fig. 8e where the value is mapped for visualization.

Note that the distance transformation value is normalized last, so that the distance between the two orthogonal neighbored pixels is 1.

3.4.2. Medial axis

The medial axis is called a skeleton and has various definitions. Generally speaking, it is a set of center points of maximal balls (Calabi and Hartnett, 1968; Vincent, 1991). Here, a maximal ball is included in the object and no other balls containing it exist (this is also included in the object). This is presented schematically in Fig. 9. There is a wide range of thinning algorithms to obtain the medial axis of a binary image (Lam et al., 1992).

A homotopic thinning method proposed by Shengxun (1984), combined with the staircase-elimination procedure (Parker, 2010; Holt et al., 1987), is used in this study. It is fast and simple to implement, and is a parallel method. Fig. 8f shows the binary image of the medial axis. Note, however, that it has branches that we do not need, and they will be removed in Section 3.4.3.

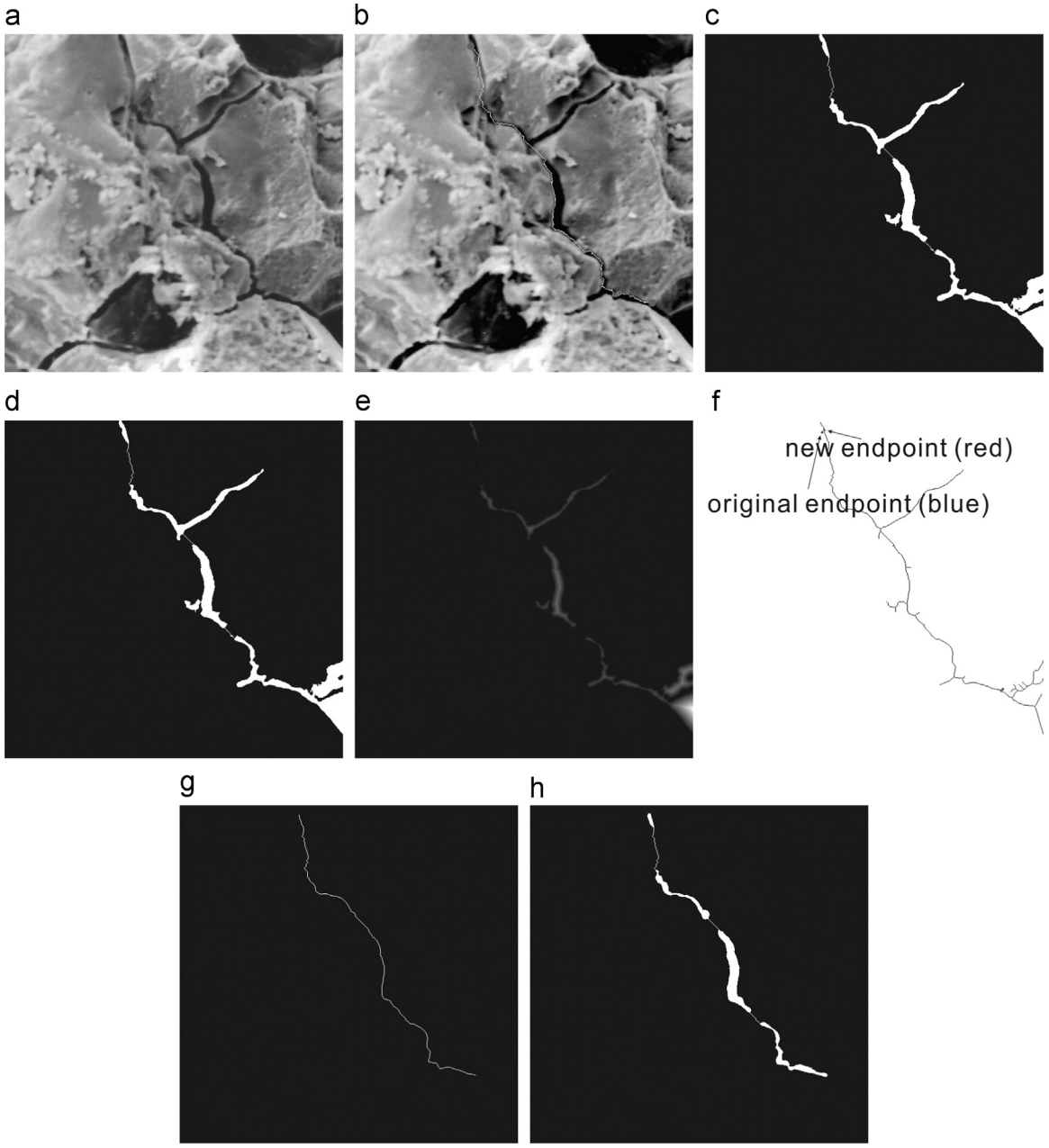

Fig. 8. Rebuilding the crack. (a) Crack SEM image, (b) path throughout the crack area with endpoints marked, (c) region growing result with a threshold value of 30, (d) result after filling the holes, (e) distance transformation result (mapped for visualization; the brighter color area represents a larger distance value to the background), (f) medial axis map is used for the A* search and the new endpoint (in red color, while the original endpoint is in blue color) can be observed, (g) medial axis after removing branches, (h) final crack result.

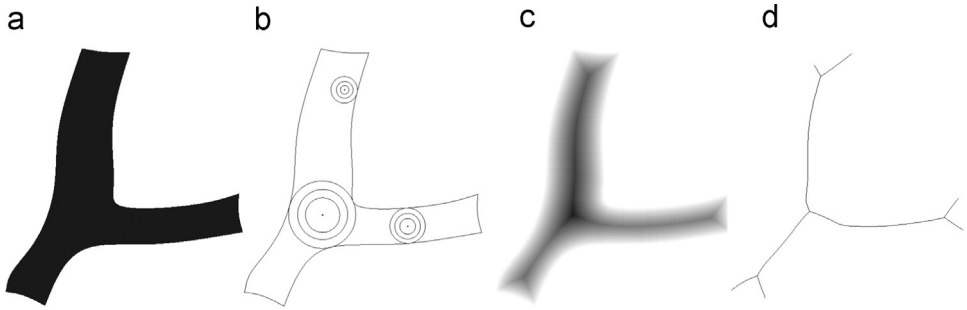

Fig. 9. Schematic of the distance transformation and medial axis. (a) A binary image of the object, (b) balls included in the object, (c) distance transformation result (the darker color corresponds to a larger distance value to the background), (d) medial axis.

a

2b	c	2a	c	2b
c	b	a	b	c
2a	a	0		

b

		0	a	2a
c	b	a	b	c
2b	c	2a	c	2b

Fig. 10. Chamfer5-7-11 masks.

