Hindawi
Advances in Civil Engineering
Volume 2020, Article ID 8896176, 23 pages
https://doi.org/10.1155/2020/8896176

Research Article

An Enhanced Percolation Method for Automatic Detection of Cracks in Concrete Bridges

Qingfei Gao,[1] **Yu Wang**,[1] **Jun Li**,[2] **Kejian Sheng**,[3] **and Chenguang Liu**[4]

[1]*School of Transportation Science and Engineering, Harbin Institute of Technology, Harbin 150090, China*
[2]*Department of Municipal and Environmental Engineering, Heilongjiang Institute of Construction Technology, Harbin 150050, China*
[3]*College of Civil and Architectural Engineering, Heilongjiang Institute of Technology, Harbin 150050, China*
[4]*School of Civil Engineering, Suzhou University of Science and Technology, Suzhou 215000, China*

Correspondence should be addressed to Qingfei Gao; gaoqingfei@hit.edu.cn

Received 7 September 2020; Revised 25 September 2020; Accepted 29 September 2020; Published 10 October 2020

Academic Editor: Yi Zhang

Copyright © 2020 Qingfei Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As cracks on concrete bridges become severer and more frequent, methods of detecting cracks on concrete bridges have aroused great concern. Conventional methods, e.g., manual detection and equipment-aided detection, suffer from subjectivity and inefficiency, which increases demands for an accurate and efficient method to detect bridge cracks. To this end, we modify the existing percolation method and propose an enhanced percolation method, which detects cracks of concrete bridges automatically. The modification includes three improvements, which are (1) employing photo expansion to eliminate boundary effects, (2) varying shape factors to increase the accuracy of percolating unclear cracks, and (3) decreasing the number of neighbouring pixels to form candidate sets. Combined with the above three improvements, three versions of enhanced percolation methods utilizing three different shape factors are put forward. The numerical experiment on detecting cracks in 200 images of the bridge surface demonstrates the outperformance of the enhanced percolation method in precision, recall, *F*-1 score, and time compared with traditional detecting methods. The proposed method can be generalized on the application of detecting other types of bridge diseases, which is an advantage for designing, maintaining, and restoring infrastructures.

1. Introduction

Bridge serves as one of the most important components in infrastructures, which imposes significant effect on the economy and social activities. With the increasing number of bridges but the inadequate funds and poor techniques for their maintenance [1–4], aging problems such as damages and deformations become severer and more frequent. According to the National Bridge Inventory (2008), there are 71,466 structurally deficient and 79,922 functionally obsolete bridges in the US [5]. On March 15, 2018, the collapse of the concrete bridge, "FIU pedestrian bridge," caused six deaths and permanent disability of one worker in Florida [6]. Although the accident is directly caused by the design error, the engineers' ignorance of cracks resulted in the miss of the best time of repairing the bridge [7]. The accident could be avoided had the civil engineers detect those growing cracks and shut down the street. Therefore, to guarantee normal operations of bridges, regular inspection and periodic maintenance are necessary. Manual inspection and equipment-aided inspection are two main kinds of bridge inspection methods [8]. However, they more or less have some limitations. In manual inspection, workers count the number of cracks and measure their sizes through crack photos. The quality and reliability of diagnosis reports on bridge cracks are highly dependent on the experience and education of the inspection workers, which suffer so much from subjectivity [9, 10]. In the equipment-aided inspection, poor technical feasibility and high expenses are two main drawbacks [8]. Besides, inefficiency of the human intervention leads to the incapability of the above two methods to be utilized in large-scale detections. Therefore, it is essential

to develop an efficient and automated method to detect bridge damages.

Recently, the development of computer vision enables researchers to apply techniques of image processing to detect damages of bridges [11]. Comprehensive reviews have been delivered by many scholars in the current literature [12–15]. Koch et al. [13] presented systematical summarization of defect detection and condition assessment of civil infrastructures, especially reinforced concrete bridges, based on image procession. They divided the detection process into 5 stages from the bottom to top level, which were preprocessing, segmentation, feature extraction, object recognition, and structural analysis. In each stage, specific methods for defect detection, classification, and assessment were given. Mohan and Poobal [14] analyzed 50 papers about crack detection from five features: the image-processing techniques, objectives, accuracy level, error level, and the image datasets. They proposed a general architecture for crack detection using the image-processing technique. Jahanshahi et al. [15] provided a survey and an evaluation of some of the promising vision-based approaches for automatic detection of cracks and corrosion in civil infrastructure systems, including edge and line detection, morphological functions, clustering, and pattern recognition. Rose et al. [9] reviewed current detection approaches for cracks on the concrete surface and classified them generally into edge detection, segmentation and percolation, machine learning methods, and morphology operations. Inspired by the above crack detection methods, we rigorously classify the relevant methods broadly as edge detection, percolation, machine learning methods, and other techniques, which are covered, respectively, in the following paragraphs.

Edge extraction is a simple way to detect bridge cracks. Due to the difference in pixel values between cracks and backgrounds of the bridge surface, cracks can be treated as edges and then extracted from backgrounds [16–20]. There are two key steps in edge extraction. Firstly, the gradient value of pixels is calculated by applying filters on target images. Then, a threshold is set, and pixels with higher gradient value are classified into edges, i.e., cracks. Sobel and Canny are two typical detectors used in edge extraction [16, 21, 22]. Lim et al. [20] employed another detector, Laplace, to identify cracks and located them on the bridge surface. However, due to keen sensitivity to the noise, edge detection very easily mistakes stains for cracks. Accordingly, many works were designed to incorporate noise reduction into edge detection. Pan et al. [23] introduced a new threshold-based denoising algorithm, and the result showed that it performed better and required less computation. Li et al [19] utilized the wavelet based-algorithm to enhance, smooth, and denoise images.

Percolation method is another way to detect bridge cracks. In this method, two features of cracks are utilized: linear shape and the difference of pixel values between cracks and backgrounds [11, 24–30]. Yamaguchi and Hashimoto [24] proposed a novel crack detection method for a concrete surface image using percolation. They evaluated whether a random pixel was crack or not based on the shape of a cluster formed using percolation processing. Later, they utilized three techniques to increase the accuracy and reduce the computational time [11, 25–27]. However, this method is still time-consuming, and some noise areas are falsely detected as crack regions. To overcome the problems above, Qu et al. [28] improved the percolation algorithm by including an accelerated algorithm and a new denoising method. Experimental results showed that the proposed algorithm could be used for accurately and efficiently detecting cracks in the image. Building on this work, they further combined genetic programming and percolation model and lining seam elimination and percolation model to take full account of the specific characteristics of the concrete surface [29, 30]. The experimental results show that the algorithm can not only quickly and accurately detect the concrete surface cracks but also eliminate the interference, such as stains and blocks.

A great many techniques from machine learning, such as neural network, have been employed in detecting cracks [31–36]. Prasanna et al. [31] utilized the support vector machine to detect cracks on the bridge slab, but the accuracy was not very high. Lattanzi and Miller [32] combined the Canny detector and K-means classification and applied them to detect bridge cracks. The accuracy was high, and the recognized speed was fast. However, the method was susceptible to light condition. With the advent of the convolution neural network (CNN), the performance of computer vision achieved a giant leap in detecting and classifying objects, as well as cracks. Kim et al. [33] used unmanned aerial vehicles to take bridge photos and reconstructed the bridge surface model. Then, they used the CNN to recognize and quantify bridge cracks. The ideas of deep learning and transfer learning were used to enhance classifier ability. The training stage and testing stage cost 14 min and 5 s, respectively, and the results were satisfying. Silva and Lucena [34] deepened the neural network. In their research, 2800 concrete bridge photos were for training, while 700 photos were for testing. The accuracy was very high, nearly to 92.3%. Li et al. [35] employed the slide window technique to enlarge their dataset. The accuracy went up to 97.9%. Jahanshahi and Masri [36] employed a 3D reconstruction technique to gain image depth and combined the neural network and support vector machine (SVM) to identify cracks. The method reduced interference of camera pose and resolution, but the accuracy was just 80%.

A broad spectrum of other approaches has been proposed for crack detection. Employing depth information of points in cracks and backgrounds is another way [36]. Cabaleiro et al. [37] presented an algorithm for the automatic detection of cracks in timber beams sampled by LiDAR data. They pointed cloud of the beam face, removed points inside the cracks, and projected them on a 2D coordinate system to identify crack outlines. Abdel-Qader et al. [38] used a PCA-based algorithm to detect bridge cracks. They compared the performance of methods using PCA alone, PCA with a linear feature process, and PCA on the local region. However, the accuracy of the three kinds of methods was affected dramatically by camera pose and distance from where images were taken. Yu et al. [39] used

robots to collect bridge crack images, but this method required to label the start and end points in cracks manually. Oh et al. [10] used a crack tracking way to extract width and length information of cracks. Zou et al. [40] proposed theF $*$ seed-growing approach for automatic crack-line detection through extending the F $*$ algorithm, and based on this, they developed CrackTree [41], which is a fully automated method to detect cracks from pavement images.

Based on the percolation method [11, 24–30], this paper incorporates three improvement techniques to increase the efficiency of the original method. Image expansion is used to eliminate the boundary effect of percolation. Linear and quadratic functions are used to replace the constant stride value to increase the accuracy of percolating unclear cracks. Instead of using eight neighbouring pixels, four neighbouring pixels are used to form the candidate area in the percolation process. To validate the performance of the three improvements, three experiments are designed accordingly.

Cracks on the surface of infrastructures are mainly divided into four types based on the type of materials used for their construction: concrete, asphalt, timber, and the combination of them [37, 42–45]. During percolation, different pixel values between cracks and backgrounds, together with the shape of the final percolated area, are two critical factors determining the final classification results. However, as shown in Figure 1(a) of an asphalt surface, the similarity of colours between cracks and backgrounds increases the difficulty of differentiating cracks from the background surface. For cracks on the timber surface shown in Figure 1(b) [13], although colours of cracks and background areas differ significantly, areas owning such significant difference are not only limited to cracks but also spread all over the timber surface, which causes percolation to mistake flawless areas for cracks and therefore decreases the detection performance. However, on the concrete surface in Figure 1(c), not only do colours of cracks have a stark contrast with backgrounds but also there are fewer noises on the concrete surface compared with the timber one. Thus, after comparing the colour difference and noise distribution among surfaces of four material types, the concrete surface is the most suitable one to apply the percolation algorithm.

2. Materials and Methods

Initially, an existing percolation-based method to detect cracks is introduced, and three experiments are designed to illustrate the effect of three critical procedures during percolation. Then, three improvements are incorporated to produce a new percolation method. After applying the enhanced method to detect photos using the 200-dataset, the effectiveness of the new method is demonstrated.

