Automatic Pavement Crack Detection by Multi-Scale Image Fusion

Haifeng Li, Dezhen Song, Yu Liu, and Binbin Li

***Abstract*—Pavement crack detection from images is a challenging problem due to intensity inhomogeneity, topology complexity, low contrast, and noisy texture background. Traditional learning-based approach has difficulty in obtain representative training samples. We propose a new unsupervised multi-scale fusion crack detection (MFCD) algorithm that does not require training data. First, we develop a windowed minimal intensity path based method to extract the candidate cracks in the image at each scale. Second, we find the crack correspondences across different scales. Finally, we develop a crack evaluation model based on a multivariate statistical hypothesis test. Our approach successfully combines strengths from both the large-scale detection (robust but poor in localization) and the small-scale detection (detail-preserving but sensitive to clutter). We analyze and experimentally test the computational complexity of our MFCD algorithm. We have implemented the algorithm and have it extensively tested on two public datasets. Compared with six existing methods, experimental results show that our method outperforms all counterparts. In particular, it increases the Precision, Recall and F1-measure over the state-of-the-art by 22%, 12% and 19%, respectively, on one public dataset.**

I. Introduction

Pavement distresses are common failures of road constructions, where cracks are the major factors and need to be monitored and evaluated reliably and periodically [1]–[3]. Traditional manual crack detection methods are very time-consuming, labor-intensive, with low-accuracy, and error prone. It is necessary to develop an automatic crack detection method for the accurate detection of pavement cracks. However, automatic crack detection is challenging due to intensity inhomogeneity, topology complexity, low contrast, and noisy backgrounds. As a result, local image-processing methods, such as intensity thresholding, and edge detection based methods can only obtain a set of disjoint crack fragments with high false positive rate. Machine learning approaches have been proposed, but the selection of parameters depends on crack variations and image quality. Moreover, the results from these learning methods depend on the quality of manually labeled training data sets which are difficult to obtain, because crack images have large variations due to different lighting conditions, surface types, and background texture.

This work was supported in part by National Science Foundation under IIS-1318638, NRI-1426752, NRI-1526200, and NRI-1748161, by National Science Foundation of China under grant 61305107, by Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) under grant MJUKF201732, and by the Fundamental Research Funds for the Central Universities under grant 3122016B006, and by Chinese Scholarship Council.

H. Li and Y. Liu are with CS Department, Civil Aviation University of China, Tianjin, 300300, China, and also with Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, 350108, China. Email: *hfli@cauc.edu.cn*.

D. Song and B. Li are with CSE Department, Texas A&M University, College Station, TX 77843, USA. Emails: *dzsong@cse.tamu.edu* and *binbinli@tamu.edu*.

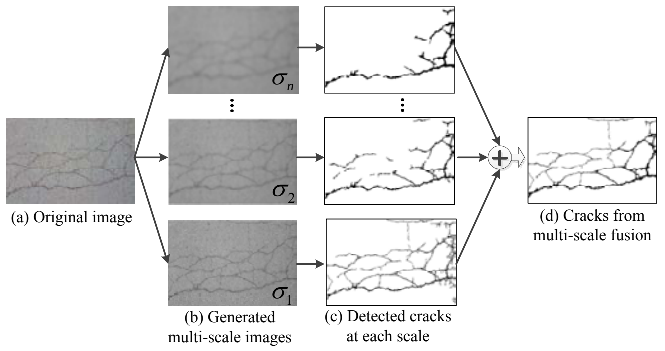

σ_n
σ_2
σ_1
(a) Original image
(b) Generated multi-scale images
(c) Detected cracks at each scale
(d) Cracks from multi-scale fusion

Fig. 1. An illustration of our MFCD algorithm outline. Given an original image, we apply Gaussian blur to generate multi-scale images with different standard deviations $\sigma_1 < \sigma_2 < \ldots < \sigma_n$ representing different scales, then detect cracks at each scale, and finally fuse and filter cracks to obtain results.

In fact, crack images exhibit different characteristics at different scales: at a large scale, crack detection is reliable, but its localization is poor and may miss small details; at a small scale, details are preserved, but detection suffers greatly from clutters in background texture. However, the key challenge is how to effectively combine the strengths of different scales to improve crack detection performance. To deal with the challenges, we propose a new unsupervised multi-scale fusion based crack detection (MFCD) algorithm that does not require labeled training data (See Fig. 1). MFCD computes the maximum average score of cracks at different scales. By extracting crack features for every probable target scale and evaluating the cracks jointly across scales, MFCD fuses and filters cracks at all scales. Another contribution of this paper is our improved minimal intensity path selection method which has two advantages. One advantage is that it only relies on statistical parameters instead of arbitrary thresholds, the other advantage is that the method has no prior training requirement which makes it easy to use in application.

We have implemented our MFCD algorithm and extensively tested it on two public datasets in comparison to six existing methods. Experimental results show that our method consistently outperforms the counterparts. More specifically, our MFCD algorithm increases Precision, Recall and F1-measure over the state-of-the-art by 22%, 12% and 19%, respectively, on one public dataset.

The rest of paper is organized as follows: we summarize the related work in Section II before we introduce our crack detection problem in Section III. We detail our algorithm design in Section IV and analyze algorithm performance in V. We test our algorithm in experiments in Section VI and conclude our paper in Section VII.

II. Related Work

Popular image-based crack detection methods can be classified as four types: intensity thresholding methods, edge detection methods, machine learning techniques, and morphological methods.

Intensity thresholding methods [4], [5] have been widely studied due to their simplicity. These methods are sensitive to noise, leading to unreliable crack detection results especially for field images with significant visual clutters under poor lighting conditions. Furthermore, selecting the appropriate threshold value is challenging.

Edge detection methods [6], [7] are also widely adopted. However, the main drawback is that edge detection methods can only detect a set of disjoint crack fragments and often fail in low-contrast and high-clutter images.

Machine learning techniques [8] have become more popular in recent years which include methods that build on techniques such as support vector machines [9], [10], random forest [11], random structured forest [12], and neural networks [13], [14]. In these learning methods, a pavement image is often divided into a number of sub-images, each of which is represented by a vector of features extracted from this sub-image. These sub-images are then used for training and classification for crack detection. However, since the training and classification are conducted at each sub-image and as local methods, they cannot exactly segment out crack curves over the whole image because it often only provides a label to the sub-image as its output. Furthermore, a supervised training stage is needed which requires accurately labeled data, a difficult requirement for applications with large lighting and scene variations.

Morphological methods [15]–[17] exploits the connectivity among crack pixels and have been successfully used in pavement crack detection research. However, their performance are usually dependent upon the parameter choices [18] which requires manual extensive parameter tuning for each data set. Our method is a variation of morphological methods but focusing on removing arbitrary threshold selection and being self-adaptive to different images because we employ statistic parameters in the threshold setting.

More specifically, our method builds on the existing minimal intensity path based techniques which find the best paths between pairs of endpoints of potential cracks. Gavilan et al. propose a seed-based approach [19] by combining multiple directional non-minimum suppression with a symmetry deck, where seeds are linked by computing paths with the lowest mean pixel intensities that meet the symmetry restrictions. Kaul et al. [20] propose a method to detect the same types of contour-like image structures with less prior knowledge about both the topology and the endpoints of the desired curves. To avoid false detections caused by low intensity loops, Amhaz et al. propose a two-stage minimal intensity path selection algorithm [17] which first selects endpoints at the local scale and then selects minimal intensity paths at the global scale. Built on this approach and to improve the accuracy and computation speed, our new method employs techniques such as crack seeds clustering & removal, windowed path generation, parameter selection, and crack grouping & refinement.

Our MFCD algorithm is also inspired by multi-scale analysis methods which capture the intrinsic geometrical structure that is key in human visual perception. Existing techniques, such as wavelet transforms [21], particle filters [22], beamlet transform [23], are essential different types of multi-scale methods. The main challenge of these multi-scale analysis methods is to select the right scale for identifying useful features or how to combine the detections at different scales to form output. Most approaches take a simplistic cue combination: they either accept results (after thresholding) at all scales, or accept results that appear at the coarsest scale. Our method fuses results from different scales using statistical hypothesis test.

III. Problem Formulation

A. Assumption

We assume a crack has a lower intensity value than that of the background image. This assumption can be accepted in most general cases since cracks always absorb more light than other areas and often appear as dark curves or tapes in the image.