3.4.3. *Removing the medial-axis branches*

To remove the branches, we once again employ the A* algorithm. The graph is constructed based on the binary image of the medial axis obtained in Section 3.4.2. The node weight is set as below: nodes (or pixels) on the medial axis are set to a weight value of 1 or some other small integer (note here that if the value is set to zero, the A* search may generate a poor result), and the other is set to a weight value of infinity, i.e., a suitable large number. The edge weight can be set exactly as per Section 3.2. Then, the A* search will generate a path, which must be on the origin of the medial axis but without branches, as illustrated in Fig. 8g. The next rebuilding step will use this path as the medial axis instead of the axis in Section 3.4.2.

Before the A* search, the two endpoint (start and goal node) locations should be modified because the initial given points may not be on the medial axis. The new endpoints should be on the medial axis and approximately nearest to the original ones. The finding algorithm dilates iteratively from the initial point until the first meeting with the pixels on the medial axis, as shown in Fig. 11. That pixel will be used as the new endpoint.

3.4.4. *Rebuilding the crack*

At this point, we have the distance transformation and the medial axis that has no branches. It can be proved that a set $X \subset Z^2$ (Z^2 refers to the discrete plane) is equal to the union of its maximal balls (Vincent, 1991). Here, it is assumed that the center of the maximal ball is located on the medial axis, and the radius of the ball is the distance transformation value of the center. This assumption is reasonable since the medial axis is the set of center points, and distance transformation is the nearest distance to the non-crack pixels. The crack is rebuilt by painting disks, i.e., the maximal balls, along the medial axis. Fig. 8h shows the final crack result after rebuilding; note that it is consecutive from beginning to end.

4. Geometrical features quantification

In this section, the geometrical features of the crack, such as the length (mean and maximum) width, area and angle, are calculated based on the extraction process.

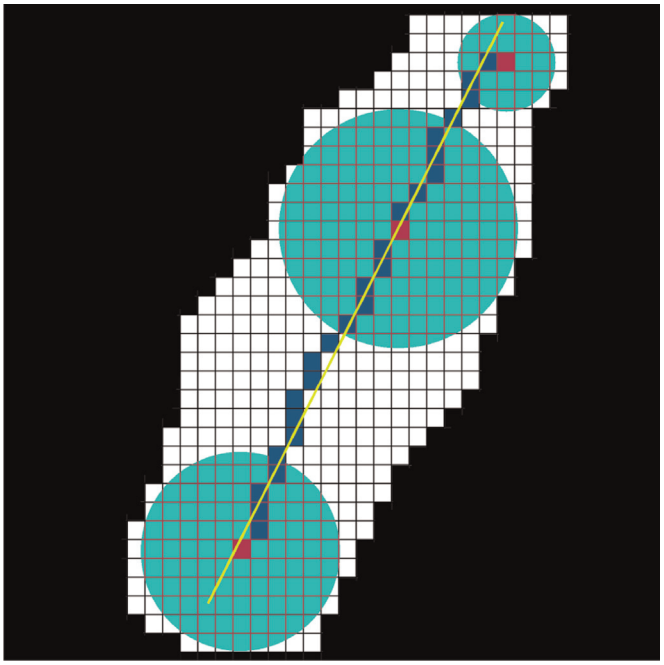

Fig. 12. Sketch of geometrical features quantification. The white squares are the crack pixels, the blue squares represent the medial axis after removing the branches, and the centers of the green disks are indicated by the red squares. The yellow line is a least square fitting line of the medial axis. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 12 shows the principle used to calculate these features, where the white squares are the crack pixels, the blue squares represent the medial axis after removing the branches and the centers of the green disks are indicated by the red squares. The radius of the green disk can be expressed by $D(c) - 1$, where $D(c)$ is the distance transformation value of the center pixel c. Note that these disks are the maximal balls included in the crack and tangent with the crack boundary; thus, they are tools with which to measure the width of the crack. The width at center pixel c can be expressed by $2_*[D(c) - 1] + 1$, namely $2_*D(c) - 1$. Since the digital image space is discrete, this formula may cause errors for a pixel width with an even value, but this error would not exceed 1 pixel in theory, and Section 5.2 indicates that the error percentage of the width decreases rapidly with an increase in the width. After the width value of all the medial axis pixels is calculated, the mean width of a crack is the mean width value of all these pixels, and the maximum width is the maximum width value. The length of a crack is considered as the curve length of the medial axis (denoted

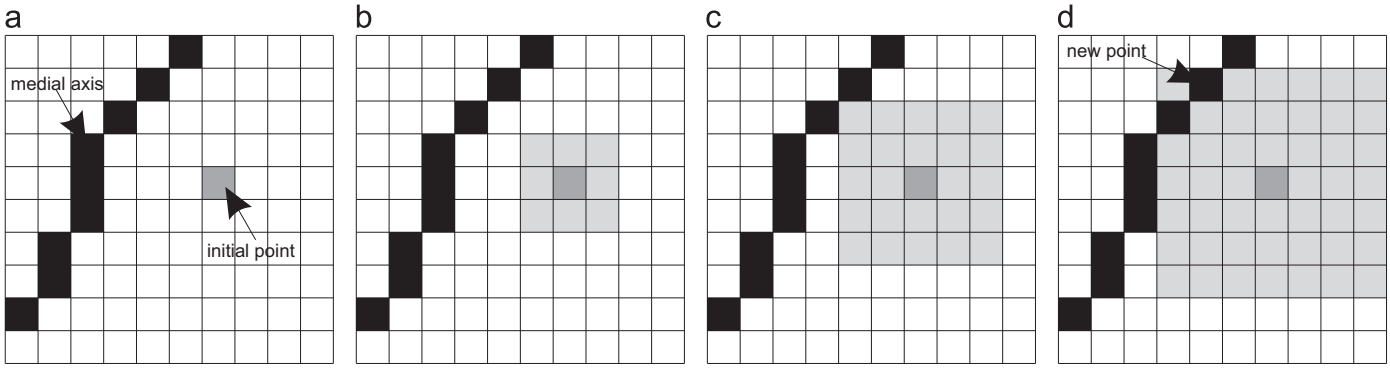

Fig. 11. Procedure of finding the nearest point. (a) Image with initial point and medial axis, (b) first search iteration, (c) second search iteration, (d) third search iteration and first meet with the medial axis.