In the whole article, photos we use to validate the improvement or to evaluate the performance of our final percolation algorithm come from the dataset containing 100 crack images and 100 background images. The dataset was collected from two primary sources: the first group was collected online through googling "Concrete Bridge Cracks" and selected carefully based on two metrics by experienced bridge engineers. Firstly, we only consider the images of concrete bridges since the linear shape of the cracks and great difference in the pixel value between cracks and backgrounds on the concrete surface are suitable for applying percolation methods. Secondly, the cracks on the images should at least be recognized by humans. The second group was collected through real shooting. We went to the lab and took photos of the surface of concrete bridge components. By such two ways, we gathered about 6000 images. Every image was checked carefully, and 200 images that were the most representative of the cracks and backgrounds of concrete bridges were selected to form the 200-dataset. Various resolution sizes of the 200 images are transformed to a unified level by employing the resize function in MATLAB. To promote the progress of the field, we uploaded our source data online for peers' convenience. The source data can be found at 6000-dataset and 200-dataset.

2.1. Existing Percolation Model. The idea of the percolation model comes from the natural phenomenon of liquid permeation. This model is used in solving problems with propagation nature, such as the spread of epidemics and fires. The connectivity of cracks enables the model to detect the spreading phenomenon through percolation. For each pixel in the image, a corresponding percolated area is generated along with its shape factor. The process is repeated on every pixel until the completion of the whole image when every pixel is assigned a percolated area as well as a shape factor. The closeness of the shape factor to 1 denotes the likelihood of the corresponding pixel belonging to backgrounds. The process is detailed in [11, 24–27]. The following three sections focus on three specific procedures used in the process.

2.1.1. Line Detection Experiment. The crack size varies from bridges to bridges, and even on the same bridge, it can be significantly different due to factors such as camera position and resolution ratio. Therefore, it is necessary to make the percolation method applicable to cracks with different sizes, which cannot be achieved with the fixed-size window [27]. The shortcoming of the fixed-size window is disclosed by a line detection experiment, where an image of 100×100 resolution ratio with three black lines and white backgrounds is used. Three black lines are in different thicknesses, and we perform percolation 6 times on each line. During the process, the size of the fixed window increases from 5 to 30 with interval 5. Fourteen points are selected from lines and backgrounds to illuminate the limitation of the fixed-size window and generality of the flexible-size window. Detecting crack areas, to some extent, is similar to the process of differentiating lines from backgrounds, and thus, replacing the fixed-size window with the flexible-size window is reasonable, as shown from the experiment in Section 3.1.1.

To speed up the percolation process, two strategies, conditional termination and skip, are used. Two experiments are designed as follows to illustrate their effects, respectively.

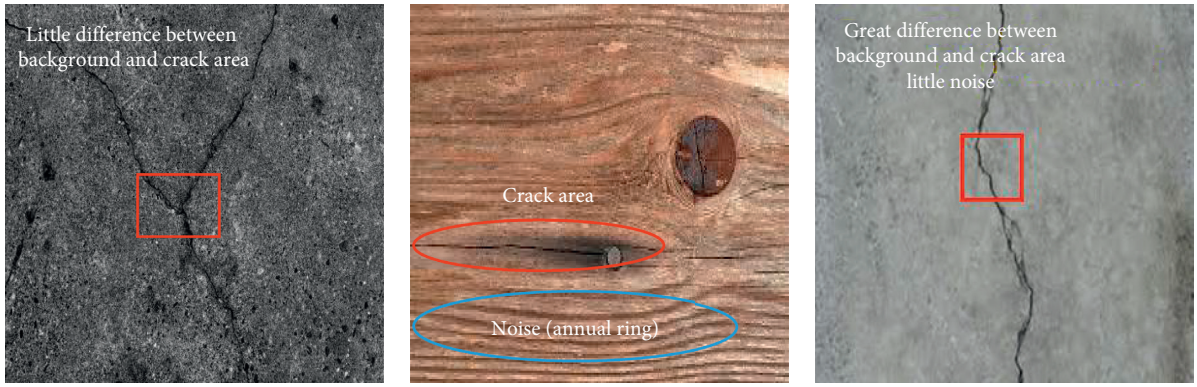


Figure 1: Cracks on different types of surfaces: (a) asphalt pavement; (b) timber; (c) concrete.

2.1.2. Conditional Termination Strategy. Every time, the percolated area reaches the boundary of the window, and the size of the window is enlarged for the percolation process to continue. However, if the shape of the current percolated area has already been circular enough, namely, the shape factor is already much closer to one, it is reasonable to classify the corresponding pixel as the background pixel and terminate the percolation process. To trigger this conditional termination, we compare the value of the shape factor after every iteration with a constant threshold value which is defined in advance. Once the value exceeds the threshold, we stop the percolation and thus save time [27].

We use three 500×500 resolution ratio images and apply percolation with and without the conditional termination procedure on four pixels in each image. One of the images is used twice. The parameters are set as follows: $N = 21$, $M = 41$, $w = 1$, and $T_s = 0.2$. Also, through tuning parameters and watching the percolation time, we investigate the effects of window size N and M on speeding up the performance of the strategy. The result is shown in Section 3.1.2.

2.1.3. Skip Strategy. During the first iteration of percolation on the background pixel, due to its high pixel value, the threshold value will also be very high, and both background pixels and crack pixels will be included in the percolated area. On the contrary, the threshold value during the first iteration of percolation towards the crack pixel will be very low, and only crack pixels can be included in the percolated area. Based on this rule, we can evaluate whether the focal pixel is a crack or not by checking whether its first-round percolated area D_p contains background pixels. If there are no background pixels, then the focal pixel might belong to the crack area, and percolation should continue. Otherwise, the percolation process is terminated, and the focal pixel is classified as a background pixel [27].

We use one 100×100 resolution ratio image and apply percolation without any strategy, with conditional termination strategy, and skip strategy on pixels in 2×11 areas. The parameters are set as follows: $N = 10$, $M = 20$, $w = 1$, and threshold value T_s is set to 0.2, and the result is shown in Section 3.1.3.

2.1.4. Percolation Using the Existing Method. The existing percolation method is applied to the 200-dataset, and four typical images with 100×100 resolution ratio are selected to illuminate the problems of the method. The parameters are set as follows: $N = 10$ and $M = 20$. Instead of comparing the F_c value with T_s and setting the pixel value as 0 or 255, we directly use $F_c \times 255$ to update the pixel value. Such change could reflect the relationship between the category of the initial pixel, the shape of its percolated area, and the F_c value. The percolated results can be seen in Section 3.1.4.

In Section 2.1, the main points of the existing percolation method are summarized, and three experiments are designed to manifest their effects. In Section 2.2, improved techniques based on problems of the existing percolation method are proposed and validated through experiments.

2.2. Improved Techniques

2.2.1. Image Expansion. From the results of detecting bridge cracks using the existing percolation method, an obvious problem can be found that those pixels along the image boundary, no matter whether they are cracks or backgrounds, are all assigned a shape factor of low value and coloured blue. Therefore, the current existing percolation method cannot differentiate cracks and backgrounds in boundary areas. One possible reason is that the percolated areas cannot extend beyond the image boundary, making themselves more likely to be linear. To eliminate the boundary effect, the image is enlarged by adding pixels on the outside of the current image boundary. Then, the original boundary areas become inside, and the percolation in these areas will not be affected by the boundary. This pixel-adding technique is called "image expansion."

Based on the value of pixels we add, three types of image expansion techniques are used: zero padding, same padding, and mean padding. Zero padding adds pixels with zero value, and same padding adds pixels with the same value as the pixels in the original boundary areas. Mean padding adds pixels with the value equaling the mean value of all pixels in the original image. To figure out which one has the best

performance in terms of eliminating the boundary effect, we use them, respectively, as the preprocessing of the existing percolation method and compare them with the one without expansion. In this experiment, two 500×500 resolution ratio crack images are used, and 20 pixels are added on the width and height side of each image. Added pixels are removed when percolation is completed. The parameters are set as follows: $N = 21$, $M = 41$, $w = 1$, and threshold value T_s is set to 0.2. The performance of each expansion technique is shown in Section 3.2.1.

2.2.2. Slacken Stride. Some of the unclear crack pixels are ignored using the existing percolation method, especially in images where the colour difference between cracks and backgrounds is little. It is because the stride parameter w takes a constant value. After a certain number of iterations, the threshold value T increases to a high level due to the contribution of w. Then, even those background pixels with high pixel values are more likely to be included in the percolated area and change its shape factor from near 0 to 1. Therefore, the constant value of w is replaced and set to be a product of the shape factor of the percolated area and the constant value [27]. Because the value of the shape factor after the first iteration is between 0 and 1, the increase of w can be well controlled by the linear function. However, it is not the only way to realize the inhibiting effect; a quadratic function and a cubic function are designed to replace the constant value. These strides are called linear, quadratic, and cubic strides, and this strategy is called the slacken stride strategy. Their equations are as follows:

$$w' = F_c \cdot w, \tag{1}$$

$$w'' = F_c^2 \cdot w, \tag{2}$$

$$w'' = F_c^3 \cdot w, \tag{3}$$

$$T = \max\left(\max_{p \in D_p}(I(p)), T\right) + w' \text{ or } w'', \tag{4}$$

$$F_c = \frac{4 \cdot C_{\text{count}}}{\pi \cdot C_{\max}^2}, \tag{5}$$

where w' is a linear function of F_c, w'' is a quadratic function of F_c, and w'' is a cubic function of F_c. In equation (5), when the shape factor grows nearer to 1, the area of the percolation area grows closer to a circular shape. Since the cracks on the surface of concrete bridges typically form in a linear shape, the pixel from which the percolation starts is highly possible to belong to the background area.

To validate the slacken stride strategy in terms of accuracy, three pixels are selected from the background, unclear crack, and clear crack areas, respectively, to be percolated using constant, linear, and quadratic strides. Other parameters are set as follows: fixed stride value $w = 1$, $N = 10$, $M = 20$, and the image resolution ratio is 100×100, and the result can be seen in Section 3.2.2.

2.2.3. Four Neighbouring Pixels' Region. In the existing percolation method, the pixels to be included in the percolated area are selected from 8 neighbouring pixels based on the assumption that cracks are connected. However, pixels on the diagonal positions of the nine-patch are usually not crack pixels even if the centre pixel is a crack one. Therefore, we modified the corresponding procedure in the existing percolation process by just using 4 adjacent pixels to form the candidate area, as can be seen in Figure 2.

2.3. Final Percolation Methods. Combined with all improved techniques detailed in Section 2.2, three final enhanced percolation methods based on linear, quadratic, and cubic strides are proposed. The only difference between these three methods is just they use three different equations, equations (1) to (3), to update the stride value w, so their flowchart and procedure are given only once, as shown in Figure 3.

