Image and Vision Computing 29 (2011) 861–872

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

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

FoSA: F* Seed-growing Approach for crack-line detection from pavement images[☆]

Qingquan Li [a,b], Qin Zou [a,b,*], Daqiang Zhang [c], Qingzhou Mao [a]

[a] Transportation Research Center, Wuhan University, Wuhan 430079, China
[b] School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China
[c] School of Computer Science, Nanjing Normal University, Nanjing 210097, China

ARTICLE INFO

Article history:
Received 28 July 2010
Received in revised form 29 March 2011
Accepted 10 October 2011

Keywords:
Line detection
Pavement crack
Seed-growing
Dynamic programming

ABSTRACT

Most existing approaches for pavement crack line detection implicitly assume that pavement cracks in images are with high contrast and good continuity. This assumption does not hold in pavement distress detection practice, where pavement cracks are often blurry and discontinuous due to particle materials of road surface, crack degradation, and unreliable crack shadows. To this end, we propose in this paper FoSA — F* Seed-growing Approach for automatic crack-line detection, which extends the F* algorithm in two aspects. It exploits a seed-growing strategy to remove the requirement that the start and end points should be set in advance. Moreover, it narrows the global searching space to the interested local space to improve its efficiency. Empirical study demonstrates the correctness, completeness and efficiency of FoSA.

1. Introduction

The detection of curvilinear structures, also referred to as lines, is a fundamental low-level operation, which has been adopted in various applications in computer vision and pattern recognition [1–3]. In general, line-detection algorithms can be classified into two types: local and global algorithms. The former exploits local features, such as intensity, gradient and local variance, to achieve goals of line enhancement and segmentation. It involves a series of edge detection operators [4, 5], morphological filter [6], steerable filter [7], and isotropic non-linear filter [2]. The later tracks and extracts lines in an overall view through dynamic programming to optimize target functions to a certain criterion. It consists of MAP statistic model [8], graphic model [9–11], snake model [12, 13], and decision tree model [3].

A pavement crack is typically with a curvilinear structure. A variety of approaches for pavement crack detection have been proposed in the last decade, but most of them cannot automatically detect cracks owing to grain-like characteristics of the road materials. They implicitly assume that speckle noises in image background are in low-level, and pavement cracks in images are with high contrast and good continuity. However, this assumption does not always hold in real world due to two reasons. One is that pavement images usually are mixed with the grain-like textured background, which acts as speckle noises that significantly affects the detection accuracy. The other is that cracks in pavement images are characterized by low Signal-to-Noise Ratio (SNR), low contrast, and bad spatial continuity.

Figure 1(a) shows a typical pavement image, Fig. 1(b) and (c) are the results from traditional local methods stemming from Canny edge detection [4] and wavelet transform [14]. As cracks are line-like structures on a large scale, these methods, that use small scale information, tend to extract only fragments of them.

In this paper, we propose FoSA — F* Seed-growing Approach for crack-line detection by extending the F* algorithm, which takes advantage of dynamic programming to track linear structures in a global view. It presents a seed-growing strategy to eliminate the requirement in the F* algorithm that the start and end points for tracking should be set beforehand. Thus, FoSA is capable of automatically identifying the start and end points. It also puts forward an interest-constrained technique which narrows the global searching space to the local space, and hence dramatically improves the efficiency of the F* algorithm. In fact, FoSA formulates the crack extraction problem as a seed-growing problem. It uses a filtering based on average path cost (i.e., APC-based filtering) over the crack element set to aggregate crack seeds with high credibility. With these seeds, FoSA presents an F* seed-growing algorithm (i.e., FoS) to collect crack strings. Finally, FoSA conducts pruning and linking operations to refine the crack strings and extract the whole identified cracks.

The rest of this paper is organized as follows. Section 2 briefly overviews the related work on pavement crack detection. Section 3 introduces the F* seed-growing algorithm. Section 4 discusses FoSA in detail. Section 5 reports our empirical study and Section 6 concludes our work by pointing out future directions.

2. Related work

Intuitively, image-based techniques are fundamental in pavement crack detection, which have received intensive attention since the

☆ This paper has been recommended for acceptance by Paolo Remagnino.

* Corresponding author at: Transportation Research Center, Wuhan University, Wuhan 430079, China Tel.: +86 2768778039; fax: +86 2768778043.
E-mail addresses: qqli@whu.edu.cn (Q. Li), qinnzou@gmail.com (Q. Zou).

0262-8856/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2011.10.003

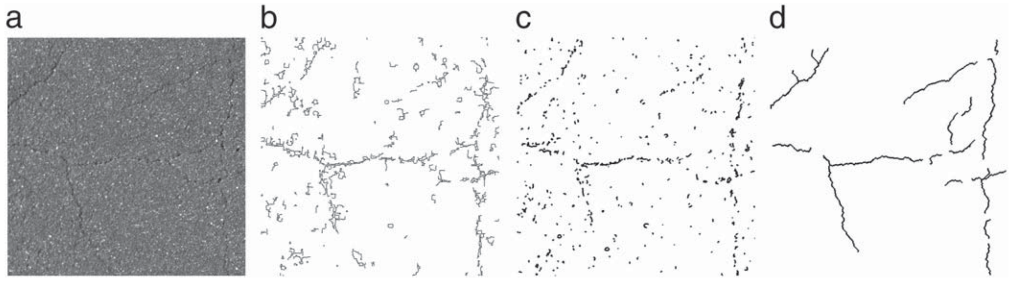

Fig. 1. Pavement crack detection with different approaches. (a)A typical pavement image, (b)Canny edge detection approach, (c)Wavelet transform based approach, (d)The proposed approach.

early of 90s of the 20th century. Various approaches have been proposed to detect cracks. This section gives a brief overview of them.

Thresholding-based segmentation is a common technique for pavement crack detection. The histogram method [15] and the iterated clipping method [16] are the initial research. Due to the diversity of the pavement environment, they have difficulty in gaining a stable performance. By using the neighboring difference information to find a threshold, [17] gets a better result than classical methods of Otsu [18] and Kapur [19] in pavement crack segmentation. A dynamic optimization-based method tested in [20] is effective for segmenting low SNR pavement images, but it suffers a high computation cost.

Edge detection is another common technique for pavement crack detection, which shares the principle that cracks detection can be achieved by identifying crack edges. In [21], the Sobel edge detector was studied to detect cracks, in which bidimensional empirical mode decomposition (BEMD) was applied to smooth pavement crack images and remove speckle noises. Wavelet transform has also been exploited in edge detection for pavement crack detection. In [14], wavelet transform was employed to decompose a pavement image into subbands. Consequently, the distresses were separated from noise and background by several statistical criteria stemming from wavelet coefficients. In [22], a 2D continuous wavelet transform was applied to create complex coefficient maps in a series of scales. With complex coefficient maps, the modulus and phase maps were built and a maxima location map was obtained for further crack segmentation. However, due to the anisotropic characteristic of wavelet, wavelet-based approaches often fall short when handling cracks with high curvature or bad continuity.

Note that a great many techniques from artificial intelligence, data mining, machine learning and neural network have been employed in checking pavement cracks [23–28]. Generally, they sample each pavement image into a number of grid sub-images, in which feature extraction was done and feature vectors were generated for training and classification. In [23], moment invariant was regarded as the class feature for different types of distresses. Then, the back-propagation (BP) neural network was used to train and classify these features. While in [24], statistical values such as mean and standard deviation were used as class features, and a curve detector was adopted for refining the results from the classification step. In [25] and [26], BP model was exploited in a slightly different manner. In class feature selection, the former used four spatial-distribution related parameters, while the later used an anisotropy measure. In [27, 28], a supervised Bayesian classifier was exploited. The training images were sampled with a size of 65×65 pixels, and labeled as crack or non-crack. Each training image was normalized for feature extraction. The mean and standard deviation of pixel intensities within image windows were regarded as class features. As crack feature extraction is restrained in the sub-image windows, these methods can be concluded as local methods based on local feature detection, which would inevitably expose short in describing the global linear features of the crack.

In addition, a wide spectrum of other approaches has also been proposed for pavement crack detection. In the idea of inspecting cracks on a best co-joint resolution in both spatial domain and frequency domain, [29, 30] presented a new image transformation using the Wigner model to detect crack defects from a textured background. Based on the assumption that crack pixels are always darker than their neighborhoods, fuzzy set theory was exploited in [31] to detect and segment cracks. An extended method based on fuzzy theory was studied in [32] to automatic pavement distress detection. Morphological filters were introduced for enhancing pavement crack images [33, 34]. In [35], crack analysis was carried out on cells with a size of 8×8 pixels. Each cell was classified as a crack or non-crack cell by using the gray scale information of border pixels and its orientation was specified by the crack string formed in the cell. Then neighboring crack strings with similar orientations were joined together and finally cracks were extracted using parameters such as contrast and length. In [36], a method based on segment extending was proposed, which achieved crack extraction by filtering and combining the crack fragments according to a bunch of rules. This method has some feasibility as it counted in some geometric features of the crack.