Fig. 13. Software user interface.

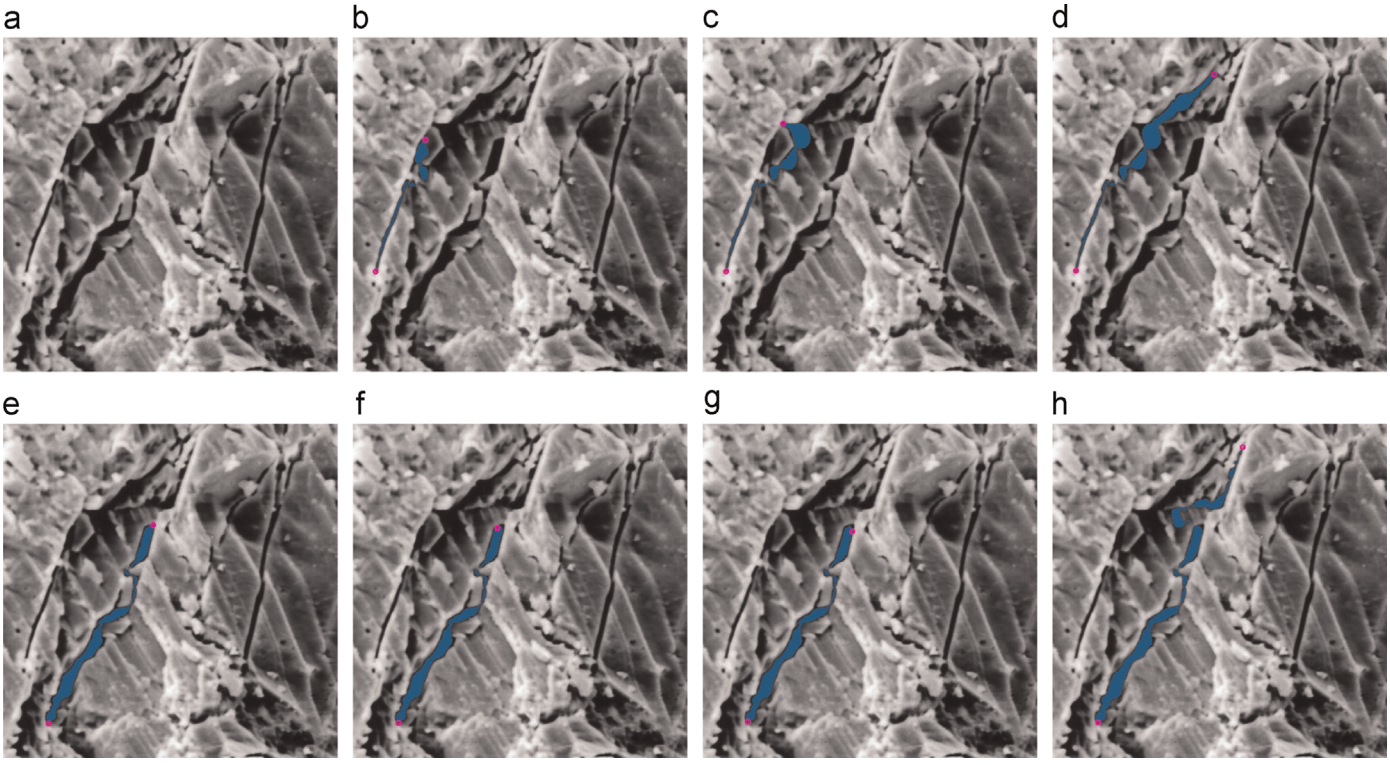

Fig. 14. Crack results of different endpoints. The colored circles are endpoints and the blue area is the crack result. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

by s) combined with the radius value of the two disks at both ends (denoted by $r1$ and $r2$), namely $s + r1 + r2$. The angle of a crack refers to the intersection angle between the straight line, which is the least square fitting of the medial axis (the yellow line in Fig. 12), and the horizontal axis towards the right. The area of a crack is the number of pixels of the rebuilding crack area during the last extraction process step, and since this rebuilding crack area is constructed by discrete disks, the area value may have errors; Section 5.2 shows how the errors change.

5. Results and discussion

To evaluate the effect of the proposed approach in this paper, we developed a software with a friendly interactive interface as