B. Notations

Common notations are defined as follows,

- $I^s, s = 0, 1, \dots, n_s - 1$, the digital image at the s-th scale with I^0 denoting the original image. All images are in grayscale.
- $\mathbf{x}_i^s = [u, v]^{\mathrm{T}}$, the i-th pixel in I^s, with (u, v) being image coordinates.
- C^s, the pixel set for the candidate cracks in I^s.
- C^*, the detected crack pixel set as the algorithm output.

C. Problem Definition

Our ultimate goal is to extract all cracks from an input pavement image by multi-scale crack fusion, as shown in Fig. 1. Thus, our crack detection problem is defined as follows,

Definition 1 (Crack detection): Given I^0, generate multi-scale images $I^s, s = 0, 1, \dots, n_s - 1$, extract the candidate cracks C^s from each I^s, then fuse C^s to obtain C^*.

IV. Multi-Scale Pavement Crack Detection

Cracks in the image tend to have one or more particular salient scales. To combine the strengths of small and large scales, we propose a three-step multi-scale crack detection algorithm as illustrated in Fig. 1. First, we generate the multi-scale images by Gaussian blurring with different kernel sizes. Second, we find the candidate cracks at each scale by utilizing

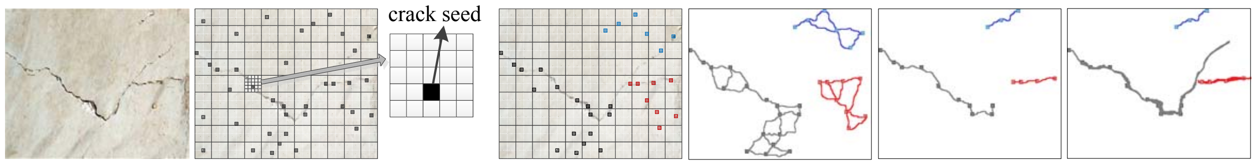


Fig. 2. An illustration of windowed minimal intensity path based crack detection method. (a) Original pavement image. (b) Crack seed extraction results. (c) Crack seed clustering and filtering illustration. (d) WMIP generation result. (e) Path verification results. (f) Crack region detection by path growing.

a windowed minimal intensity path selection based method. Finally, we determine the cracks correspondence across different scales and propose a statistical crack evaluation model to compute the average score of each crack at different scales for crack fusion.

A. Multi-Scale Images Generation

Using Gaussian blurring technique in computer vision [24], the image at the s-th scale level, I^s, can be produced from the convolution of a variable-scale Gaussian, $G(u, v, \sigma_s)$, with an input image I^0.

$$G(u, v, \sigma_s) = \frac{1}{2\pi\sigma_s^2} e^{-(u^2+v^2)/2\sigma_s^2}. \tag{1}$$

Thus, a set of images with different scales are obtained with different σ_s, which serve as the input to the following steps.

B. Candidate Crack Extraction at Each Scale

As shown in Fig. 2, we propose a Windowed Minimal Intensity Path (WMIP) method to find candidate cracks at each scale.

1) Crack seed extraction (Fig. 2(b)): Let $g(\mathbf{x})$ be the intensity value of pixel $\mathbf{x}$, and T_e be a threshold. We divide the whole image I^s into grid cells, denoted as $Y_i^s, i = 1, 2, \ldots, n_y$. Each cell is a set of $m \times m$ pixels. In each grid cell Y_i^s, we select a pixel $\mathbf{x}_i^s$ as crack seed when the following two conditions are satisfied: 1) $\mathbf{x}_i^s$ is the darkest pixel in Y_i^s, and 2) $\mathbf{x}_i^s$ is within the top T_e percent darkest pixels in I^s.

$$\begin{aligned} \mathbf{x}_i^s &= \arg\min_{\mathbf{x}} g(\mathbf{x}), \\ s.t. \quad & \mathbf{x} \in Y_i^s, \quad g(\mathbf{x}) < g_e, \end{aligned} \tag{2}$$

where g_e is the pixel intensity value of the top T_e percent darkest pixels in I^s.

The choice of T_e is important. A bigger T_e leads to more extracted seeds which increase computation load in the following steps. On the other hand, decreasing values of T_e may remove true crack pixels. Our empirical results show that a T_e value of 0.2% reaches a good balance between the computation load and the resulting false negative rate. As the output of the step, let us denote E^s as the set of crack seeds.

2) Crack seed clustering and filtering (see Fig. 2(c)): With E^s obtained, we use DBSCAN algorithm [25] to group crack seeds into clusters, meanwhile, find the isolated crack seeds that need to be removed.

Denote the generated clusters as $G_n, n = 1, 2, \ldots, n_c$. This clustering step can help us reduce the searching region while finding the WMIP between crack seeds in the following step.

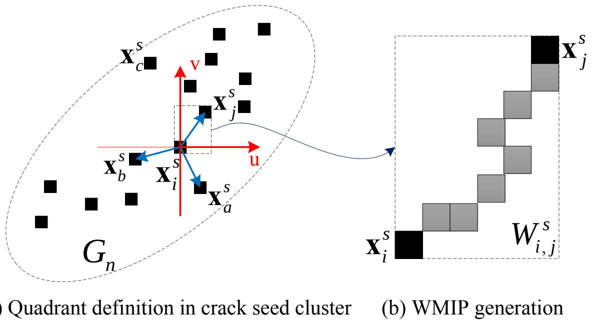


Fig. 3. WMIP generation.

3) WMIP generation (Fig. 2(d)): For each crack seed $\mathbf{x}_i^s$, we define D_i^s be the set of crack seeds which need to be connected with $\mathbf{x}_i^s$. We first find D_i^s and then employ a WMIP based method to connect $\mathbf{x}_i^s$ with each crack seed in D_i^s.

To determine D_i^s, we define the local coordinate system of $\mathbf{x}_i^s$ at first, with $\mathbf{x}_i^s$ as the origin and two axes parallel with the two axes of I^s, respectively, as shown in Fig. 3(a). Denote $q(\mathbf{x}_i^s, \mathbf{x}_j^s)$ as the quadrant number of $\mathbf{x}_j^s$ in $\mathbf{x}_i^s$'s local coordinate system.

$$q(\mathbf{x}_i^s, \mathbf{x}_j^s) = \begin{cases} 1, & u_j - u_i \geq 0 \quad \text{and} \quad \mathrm{v_j - v_i} > 0, \\ 2, & u_j - u_i < 0 \quad \text{and} \quad \mathrm{v_j - v_i} \geq 0, \\ 3, & u_j - u_i \leq 0 \quad \text{and} \quad \mathrm{v_j - v_i} < 0, \\ 4, & u_j - u_i > 0 \quad \text{and} \quad \mathrm{v_j - v_i} \leq 0. \end{cases} \tag{3}$$

where (u_i, v_i) and (u_j, v_j) are the image coordinates of $\mathbf{x}_i^s$ and $\mathbf{x}_j^s$, respectively.

For crack seed $\mathbf{x}_i^s$, we connect it with another crack seed $\mathbf{x}_j^s$ if 1) they are in the same cluster, 2) their geometric distance is smaller than the threshold T_p, and 3) $\mathbf{x}_j^s$ is the nearest crack pixel to $\mathbf{x}_i^s$ in its own quadrant, which means each crack seed can be connected with 4 other crack seeds at most. Summarizing the conditions, for each crack seed $\mathbf{x}_i^s \in G_n$, we can write D_i^s as,

$$\begin{aligned} D_i^s := \Big\{ \mathbf{x}_j^s \Big| & \mathbf{x}_j^s \in G_n, \|\mathbf{x}_i^s - \mathbf{x}_j^s\| < T_p, \|\mathbf{x}_i^s - \mathbf{x}_j^s\| \leq \|\mathbf{x}_i^s - \mathbf{x}_m^s\|, \\ & \forall \mathbf{x}_m^s \in G_n, q(\mathbf{x}_i^s, \mathbf{x}_j^s) = q(\mathbf{x}_i^s, \mathbf{x}_m^s) \Big\}. \end{aligned} \tag{4}$$

An example is shown in Fig. 3(a), where $\mathbf{x}_i^s$ is connected with three crack seeds: $\mathbf{x}_j^s$, $\mathbf{x}_a^s$ and $\mathbf{x}_b^s$. Thus, $D_i^s = \{\mathbf{x}_j^s, \mathbf{x}_a^s, \mathbf{x}_b^s\}$. We do not connect $\mathbf{x}_i^s$ and $\mathbf{x}_c^s$ because their distance is bigger than T_p.