2.3.1. Comparison with Classic Crack Detection Methods. To evaluate the test result, we randomly select 50 images from the 200-dataset and ask experienced bridge engineers to mark crack pixels on each image. Three measures, precision, recall, and F-measure, are computed by comparing the detected crack curves against the human-annotated ground-truth crack curves. Because the cracks in concrete bridge images have a certain width, we allow a certain error in measuring the coincidence between the detected crack curves and the ground-truth crack curves. More specifically, the average crack width in our dataset is around 3 pixels. Therefore, a detected crack pixel is still considered to be a true positive if it is located no more than 3 pixels away from human-annotated crack curves. For the parameters, we set $w = 1$, $N = 10$, $M = 20$, and $T_s = 0.2$. The three methods are compared with each other in terms of precision, recall, F-measure, and time. Then, they are further compared with the other five classic methods in [41]. The results can be seen in Section 3.3.1.

2.3.2. Practical Test. To test the practical performance of our three enhanced algorithms, 200 concrete bridge images of 100×100 resolution ratio are percolated. The parameters are set as follows: $w = 1$, $N = 10$, $M = 20$, and $T_s = 0.2$. Zero-padding image expansion technique is used with the number of pixels added on each side equaling 10. The results are included in Section 3.3.2.

(i) Firstly, image expansion and monochrome are performed. The size of the local window is fixed as $N \times N$, and the maximum window is defined as $M \times M$. The initial stride is set as w. A pixel is selected randomly in the image as the initial pixel, and it is included in the percolated area D_p. The threshold value T is set as the initial pixel value.

(ii) The threshold T is updated using equation (4), and at the first iteration, F_c is set to zero.

(iii) The four neighbouring pixels of all pixels on the boundary of D_p are selected as the precandidate set

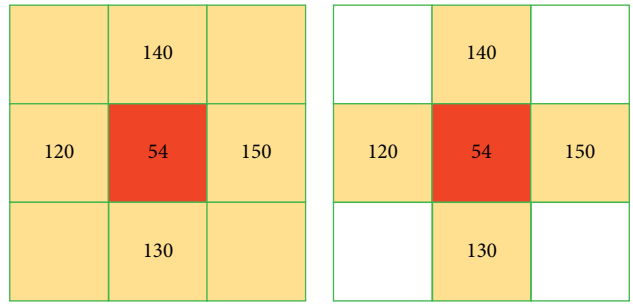


Figure 2: The selection of the candidate area during percolation: (a) eight neighbouring pixels' region; (b) four neighbouring pixels' region.

D_{pc}. The repeated pixels in D_{pc} are eliminated, and the rest of them are defined as the candidate set D_c.

(iv) In D_c, the pixels with values lower than the threshold T are percolated and included in D_p. If there are no such pixels in D_c, then pixels with the lowest value are included in D_p.

(v) If the process is in its first iteration, it moves to step (vi); otherwise, the process moves to step (vii).

(vi) Whether D_p contains background pixels is checked; if so, the process goes to step (vii); otherwise, F_c is set to 1, and the program turns to step (xv).

(vii) Percolated area D_p is checked. If it reaches the boundary of the local window, the process goes to step (viii). Otherwise, it returns to step (ii).

(viii) N is incremented to $N+2$.

(ix) F_c value is calculated using equation (5), and the threshold value T is updated using equation (4).

(x) F_c value is compared with T_s. If it is bigger than T_s, the process ends, and F_c is set to 1. Otherwise, the process moves to step (xi).

(xi) The four neighbouring pixels of all pixels on the edge of the D_p area are selected as the precandidate set D_{pc}. The same pixels in D_{pc} are eliminated, and the rest of them are defined as the candidate set D_c.

(xii) In D_c, the pixels with values lower than the threshold T are percolated and included in D_p, and the process moves to step (xiii). If there are no such pixels, the F_c value is calculated using equation (5), and the process terminates.

(xiii) Whether percolated area D_p reaches the boundary of the updated local window is checked; the process returns to step (ix) if no. Otherwise, the process goes to step (xiv).

(xiv) N is incremented to $N+2$ and compared with the maximum window size M. If N is larger than M, the F_c value is calculated based on equation (5), and the process goes to step (xv). Otherwise, the process returns to step (ix).

(xv) If the F_c value is smaller than the threshold value, the corresponding initial pixel is classified as the crack pixel, and its pixel value is updated to 0 and coloured black. Otherwise, the pixel is treated as the background pixel, and its pixel value is updated to 255 and coloured white.

(xvi) Whether all pixels in the image are percolated is checked. If so, the program terminates; otherwise, the process returns to step (i).

3. Results

Based on the existing percolation method and the improvements discussed in Section 2, their experiment results are discussed and analyzed in this section.

3.1. Crack Detection Using the Existing Percolation Model. In this part, the results of three experiments introduced in Section 2.1 are given and analyzed. Then, crack detection results by the existing percolation method are presented, and their problems are specified.

3.1.1. Results of the Line Detection Experiment. The percolation results of 14 pixels using different window sizes can be seen in Figures 4 and 5. As shown in Figure 4(a), when the window size is set to 5, the percolated regions of pixels 5 and 8 inside thick and middle lines are a near-circle, making their colour as white as background pixel 2. However, in terms of pixels in the thin line such as 12 and 13, the percolated regions are linear and coloured black. All these pixels are located among line areas, but they have different percolated results. It is because when the window size is too small compared with the size of the target to be percolated, the enlarging percolated region touches the window before reaching the target's boundary. Therefore, the region's shape factor value cannot reflect the shape of the target very well, and using this value to classify the pixel will produce errors. The analysis can be further proved by the changing values of shape factors as window size increases, which are shown in Figure 6 and Table 1. As the window size increases, the shape factors of those crack pixels decrease, such as pixels 3 to 14. In addition, those background pixels remain high such as 1 to 2, which means the fixed-size window is not suitable for percolating pixels of various sizes. Therefore, we should replace the fixed-size window with a flexible-size window to increase the percolation performance.

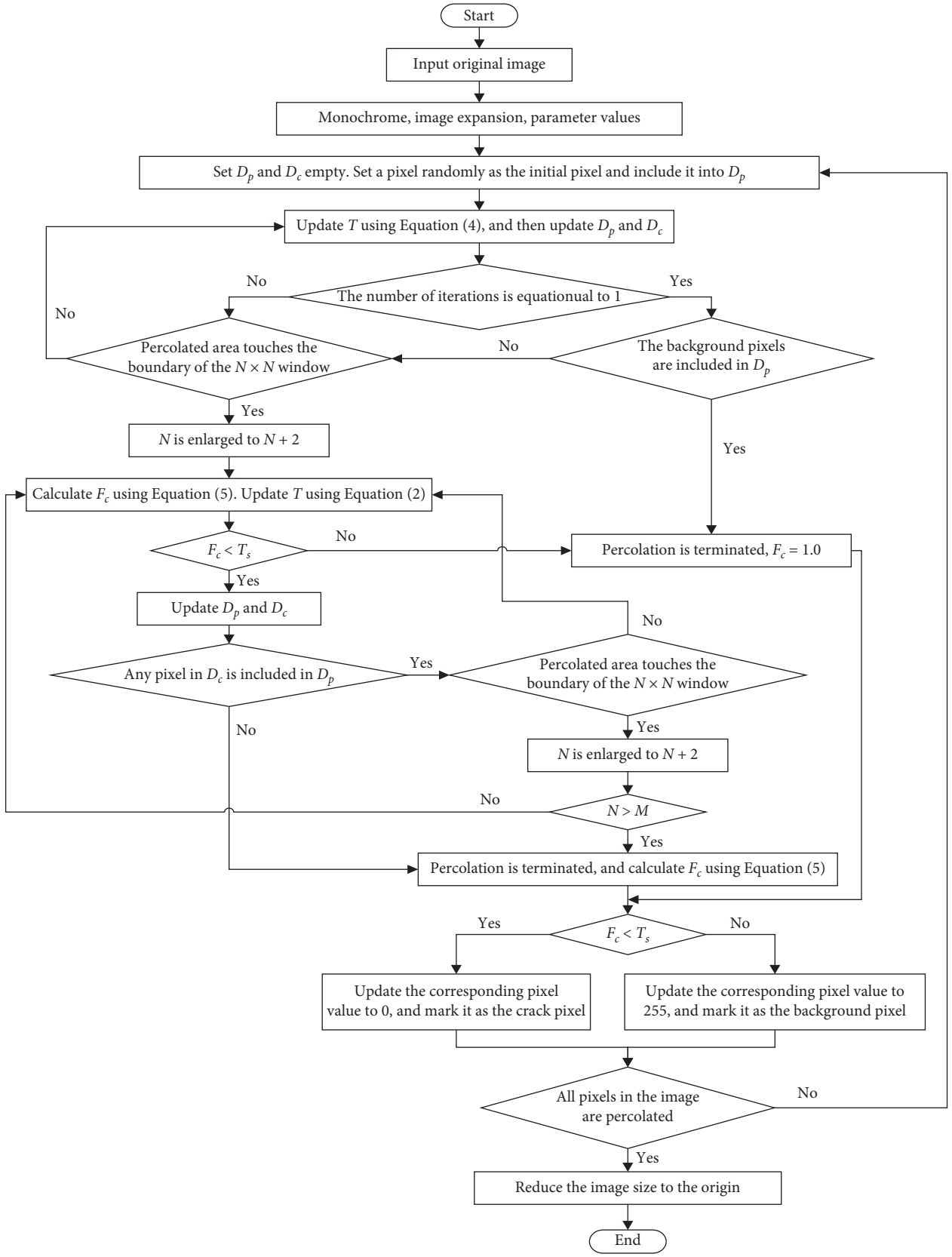
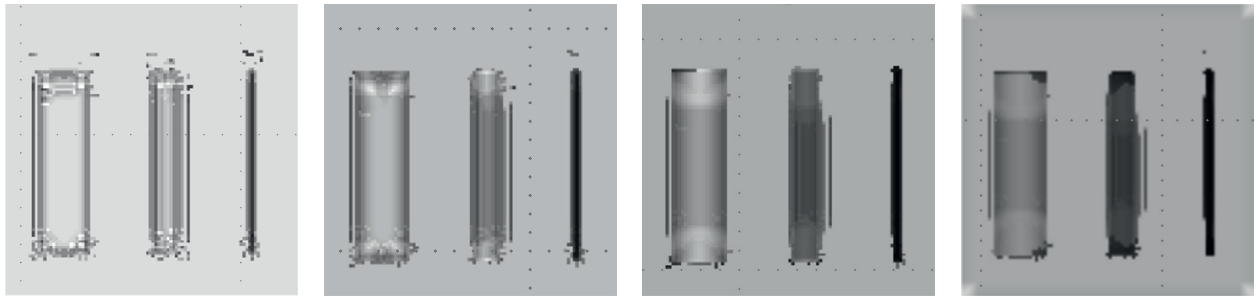
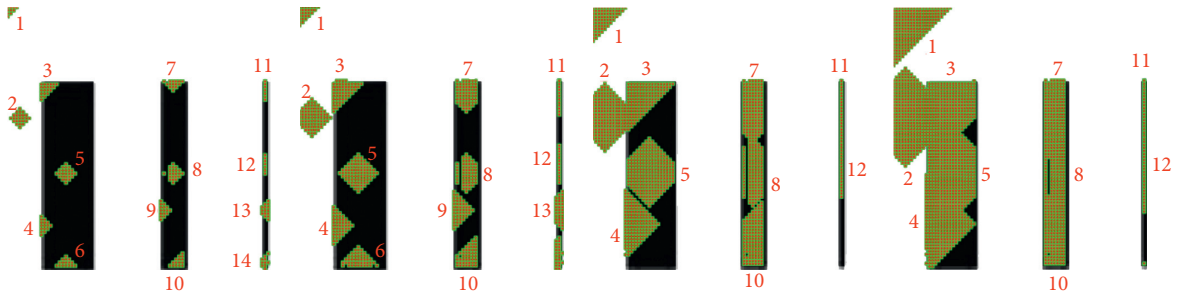


Figure 3: Flowchart of the enhanced percolation method.