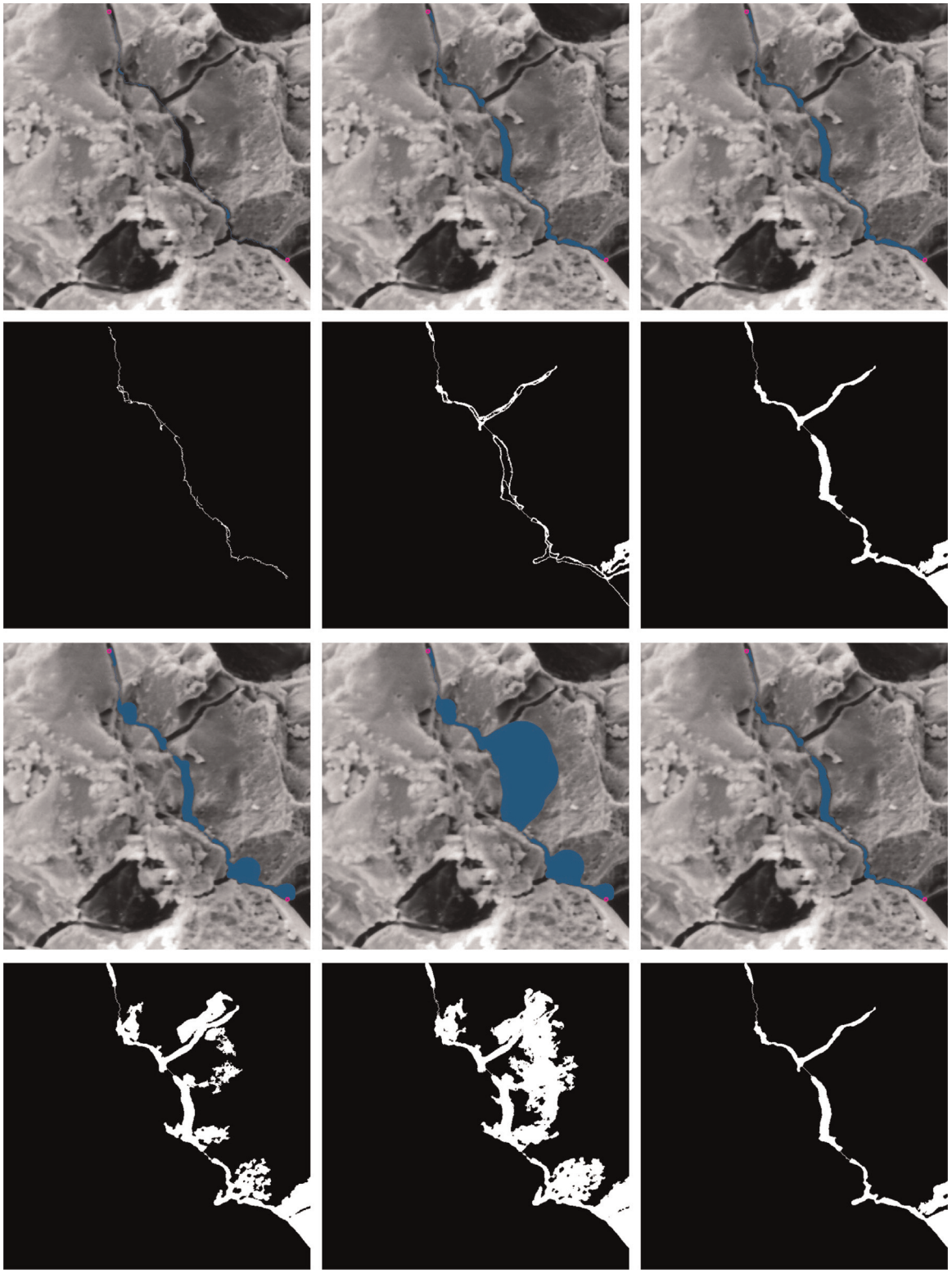

Fig. 15. Crack result and region growing result of different region growing threshold value ε_e. Rows 1 and 3 denote the crack results of ε_e = 5, 15, 25, 40, 45, 24.14 respectively and 24.14 is the default value of the software calculated using the standard deviation of the seed pixels. Rows 2 and 4 represent the results after the region growing operation, which correspond to the above images.

shown in Fig. 13. The software was developed using the Microsoft Visual Studio 2010 platform; the interface was coded in C# (WPF environment), and the core algorithm was coded in C++ and tested on a computer running Windows 8.1 operating system (.net framework 4.0 above is required) and equipped with 4CPU (2.4 GHz) and 8 GB of RAM.

5.1. *Effects of parameters*

The input parameters in the extraction process include the two endpoints of a crack (Section 3.2) and the region growing threshold ε_e (Section 3.3), and they have a significant impact on the crack result.

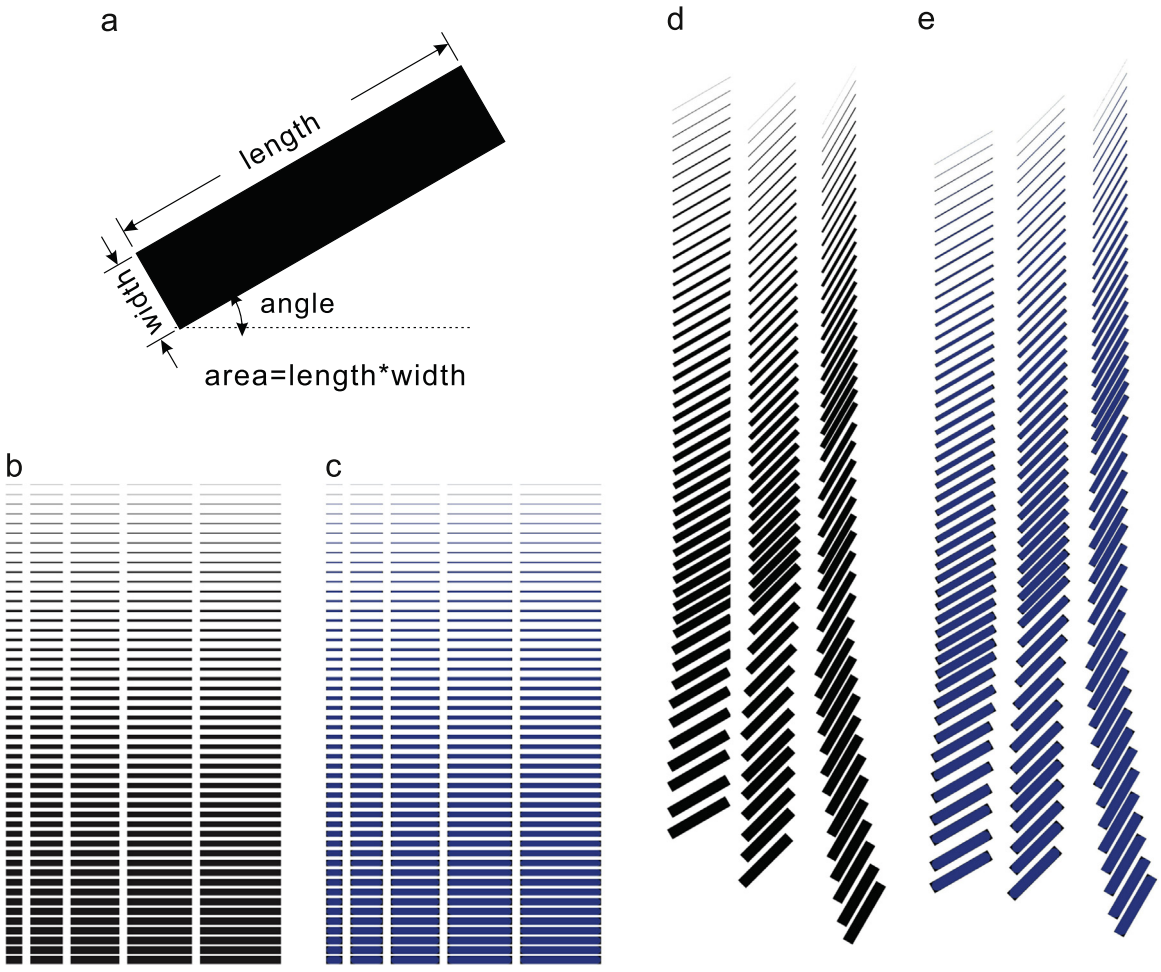

Fig. 16. Images created manually and the results of extraction. (a) Measurement of the actual length, width, area, and angle of a crack, (b) five groups of horizontal cracks. The actual length is 100, 200, 300, 400, and 500 from left to right, and the actual width is 1–50, from top to bottom. (c) Crack results of extraction. One blue colored area represents a crack result. (d) Three groups of cracks. The actual length is 300 and the actual angle is 30°, 45°, and 60° from left to right, and the actual width is 1–50, from top to bottom. (e) Crack results of extraction. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

As the endpoints are updated in the removing branches step (Section 3.4.3) to make sure that the new endpoints are on the medial axis, the location of the original endpoints does not affect the crack result within a small range, as shown in Fig. 14e–g. However, the original endpoints are selected manually by the user; these contain important general location information of the crack and they dominate the initial A* searching path that is the base of next steps. If they change considerably, the crack result would be much different, as illustrated in Fig. 14b–d.