For $\mathbf{x}_i^s$ and each crack seed $\mathbf{x}_j^s \in D_i^s$, we use WMIP to connect them. We consider an image as a bidirectional weighted graph of pixels. Pixels are vertices. If and only if two pixels $\mathbf{x}_a^s$ and $\mathbf{x}_b^s$ are adjacent in the image, we build a bidirectional edge between them, denoted as $(\mathbf{x}_a^s, \mathbf{x}_b^s)$. Recall the pixel intensity value of $\mathbf{x}$ is $g(\mathbf{x})$. The edge weight from vertex $\mathbf{x}_a^s$ to $\mathbf{x}_b^s$ is $g(\mathbf{x}_b^s)$, while edge weight from $\mathbf{x}_b^s$ to $\mathbf{x}_a^s$ is $g(\mathbf{x}_a^s)$. $g(\mathbf{x}_a^s)$ and $g(\mathbf{x}_b^s)$ are often not the same value.

We denote $P_{i,j}^s$ as a path connecting $\mathbf{x}_i^s$ and $\mathbf{x}_j^s$ in the form of a sequence of pixels,

$$P_{i,j}^s := \{\mathbf{x}_i^s, \mathbf{x}_k^s, \mathbf{x}_{k+1}^s, \ldots, \mathbf{x}_{k+n}^s, \mathbf{x}_j^s\}, \tag{5}$$

such that $||\mathbf{x}_i^s - \mathbf{x}_k^s|| = ||\mathbf{x}_{k+n}^s - \mathbf{x}_j^s|| = ||\mathbf{x}_{k+m}^s - \mathbf{x}_{k+m+1}^s|| = 1$, for $m = 0, 1, \ldots, n-1$ because they are adjacent pixels.

We also define $\mathcal{P}_{i,j}^s := \{P_{i,j}^s | P_{i,j}^s \subseteq W_{i,j}^s\}$ as a set of all possible paths in the window set $W_{i,j}^s$ where $W_{i,j}^s$ is the rectangular window with $\mathbf{x}_i^s = [u_i, v_i]^\mathrm{T}$ and $\mathbf{x}_j^s = [u_j, v_j]^\mathrm{T}$ as the two opposite vertices,

$$\begin{aligned} W_{i,j}^s := \Big\{\mathbf{x}_b^s \Big| &\min(u_i, u_j) \le u_b \le \max(u_i, u_j), \\ &\min(v_i, v_j) \le v_b \le \max(v_i, v_j)\Big\}. \end{aligned} \tag{6}$$

For each pair of crack seeds $\mathbf{x}_i^s$ and $\mathbf{x}_j^s \in D_i^s$, the WMIP is the path minimizing the sum of the intensities of pixels along path (see Fig. 3(b)). Let $P_{i,j}^s$ be the WMIP, then

$$P_{i,j}^s = \operatorname*{arg\,min}_{P \in \mathcal{P}_{i,j}^s} \sum_{\mathbf{x}_a^s \in P} g(\mathbf{x}_a^s), \tag{7}$$

where $\mathbf{x}_a^s$ represents a pixel in a candidate path P. We apply Dijkstra algorithm [26] to solve the optimization problem in (7). Finally, we obtain the set of WMIPs, denoted as $\mathcal{P}^s$, for all (i, j) pairs.

4) Path verification (Fig. 2(e)): Not all WMIPs in $\mathcal{P}^s$ represent real cracks. To remove false WMIPs from $\mathcal{P}^s$, a verification step is performed by calculating the mean intensity, $m(P_{i,j}^s)$, of each $P_{i,j}^s \in \mathcal{P}^s$,

$$m(P_{i,j}^s) = \frac{\sum\limits_{\mathbf{x}_a^s \in P_{i,j}^s} g(\mathbf{x}_a^s)}{|P_{i,j}^s|}, \tag{8}$$

where set cardinality operator $|\cdot|$ counts the number of pixels. If the mean intensity of a WMIP is bigger than a given threshold, g_m, we consider the case as false-positive detection and discard the WMIP. Threshold g_m is set as the intensity value of the top T_v percent darkest pixels in I^s.

5) Crack region detection by path growing (Fig. 2(f)): The previous steps may miss some outside pixels due to its focus on connectivity. This step is to absorb neighboring dark pixels to grow paths. The neighboring relationship is not limited to adjacent pixels, two pixels with distance smaller than a threshold are considered as neighbors. Define $L(\mathbf{x}_i^s) := \{\mathbf{x}_j^s | g(\mathbf{x}_j^s) < g(\mathbf{x}_i^s)\}$ be the set of pixels whose intensities are less than $g(\mathbf{x}_i^s)$, κ be the total amount of pixels in I^s, C_i^s be the i-th pixel set for the candidate crack in I^s. Initially, we set $C_i^s = P_{i,j}^s$. Denote $\mathcal{G}_i^s$ as the new found crack pixel set which is generated from C_i^s. $\mathcal{G}_i^s$ can be computed as

$$\mathcal{G}_i^s := \left\{\mathbf{x}_a^s \middle| \|\mathbf{x}_a^s - \mathbf{x}_b^s\| < r, \mathbf{x}_b^s \in C_i^s, \frac{|L(\mathbf{x}_a^s)|}{\kappa} < T_r \right\}, \tag{9}$$

where r is the distance tolerance for path growing, T_r is a threshold. After obtaining $\mathcal{G}_i^s$ from C_i^s, we add $\mathcal{G}_i^s$ into C_i^s.

$$C_i^s = C_i^s \cup \mathcal{G}_i^s. \tag{10}$$

This aggregation process is repeatedly performed, until there are no further pixels to add.

6) Crack region grouping: One real crack may correspond to several crack regions in the image due to noise. We want to group the detected crack regions according to their geometric distances between each other. The distance between two candidate crack pixel sets, C_i^s and C_j^s, can be computed as

$$d_{i,j} = \min \|\mathbf{x}_a^s - \mathbf{x}_b^s\|, \text{s.t. } \mathbf{x}_a^s \in C_i^s, \mathbf{x}_b^s \in C_j^s. \tag{11}$$

For a given threshold T_g, if $d_{i,j} < T_g$ is satisfied, this pair of crack regions, C_i^s and C_j^s, are labeled as the same group. All crack regions may be divided into several groups. For brevity, we use "one crack" to represent a crack group in the rest of this paper.

7) Small crack removal: If a crack size (total pixel number) is smaller than a threshold T_s, the crack is regarded as noises and discarded.

After detecting candidate cracks at each scale, we are ready for multi-scale fusion.

C. Crack Correspondences Matching

Now let us find the crack correspondences across different scales. If two cracks at different scales correspond to the same real crack, they are called as a pair of crack correspondence. The problem of crack matching is defined as follows.

Definition 2 (Crack matching): Given C^{s_i} and C^{s_j}, for each $C_i^{s_i} \subseteq C^{s_i}$ and $C_j^{s_j} \subseteq C^{s_j}$, find $C_m^{s_i} \subseteq C_i^{s_i}$ and $C_n^{s_j} \subseteq C_j^{s_j}$, such that $C_m^{s_i}$ corresponds to $C_n^{s_j}$.

Then, for each $C_i^{s_i} \subseteq C^{s_i}$ and $C_j^{s_j} \subseteq C^{s_j}$, we find the overlapping region with a distance tolerance as the matched segments between $C_i^{s_i}$ and $C_j^{s_j}$. Let us define the distance from the pixel $\mathbf{x}_b^{s_j}$ to crack $C_i^{s_i}$ as $d(\mathbf{x}_b^{s_j}, C_i^{s_i})$,

$$d(\mathbf{x}_b^{s_j}, C_i^{s_i}) = \min \|\mathbf{x}_b^{s_j} - \mathbf{x}_a^{s_i}\|, \text{s.t. } \mathbf{x}_a^{s_i} \in C_i^{s_i}. \tag{12}$$

We want to find the overlapping region of $C_i^{s_i}$ and $C_j^{s_j}$ within a distance tolerance as their correspondence. Initially, we set $C_m^{s_i} = C_n^{s_j} = \varnothing$. Then, for each pixel $\mathbf{x}_b^{s_j} \in C_j^{s_j}$, if the criteria, $d(\mathbf{x}_b^{s_j}, C_i^{s_i}) < T_d$, is satisfied, we add $\mathbf{x}_b^{s_j}$ into $C_n^{s_j}$, where T_d is a threshold. Similarly, for each point $\mathbf{x}_a^{s_i} \in C_i^{s_i}$, if $d(\mathbf{x}_a^{s_i}, C_j^{s_j}) < T_d$ is satisfied, add $\mathbf{x}_a^{s_i}$ into $C_m^{s_i}$. When $C_m^{s_i}$ and $C_n^{s_j}$ are unchanged, we obtain the correspondence $C_m^{s_i}$ and $C_n^{s_j}$.