(a) (b) (c) (d)

Figure 4: Percolation result of 100 resolution images using different window sizes: (a) 5 window size; (b) 10 window size; (c) 20 window size; (d) 30 window size.



(a) (b) (c) (d)

Figure 5: Percolation areas of 14 pixels using different window sizes: (a) 5 window size; (b) 10 window size; (c) 20 window size; (d) 30 window size.

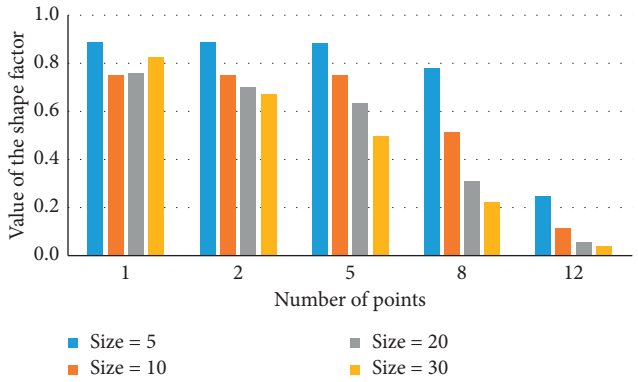


Figure 6: The relationship between the value of the shape factor and the size of the percolation window.

Table 1: F_c value of percolation regions of different pixels in 100 resolution images.

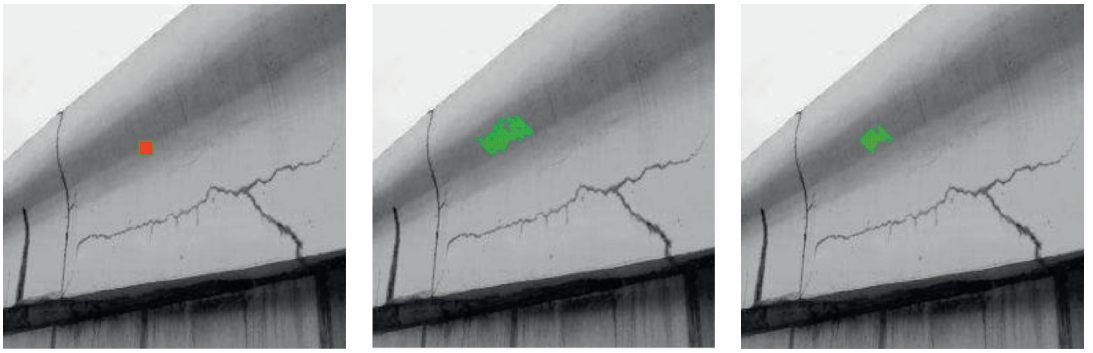
Pixel/window size	5	10	20	30
1	0.88	0.75	0.76	0.83
2	0.88	0.75	0.70	0.67
3	0.73	0.58	0.49	0.44
4	0.57	0.43	0.41	0.40
5	0.88	0.75	0.63	0.50
6	0.57	0.48	0.75	0.58
7	0.53	0.75	0.42	0.16
8	0.78	0.51	0.31	0.22
9	0.57	0.43	0.36	0.28
10	0.6	0.52	0.37	0.15
11	0.25	0.13	0.06	0.03
12	0.25	0.11	0.06	0.04
13	0.53	0.29	0.17	0.13
14	0.49	0.29	0.14	0.06

3.1.2. Performance of the Conditional Termination Strategy. The percolation time, iteration number, and shape factor value of the four pixels with and without the conditional termination procedure are shown in Table 2. The reduced time and iteration number together with the almost unchanged shape factor value demonstrate that the strategy can not only accelerate the percolation method but also has no effect on its accuracy. From Figures 7–10, we can also see that the size of the percolated area of each pixel decreases, and their whole shapes remain the same.

Besides, by tuning the upper and lower bound of the window, the effect of them on the percolation performance is illustrated. In Figure 11(a), as the upper bound of the

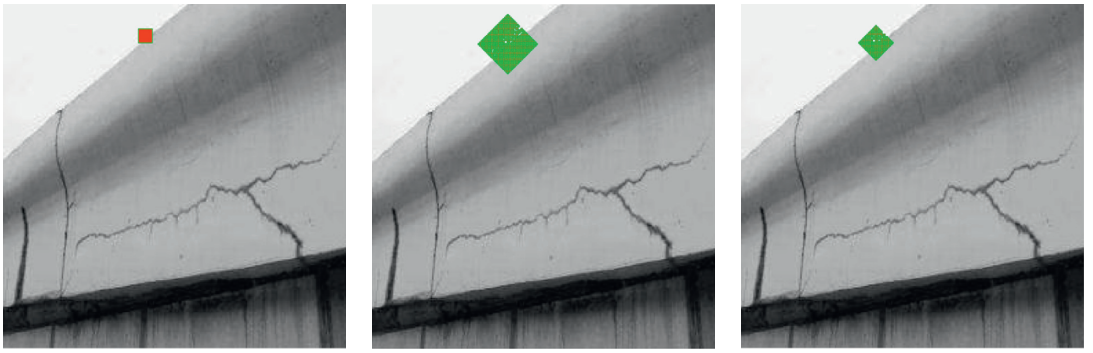
Table 2: The performance of the conditional termination strategy.

Image	Pixel coordinates	Time (s)		Iterations		F_c value	
		Before	After	Before	After	Before	After
Crack 3	(100, 100)	1.09	0.1062	24	13	0.3291	0.4905
Crack 3	(20, 100)	2.77	0.21	22	13	0.6345	0.6433
Crack 23	(100, 115)	1.22	0.0781	23	13	0.4211	0.3470
Crack 5	(20, 100)	2.71	0.2182	22	13	0.6323	0.6477



(a) (b) (c)

Figure 7: Percolation regions of pixel (100, 100) in image crack 3: (a) raw image; (b) without termination; (c) with termination.

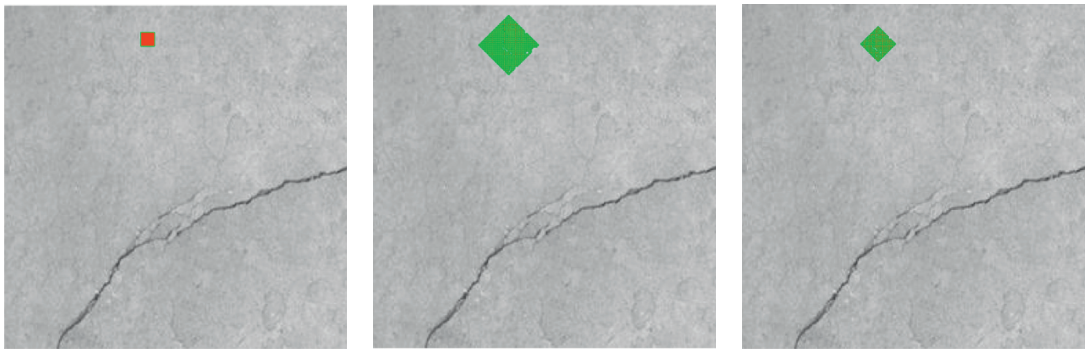


(a) (b) (c)

Figure 8: Percolation regions of pixel (20, 100) in image crack 3: (a) raw image; (b) without termination; (c) with termination.

(a) (b) (c)

Figure 9: Percolation regions of pixel (100, 115) in image crack 23: (a) raw image; (b) without termination; (c) with termination.



(a) (b) (c)

Figure 10: Percolation regions of pixel (20, 100) in image crack 5: (a) raw image; (b) without termination; (c) with termination.

window increases, the time and iterations of the percolating single pixel without the conditional termination increase but stay the same with the termination. This is because before using the conditional termination, the higher the window upper bound is, the more iterations the percolated area D_p need to reach the boundary and thus the more time the percolation takes. However, after improvements, the end of the process does not depend on the window upper bound any more but on whether the shape factor exceeds value T_s. As a result, as long as T_s remains unchanged, the time and iteration number will not change. In Figure 11(b), as the lower bound of the window increases, the time and iterations without the termination remain at a high level but increase with the termination. Once the window lower bound exceeds a certain degree, the time and iterations will be the same as the ones without improvements. This is because the conditional termination strategy is only effective after the percolation area exceeds the lower bound of the window. So, the increase of the lower bound delays the first work of the termination strategy, extending the time and iteration number of the whole process. If the lower bound goes up very close to the upper bound, then the first stage that does not contain the termination procedure will last a long time, and the whole process will end immediately when it enters the second stage where the termination works.

3.1.3. Performance of the Skip Strategy. After percolation towards 22 pixels under the three situations mentioned in Section 2.1.3, the results and their corresponding percolated areas are shown in Figures 12–15. Apart from the speeding-up effect of conditional termination, we can also see the percolation process towards pixels 42, 48, 49, and 50 is skipped directly in Figure 16, where the percolation time and iteration number are nearly zero. It should be noted that both strategies do not work for pixels such as 43 to 47. This is because all of these pixels are among crack areas, which cannot trigger the strategies, but the extra procedures to realize them will cause extra time burden. Fortunately, the time burden can be ignored compared to the huge contributions made by the two strategies. From Figure 17, using different strategies to percolate the same image of different resolution ratios, the performance of combining two strategies is better than using either of them alone. In addition, the clearer the image is, the better the speeding-up effect is. By visualizing the percolated area in each situation, Figures 12–15 demonstrate how the two strategies work. The percolated region of each pixel in Figure 14 is much smaller than the one in Figure 13, which means the termination strategy is triggered. Percolated regions of some pixels in Figure 15 almost disappear, which means the skip procedure works.