Cracks in pavement images are often mixed with speckle noises, and with low contrast and bad spatial continuity owing to particle materials of road surface, crack degradation and unreliable crack shadows. This incurs the ineffectiveness of most existing approaches of pavement crack detection.

3. FoS: F* Seed-growing

In this section, we first give an introduction to the background of F* algorithm, and then present the FoS — F* Seed-growing algorithm.

3.1. F* algorithm

F* algorithm was originally proposed by Ford for computing the minimum cost path between vertices in a graphic network [37]. Here we introduce its application in an image-generated grid graph.

Let $\{(i,j)|0 \le i < row, 0 \le j < col\}$ be the node set in a grid graphic network generated from an image $I(row,col)$, P_s and P_e be the start point and the end point, $c_{i,j}$ be the cost on node (i,j), and $x_{i,j}$ be the path cost from node (i,j) to the start node. Then the iterative algorithm to find the minimum cost path could be described as follows.

i). Set a start point. Set $P_s(m,n)$ as the start point, and initialize the path cost matrix X by

$$x_{i,j} = \begin{cases} 0, & \text{if } i = m \&\& j = n \\ \infty, & otherwise \end{cases}. \tag{1}$$

ii). Calculate the path cost matrix X. To achieve this, we update the matrix X by Eq. (2) until all nodes in X reach stable.

$$x_{i,j} = \begin{cases} x_{i+a,j+b} + c_{i,j}, & if \ x_{i,j} - x_{i+a,j+b} > c_{i,j} \\ x_{i,j}, & otherwise \end{cases} \tag{2}$$

where $(i+a,j+b)$ denotes a neighbor node of (i,j), and the values of a and b are -1, 0 or 1.

iii). Get the minimum cost path from P_e to P_s. The minimum cost path could be obtained by a node moving from P_e backwards to P_s in the direction of path cost decreasing in X.

Figure 2 illustrates path tracking by using the F* algorithm. Fig. 2(a) is a vertex cost matrix, constructed by an image-generated grid graph. Once the right bottom corner point is set as the starting point, the path cost matrix could be initialized as in Fig. 2(b). After a four round iteration, the path cost matrix becomes stable, as shown in Fig. 2(e). Then, the minimum cost path can be obtained through a backward tracking, and the tracked path is shown in Fig. 2(f).

F* algorithm attracted extensive attention due to its high performance on linear structure detection from remote sensing images [10]. It was improved in [11] to extract roads and ridges in SPOT images with complex background. Meanwhile, it was also used to enhance low-contrast curvilinear features [38]. Through searching a minimum cost path, F* algorithm tracks linear structures accurately, even in the presence of heavy noise. However, it still involves several problems to apply F* algorithm to automatic crack line detection, e.g., the setting of the start and end points for the tracking, the low efficiency when handling large-scale images, etc.

3.2. FoS

FoS (i.e., F* seed-growing) is a seed-growing algorithm based on F* algorithm. It aims at searching a path through the seed point S and across the image with a minimum average path cost. To describe FoS, we initially give a definition to the average path cost.

Definition 1. *Let $x_{i,j}$ be the minimum path cost from point (i,j) to the start point P_s, $l_{i,j}$ be the number of nodes which make up the according path, define $\hat{x}_{i,j}=x_{i,j}/l_{i,j}$ be the average path cost from (i,j) to P_s.*

Suppose I be an image with a width of len (for convenience, we assume it is a square image), S be a seed point (start point), X be the minimum path cost matrix, L be the matrix recording the number of nodes on each path in accord with the element in X, $\hat{X}$ be the average path cost matrix, and some functions be described as follows, then FoS can be illustrated by Algorithm 1.

- $X,L \leftarrow F^*(I,S)$. By using F* algorithm, it gets X and L through the input I and S.

```
Algorithm 1 FoS: F* Seed-growing algorithm
1: procedure FoS
2: input: $I$: an image, $S$: a seed point
3: output: $P_{e1}$, $P_{e2}$: two endpoints from seed growing
4: $PH_{Pe1-S-Pe2}$: the seed-growing path
5: $X, L \leftarrow F^*(I, S)$
6: $\hat{X} \leftarrow getAvePathCost(X, L)$
7: $B \leftarrow getBoundary(\hat{X})$
8: for each $B_i \in B$ do
9:     $N_i \leftarrow getNeighbors(B_i, len/2)$
10:    if $B_i$ is the lowest one in $N_i$ then
11:        Add $B_i$ into the candidate endpoint set $\mathcal{M}$
12:    end if
13: end for
14: $P_{e1}, P_{e2} \leftarrow getEndpoints(\mathcal{M})$
15: $PH_{Pe1-S-Pe2} \leftarrow getPath(X, S, P_{e1}, P_{e2})$
16: end procedure
```

- $\hat{X} \leftarrow getAvePathCost(X,L)$. Once X and I are gained, the average path cost could be achieved according to Definition 1.
- $B \leftarrow getBoundary\left(\hat{X}\right)$. It gains the boundary elements of $\hat{X}$.
- $N_i \leftarrow getNeighbors(B_i, len/2)$. Taking B_i as the center, it gets len neighboring points. Exactly, $len/2$ points on the left and $len/2$ points on the right.
- $P_{e1}, P_{e2} \leftarrow getEndpoints(\mathcal{M})$. It gets two endpoints from the candidate endpoints set $\mathcal{M}$ by using Eq. (3), where $Apc()$ means to calculate the average

$$\{P_{e1},P_{e2}\} = arg \min_{\{P_i,P_j\}} \left\{Apc\left(PH_{P_i-S-P_j}\right)\middle| P_i,P_j \in \mathcal{M}, and\ i \neq j\right\} \quad (3)$$

path cost on the corresponding path. Note that FoS assumes that the path crosses the image, we constrain P_{e1} and P_{e2} to be on the different edges of the image boundary.
- $PH_{P_{e1}-S-Pe2} \leftarrow getPath(X,S,P_{e1},P_{e2})$. It respectively tracks the path from P_{e1} and P_{e2} to S, and thus get the seed-growing result.

Figure 3 illustrates FoS by an example. Fig. 3(a) is an original image. When a seed point S is set, we can calculate the average path cost matrix, and display it with a gray image shown in Fig. 3(b). We stretch the intensity range of image (b) to [0, 255] and generate a stretched image, Fig. 3(c), from which we can clearer observe the difference in brightness over the image, and find that the dark area matches well with the crack area in Fig. 3(a). After finding out the two endpoints e_1 and e_2 on image (b)'s boundary, we then, with a backward tracking, could gain the seed growing path, as shown in Fig. 3(d).

4. FoSA: F* Seed-growing Approach for crack detection

In this section, we propose FoSA — F* Seed-growing Approach based on the F* algorithm for automatic crack detection. We start with an overview of its design, followed by detail introduction to its each part.

4.1. An overview of FoSA

The main idea of the proposed approach is to detect the cracks with a seed-growing strategy. Specifically, it employs the FoS (i.e., F* seed growing) algorithm presented in Section 3.2. Thus, we concisely name the proposed crack detection approach FoSA for convenience.

As an irregular linear target, a crack is composed of a number of short crack sections, and each crack section typically has a certain depth and direction. When imaging under certain illumination directions, there will be a shadow over the crack section that displays darker than the background, otherwise there will not be any distinct shadow and the crack section shows a low contrast to the background. In this study, the crack sections that could generate shadows in exposure are labeled as exposed crack sections (e-sections), the others are labeled as hidden crack sections (h-sections).

Consequently, a crack consists of e-sections and h-sections. The e-sections and h-sections are alternative, and all the e-sections constitute a relatively dark linear target — a crack. It is desirable to gather crack seeds from the e-sections and apply FoS algorithm to discover the whole crack. Therefore, we design FoSA as a two-step method, and Fig. 4 shows the flowchart. In the first step, FoSA collects crack seeds automatically, involving the top three operations — aggregating crack sections, creating crack elements and APC-based filtering (i.e., average-path-cost-based filtering), whereas in the second step, FoSA applies FoS (i.e., F* Seed-growing) to get crack strings and conducts post-processing, e.g., linking and pruning.

4.2. Aggregating crack sections

Pavement images are sometimes captured with an uneven illuminance. As such, an illuminance-balancing process is a preliminary step. In order to make our study more focused, all pavement images referred to in this paper have been previously balanced in illuminance. Crack sections, particularly the exposed crack sections i.e., e-sections, will be later used for collecting crack seeds. Note that FoSA employs our prior work NDHM, i.e., neighboring difference histogram method [17], to obtain the e-sections.