The steps above are applied for each pair of cracks at different scales until all crack correspondences are found. In the rest of the paper, we renumber all cracks at each scale so that cracks with the same subscript are a group of correspondences.

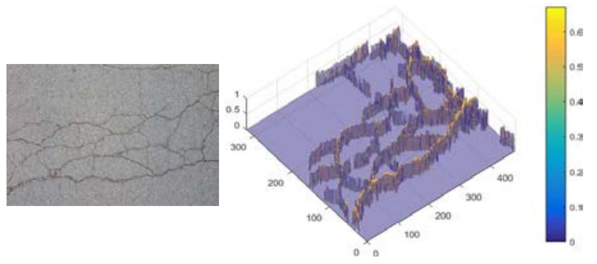

(a) Original image (b) Score

Fig. 4. An example output of score computation. (Best viewed in color).

D. Crack Selection and Verification by Multi-Scale Fusion

The key issue here is how to optimally integrate all candidate cracks C^s to obtain C^*. First, we design the model of average score to evaluate the probability of each detected crack from a true crack. Then, we develop statistical hypothesis test to remove false-positive cracks.

We propose a metric/scoring mechanism to evaluate a pixel's probability to be a crack pixel. A dark pixel or a pixel with many dark neighboring pixels has higher probability to be a crack pixel than otherwise. Thus, the score for each pixel is defined as the aggregation between itself and from all neighboring pixels. We model the probability distribution of each crack pixel to be a weighted Gaussian function. The score of a pixel, $\mathbf{x}_a^s$ which is defined as $s(\mathbf{x}_a^s)$,

$$s(\mathbf{x}_a^s) = \sum_{\mathbf{x}_b^s \in N_e(\mathbf{x}_a^s)} w_b \times \frac{1}{2\pi\sigma_p^2} e^{-\frac{(u_a-u_b)^2+(v_a-v_b)^2}{2\sigma_p^2}}, \quad (13)$$

where σ_p is the standard deviation of the Gaussian distribution, $N_e(\mathbf{x}_a^s)$ is the neighboring window with the size of $(3\sigma_p+1)\times(3\sigma_p+1)$ centering as $\mathbf{x}_a^s$, and w_b is a weight. We adopt the intensity weight in the score computation. A pixel's intensity weight is inversely proportional to its intensity value, i.e., a pixel with a lower intensity is given a greater intensity weight,

$$w_b = 1 - \frac{|L(\mathbf{x}_b^s)|}{\kappa}. \quad (14)$$

Fig. 4 illustrates an example of score distribution where we can see that the crack pixel has a larger score at this scale.

With the score of pixel defined, we can compute the average score of crack C_i^s as follows,

$$\rho(C_i^s) = \frac{\sum_{\mathbf{x}_a^s \in C_i^s} s(\mathbf{x}_a^s)}{|C_i^s|}. \quad (15)$$

With crack's average score defined, we perform statistical hypothesis test to remove false-positive cracks. From (13) and (15), we know that $\rho(C_i^s) \sim N(\mu_s, \sigma_s^2)$ is a random variable following Gaussian distribution with mean being μ_s and variance being σ_s^2. Hence $(\rho(C_i^s)-\mu_s)/\sigma_s \sim N(0,1)$ is a random variable following the normal distribution with zero mean and unit variance. Define $\mathcal{C}_i$ be the union pixel set of the i-th matched cracks across all scales, thus

$$\mathcal{C}_i = \bigcup_{s=0}^{n_s-1} C_i^s. \quad (16)$$

We define function $f(\mathcal{C}_i)$ to evaluate the probability of $\mathcal{C}_i$ to be a real crack or not.

$$f(\mathcal{C}_i) = \sum_{s=0}^{n_s-1} (\frac{\rho(C_i^s)-\mu_s}{\sigma_s})^2. \quad (17)$$

For each candidate crack $\mathcal{C}_i$, we set up two hypotheses:

$$H_0 : \mathcal{C}_i \text{ is a crack.}$$
$$H_1 : \mathcal{C}_i \text{ is not a crack.}$$

Since $f(\mathcal{C}_i)$ is the sum of squares of multiple normal distributions according to (17), $f(\mathcal{C}_i)$ follows χ^2 distribution with n_s degrees of freedom. Define $F(x, n_s)$ be the cumulative distribution function of χ^2 distribution with n_s degrees of freedom, the probability of a value from χ^2 distribution larger than x is

$$P\{f(\mathcal{C}_i) \geq x\} = 1 - F(x, n_s). \quad (18)$$

By setting the significance level as α, we can obtain

$$P\{f(\mathcal{C}_i) \geq x\} = \alpha. \quad (19)$$

Since the function F is continuous and strictly monotonically increasing, combining (18) and (19), we can obtain

$$x = F^{-1}(1-\alpha, n_s), \quad (20)$$

where $F^{-1}(\cdot)$ is the inverse function of $F(\cdot)$. Thus, we reject H_0 if

$$f(\mathcal{C}_i) \leq F^{-1}(1-\alpha, n_s). \quad (21)$$

After the statistical hypothesis test, we remove the false-positive cracks. Combine all the cracks remained to obtain C^*.

V. Algorithm Analysis

We summarize the proposed WMIP based candidate crack detection method in Algorithm 1 to facilitate our analysis.

Recall κ is the total amount of pixels in I^s. For crack seed extraction, since at most κ/m^2 cells are obtained, with each cell containing m^2 pixels, the crack seeds can be detected in $O(\kappa/m^2 \times m^2) = O(\kappa)$ time. Crack seed clustering takes $O(|E^s|\log|E^s|)$ [25]. Obviously, $|E^s| \leq \kappa/m^2 \leq \kappa$, thus, crack seed cluster has a time complexity $O(\kappa\log\kappa)$. Since we only find the WMIP within a rectangular window $W_{i,j}^s$ whose diagonal length is smaller than T_p, the maximum size of $W_{i,j}^s$ is $\frac{1}{2}T_p^2$, thus, when proceeding the Dijkstra algorithm, the number of vertex is smaller than $\frac{1}{2}T_p^2$, and the number of edges is no more than $4T_p^2$. So the time complexity of each WMIP generation is $O(T_p^2\log T_p)$. There are at most κ^2/m^4 paths in all because $|E^s| \leq \kappa/m^2$. Thus, the time complexity for all WMIPs generation is $\frac{T_p^2\log T_p}{m^4}\kappa^2$. Since the length of $P_{i,j}^s$ is no more than the size of $W_{i,j}^s$, we have $|P_{i,j}^s| \leq \frac{1}{2}T_p^2$. Therefore, we can compute $m(P_{i,j}^s)$ in $O(T_p^2)$

time. Crack region detection by path growing takes $O(\kappa \log \kappa)$ time. Computing $d_{i,j}$ between two sets with size $|C_i^s|$ and $|C_j^s|$ needs $O(|C_i^s||C_j^s|)$ time. Usually, the total number of crack regions is very small and can be considered as constant. Thus, the time complexity of crack grouping is also $O(|C_i^s||C_j^s|)$. To summarize, since $|C_i^s||C_j^s| < \kappa^2$, the most computationally expensive step in Algorithm 1 is WMIP generation using Dijkstra algorithm. Thus, the computational complexity of our WMIP based crack detection algorithm is $O(\frac{T_p^2 \log T_p}{m^4}\kappa^2)$.

Theorem 1: The computational complexity of the proposed WMIP based crack detection algorithm is $O(\frac{T_p^2 \log T_p}{m^4}\kappa^2)$.