3.1.4. Percolation Result of Our Dataset. Following the existing percolation process, cracks are detected using our own dataset, and four images are listed in Figure 18 to show the results. In Figure 19, we can see that most crack pixels are coloured blue, while background pixels are coloured yellow, which means the crack area can be differentiated from the background area by the method. However, there are still two problems to notice. Firstly, in boundary areas, no matter whether the pixels belong to crack or not, they are all coloured blue, as shown in Figures 19(a)–19(d). Secondly, in unclear crack images, many crack pixels are ignored, and many background pixels are mistaken for crack pixels as shown in Figures 19(a)–19(d). Such problems will be discussed and solved in Section 3.2.

3.2. Performance of Improved Techniques

3.2.1. Image Expansion. To eliminate the boundary effect of the existing percolation method shown in Figures 19 and 20, three kinds of image expansion techniques are used, and the

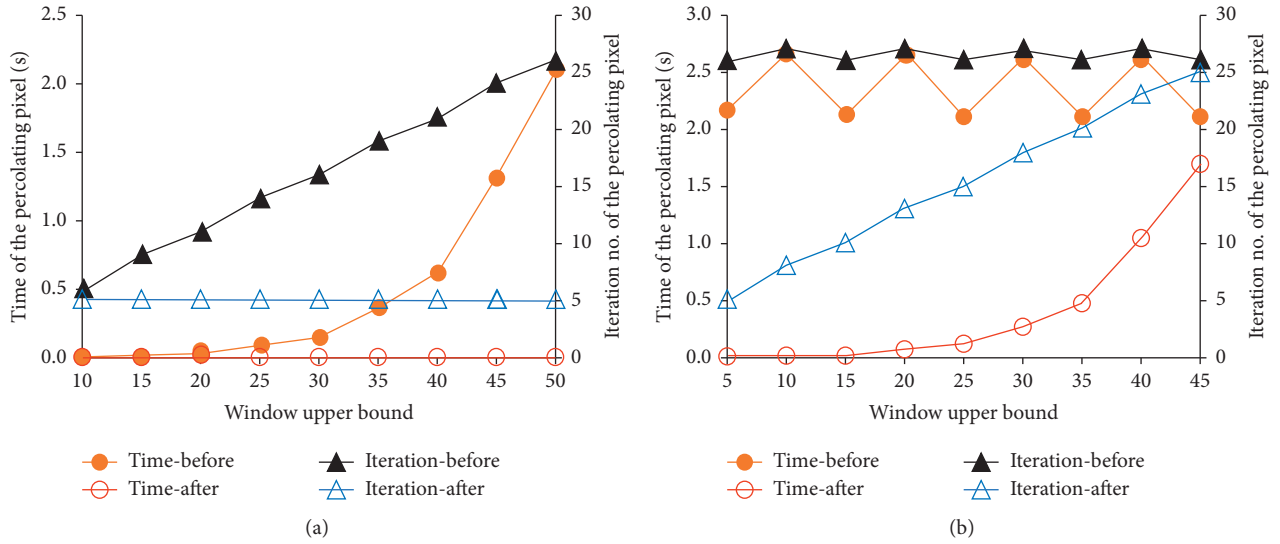
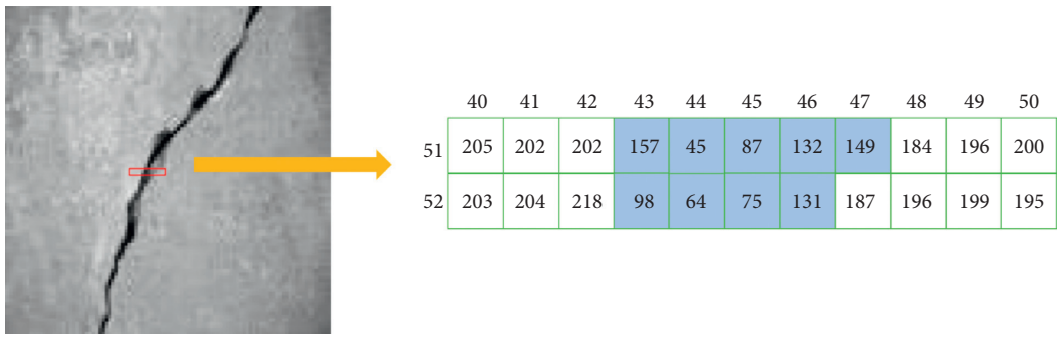


Figure 11: The effect of changing parameters on time and iterations of percolation on the single pixel: (a) window upper bound; (b) window lower bound.



	40	41	42	43	44	45	46	47	48	49	50
51	205	202	202	157	45	87	132	149	184	196	200
52	203	204	218	98	64	75	131	187	196	199	195

Figure 12: Percolating area we focus on.

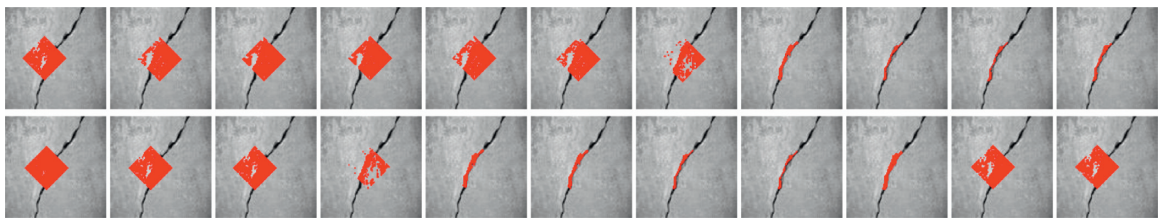


Figure 13: Percolated results using no strategies.

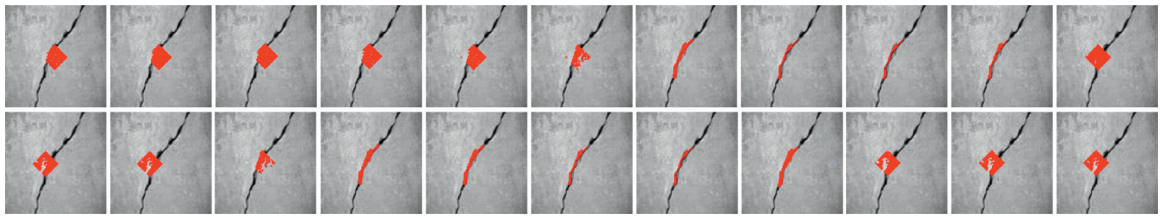


Figure 14: Percolated results using the conditional termination strategy.

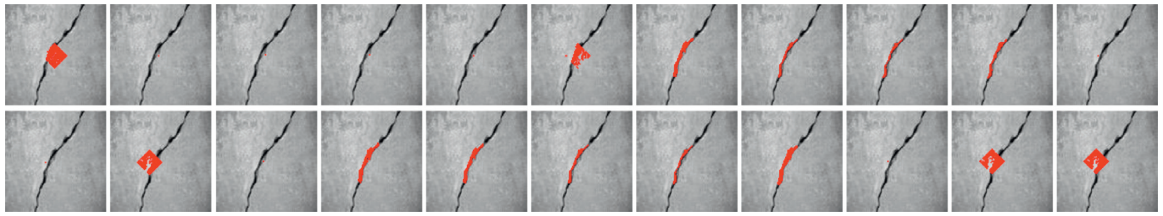


Figure 15: Percolated results using conditional termination and skip strategies.

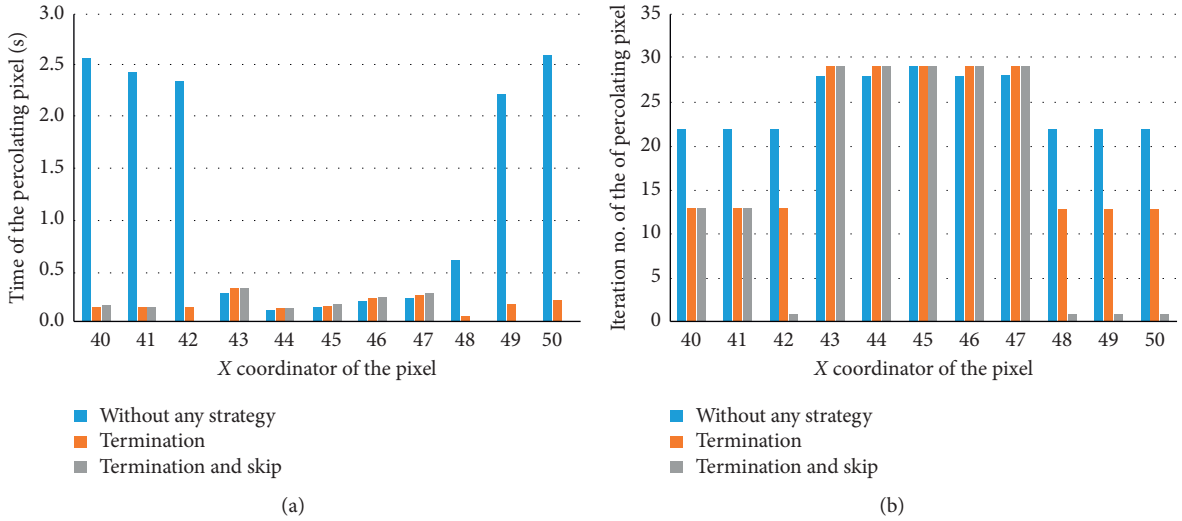


Figure 16: Results of the percolating single pixel using different speeding-up strategies: (a) percolation time; (b) percolation iterations.