The region growing threshold ε_e controls the initial crack area size. If ε_e is too small, the area size will be small and the actual crack pixels will be lost. Alternatively, the size may be too large and over identified. The region growing crack results of different values of ε_e are shown in Fig. 15. Here, we present a relatively compromising method to choose ε_e, although it is not an optimal value for all cracks. The A* path (Section 3.2) goes through a relatively dark place, and each pixel on the path represents a sample of the crack and may have a different gray level. We calculate the standard deviation value (denoted by sd) of the pixels on the A* path and then ε_e can be expressed as

$$\varepsilon_e = \begin{cases} 10 & \text{if } sd \le 10 \\ sd & \text{otherwise} \end{cases} \quad (2)$$

Here, ε_e is set to 10 if sd does not exceed 10 in the case that the pixel value of the crack is very close but mixed with noise. This ε_e value is set to the default threshold in our software, and the last result in Fig. 15 was extracted using this threshold, which is 24.14.

5.2. Error analysis

To check the accuracy of our algorithm, we tested it on "good condition" images that were manually created in Adobe Photoshop CS4, since we can obtain the exact feature value of the cracks for comparison, which is measured as shown in Fig. 16a. This figure indicates that the tested areas are characterized by long and thin shapes to simulate cracks. Fig. 16b shows five groups of cracks with horizontal orientation, and each group has 50 cracks with an actual pixel width of 1–50 pixels, and the length of the cracks in each group increases from 100 to 500 pixels. The two endpoints of each crack are chosen at the center of two width ends automatically by script, and the region growing threshold ε_e is the default value of the software. The length and angle of the cracks generated by the proposed approach are very close to the actual values, where the mean length of each group is 97.232, 196.632, 297.232, 396.632, 497.232, and the angle is $0 \pm 0.1°$. The maximum width, mean width and area value of the cracks with small even width have numerous errors, as shown in Fig. 17a–c, but they quickly reduced to a very low level as the actual width increased.

Fig. 16 d shows the three groups of cracks with the same length as 300 pixels; each group has 50 cracks with an actual pixel width of 1–50 pixels, and the actual angle of the cracks of each group is

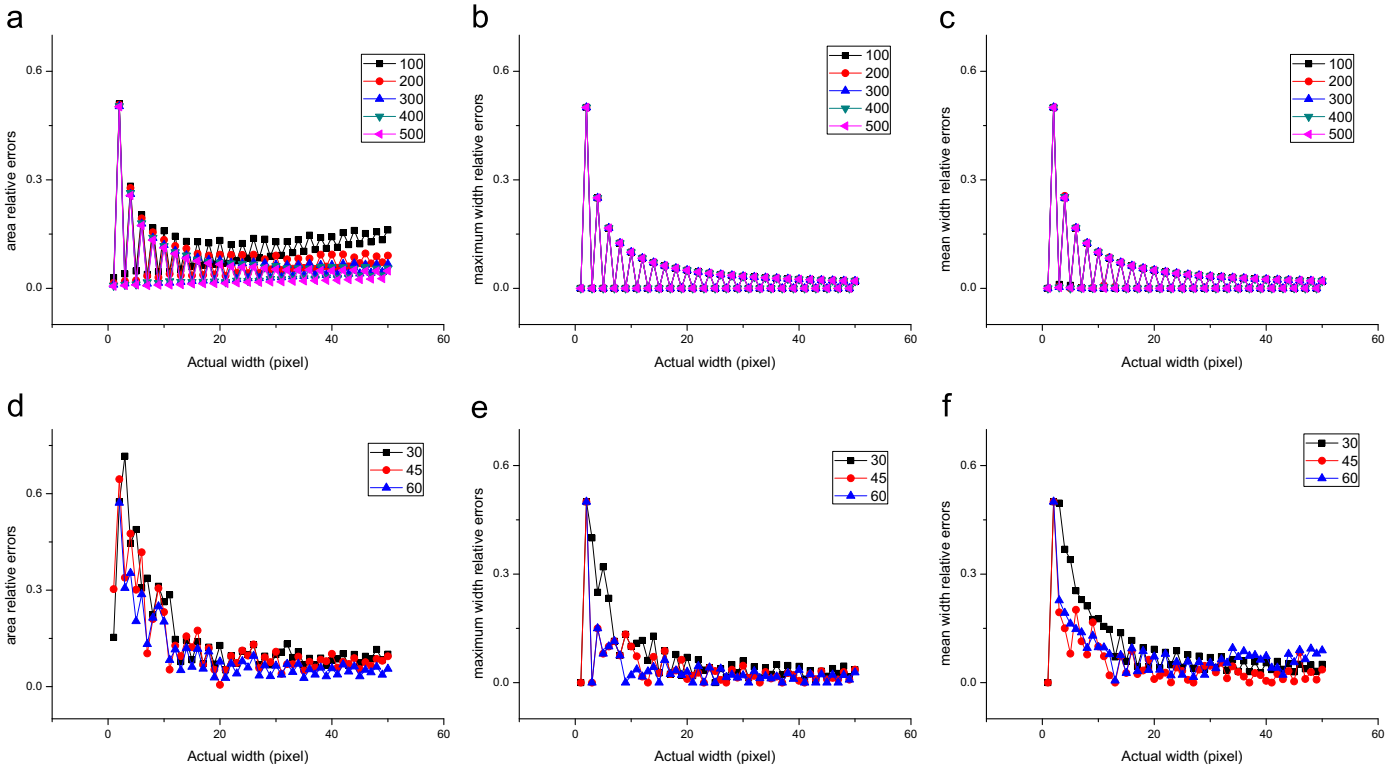

Fig. 17. Errors data graph. (a) Errors of the area values of the 5 groups, (b) errors of the maximum width values of the 5 groups, (c) errors of the mean width values of the 5 groups, (d) errors of the area values of the 3 groups, (e) errors of the maximum width values of the 3 groups, (f) errors of the mean width values of the 3 groups.