Algorithm 1: WMIP based Crack Detection

```
input : $I^s$
output: $C^s$
1  Detect all crack seeds from $I^s$ to generate $E^s$;          $O(\kappa)$
2  Cluster and filter $E^s$ using DBSCAN;                        $O(\kappa \log \kappa)$
3  foreach $\mathbf{x}_i^s \in E^s$ and $\mathbf{x}_j^s \in E^s$ do      $O(\frac{T_p^2 \log T_p}{m^4}\kappa^2)$
4      if $\mathbf{x}_j^s \in D_i^s$ then
5          Compute $P_{i,j}^s$ by solving (7) using Dijkstra
             algorithm;                                          $O(T_p^2 \log T_p)$
6      Compute mean intensity $m(P_{i,j}^s)$ using (8);          $O(T_p^2)$
7      if $m(P_{i,j}^s) < g_m$ then
8          Discard $P_{i,j}^s$;                                  $O(1)$
9  foreach $P_{i,j}^s \in \mathcal{P}^s$ do                      $O(\kappa \log \kappa)$
10     repeat                                                    $O(\kappa \log \kappa)$
11         Compute $\mathcal{G}_i^s$ using (9);                  $O(n_i)$
12         $C_i^s = C_i^s \cup \mathcal{G}_i^s$;                 $O(1)$
13     until $C_i^s$ is unchanged;
14 foreach $C_i^s$ and $C_j^s$ do                                $O(|C_i^s||C_j^s|)$
15     Compute $d_{i,j}$ using (11);                             $O(|C_i^s||C_j^s|)$
16     if $d_{i,j} < T_g$ then
17         Merge $C_i^s$ and $C_j^s$ into the same group;
             $O(|C_i^s| + |C_j^s|)$
18 foreach $C_i^s \in C^s$ do                                    $O(|C_i^s|)$
19     if $|C_i^s| < T_s$ then
20         Remove $C_i^s$ from $C^s$;                            $O(|C_i^s|)$
21 return $C^s$;                                                 $O(1)$
```

We present our MFCD algorithm in Algorithm 2, and facilitate the computational complexity analysis as follows. Matching two cracks $C_i^{s_i}$ and $C_j^{s_j}$ takes in $O(|C_i^{s_i}||C_j^{s_j}|)$ time. Generally, the total number of cracks in I^s is small and can be considered as constant. Thus, crack matching across two scales has a time complexity $O(|C_i^{s_i}||C_j^{s_j}|)$. Computing all crack correspondences across each pair of adjacent scales takes $O(n_s|C_i^{s_i}||C_j^{s_j}|)$ time. When computing the average score of any crack C_i^s, the scores of $|C_i^s|$ pixels in C_i^s need to be calculated firstly, as shown in Eq. (15). The neighboring window for each pixel is $(3\sigma_p+1)\times(3\sigma_p+1)$. Thus, the overall computation of average score of C_i^s takes $O((3\sigma_p+1)^2|C_i^s|)$ time. Considering σ_p is a constant, computing $\rho(C_i^s)$ has a time complexity $O(|C_i^s|)$. Since the total number of crack pixels in each I^s is smaller than κ, merging cracks across n_s scales takes in $O(n_s\kappa)$ time. Define the total number of $\mathcal{C}_i$ be n. Considering $|\mathcal{C}_i| \leq \kappa$, the time complexity of hypothesis test for all merged cracks is $O(n\kappa)$. Thus, the crack selection and verification for each two scales takes in $O(\max(n_i^{s_i}, n_j^{s_j}))$ time. Since both $|C_i^s|$ and n are much smaller than κ, the computational complexity of our MFCD algorithm is $O(n_s\frac{T_p^2 \log T_p}{m^4}\kappa^2)$.

Theorem 2: Our MFCD algorithm runs in $O(n_s\frac{T_p^2 \log T_p}{m^4}\kappa^2)$ time.

Algorithm 2: MFCD Algorithm

```
input : $I^s, s = 0, 1, \ldots, n_s - 1$
output: $C^*$
1  foreach $I^s$ do                                              $O(n_s\frac{T_p^2 \log T_p}{m^4}\kappa^2)$
2      Extract $C^s$ using WMIP algorithm;                       $O(\frac{T_p^2 \log T_p}{m^4}\kappa^2)$
3  foreach $C^{s_i}$ and $C^{s_j}$ do                            $O(n_s|C_i^{s_i}||C_j^{s_j}|)$
4      foreach $C_i^{s_i} \subseteq C^{s_i}$ and $C_j^{s_j} \subseteq C^{s_j}$ do   $O(|C_i^{s_i}||C_j^{s_j}|)$
5          Match $C_i^{s_i}$ and $C_j^{s_j}$;                    $O(|C_i^{s_i}||C_j^{s_j}|)$
6  foreach $C^s$ do                                              $O(n_s|C_i^s|)$
7      foreach $C_i^s \subseteq C^s$ do                          $O(|C_i^s|)$
8          Compute $\rho(C_i^s)$ using (15);                     $O(|C_i^s|)$
9  Merge matched cracks using (16);                              $O(n_s\kappa)$
10 foreach $\mathcal{C}_i$ do                                    $O(n\kappa)$
11     Compute $f(\mathcal{C}_i)$ using (17);                    $O(n_s)$
12     if $f(\mathcal{C}_i) \geq F^{-1}(1-\alpha, n_s)$ then
13         Add $\mathcal{C}_i$ into $C^*$;                       $O(|\mathcal{C}_i|)$
14 return $C^*$;                                                 $O(1)$
```

VI. Experiments

We have implemented our algorithm using MATLAB under a PC with an operating system of Windows 8, which has an Intel(R) Core$^{\text{TM}}$2 i5-4200U CPU@1.60GHz and 4 GB memory. We evaluate the performance of our MFCD method on two public datasets and compare it with six state-of-the-art algorithms.

A. Parameter Settings

It is important to determine the values of σ_s in multi-scale image generation in Section IV-A. We have to find a trade-off between efficiency and completeness. Define $\sigma_s = k\sigma, k \in \mathbb{N}$. First, we fix $\sigma = 1$, and can adjust the number of scales from 1 to 5. By setting $k = 0, 1, \ldots, n$, we obtain $n+1$ images with different scales. We select 10 representative images for the experiments. We find that when the number of scales exceeds 3, the detected cracks are almost unchanged. However, the cost of computation increases with this number. Therefore, we use 3 scale images. Then, by fixing $k = 0, 1, 2$, we find $\sigma = 0.8$

TABLE I
PARAMETER SETTING

Parameter	Value	Description
σ_s	0.8,1.6	standard deviation of Gaussian kernel
m	8	size of grid cell
T_e	0.2%	threshold for crack seed extraction
T_p	32	threshold for path generation
T_v	1%	threshold for path verification
T_r	2%	threshold for path growing
r	4	distance tolerance for path growing
T_g	64	threshold for crack region grouping
T_s	64	threshold for small crack remove
T_d	8	distance tolerance for crack matching
σ_p	1	standard deviation in pixel score computation
α	0.05	significance level

experimentally to be the best choice. Thus, the 3 scale images are the original image, $\sigma_s = 0.8$, and $\sigma_s = 1.6$.

Referring to [27], the size of grid cell, m, is set as 8. Other parameters are set according to the experiments empirically as shown in Tab. I.

B. Datasets, Counterparts, and Metrics

Two public datasets are tested in our experiments.

- AigleRN dataset [17]. It contains 38 French pavement images with ground truth. The resolution is 991×462 pixels. They have been pre-processed to mitigate the influence of non-uniform lighting conditions.
- CFD dataset [12]. It is composed of 118 images with a resolution of 480×320 pixels, which can generally represent urban road surface conditions in Beijing, China. Each image has hand labeled ground truth. The images contain noises such as shadows, oil spots, and water stains.

The six existing methods which we compare our algorithm to are:

- Canny [28]. Canny is a traditional edge detection method.
- MS [18]. Markov Segmentation (MS) is a crack detection method based on a multi-scale extraction and a Markov segmentation.
- GC [29]. GC is a geodesic contour method with automatic selection of points of interest based on auto-correlation.
- FFA [30]. Free Form Anisotropy (FFA) is based on the estimation of minimal intensity paths at each pixel in four directions, and the pixel is recognized as a crack if the path cost greatly varies with the direction.
- MPS [17]. Minimal Path Selection (MPS) is a crack detection algorithm based on the original minimal intensity path selection.
- CrackForest [12]. CrackForest is a road crack detection framework based on random structured forests, by learning the inherent structured information of cracks.