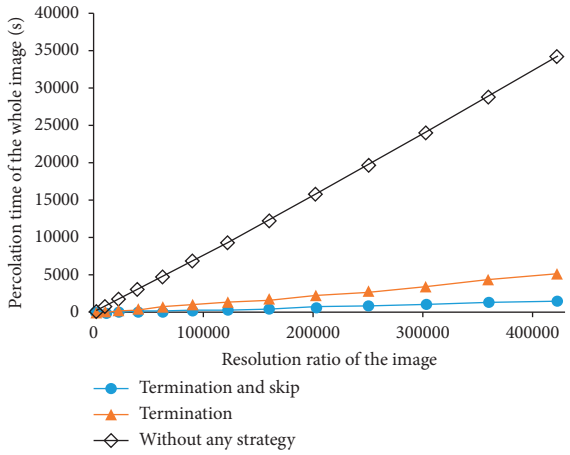
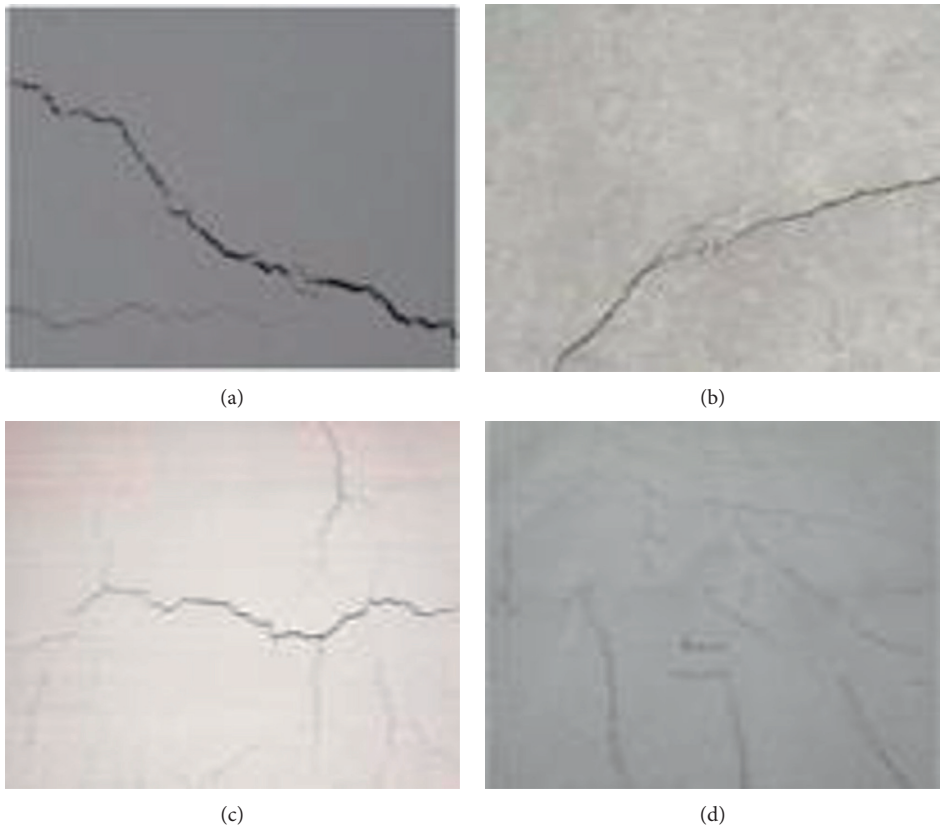


Figure 17: Time to percolate the whole image with different resolution ratios using different strategies.

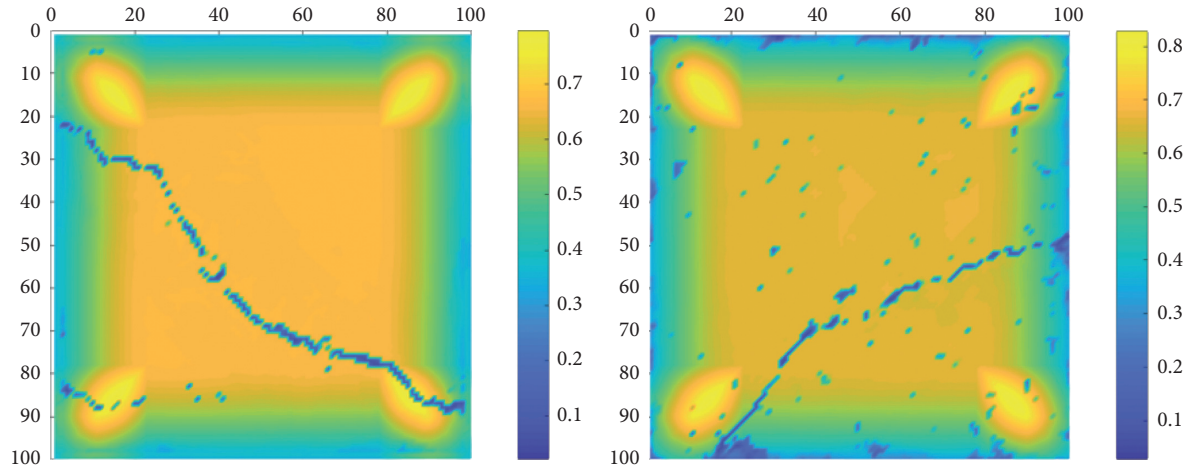
percolation results of crack images in Figure 21 are given as follows. In Figure 22, the boundary effect is lessened, but there are still a few tracks of boundary areas which remained. In Figure 23, the boundary effect on percolation towards crack 2 is removed but remains on the horizontal side of crack 1. In Figure 24, the boundary effect is eliminated in



(a) (b)

(c) (d)

Figure 18: Four crack images to be percolated: (a) image 1; (b) image 2; (c) image 3; (d) image 4.



(a) (b)

Figure 19: Continued.

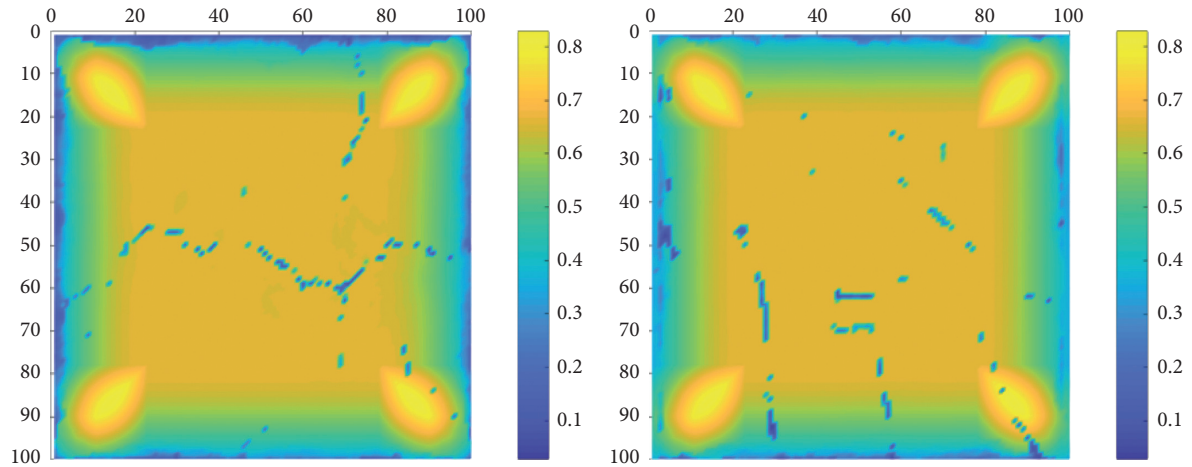


Figure 19: Percolation result of 100 resolution images using the existing percolation method: (a) image 1; (b) image 2; (c) image 3; (d) image 4.

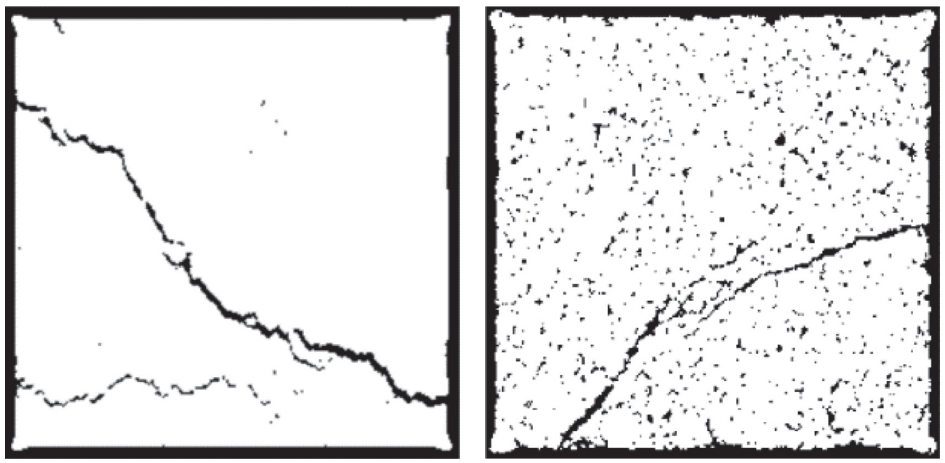


Figure 20: Percolation results without any image expansion techniques: (a) crack 1; (b) crack 2.

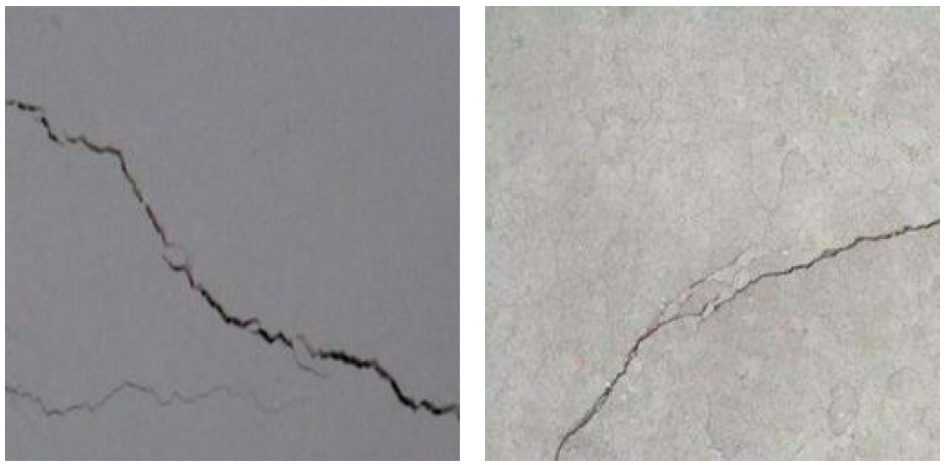
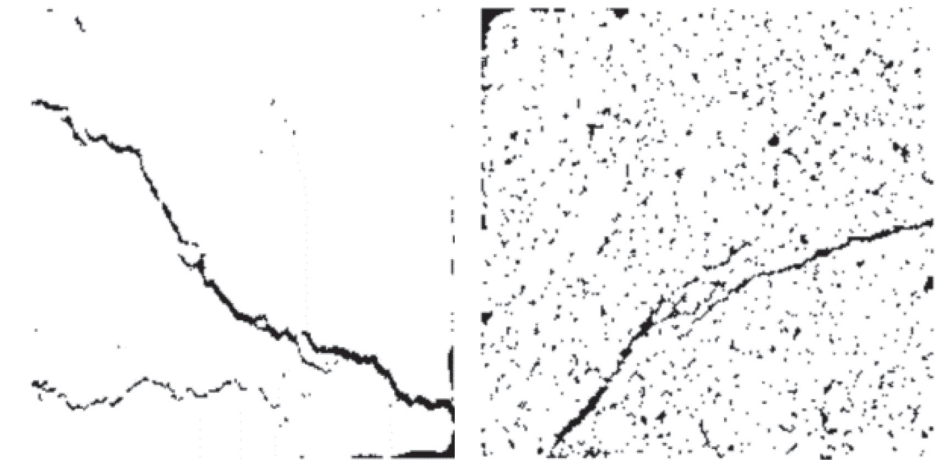
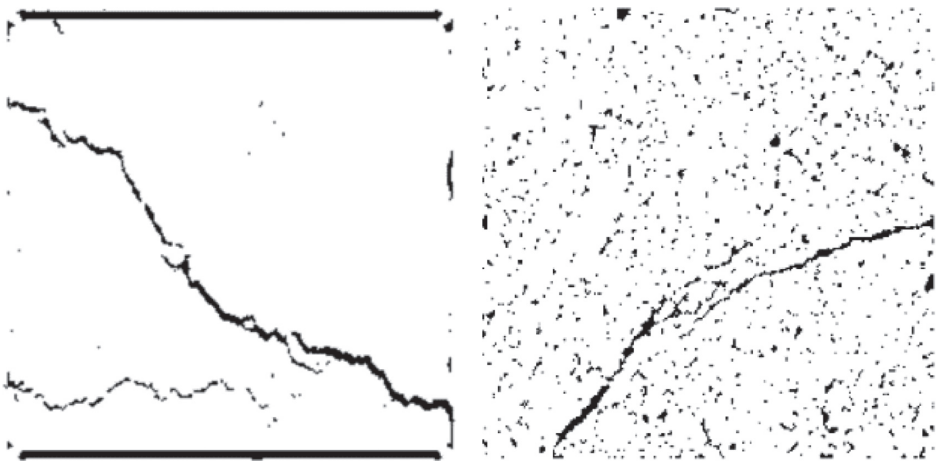