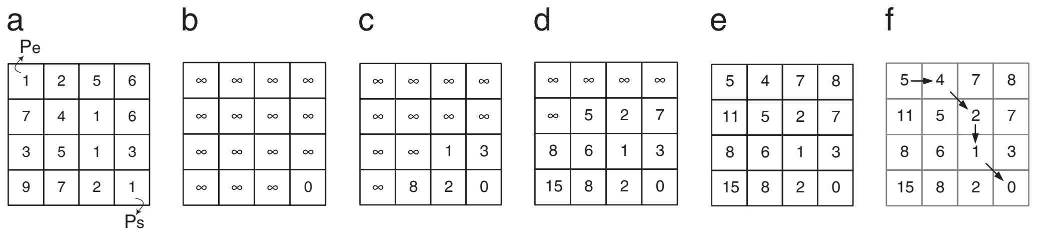

Fig. 2. An example for F* algorithm. (a)A vertex cost matrix, (b)Path cost matrix initialization, (c)The 1st iteration, (d)The 2nd iteration, (e)The 4th iteration, (f)Path tracking result.

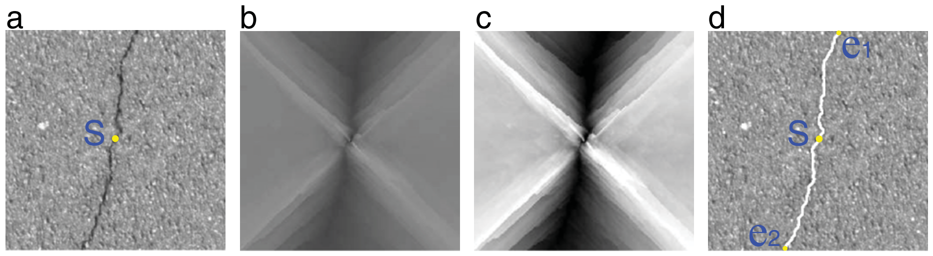

Fig. 3. An example for FoS. (a) The input image with a seed point *S*, (b) Average path cost matrix, (c) A stretched view of (b), (d) The seed-growing path and the two endpoints.

Suppose $p_0(x,y)$ be a pixel in an image, $p_j(x,y)$ be its neighbor, $\psi(x,y)$ be the neighboring difference value of $p_0(x,y)$, then NDHM defines $\psi(x,y)$ by Eq. (4)

$$\psi(x,y) = \sum_{j=1}^{N} \left[p_j(x,y) - p_0(x,y)\right]\delta_j \tag{4}$$

where N denotes the number of neighbors to each pixel, and δ_j is a normalized weight to restrain the impact of speckle noises, as defined by Eq. (5).

$$\delta_j = \frac{p_j(x,y) - p_0(x,y)}{\sum_{j=1}^{N} \left|p_j(x,y) - p_0(x,y)\right|}. \tag{5}$$

NDHM then accumulates the difference value of each gray-level of the image pixels and hence builds a difference histogram, that is, x-axis denotes the gray level, and y-axis denotes the accumulated difference value. The e-sections keep a relatively high contrast with the background, and generally they are inclined to have an overlap in their intensity ranges, therefore, NDHM takes the gray level who holds a max value in the difference histogram as a threshold to get the e-sections.

4.3. Creating crack elements

A crack element is a data object used for storing crack sections, and facilitates the follow-up operations, e.g., crack element filtering, crack string acquisition. This sub-section firstly gives the data structure of the crack element, then introduces the processes to transform a crack section into crack elements.

4.3.1. Crack element data structure

The data structure is modeled as follows, containing the information of start and end points, average path cost and the according path.

```
typedef struct tagCrackElement{
POINT PtStart, PtEnd; // two endpoints of the crack element
INT nApc; // average path cost from the start point to the end point
POINT* pPtPath; // the minimum cost path between the two endpoints
}CE;
```

4.3.2. Crack section processing

A simple way to transform a crack section into a crack element object could be realized by the following two steps — extracting the endpoints of the crack section, and searching the path through each pair of endpoints by using the F* algorithm. However, this approach will fail when a crack section has line junctions, as illustrated in Fig. 5(a). To deal with such case, we divide the crack section by junction points. To be specific, this step will firstly conduct thinning operation to get the skeletons of crack sections, and then remove cross-points (i.e., junction points) on the skeletons, extract endpoints and assign crack element objects.

Note that we use the thinning algorithm proposed in [39]. Emphatically, the thinning should be implemented strictly in the 8-connection case. FoSA aims at getting rid of junction points with three or more branches, which can be achieved by connectivity analysis. Fig. 5 gives an example for transforming a crack section into crack elements. Notably, the crack section processing solves the problem of automatic selection of start and end points for F* tracking in the follow-up operation.

Through crack section processing, each crack section will generate several pairs of endpoints. Each pair of endpoints dedicates to a crack element, and the endpoints are assigned to *PtStart* and *PtEnd*. Meanwhile, each pair of endpoints corresponds to a rectangle shown in Fig. 5(f), through which a sub-image could be sampled from the original image. Then a minimum cost path from *PtStart* to *PtEnd* can be gained through F* path tracking on the sub-image, and the average path cost *nApc* can be calculated, as well as the path coordinates for *pPtPath*. Fig. 5(g) displays the three sub-images corresponding to the crack elements. With the sub-images and endpoints, the F* algorithm tracks out the element paths, as shown in Fig. 5(h).

4.4. APC-based filtering

So far, a large number of crack seed candidates are available in the crack elements. In order to provide crack seeds with high credibility, a filtering operation is needed to remove some false positives. As crack elements could be divided into two categories — the true e-sections and the fake e-sections (the background), a two-class clustering method proposed by [18] is used. The e-sections of the crack are darker than background so that the average path cost is of characteristic significance. Therefore, in the filtering, we select the average path cost value of the crack elements as the class feature.

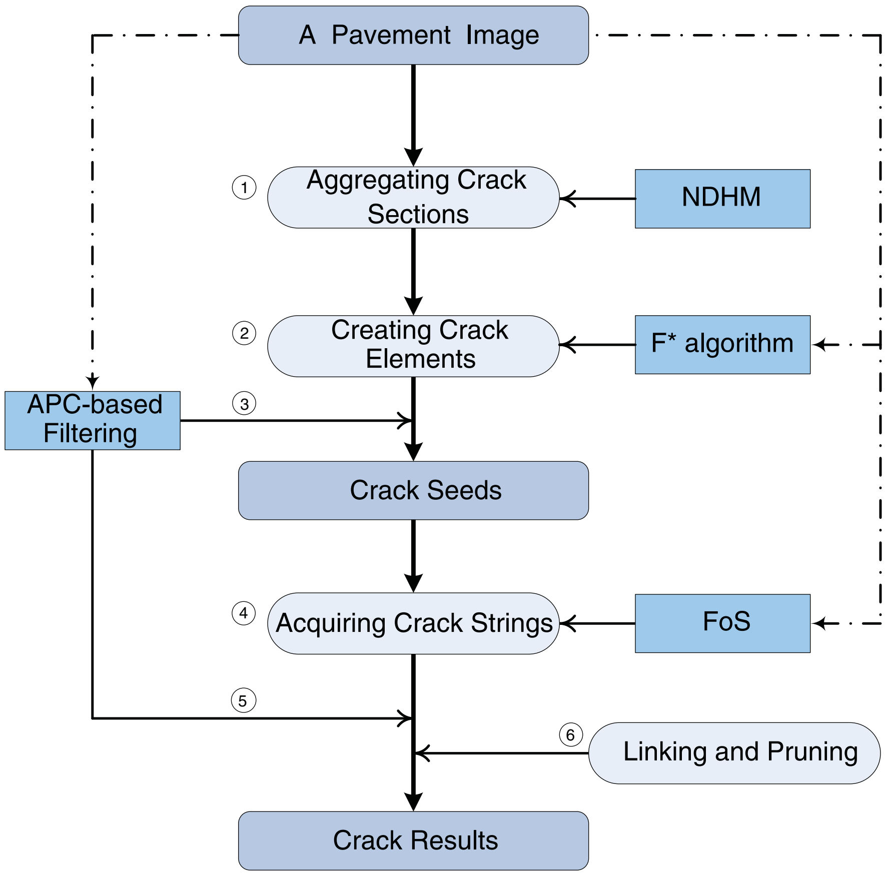

Fig. 4. FoSA flowchart.