To evaluate the performance of different crack detection algorithms quantitatively, three metrics, including Precision, Recall and F1-measure, are employed. These three metrics can be computed based on true positive (TP), false negative (FN), and false positive (FP),

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{22}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{23}$$

$$\text{F1-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

Since acquiring a high quality ground truth is difficult for real images, we allow a tolerance margin in measuring the coincidence between the detected cracks and the ground truth. As comparisons, according to the experiment settings in [17] and [12], we assume that TP pixels are included within a 2 and 5 pixel vicinity of the ground truth on AigleRN and CFD, respectively.

C. Computation Speed Test

To validate the computation complexity analysis in theorem 2, we perform the speed test on our MFCD algorithm. According to theorem 2, the computation complexity of MFCD algorithm is related to four parameters: n_s, κ, m and T_p. We test the speed of MFCD algorithm with different settings of the four parameters. Each time, We fix three parameters of them and change the remaining one to see the trend of run time. The results are shown in Fig. 5, where for each setting, 10 pavement images from CFD dataset are carried out for averaged performance. From Fig. 5 we can see that these timing results are consistent with the theoretical analysis in general.

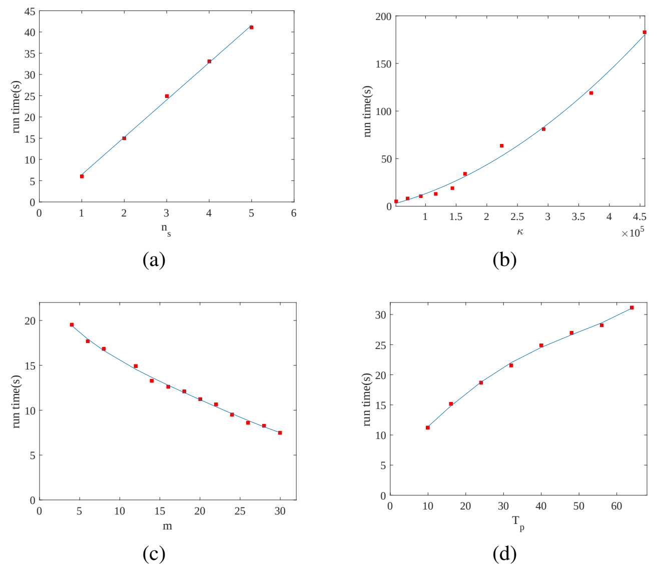


Fig. 5. Computation speed test results of our MFCD algorithm. Each data point in this figure is an average of results from ten pavement images.

D. Results on AigleRN Dataset

Fig. 6 illustrates two typical sample detections on AigleRN. The final detection results (row 5) are obtained by fusing the

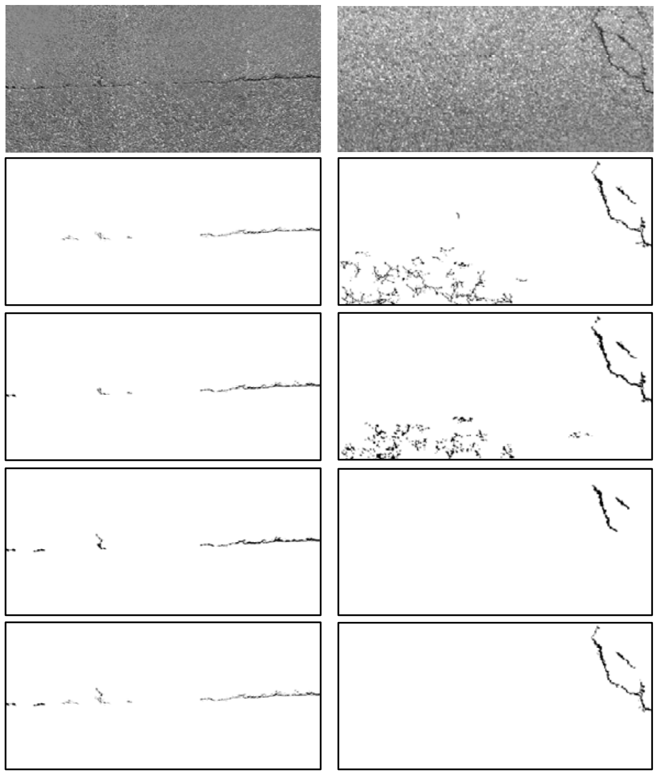


Fig. 6. Representative sample results of crack detection using our proposed method. The first row lists the original images, rows 2-4 are detection results from scale 1-3, the last row is the final detection results.

results from different scales (row 2-4). It is clear that multi-scale fusion improves the overall performance.

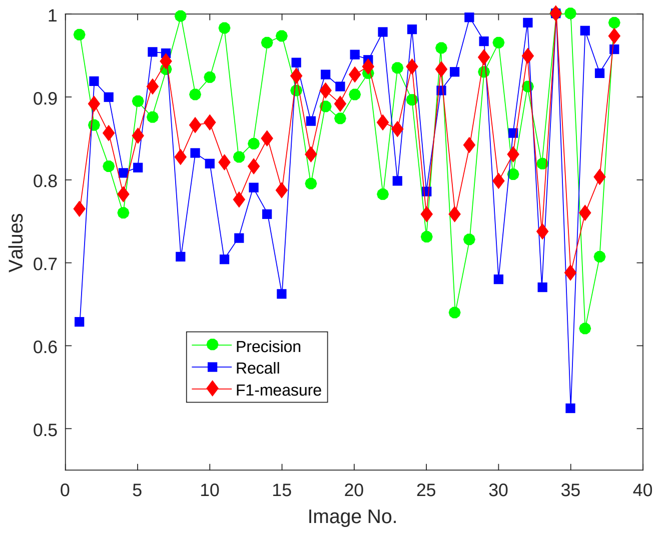


Fig. 7. Crack detection results of our proposed MFCD algorithm on AigleRN dataset.

The crack detection result of our algorithm is shown in Fig. 7. The summary statistics for comparisons are presented in Fig. 8, and representative sample results are shown in Fig. 9. It is clear that our MFCD method outperforms the counterparts.

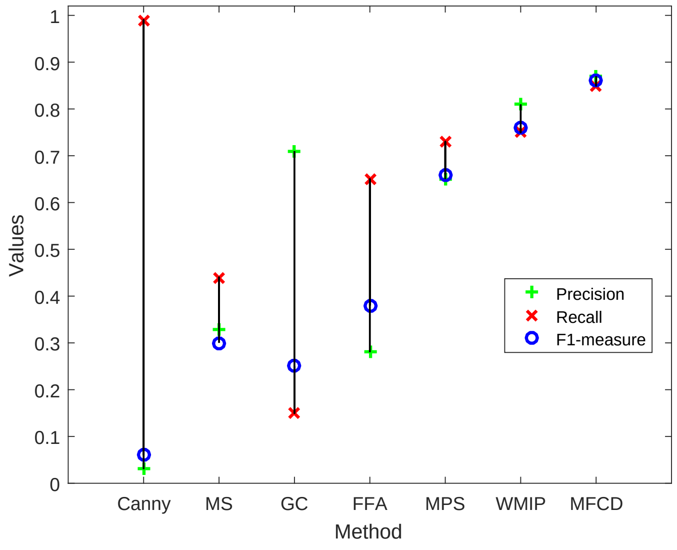


Fig. 8. Averaged values of Precision, Recall, and F1-measure for all the images in AigleRN. The values of MS, GC, FFA, MPS are from reference [17]. Here, WMIP indicates WMIP based crack detection method on I^0.

TABLE II
CRACK DETECTION RESULTS EVALUATION ON CFD DATASET

Method	Precision	Recall	F1-measure
Canny	12.23%	22.15%	15.76%
FFA	78.56%	68.43%	73.15%
CrackForest	82.28%	89.44%	85.71%
MFCD	**89.90%**	**89.47%**	**88.04%**

Traditional edge detection method Canny is not suitable for pavement crack detection due to its overly high sensitivity. MS is too sensitive to the background texture. GC can only find a small part of cracks. FFA outputs the results as a thicker line, implying high FP rate. As for MPS, for the image with serious noise, the values of FP and FN become an issue.

Furthermore, to validate whether the multi-scale fusion can improve crack detection, we compare the single scale crack detection result on I^0 with MFCD method. It is clear that the multi-scale fusion indeed improves the performance of single-scale detections.