30°, 45°, and 60°. The two endpoints of each crack are selected by mouse clicking close to the ends of the crack. ε_e is set to 100 because we use the anti-aliased method to create these cracks, and boundary pixels of the crack have larger gray level and ε_e values that are too small may lead to the loss of these pixels. The mean length of the 30°, 45°, and 60° groups is 313.152, 294.5, 322.620, and the angle is between [29.9, 30.28], [44.29, 45.07], and [58.33, 60.34] respectively. The maximum width, mean width and area value also reduced with an increase in the width.

The above experiments' results indicate that the width and area feature values may have errors that are too large to be used for the small width cracks. However, as the actual width increases, these errors can be ignored, which means that a higher resolution (dpi) of the image can also reduce the errors. The rebuilt crack area is nearly identical to the original, as shown in Fig. 16c and e, except for the sharp corners.

5.3. Assessment of SEM images of rocks

The SEM images are acquired from a sample of load-cracked marble. In order to maintain the original appearance of microcracks, the specimens were not polished. However, this results in unfavorable crack conditions with spatial discontinuity owing to fragments, as well as unreliable crack shadows owing to uneven surfaces, and low contrast in some local areas, as illustrated in Fig. 18.

If there is more than one crack in an SEM image, the user should choose the two endpoints of a crack and then confirm these to search the crack; this operation should be repeated to extract the next crack. The region growing threshold for each crack can be set to the custom value. To compare the extraction performance of the proposed approach, we also use the traditional Otsu (1975) method to segment the image. In order to compare the difference between the extraction result and the original image more clearly, the software uses different colors to identify the crack results and uses an overlay algorithm to combine the colored crack area with the original image, which makes the background pixel in the colored crack area appear brighter. Fig. 18 shows the results of different methods; the results of the Otsu method are over identified, whereas the results of our method are quite consistent with the actual darker area and each crack result is continuous, which is good for crack analysis. Since we optimize the underlying code of the software, the runtime of searching one crack is controlled in approximately 2000 ms.

However, this method failed in certain situations, as illustrated in Fig. 19, because the background pixel gray level is very similar to the crack, and the location is close to the crack. This leads to A* searching for a less costly path through this area, which finally causes a wrong crack result.

6. Conclusions

In this paper, a new extraction method was proposed for microcracks in rocks; this method is based on a graph searching technique. This technology is introduced to extract crack objects in images for the first time, based on the currently available literature.

A fully automatic algorithm is certainly desirable, but not completely adaptable for the microcrack situation here. Thus, some a priori knowledge of the crack is required in our method, namely the two endpoints of a crack. A path throughout the crack region, which is determined by the given endpoints, is used as the region growing seed. After the crack area is restored by region growing, the algorithm employs the endpoint information to separate the crack network to extract a single crack without branches. This isolated crack result reflects the actual microcrack more closely to some extent.

It is, however, recognized that the crack result does not appear to match perfectly with the origin in certain sharp places, partly due to the fact that the cracks are rebuilt using disks. In addition, the region

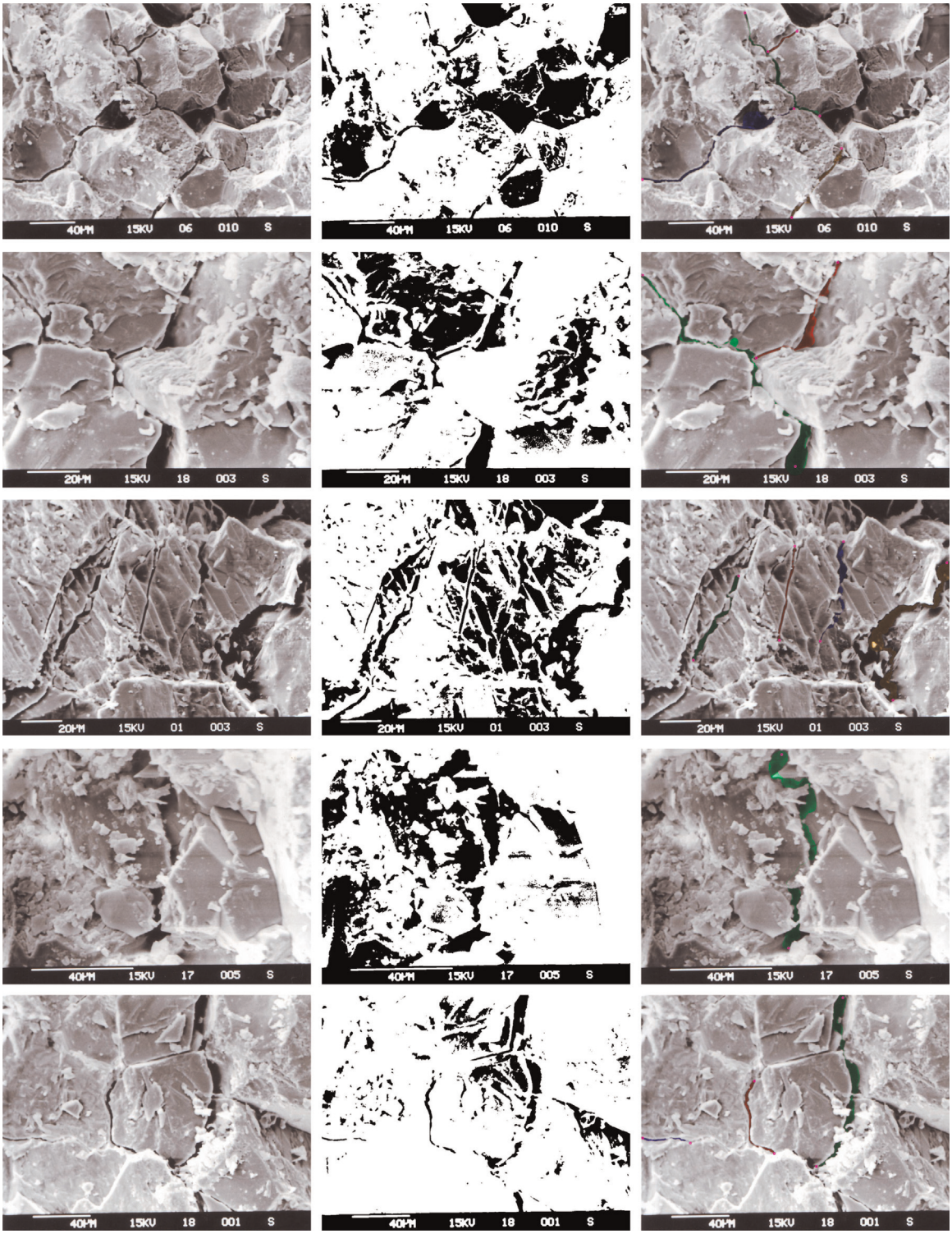

Fig. 18. Results of SEM images using the Otsu method and the proposed approach. Column 1 denotes the original SEM images, Column 2 contains the result of the Otsu method, and Column 3 is the result of the approach adopted in this paper, which identifies the cracks with different colors. The colored circles are the chosen endpoints. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

growing threshold affects the crack result as well. It should be also noted that we have to generate the two endpoints for every crack in an image manually, although the algorithm is much more effective compared with traditional manual methods. Hence, this method may not be suitable for images with a large number of cracks.