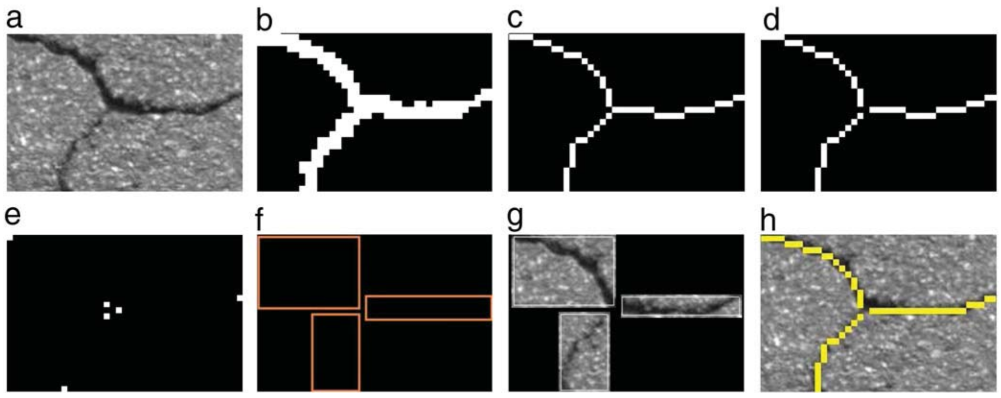

Fig. 5. Crack section processing. (a)A crack image, (b)Crack section corresponding to (a), (c)Thinning result, (d)Cross point removal, (e)Three couples of endpoints, (f)Three crack element rectangles, (g)Three sub-images, (h)Crack element path.

Let CE_i be a crack element, then we can form a crack element set $C=\{CE_i|i=1,\ldots,N\}$, where N denotes the number of the crack elements. Note that each crack element CE_i has an average path cost value, $nApc_i$ $(0\le nApc_i\le 255)$. Suppose the average path cost value of crack elements in C can be represented by L levels $[1,2,\ldots,L]$, and C_1 denotes crack elements with $nApc\in[1,2,\ldots,k]$, C_2 denotes crack elements with $nApc\in[k+1,\ldots,L]$, obviously $C=C_1+C_2$, then we can divide C by searching a k who maximizes the inter-class variance between C_1 and C_2 [18].

As a smaller *nApc* simply denotes a higher probability that the crack element is a true e-section, we remove crack elements in C_2, who hold a *nApc* larger than k. Fig. 6 gives an example for the APC-based filtering.

4.5. Acquiring crack strings

Once crack elements with high credibility are gained, crack seeds can be collected and subsequently FoS (F* seed-growing, which is presented in Section 3) algorithm is applied on the crack seeds to obtain the crack strings.

Suppose C' be the crack element set generated from APC-based filtering, then for each $CE_i\in C'$, we conduct the following steps to gain the crack strings.

1. Crop a square sub-image from the original pavement image, which meets the conditions that CE_i's start point, *PtStart*, is the square center, $2r+1$ is the square width, where r is the seed-growing radius.

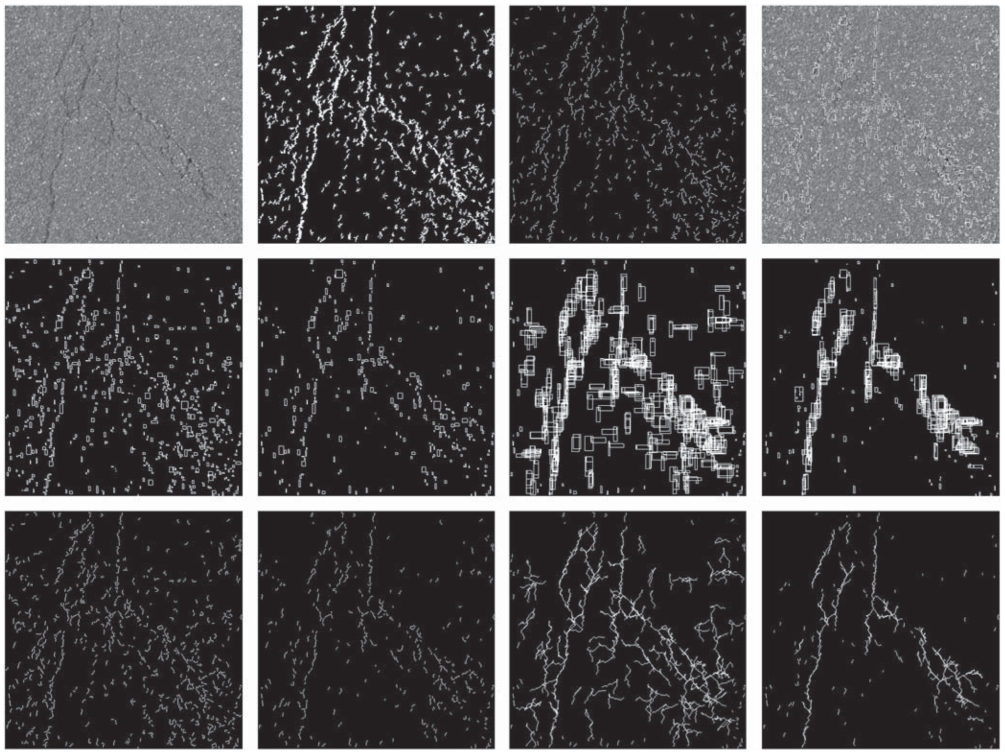

Fig. 6. APC-based filtering. Row 1 gives an original pavement image and the corresponding results of crack section collection, crack element generation, and an overlay display. Rows 2 and 3 display the crack elements with rectangles and paths respectively, and show the procedure of APC-based filtering, in which, column 1: crack elements at the beginning, column 2: APC-based filtering results, column 3: FoS results, and column 4: APC-based filtering results again.

2. Set *PtStart* as the seed point, use FoS to get the seed-growing path, then take the path as one crack string of CE_i.
3. Apply steps i) and ii) on CE_i's end point *PtEnd*, and get another crack string.

Through seed growing, the crack element set is refreshed and the crack element number is three times of that before seed growing. As the crack elements gained from seed growing cannot be guaranteed to be the exact crack strings, the APC-based filtering is applied again to remove the false crack strings. Fig. 7(c) shows the crack seeds gained from image n1, Fig. 7(d) and (g) is the FoS result at $r=24$ and $r=48$, while Fig. 7(e) and (h) is the corresponding APC-based filtering result.

4.6. Linking and pruning

Up to now, FoSA has gained crack elements with high credibility. Given that crack elements may be disconnected or false connected, FoSA puts forward a linking operation to connect adjacent crack elements, and a pruning operation to remove short branches of the linear structures and extract the main crack globally.

In the linking stage, a crack element has two endpoints, and all crack elements constitute the endpoint set E. For each endpoint $P_k \in E$, we calculate its mapping points P_{k1} and P_{k2} through Eq. (6), where $d(P_i, P_k)$ denotes the Euclidian distance between the two points. Then, we get P_c as the link point of P_k by Eq. (7).

$$\{P_{k1}, P_{k2}\} = \{P_i | min2\{d(P_k, P_i) | P_i \in E\}\} \tag{6}$$

where $min2\{\}$ refers to extracting the top two smallest elements.

$$P_c = \begin{cases} P_{kn}, if\ P_{kn} = arg \min\limits_{P_{ki}} \left\{Apc\left(Path_{P_{ki} \to P_e}\right) \middle| i = 1, 2\right\}, \\ \quad \&d(P_{kn}, P_e) < D_{tol} \\ P_k, otherwise. \end{cases} \tag{7}$$

In Eq. (7), P_{kn} is the mapping point of P_k, *Apc()* refers to the average path cost, and D_{tol} is the distance tolerance, empirically set as $r/2$.

In the pruning stage, FoSA prunes cracks by using an algorithm based on minimum spanning tree. It creates a graph to describe the topologies of the spatial crack points, and then calculates the minimum spanning tree of the graph. Finally, it prunes the short branches on this spanning tree, as illustrated in Fig. 7.

5. Experiments and discussions

In order to evaluate our proposed approach, we carried out a series of experiments. In particular, we tried to answer the following questions:

- How effective is the FoS? Is it more effective than the pure-F* algorithm?
- What is the overall performance of FoSA? Does it work better than state-of-the-art edge detection based, wavelet transform based approaches?
- How does the parameter r influence the performance of FoSA?