E. Results on CFD Dataset

Our algorithm also succeeds on CFD dataset. Fig. 10 presents some representative images and their detection results using our MFCD algorithm, where we can see that the images in CFD are quite noisy. The performance of our MFCD algorithm degrades when the contrast between cracks and backgrounds is low (as shown in the 2nd row in Fig. 10), or there are dark regions, e.g. oil spots, in the pavement images (as shown in the 5th and 9th rows in Fig. 10). The crack detection results on the CFD dataset are summarized in Tab. II where the results of Canny, FFA and CrackForest are from [12]. Again, our algorithm outperforms all counterparts.

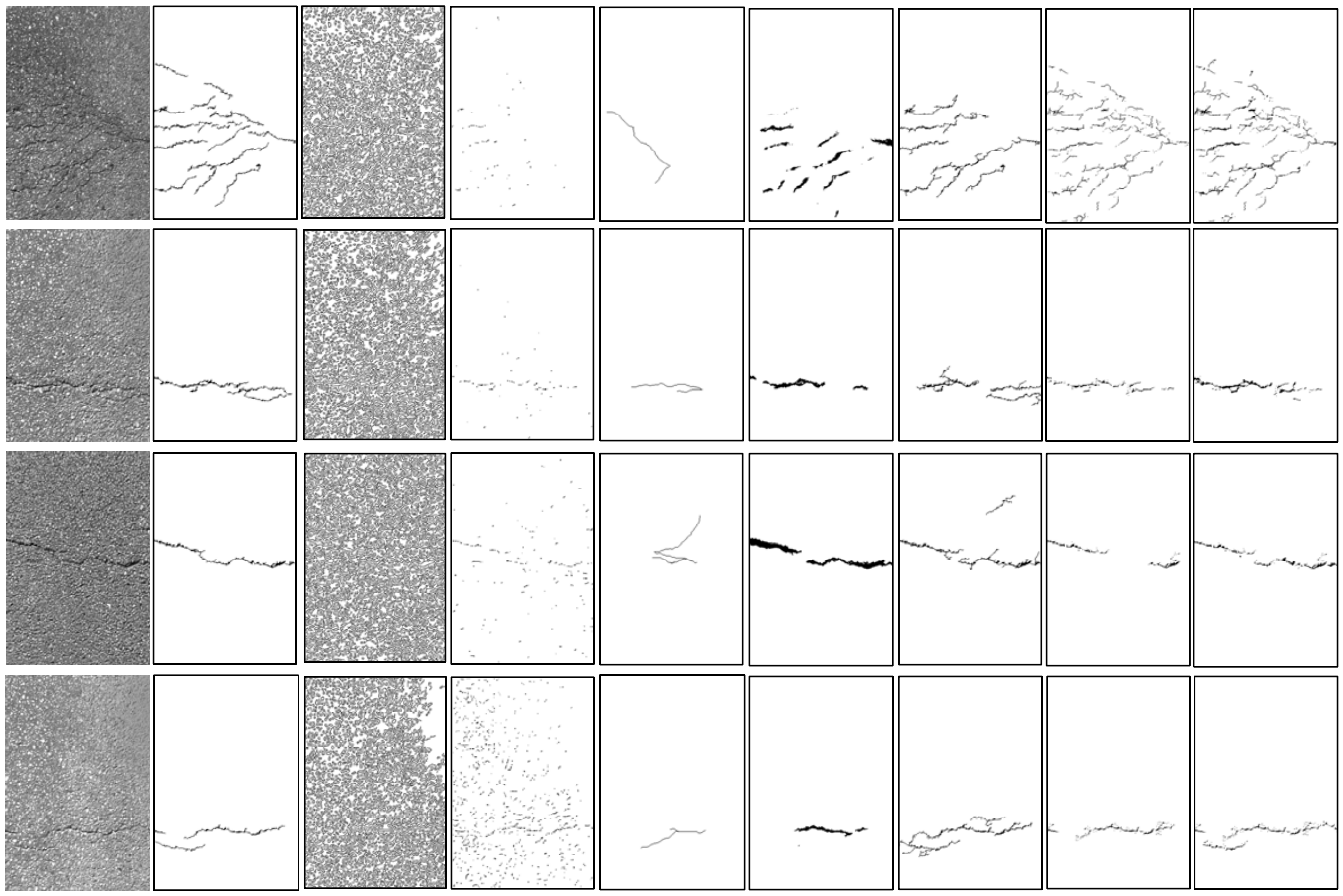


Fig. 9. Example results of different algorithms on AiGleRN (from left to right: original image, ground truth, Canny, MS, GC, FFA, MPS, WMIP based method on I^0, and MFCD).

The whole set of original images and detection results are available on the following web page: http://telerobot.cs.tamu.edu/bridge/Datasets.html.

VII. Conclusion and Future Work

We developed a new MFCD method for detecting the cracks from pavement images. First, we proposed a WMIP based method to detect the cracks at each single scale. Second, we matched the corresponding cracks across different scales. Finally, a crack evaluation model was built, and the crack was selected as the detected crack if it passed statistical hypothesis test. Our MFCD algorithm takes in $O(n_s \frac{T_p^2 \log T_p}{m^4} \kappa^2)$ time. We have implemented the proposed algorithm and experimental results are consistent with our complexity analysis. We tested MFCD algorithm on two public datasets. Experimental results showed that 1) The proposed MFCD algorithm improves the crack detection performance compared with the single scale WMIP based method. 2) Our MFCD algorithm outperforms six existing methods.

In the future, we will consider fusing the camera inputs with ground penetration radar and laser ranger finder inputs. We will study how to perform crack detection using multi-modal inputs.

Acknowledgment

We would like to thank Guimu Robot Co. Ltd. for their feedback. We are also grateful to C. Chou, H. Cheng, S. Yeh, A. Kingery, A. Angert, T. Sun, D. Wang, Y. Sun, Y. Yu, J. Gong, and M. Momin for their inputs and contributions to the Networked Robots Laboratory at Texas A&M University.

References

[1] H. La, R. Lim, B. Basily, et al., “Mechatronic systems design for an autonomous robotic system for high-efficiency bridge deck inspection and evaluation,” *IEEE/ASME Trans. on Mechatronics*, vol. 18, no. 6, pp.1655-1664, Dec. 2013.

[2] H. La, R. Lim, B. Basily, et al., “Autonomous robotic system for high-efficiency non-destructive bridge deck inspection and evaluation,” in *Proc. Int. Conf. on Automation Science and Engineering.*, 2013, pp. 1065-1070.

[3] H. La, N.Gucunski, S. Kee, and L. Nguyen, “Data analysis and visualization for the bridge deck inspection and evaluation robotic system,” *Springer Journal of Visualization in Engineering*, no. 3:6, pp.1-16, Feb. 2015.

[4] Y. Hu, and C. Zhao, “Alocal binary pattern based methods for pavement crack detection,” *Journal of Pattern Recognition Research*, vol. 5, no. 1, pp.140-147, Sep. 2010.

[5] J. Tang, and Y. Gu, “Automatic crack detection and segmentation using a hybrid algorithm for road distress analysis,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2013, pp. 3026-3030.

[6] R. Lim, H. La, W. Sheng, and Z. Shan, “Developing a crack inspection robot for bridge maintenance,” in *IEEE International Conference on Robotics and Automation.*, 2011, pp. 6288-6293.