Finally, the proposed algorithm can be used for microcrack extraction and can also potentially be used for any type of linear structural object extraction, such as macrocracks on pavement images or soil desiccation cracks.

Acknowledgments

This research was partially carried out with financial support from the National Basic Research Program of China (973 Program, Grant no. 2011CB013504 and no. 2015CB057903), and project

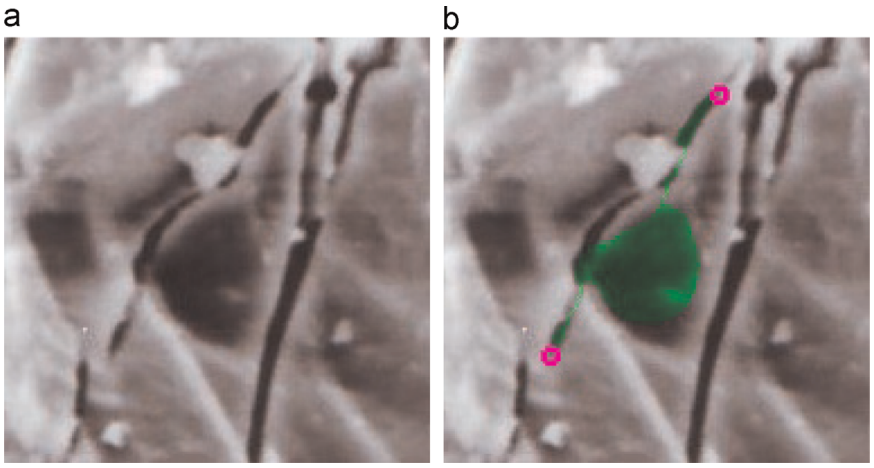

Fig. 19. Failure situation of extraction. (a) The original image, (b) the extraction result colored with green. The crack result is not what we want since there are background pixels, which may be darker than the crack, close to the crack. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

support from the National Natural Science Foundation of China (NSFC, Grant no. 51379065, no. 41272329 and no. 51309089) and the Fundamental Research Funds for the Central Universities (Grant no. 2015B06014 and no. 2014B33614). We thank LetPub (www.letpub.com for its linguistic assistance during the preparation of this paper.

Appendix A. The A* algorithm

The A* algorithm uses a cost function to determine the order in which the search visits nodes, and the cost function (denoted by F) is the sum of two functions: one is the past path-cost function (denoted by G), which is the known distance cost from the start node to the current node; the other is the future path-cost function (denoted by H), which is an admissible heuristic estimate of the distance cost from the current node to the goal node. The relationship can be expressed as $F = G + H$. These functions are indicated by a two-dimensional matrix in the algorithm implementation, and the size of these matrices is the same as the input image.

In the visiting process, a matrix (denoted by *STATUS*) is used to indicate the node status, which is specified by *unchecked*, *opened*, and *closed*. The status of all nodes is initialized as *unchecked*, and the future path-cost function H is precomputed using a straight-