5.1. Dataset

Our study is a part of the research and development of a landborne road testing and measurement system. The pavement images are collected by a line-scanning camera equipped on a vehicle. All images in the experiment have a ground sample distance (GSD) of 1 mm (i.e., millimeter). They also have a size of 512×512 pixels, and have been preprocessed, such as geometric correction, illuminance balancing and contrast stretching. FoSA randomly selected 5 images (see Fig. 11) from our crack database. Among them, images from n1 to n4 are freeway pavement images, and image n5 is a highway pavement image. Table 1 summarizes the severity rank of the cracks in them. The longitudinal crack in image n2 has comparative higher contrast and better continuity than cracks in the remaining images. Cracks in the other four

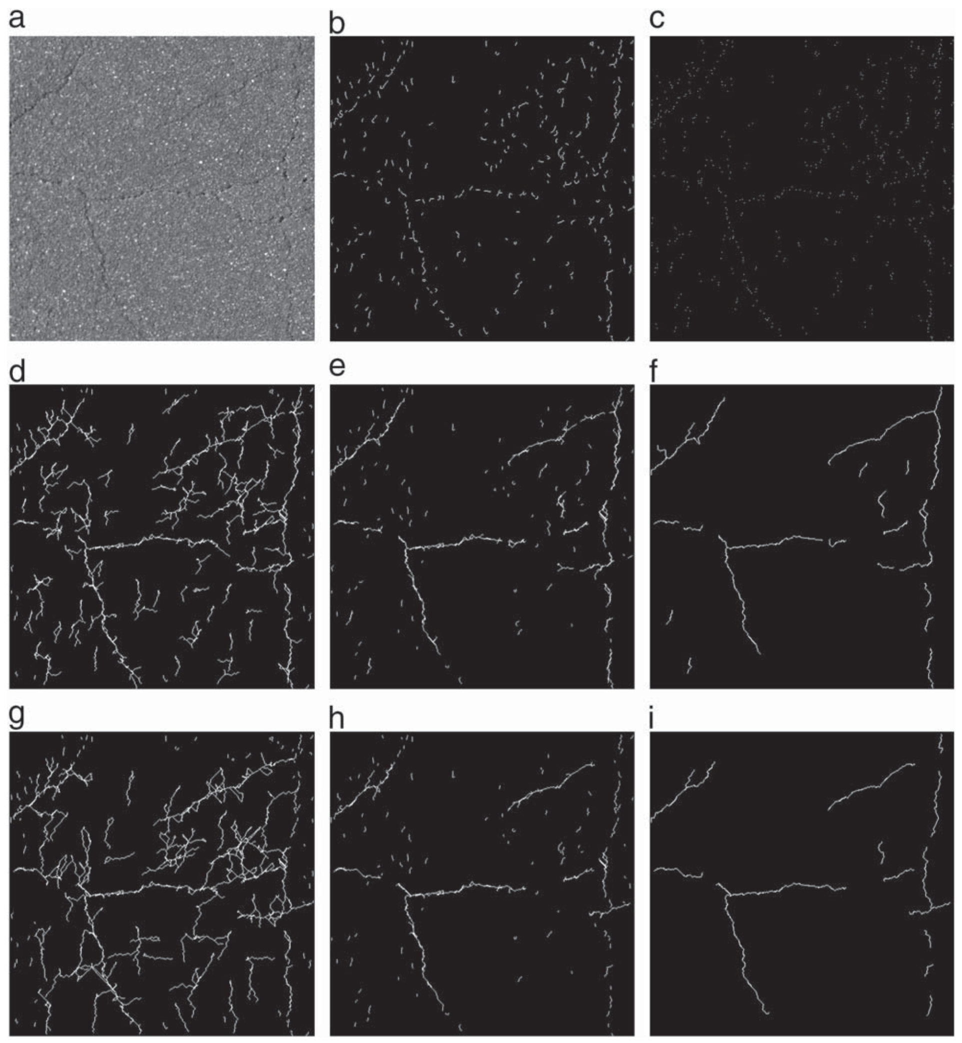

Fig. 7. An example for crack string acquisition and Linking-pruning. (a)Pavement image n1, (b)Crack elements, (c)Crack seeds, (d)FoS result at $r=24$, (e)APC-based filtering result on (d), (f)Linking-pruning result on (e), (g)FoS result at $r=48$, (h)APC-based filtering result on (g), (i)Linking-pruning result on (h).

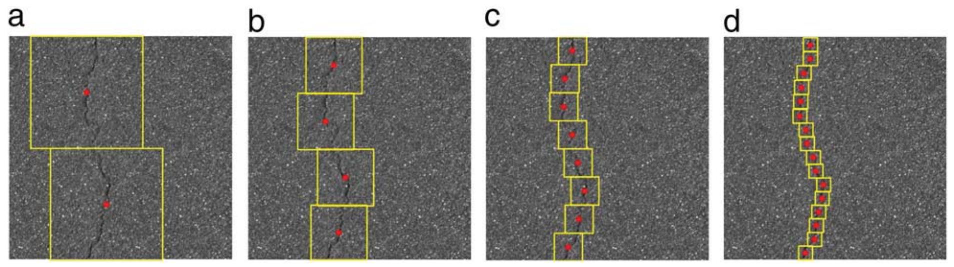

Fig. 8. Seed-growing with different r. (a) $r=128$, (b) $r=64$, (c) $r=32$, (d) $r=16$.

images are much complex due to serious crack degradation. Our algorithm is coded in C++, tested on a computer running with Windows Enterprise 7, CPU 2.5 GHz and RAM 2 GB.

5.2. Efficiency of FoS

An experiment was conducted to compare the efficiency of pure-F* and FoS. Pure-F* means tracking the crack by using F* algorithm purely, where two endpoints are set by human-interface in advance. To facilitate the comparison, the seed-growing radius r for FoS was set as 2^N, that is 128, 64, 32 and 16, and all seeds were set manually (see Fig. 8). As the computing complexity of pure-F* at a 512×512 pixels image is equal to that of FoS at $r=256$ on the same image, we also tested FoS at a radius $r=256$ and used it to denote the performance of pure-F*. At a certain r, the total running time and average running time of FoS can be formulated by Eqs. (8) and (9) respectively, where T_{total} denotes the total running time of FoS, k refers to the number of seeds, T_i is the time consuming of FoS at the i_{th} seed, and T_{ave} denotes the average running time. In Fig. 8, the extraction of the crack was achieved by using FoS at a number of seeds, which were 2 in (a), 4 in (b), 8 in (c), and 16 in (d).

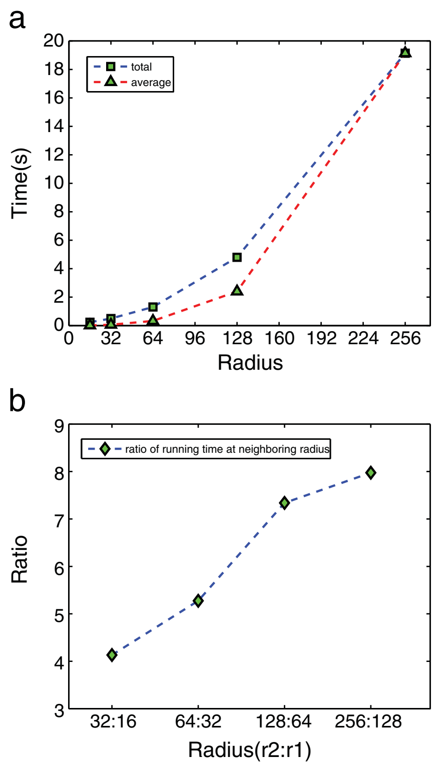

Fig. 9. Running time of FoS. (a)Running time at different *r*, (b)Running time ratio at neighboring *r*.

The running time of FoS at different *r* was recorded and listed in Table 2, and a chart was shown in Fig. 9(a) correspondingly.

$$T^r_{total} = \sum_{i=1}^{k} T^r_i \qquad (8)$$

$$T^r_{ave} = T^r_{total}/k \qquad (9)$$

From Table 2 and Fig. 9(a), both T_{total} and T_{ave} soar with the increase of *r*. The running time at $r=128$ is much less than that at $r=256$, indicating that the FoS is more efficient than the pure-F* when the value of *r* is less than *len*/2, where *len* is the width of the image.

$$Ratio^{r2:r1} = T^{r2}_{ave}/T^{r1}_{ave}. \qquad (10)$$

To evaluate how much parameter *r* affects the efficiency of FoS, the ratio of running time between neighboring radius(r2:r1), $Ratio^{r2:r1}$, is calculated by Eq. (10). Then four $Ratio^{r2:r1}$ values are figured out. The corresponding chart is shown in Fig. 9(b) and demonstrates that the larger the neighboring radiuses are, the larger running time ratio is.

Generally, the complexity of the F* algorithm is $o(n^2)$. However, as a dynamic programming algorithm, its time efficiency is highly related to the convergence speed, namely the iteration times. A parameter κ is introduced to denote the iteration times. Thus, the complexity of the F* algorithm is described as $o(\kappa n^2)$, where *n* can be treated as the width of the image. Taking into account the fact that κ is up approaching to $2n$ in the worst case, we get $o(\kappa n^2) \approx o(n^3)$. On the other hand, we have $o(\kappa n^2) \approx o(n^2)$ in the best case. In Fig. 9(b), $Ratio^{256:128} = 7.975$, is the largest one in all ratio values. Meanwhile, $Ratio^{256:128}$ and $Ratio^{128:64}$ are approaching to 2^3, with an algorithm complexity of $o(n^3)$. Note that $Ratio^{32:16} = 4.133$ and $Ratio^{64:32} = 5.274$ are down approaching to 2^2, with the complexity of $o(n^2)$. This implies that the larger the radius is, the more difficult the algorithm can converge. At a large radius, κ is linear to the image width *n* in the complexity evaluation, while at a small radius, κ can be treated as a constant. By constraining the F* algorithm in a series of interested sub-images with a small radius, FoS gains a much higher efficiency than pure-F*.