Figure 21: Original images: (a) crack 1; (b) crack 2.



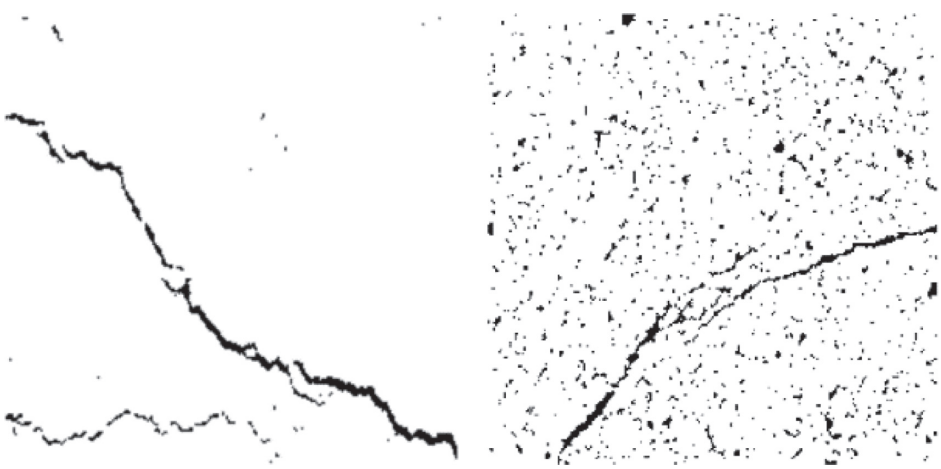
(a) (b)

Figure 22: Percolation results using the mean-padding technique: (a) crack 1; (b) crack 2.



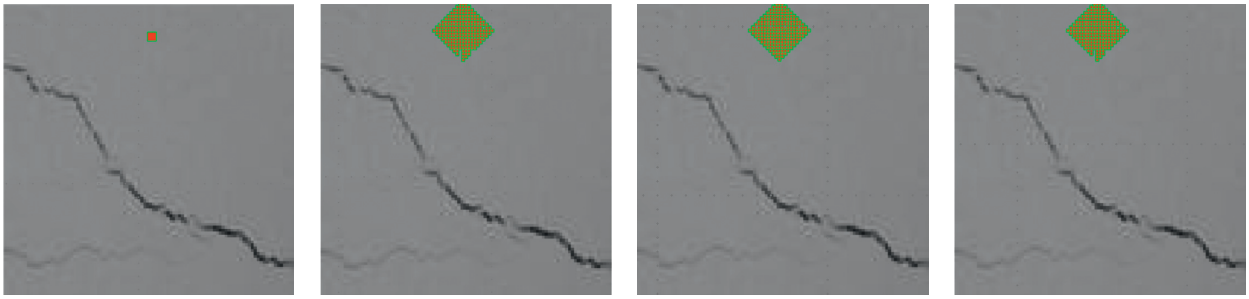
(a) (b)

Figure 23: Percolation results using the same-padding technique: (a) crack 1; (b) crack 2.



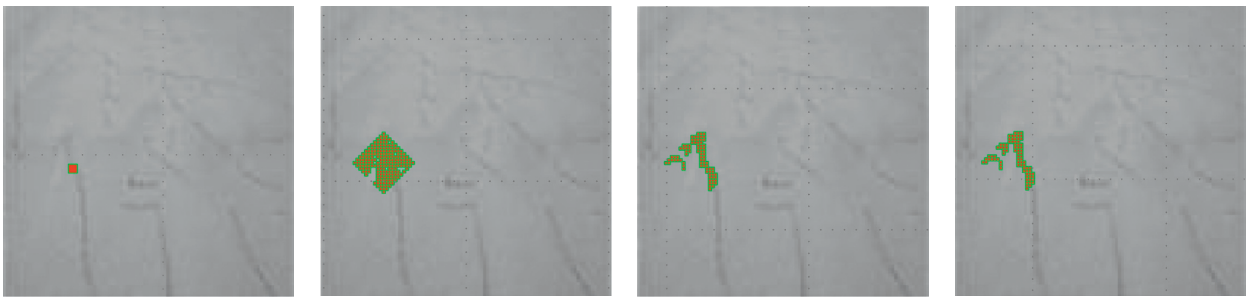
(a) (b)

Figure 24: Percolation results using the zero-padding technique: (a) crack 1; (b) crack 2.



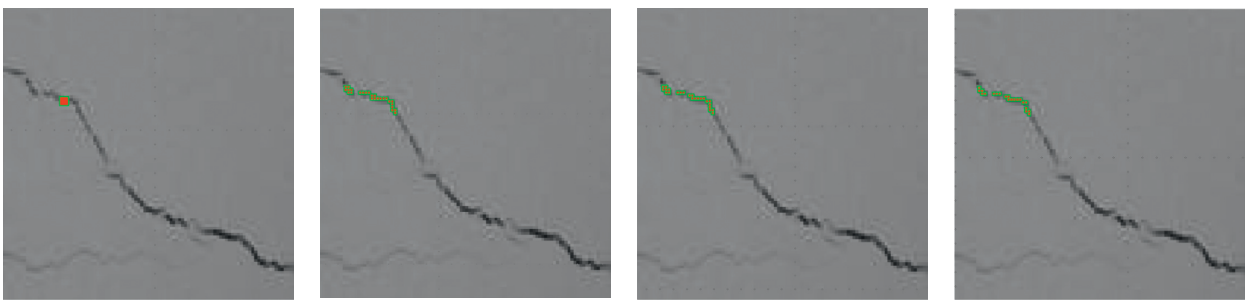
(a) (b) (c) (d)

Figure 25: Percolation result of a background pixel using different striding strategies: (a) raw image; (b) fixed stride percolation; (c) linear stride percolation; (d) quadratic stride percolation.



(a) (b) (c) (d)

Figure 26: Percolation result of an unclear crack pixel using different striding strategies: (a) raw image; (b) fixed stride percolation; (c) linear stride percolation; (d) quadratic stride percolation.



(a) (b) (c) (d)

Figure 27: Percolation result of a clear crack pixel using different striding strategies: (a) raw image; (b) fixed stride percolation; (c) linear stride percolation; (d) quadratic stride percolation.

both directions of both images. Therefore, zero padding is chosen as the best image expansion technique and used as the preprocessor of percolation. The boundary effect is due to the near-linear shape of the percolation area initiated at the boundary of the image. If we pad zero pixels outsize the original image, the percolation area of the boundary pixel

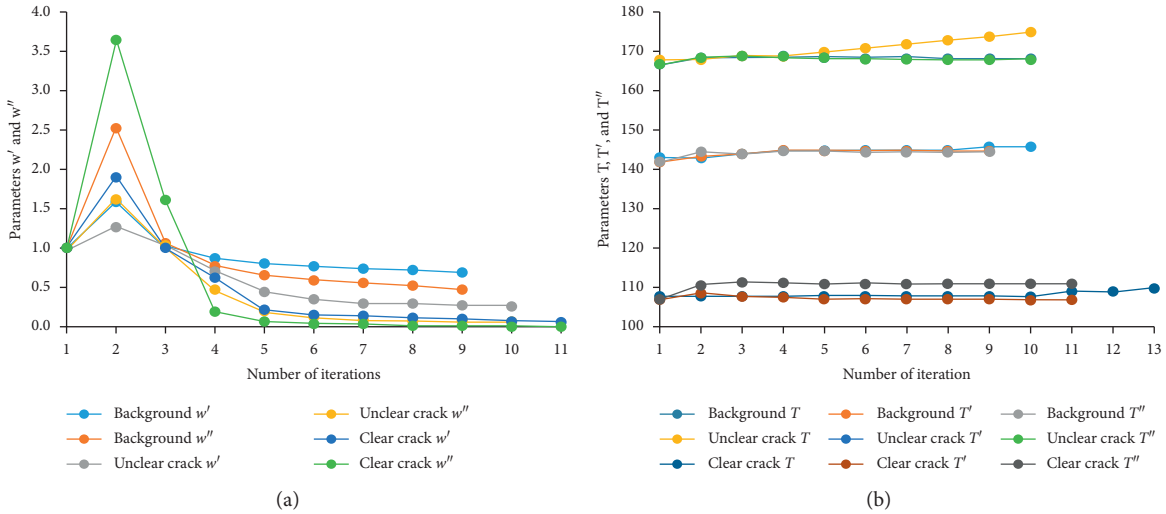


Figure 28: The effectiveness of the slacken stride strategies: (a) alteration of the stride parameter; (b) alteration of the threshold value.

grows into the circular shape rather than the linear shape, which therefore is deemed as the background pixel. One obvious advantage of the zero-padding technique over mean-padding and the same-padding techniques is that it eliminates the boundary effects without adding other potential noises. However, the mean padding and the same padding will impose pixels with nonzero values outside the image. The difference between the nonzero pixels and the zero pixels lying on the original boundary forms the percolated area to be the linear shape, which is out of our expectation.