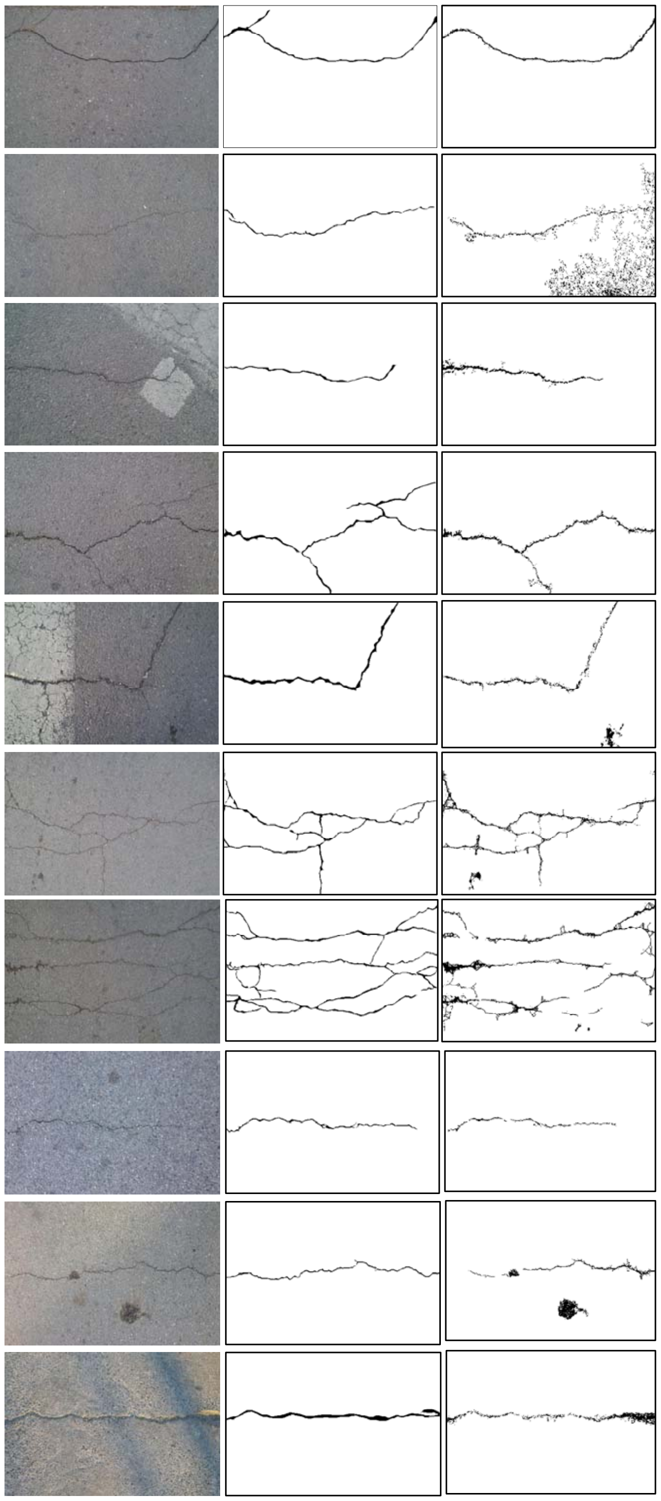


Fig. 10. Some experimental results using our MFCD algorithm on CFD dataset (from left to right: original image, ground truth, and results from MFCD).

[7] R. Lim, M. Hung, and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp.367-378, April 2014.

[8] H. Oliveira, and P. L. Correia, "Automatic road crack detection and characterization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp.155-168, March 2013.

[9] Y. Hu, C. Zhao, and H. Wang, "Automatic pavement crack detection using texture and shape descriptors," *IETE Technical Review*, vol. 27, no. 5, pp.398-405, Sep. 2010.

[10] M. Jahanshahi, S. Masri, C. Padgett, and G. Sukhatme, "An innovative methodology for detection and quantification of cracks through incorporation of depth perception," *Machine Vision Application*, vol. 24, no. 2, pp.227-241, 2013.

[11] P. Prasanna, K. J. Dana, N. Gucunski, et al. ,"Automatic crack detection on concrete bridges," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp.591-599, April 2016.

[12] Y. Shi, L. Cui, Z. Qi, F. Meng,and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp.3434-3445, Dec. 2016.

[13] A. Cord, and S. Chambon, "Automatic road defect detection by textural pattern recognition based on AdaBoost," *Computer-Aided Civil Infrastructure Engineering*, vol. 27, no. 4, pp.244-249, April 2011.

[14] B. Lee, Y. Kim, S. Yi, and J. Kim, "Automated image processing technique for detecting and analyzing concrete surface cracks," *Structure Infrastructure Engineering*, vol. 9, no. 6, pp.567-577, 2013.

[15] H. La, N. Gucunski, K. Dana, and S. Kee, "Development of an autonomous bridge deck inspection robotic system," *Journal of Field Robotics*, in press, Apr. 2017.

[16] D. Zhang, Q. Li, Y. Chen, et al., "An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection," *Image and Vision Computing*, vol. 2017, no. 57, pp.130-146, 2017.

[17] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: an algorithm based on minimal path selection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp.2718-2729, Oct. 2016.

[18] S. Chambon, and J. M. Moliard, "Automatic road pavement assessment with image processing: Review and comparison," *International Journal of Geophysics*, vol. 2011, pp.1-20, 2011.

[19] M. Gavilan, D. Balcones, O. Marcos, et al., "Adaptive road crack detection system by pavement classification," *Sensors*, vol. 11, no. 10, pp.9628-9657, Oct. 2011.

[20] V. Kaul, A. Yezzi, and Y. Tsai, "Detecting curves with unknown endpoints and arbitrary topology using minimal paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp.1952-1965, Oct. 2012.

[21] Y. Jeon, J. Yun, D. Choi, and S. Kim, "Defect Detection algorithm for corner cracks in steel billet using discrete wavelet transform," in *Proc. Int. Conf. Control, Autom., Syst.*, 2009, pp. 2769-2773.

[22] R. G. Lins, and S. N. Givigi, "Automatic crack detection and measurement based on image analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 3, pp.583-590, March 2016.

[23] L. Ying, and E. Salari, "Beamlet transform-based technique for pavement crack detection and classification," *Computer-Aided Civil and Infrastructure Engineering*, vol. 25, no. 8, pp.572-580,Nov. 2010.

[24] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing (4th Edition)*. London, U.K.: Pearson, 2017.

[25] M. Ester, H. P. Kriegel, J. Sander, et al, "A density-based algorithm for discovering clusters in large spatial databases with noise", *Kdd*, vol. 96, no. 34, pp. 226-231, 1996.

[26] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp.269-271, 1959.

[27] Y. Huang, and B. Xu, "Automatic inspection of pavement cracking distress," *Journal of Electronic Imaging*, vol. 15, no. 1, pp.1-6, Mar. 2006.

[28] J. Canny, "A computational approach to edge detection," *IEEE Trans. on Pattern Anal. Mach. Intell.*, no. 6, pp.679-698, 1986.

[29] S. Chambon, "Detection of points of interest for geodesic contours: Application on road images for crack detection," in *Proc. Int. Conf. Comput. VISAPP*, 2011, pp. 1-4.

[30] T. S. Nguyen, S. Begot, F. Duculty, and M. Avila, "Free-form anisotropy: A new method for crack detection on pavement surface images," in *Proc. Int. Conf. Image Process.*, 2011, pp. 1069-1072.

Haifeng Li received the Ph.D. degree in control theory and control engineering from Nankai University, Tianjin, China, in 2012.

He is an associate professor with Department of Computer Science and Technology, Civil Aviation University of China, Tianjin, China. He has authored or coauthored over 30 technical articles. His research interests include computer vision, image processing, robotic sensing, multisensor fusion, robot localization and navigation.

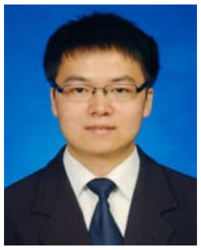
Dezhen Song (S'02-M'04-SM'09) received the Ph.D. degree in industrial engineering from University of California, Berkeley, CA, US, in 2004.

He is a Professor with Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA. His research interests include networked robotics, distributed sensing, computer vision, surveillance, and stochastic modeling.

Dr. Song received the Kayamori Best Paper Award of the 2005 IEEE International Conference on Robotics and Automation (with J. Yi and S. Ding). He received NSF Faculty Early Career Development (CAREER) Award in 2007. From 2008 to 2012, Song was an associate editor of IEEE Transactions on Robotics. From 2010 to 2014, Song was an Associate Editor of IEEE Transactions on Automation Science and Engineering. He is currently a Senior Editor for IEEE Robotics and Automation Letters (RA-L), a new flagship journal from IEEE Robotics and Automation Society. He is also an author and a Multimedia Editor for Springer Handbook for Robotics.

Yu Liu received the B.S. degree from the Department of Computer Science and Technology from Taiyuan University of Technology, Taiyuan, China, in 2015. He is currently working toward the Master degree in computer science and technology with Civil Aviation University of China, Tianjin, China.

His current research interests include the areas of computer vision, image processing and understanding.



Binbin Li received the B.S. degree from the Department of Electrical Engineering and Automation from Harbin Institute of Technology, Harbin, China, in 2012. He is currently working toward the Ph.D. degree in computer engineering with Texas A&M University, College Station, TX, USA.

His current research interests include the areas of robot vision, visual tracking and recognition.