5.3. Accuracy

5.3.1. Measures

In order to evaluate the crack detection results, we compute three measures: completeness index *cpt*, correctness index *crt* [40] and F-measure *F* [41] by comparing the detected crack with the human labeled ground truth (GT). *cpt* describes how much the extraction task has been completed, while *crt* describes how much the extraction work done is valid. Considering cracks in our database are not wider than 3 mm (3 pixels), we choose a distance buffer D_{buf}[1] of 3 pixels when computing the measures. If the distance of the detected crack point to the GT is equal or smaller than D_{buf}, that point is considered as a true positive. The *cpt* and *crt* are defined by Eqs. (11) and (12), respectively.

$$cpt = \frac{L_r}{L_{gt}} \qquad (11)$$

$$crt = \frac{L_r}{L_N} \qquad (12)$$

where L_r denotes the length of the extracted results belonging to actual cracks, namely the number of true positive points, L_{gt} is the crack length on ground truth that is obtained by site investigation and manual editing, and L_N represents the total length of the extracted results. The F-measure *F* acts as an overall score, and is defined by Eq. (13).

$$F = 2 \cdot \frac{cpt \cdot crt}{cpt + crt} \qquad (13)$$

5.3.2. Overall performance

We conducted a series of experiments to evaluate FoSA. To better illustrate the performance of FoSA, we compare it with a newly proposed crack detection method Seg-ext [36]. In addition, since crack

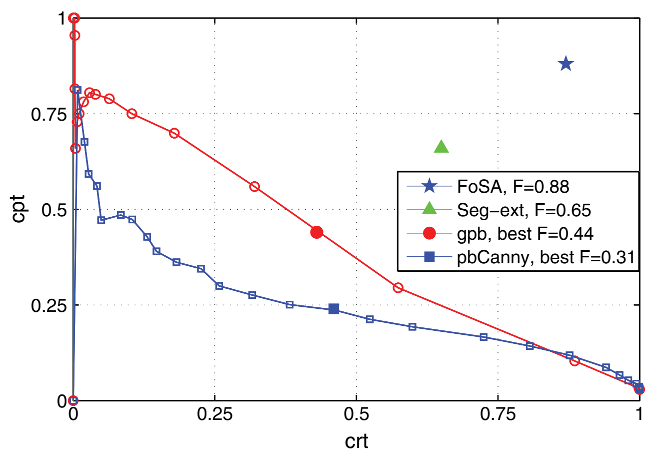

Fig. 10. Performance of FoSA and other three approaches.

[1] The buffer distance is usually empirically selected by considering the ground sample distance (GSD) of the image and the width of the cracks.

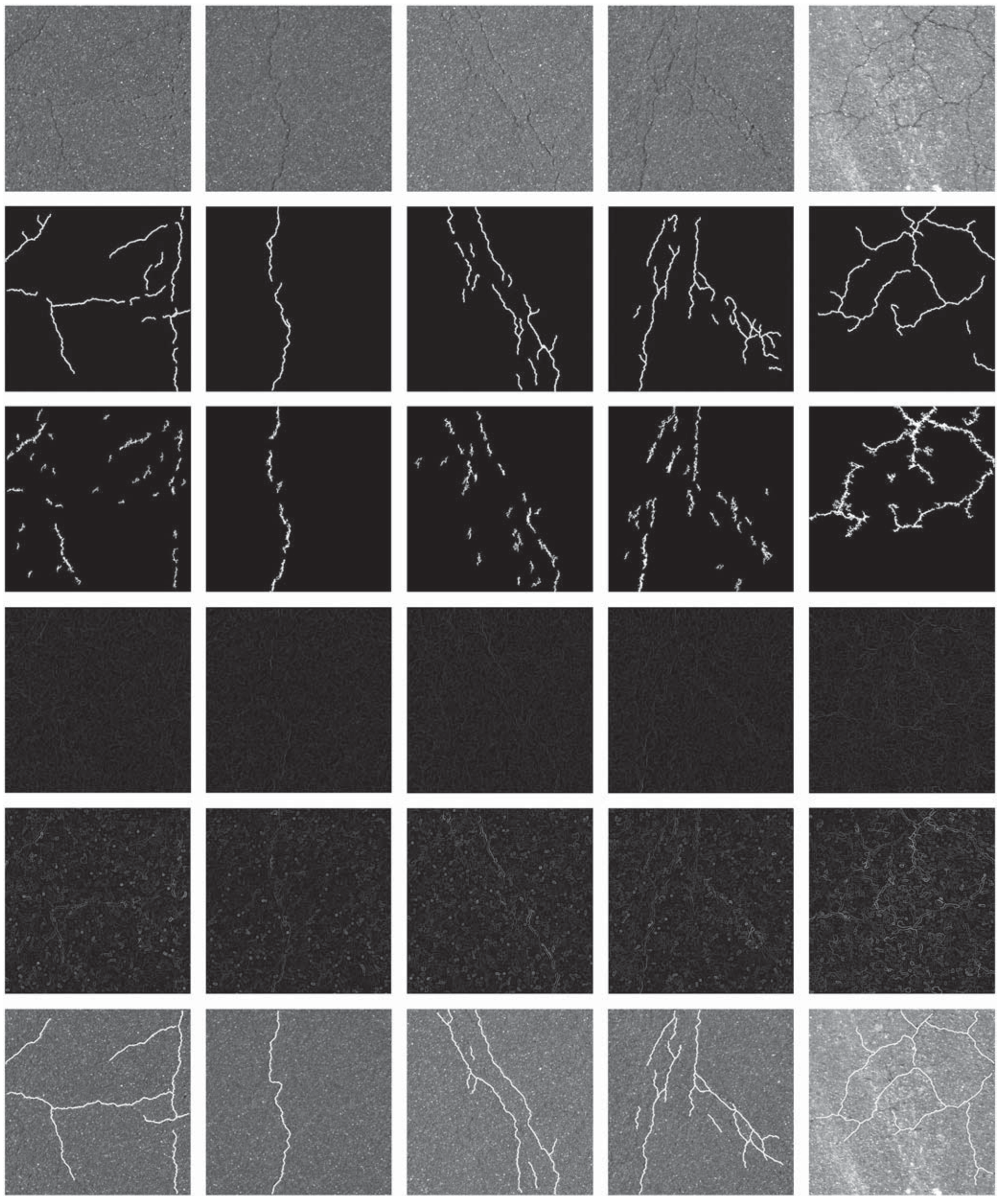

Fig. 11. Comparison of FoSA and three other approaches. Row 1: pavement image n1-n5, row 2: FoSA($r = 32$), row 3: Seg-ext method, row 4: gpb, row 5: pbCanny, row 6: crack ground truth.

line detection can be degenerated into an edge detection problem especially when dealing with low resolution images, we also select two state-of-the-art edge detection methods, namely global pb (gpb) and pbCanny [41] for comparison.

Figure 11 displays the results from the four approaches on images n1–n5. Cracks detected by FoSA have a better consistency with the crack ground truth than Seg-ext, especially on images n1, n3, n4 and n5. Table 3 lists the *cpt* and *crt* values. The results show that the FoSA considerably outperforms Seg-ext. In FoSA, the *cpt* values are generally over 0.84, and the *crt* values are higher than 0.83. While in Seg-ext, except for image n2, it has much lower *cpt* and *crt* values. Noted that a simple longitudinal crack in image n2 is easier to detect, FoSA shows much higher performance than Seg-ext when handling complex cracks with low contrast and bad continuity. Considering gpb and pbCanny only generate soft edges, we hereby compare

Table 1
Crack descriptions in image n1-n5.

Image	Crack type	Low contrast	Bad continuity
n1	Complex	++++	+++++
n2	Longitudinal	++	+
n3	Complex	+++++	+++
n4	Complex	++++	+++
n5	Alligator	+++	++

Note: '+' denotes the severity.

Table 2
Running time of FoS at different *r*.

Radius	$r = 256$	$r = 128$	$r = 64$	$r = 32$	$r = 16$
$T_{total}(s)$	19.141	4.800	1.308	0.496	0.240
$T_{ave}(s)$	19.141	2.400	0.327	0.062	0.015

Table 3
Evaluation and comparison of FoSA method and Seg-ext method.

Measure	Method	Image n1	Image n2	Image n3	Image n4	Image n5
cpt	FoSA	0.860	0.948	0.885	0.852	0.849
	Seg-ext	0.493	0.835	0.560	0.621	0.751
crt	FoSA	0.837	0.924	0.876	0.885	0.913
	Seg-ext	0.601	0.772	0.637	0.690	0.604

their best F-measure with the average F-measure of FoSA and Seg-ext. The comparison results are shown in Fig. 10, from which we can see FoSA holds an F-measure of 0.88, much higher than that of Seg-ext, or even the best F-measure of gpb and pbCanny.