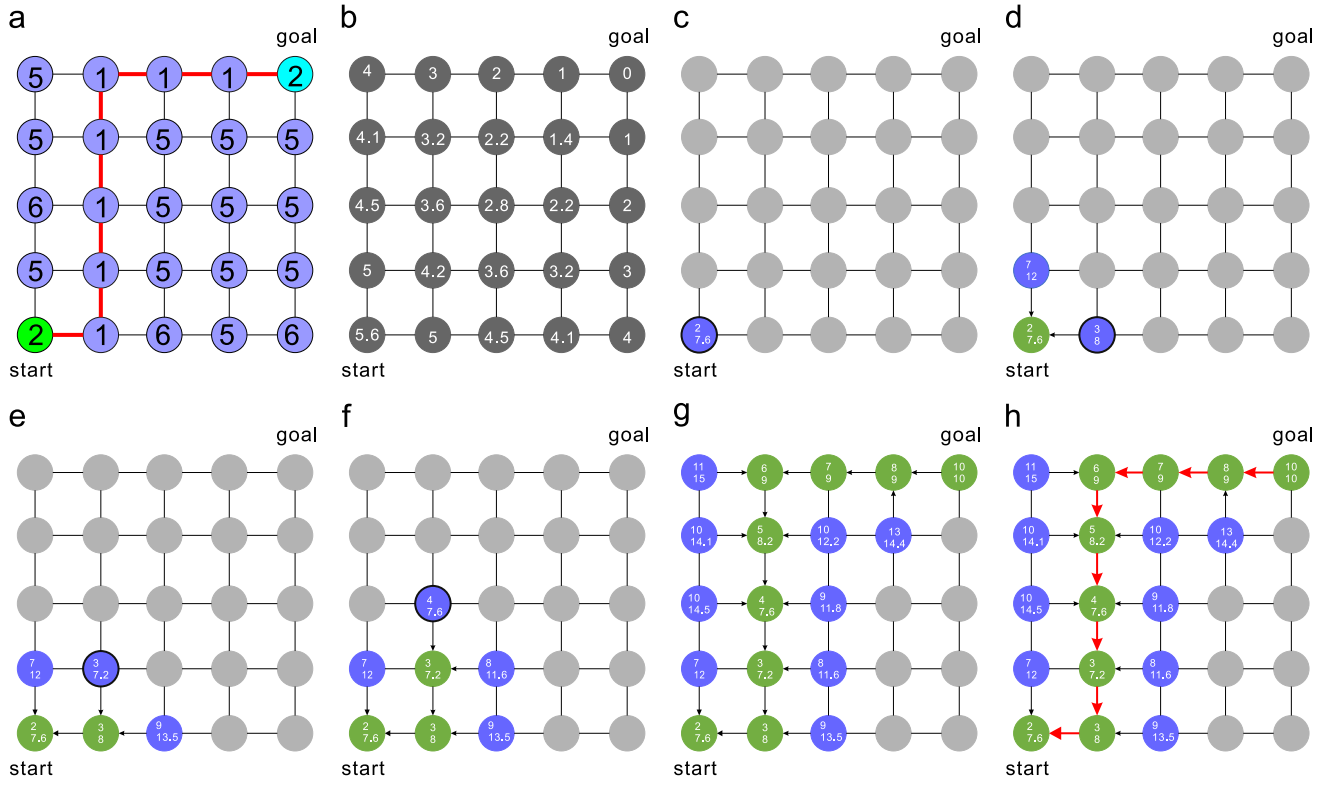

Fig. A1. Overview of the A* algorithm process. (a) An image-generated graph, (b) future path-cost function F, which is computed using the Euclidean straight-line distance to the goal node, (c) the initial visiting process, (d) the second visiting, (e) the third visiting, (f) the fourth visiting, (g) the last visiting, (h) backtrack the least-cost path. In (c)–(h), the green color represents the status of *closed*, and the blue color is used for *opened*, and the gray color is used for *unchecked*. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

line distance to the goal node, as illustrated in Fig. A1b.

Let p be the current node that is being visited, and the start node is the initial current node. Then, the visiting process follows as outlined below.

Repeat until the goal node is *closed*:

(Step 1) Set the status of neighbors (not *closed*) of p to *opened* (If it is already *opened*, consider whether the updated G value by step 2) is smaller than its existing G value? If true, update the G, F value and its parent, otherwise, do nothing).

(Step 2) Update the G value of the neighbors (for a node weighted graph, it is namely to add the node weight of the neighbors to the G value of p).

(Step 3) Update the F value of the neighbors ($F = G + H$).

(Step 4) Update the parent of the neighbors (Its parent here represents the node that recently made it *opened*).

(Step 5) Set the status of the current node p to *closed*.

(Step 6) Compare the F value of all *opened* nodes, and the node with a minimum F value is set to the next current node.

Then, the path can be obtained by backtracking the parent from the goal node. Fig. A1 shows the process of the A* algorithm on a simple node-weighted graph.

Appendix B. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2015.08.013.

References

Arena, A., DellePiane, C., Sarout, J., 2014. A new computational approach to cracks quantification from 2d image analysis: application to micro-cracks description in rocks. Comput. Geosci. 66, 106–120.

Borgefors, G., 1986. Distance transformations in digital images. Comput. Vis. Graph. Image Process. 34 (3), 344–371.

Calabi, L., Hartnett, W.E., 1968. Shape recognition, prairie fires, convex deficiencies and skeletons. Am. Math. Mon., 335–342.

Cuisenaire, O., Macq, B., 1999. Fast Euclidean distance transformation by propagation using multiple neighborhoods. Comput. Vis. Image Underst. 76 (2), 163–172.

Danielsson, P.-E., 1980. Euclidean distance mapping. Comput. Graph. Image Process. 14 (3), 227–248.

Dare, P., Hanley, H., Fraser, C., Riedel, B., Niemeier, W., 2002. An operational application of automatic feature extraction: the measurement of cracks in concrete structures. Photogramm. Rec. 17 (99), 453–464.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

Gonzales, R.C., Woods, R.E., 2002. Digital Image Processing, 2nd ed. Prentice Hall, Upper Saddle River, New Jersey.

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107.

Holt, C.M., Stewart, A., Clint, M., Perrott, R.H., 1987. An improved parallel thinning algorithm. Commun. ACM 30 (2), 156–160.

Jahanshahi, M., Masri, S., Padgett, C., Sukhatme, G., 2013. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. Mach. Vis. Appl. 24 (2), 227–241.

Kranz, R.L., 1983. Microcracks in rocks: a review. Tectonophysics 100 (1), 449–480.

Lam, L., Lee, S.-W., Suen, C.Y., 1992. Thinning methodologies—a comprehensive survey. IEEE Trans. Pattern Anal. Mach. Intell. 14 (9), 869–885.

LeRoux, S., Medjedoub, F., Dour, G., Rézaï-Aria, F., 2013. Image analysis of microscopic crack patterns applied to thermal fatigue heat-checking of high temperature tool steels. Micron 44, 347–358.

Li, Q., Zou, Q., Zhang, D., Mao, Q., 2011. Fosa: F* seed-growing approach for crack-line detection from pavement images. Image Vis. Comput. 29 (12), 861–872.

Liu, C., Shi, B., Zhou, J., Tang, C., 2011. Quantification and characterization of microporosity by image processing, geometric measurement and statistical methods: application on sem images of clay materials. Appl. Clay Sci. 54 (1), 97–106.

Liu, C., Tang, C.-S., Shi, B., Suo, W.-B., 2013. Automatic quantification of crack patterns by image processing. Comput. Geosci. 57, 77–80.

Martelli, A., 1972. Edge detection using heuristic search methods. Comput. Graph. Image Process. 1 (2), 169–182.

Nilsson, N.J., 1982. Principles of Artificial Intelligence. Springer, New York.

Otsu, N., 1975. A threshold selection method from gray-level histograms. Automatica 11 (285–296), 23–27.

Parker, J.R., 2010. Algorithms for Image Processing and Computer Vision, 2nd ed. John Wiley & Sons, Indianapolis, Indiana.

Pu, Y., Wang, W., Zhou, J., Wang, Y., Jia, H., 2008. Fractional differential approach to detecting textural features of digital image and its fractional differential filter implementation. Sci. China Ser. F: Inf. Sci. 51 (9), 1319–1339.

Shengxun, Z., 1984. A thinning algorithm for discrete binary image. J. Zhejiang Univ. (Eng. Sci.) 1, 001.

Sinha, S.K., Fieguth, P.W., 2006. Automated detection of cracks in buried concrete pipe images. Autom. Constr. 15 (1), 58–72.

Valença, J., Dias-da Costa, D., Júlio, E., 2012. Characterisation of concrete cracking during laboratorial tests using image processing. Constr. Build. Mater. 28 (1), 607–615.

Vincent, L.M., 1991. Efficient computation of various types of skeletons. In: Medical Imaging V: Image Processing. International Society for Optics and Photonics, Bellingham, Washington, pp. 297–311.

Zou, Q., Cao, Y., Li, Q., Mao, Q., Wang, S., 2012. Cracktree: automatic crack detection from pavement images. Pattern Recognit. Lett. 33 (3), 227–238.