5.3.3. Sensitivity with parameter *r*

Parameter *r* has a significant impact on the performance of FoSA. Thus, we designed several experiments by varying the value of radius *r* from 8 to 48 with an interval of 8. Fig. 13 shows the experiment results on these 5 images. The statistical data is listed in Table 4 and plotted in charts as shown in Fig. 12. We can see from Fig. 12(a), when *r* is between 24 and 48, FoSA has a stable high performance with a *cpt* higher than 0.74. The *cpt* rises quickly with radius *r* increasing from 8 to 32. This is because a larger seed-growing radius would bridge a bigger gap between two crack seeds. However, the *cpt* slowly decreases when radius is over 32, indicating that an excessive growing can be counterproductive. In fact, at a certain resolution, the $Index_{cpt}$ would reach its peak at certain *r*. For example, with GSD = 1 mm in our study, the best *cpt* could be gained at $r = 32$.

Figure 12(b) shows that the *crt* keeps a considerable high value between 0.83 and 0.94 at all radii, which indicates that the FoSA is rather stable in extracting cracks with a low rate of false alarm.

Moreover, Fig. 12(c) gives the running time of FoSA when handling the five images. The running time soars with an increasing radius. From Fig. 12(d) we find that, it takes about 3 s for FoSA at $r = 32$, which is much less than an average running time of about 20 s of pure-F*. This is because FoSA uses a seed-growing strategy which narrows the global searching space to the interested local space and hence greatly improves its efficiency.

6. Conclusion

In this paper, we have proposed FoSA — F* Seed-growing Approach for pavement crack line detection, which is built on top of the crack seed growing strategies. In the seed-aggregating stage, crack seeds with high credibility are gained by taking binary-section information back to the original gray-scale images, and making a full use of the original images. While in the seed-growing stage, crack strings are extracted by an efficient FoS algorithm. The main contributions of this paper are three-fold. First, it is the first to introduce the F* algorithm to solve the problem of crack line detection. Second, through a seed-growing strategy, it solves the problem of automatic selection of the start and end points for F* algorithm. Third, it improves the efficiency by narrowing the global searching space into the interested local space. Experimental results show that the proposed method has strong anti-speckle-noise capability to extract crack lines from pavement images, with high efficiency and reliability.

Currently, FoSA could be further improved. Since FoSA assumes that crack sections are darker than background, it cannot handle “white cracks” at present. But fortunately, in recent research, we have been making progresses in uniformizing the “black crack” and the “white crack” by using a crack-map framework. In addition, we will study to gain more crack information based on the FoSA results, such as the crack width, the crack length, and the crack type, which

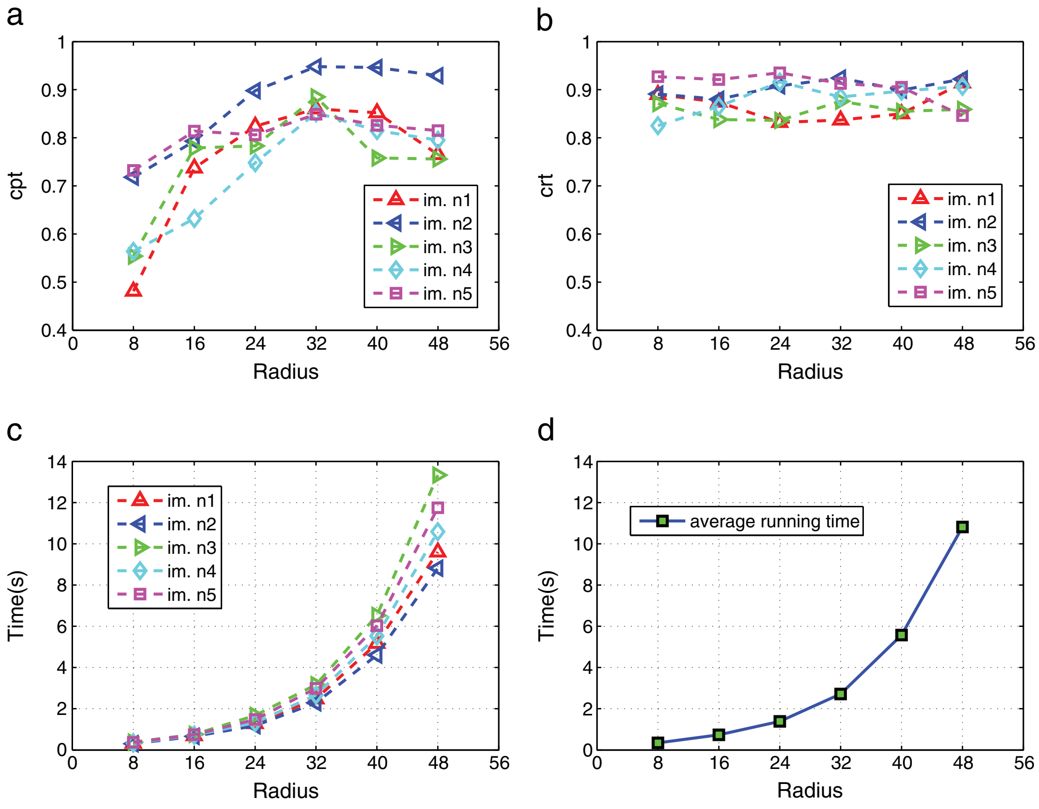

Fig. 12. Impact of *r* on FoSA results. (a)Completeness index at different *r*, (b)Correctness index at different *r*, (c)Running time at different *r* for image n1-n5, (d) Average running time.

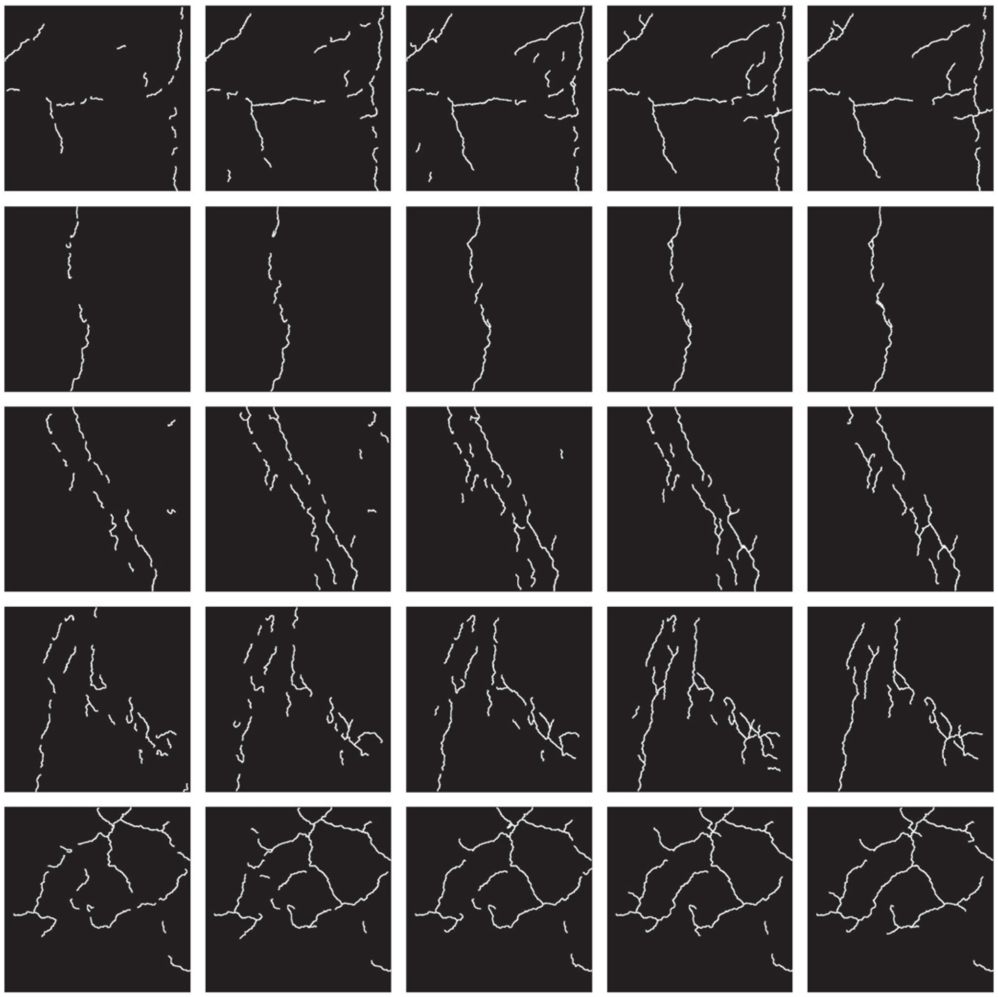

Fig. 13. FoSA results at different *r*. Row 1–5 denote the results of image n1-n5 respectively, column 1: $r=8$, column 2: $r=16$, column 3: $r=24$, column 4: $r=32$, and column 5: $r=40$.

are also very useful in the pavement testing practices. Moreover, we will study how FoSA can further improve its efficiency in a parallel manner.

Acknowledgment

The author Q. Q. Li acknowledges the supports from the National Natural Science Foundation of China on Innovation Team Program under grant no. 40721001, and the Doctoral Foundation Program under grant no. 20070486001. The author Q. Zou thanks the supports from the Fundamental Research Funds for the Central Universities under grant No. 20102130101000130.

Table 4
Evaluation values at different *r*.

Image	Measure	$r=8$	$r=16$	$r=24$	$r=32$	$r=40$	$r=48$
n1	cpt	0.481	0.737	0.824	0.860	0.852	0.765
	crt	0.890	0.874	0.832	0.837	0.850	0.914
n2	cpt	0.718	0.793	0.898	0.948	0.946	0.929
	crt	0.891	0.880	0.908	0.924	0.899	0.922
n3	cpt	0.554	0.779	0.783	0.885	0.758	0.756
	crt	0.871	0.838	0.836	0.876	0.855	0.859
n4	cpt	0.564	0.632	0.748	0.852	0.815	0.795
	crt	0.825	0.865	0.916	0.885	0.897	0.907
n5	cpt	0.732	0.814	0.806	0.849	0.826	0.815
	crt	0.927	0.921	0.935	0.913	0.905	0.846

References

[1] A. Plaza, E. Cernadas, M.L. Durn, P.G. Rodrguez, M.J. Petrn, Multi-scale detection of curvilinear structures with high contour accuracy, Proc. the 5th Iberoamerican Symp. Pattern Recognit., Lisbon, Portugal, 2000, pp. 405–412.

[2] L. Liu, D. Zhang, J. You, Detecting wide lines using isotropic nonlinear filter, IEEE Trans. Image Process. 16 (6) (2007) 1584–1595.

[3] D. Geman, B. Jedynak, An active testing model for tracking roads in satellite images, IEEE Trans. Pattern Anal. Mach. Intell. 18 (1) (1996) 1–14.

[4] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (1986) 679–698.

[5] F. Wang, R. Newkirk, A knowledge-based system for highway network extraction, IEEE Trans. Geosci. Remote. Sens. 26 (5) (1988) 525–531.

[6] C. Barat, C. Ducottet, M. Jourlin, Line pattern segmentation using morphological probing, Proc. of Int. Symp. Image Signal Process., Italy, vol. 1, 2003, pp. 417–422.

[7] W.T. Freeman, E.H. Adelson, The design and use of steerable filters, IEEE Trans. Pattern Anal. Mach. Intell. 13 (9) (1991) 891–905.

[8] R. Samadani, J.F. Vesecky, Finding curvilinear features in speckled images, IEEE Trans. Geosci. Remote. Sens. 28 (4) (1990) 669–673.

[9] M.A. Fischler, J. Tenenbaum, H. Wolf, Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique, Comput. Graph. Image Process. 15 (1981) 201–223.

[10] N. Merlet, J. Zerubia, Classical mechanics and road detection in SPOT images, Joint Res. Rep. No. 1889, INRIA/Hebrew University, 1993.

[11] N. Merlet, J. Zerubia, New prospects in line detection by dynamic programming, IEEE Trans. Pattern Anal. Mach. Intell. 18 (4) (1996) 426–431.

[12] A. Gruen, H.H. Li, Semi-automatic linear feature extraction by dynamic programming and LSB-snakes, Photogramm. Eng. Remote. Sens. 63 (8) (1997) 985–995.

[13] R. Marikhu, M.N. Dailey, S. Makhanov, K. Honda, A family of quadratic snakes for road extraction, Proc. Asian Conf. Comput. Vis. (ACCV'07), 2007, pp. 85–94.

[14] J. Zhou, P.S. Huang, F.P. Chiang, Wavelet-based pavement distress detection and evaluation, Opt. Eng. 45 (2) (2006) 027007.1-027007.10.

[15] K.R. Kirschke, S.A. Velinsky, Histogram-based approach for automated pavement-crack sensing, J. Transp. Eng. 118 (5) (1992) 700–710.

[16] H. Oh, N.W. Garrick, L.E.K. Achenie, Segmentation algorithm using iterated clipping for processing noisy pavement images, Proc. Int. Conf. Imaging Tech.: Tech. and Appl. in Civ. Eng., ASCE, 1997, pp. 138–147.
[17] Q.Q. Li, X.L. Liu, Novel approach to pavement image segmentation based on neighboring difference histogram method, Proc. Int. Cong. Image Signal Process, 2008, pp. 792–796.
[18] N. Otsu, A threshold selection method from gray-level histogram, IEEE Trans. Syst., Man and Cybern. SMC-9 (1) (1979) 62–66.
[19] J.N. Kapur, P.K. Sahoo, A.K.C. Wong, A new method for gray-level picture thresholding using entropy of the histogram, Comput. Vis. Graph. Image Process. 29 (1985) 273–285.
[20] Y. Tsai, V. Kaul, R.M. Mersereau, Critical assessment of pavement distress segmentation methods, J. Transp. Eng. 136 (1) (2010) 11–19.
[21] A. Ayenu-Prah, N. Attoh-Okine, Evaluating pavement cracks with bidimensional empirical mode decomposition, EURASIP J. Adv. Signal Process. (2008) 1–7.
[22] P. Subirats, J. Dumoulin, V. Legeay, D. Barba, Automation of pavement surface crack detection using the continuous wavelet transform, Proc. Int. Conf. Image Process. (ICIP'06), 2006, pp. 3037–3040.
[23] J. Chou, W.A. O'iNeill, H.D. Cheng, Pavement distress evaluation using fuzzy logic and moments invariants, Transp. Res. Rec. 1505 (1995) 39–46.
[24] H.D. Cheng, J.L. Wang, Y.G. Hu, C. Glazier, X.J. Shi, X.W. Chen, Novel approach to pavement cracking detection based on neural network, Transp. Res. Rec. 1764 (2001) 119–127.
[25] G.A. Xu, J.L. Ma, F.F. Liu, X.X. Niu, Automatic recognition of pavement surface crack based on BP neural networks, Proc. Int. Conf. Comput. Electrical Eng. (ICCEE'08), 2008, pp. 19–22.
[26] T.S. Nguyen, M. Avila, S. Begot, Automatic detection and classification of defect on road pavement using anisotropy measure, Proc. European Signal Process. Conf. (EUSIPCO'09), 2009, pp. 617–621.
[27] H. Oliveira, P.L. Correia, Identifying and retrieving distress images from road pavement surveys, Proc. Int. Conf. Image Process. (ICIP'08), 2008, pp. 57–60.
[28] H. Oliveira, P.L. Correia, Supervised strategies for crack detection in images of road pavement flexible surfaces, Proc. European Signal Process. Conf. (EUSIPCO'07), 2008, pp. 25–29.
[29] K.Y. Song, M. Petrou, J. Kittler, Texture crack detection, Mach. Vis. Appl. 8 (1995) 63–76.
[30] M. Petrou, J. Kittler, K.Y. Song, Automatic surface crack detection on textured materials, J. Mater. Process. Tech. 56 (1996) 158–167.
[31] H.D. Cheng, J.R. Chen, C. Glazier, Y.G. Hu, Novel approach to pavement cracking detection based on fuzzy set theory, J. Comput. Civ. Eng. 13 (4) (1999) 270–280.
[32] A. Hassani, H.G. Tehrani, Crack detection and classification in asphalt pavement using image processing, Proc. Int Conf. Cracking in Pavements, RILEM, 2008, pp. 891–896.
[33] M.D. Yan, S.B. Bo, K. Xu, Y.Y. He, Pavement crack detection and analysis for high-grade highway, Proc. Int. Conf. Electron. Measure. Instrum. (ICEMI'07), 2007, 4-548-4-552.
[34] H. Oliveira, P.L. Correia, Automatic road crack segmentation using entropy and image dynamic thresholding, Proc. European Signal Process. Conf. (EUSIPCO'09), 2009, pp. 622–626.
[35] Y.X. Huang, B.G. Xu, Automatic inspection of pavement cracking distress, J. Electron. Imaging 15 (1) (2006) 013017.1-013017.6.
[36] F.F. Liu, G.A. Xu, Y.X. Yang, X.X. Niu, Y.L. Pan, Novel approach to pavement cracking automatic detection based on segment extending, Proc. Int. Symp. Knowl. Acquis. and Model, 2008, pp. 610–614.
[37] L.R. Ford, Network Flow Theory, Rand Tech. Rep. P-923, 1956.
[38] M.J. Carlotto, Enhancement of low-contrast curvilinear feature in imagery, IEEE Trans. Image Process. 16 (1) (2007) 221–228.
[39] C.J. Hilditch, Linear skeletons from square cupboards, Mach. Intell. 4 (1969) 403–420.
[40] C. Wiedemann, H. Ebner, Automatic completion and evaluation of road networks, Int. Arch. Photogramm. Remote Sens. 33 (2000) 976–986.
[41] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, Proc. Int. Conf. Comput. Vis. (ICCV'01), 2001, pp. 416–423.