3.2.2. Slacken Stride. The percolation areas of the three pixels using constant, linear, and quadratic strides are shown in Figures 25–27. In Figures 25 and 27, it is reasonable to see the percolation areas are in circular and linear shapes. In Figure 26(b), the percolation area, which should have been linear, is circular, and thus, the corresponding pixel is mistaken for the background pixel, illustrating the problem of using the constant stride. In Figure 26(c) and 26(d), the linear shape of the percolation area matches our expectation and validates the effect of the slacken stride strategy.

Figure 28 shows the alterations of parameters w, w', and w'' and threshold values T, T', and T''. From Figure 28(a), the stride values w' and w'' in the unclear and clear crack case increase at first and then decrease close to 0. Both of them decrease less than 1 after a number of iterations and become near 0 eventually, whereas in the background case, the stride values w' and w'' are not close to 0, which reflects that the percolation shape is near-circular. From Figure 28(b), the improved method can be further verified through the alteration of the threshold value T. In the situations of clear and unclear crack, T' and T'' stay almost unchanged, while T gradually increases in all situations. Therefore, the unclear cracks are mistaken for background in the fixed stride case but are still regarded as cracks in another two cases. Note that the entire stride values increase during the first iteration, which seems unusual. It is because during the first iteration, the square value of the diameter of the percolated area is far smaller than the number of pixels among it, making the F_c value bigger than 1 and causing such unusual increase.

3.3. Performance of the Final Improved Percolation Model

3.3.1. Performance Evaluation. The precision, recall, F-1 score, and time of percolating each image are presented in Figures 29–33. We can see that, in linear and quadratic cases, precisions are almost up to 90%, recalls are between 50% and 90%, and F-1 scores are up to 80%, while in the cubic case, all of three indexes are lower. Besides, in Figure 32, the percolation time using linear and quadratic strides is on average smaller than using cubic strides. Furthermore, we define a parameter T* by equation (6), which represents the relative percolation time of each stride. In equation (6), the numerator is the average value of the percolation time over 50 images, and the denominator is the maximum of the average value among three strategies. By computing the average values of the first 3 indexes over the 50 images and the relative percolation

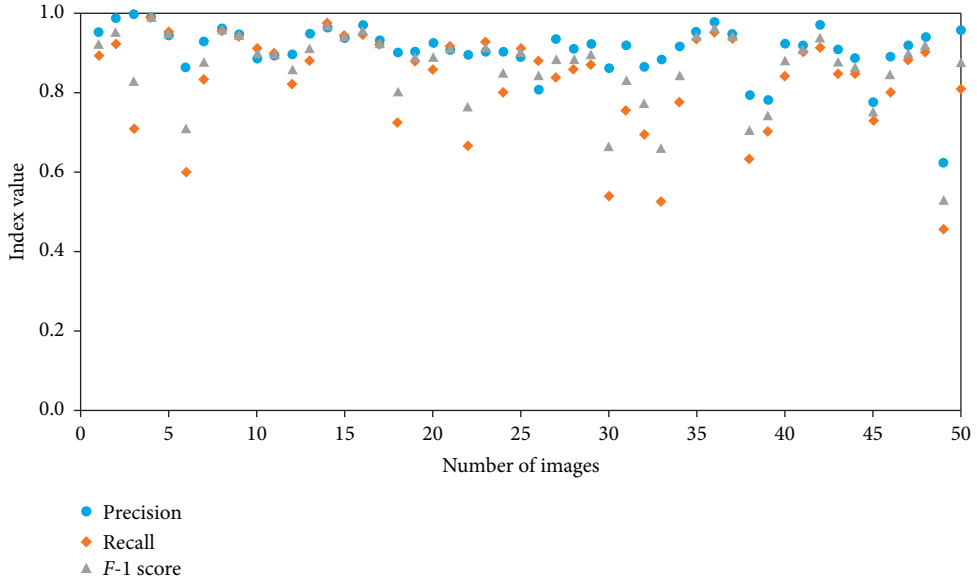


Figure 29: Three evaluation indexes of 50 images from the 200-dataset using the linear stride.

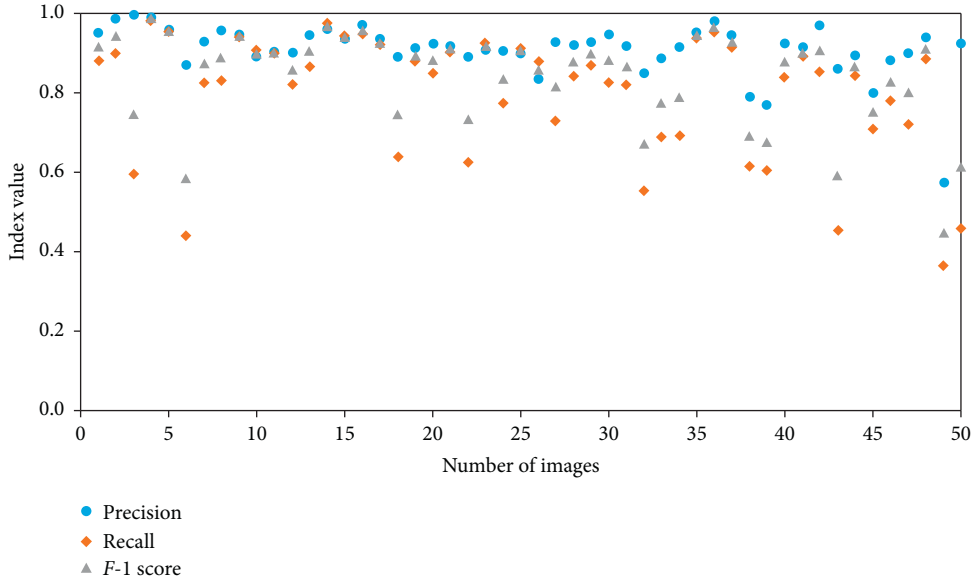


Figure 30: Three evaluation indexes of 50 images from the 200-dataset using the quadratic stride.

time, a comparison result between the three stride strategies is given in Figure 33. The linear and quadratic strides are better than the cubic one.

$$T^*_{\text{type}} = \frac{1/50 \sum_{i=1}^{50} t^i_{\text{type}}}{\max\left(1/50 \sum_{i=1}^{50} t^i_{\text{type}}\right)}, \text{type} = \text{linear, quadratic, and cubic.} \tag{6}$$

In addition, their performances are further compared with 5 other classic crack detection methods, and the results are shown in Table 3. We can see that both linear and quadratic percolation methods have a higher precision but a lower recall than CrackTree [41]. Moreover, their F-1 scores are roughly the same, which means our enhanced percolation methods, in some level, achieve a better performance than the classical method.

3.3.2. Real Test Result. Due to space limitation, here we just present percolation results of 27 images using linear, quadratic, and cubic stride strategies. The results of our whole 200-dataset are uploaded and can be seen in the

Figure 31: Three evaluation indexes of 50 images from the 200-dataset using the cubic stride.

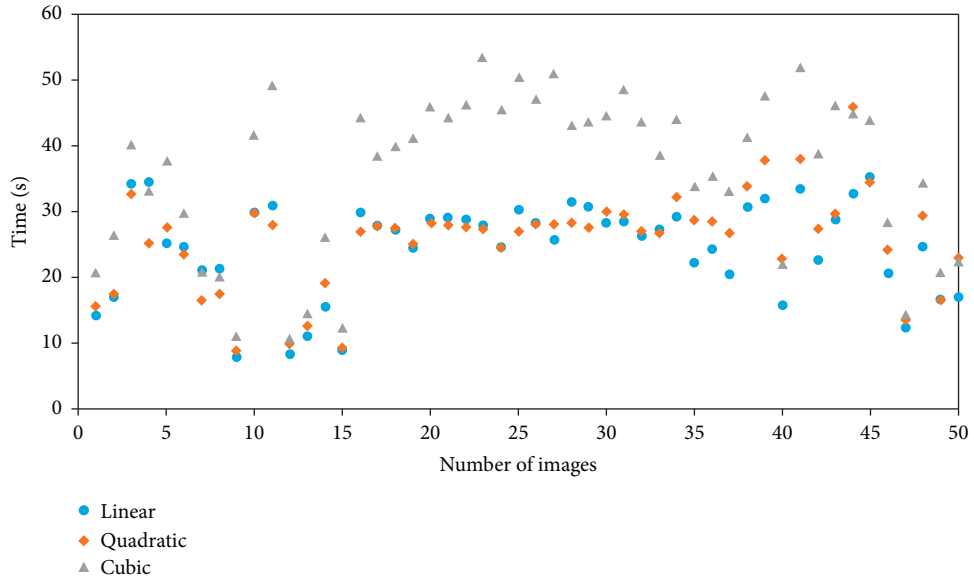


Figure 32: Time to percolate 50 images from the 200-dataset using three types of strides.

same link of our 200-dataset. In Figures 34–36, each row lists several images to be percolated, and their corresponding percolation results are listed in the following row. We can see that almost all cracks are detected and localized successfully in linear and quadratic cases. However, there are also some problems to be noticed. In the cubic situation, many unclear cracks are ignored and treated as backgrounds. When the raw image contains noise or some other objects with similar properties to cracks, the method will mistake them for cracks and mark them too in the results. Besides, for those extremely unclear cracks, even though the method can detect them, they cannot be localized precisely. Such problems will be discussed further in Section 4.

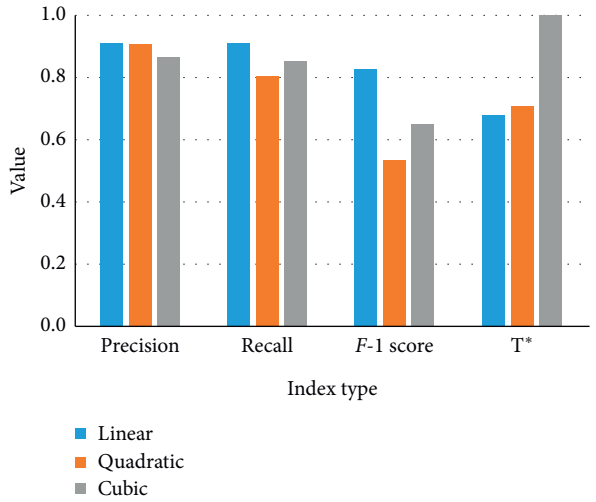


Figure 33: The percolation time of images using the 200-dataset with different stride values.

Table 3: Comparison of the crack detection performance between eight different methods [41].

Method	pbCGTG	gpb	pbCanny	Seg-ext	CrackTree	Linear	Quadratic	Cubic
Precision	0.34	0.36	0.30	0.57	0.79	**0.91**	**0.91**	0.83
Recall	0.36	0.49	0.21	0.63	0.92	**0.83**	**0.81**	0.53
F-1 score	0.35	0.41	0.25	0.59	0.85	**0.87**	**0.85**	0.65

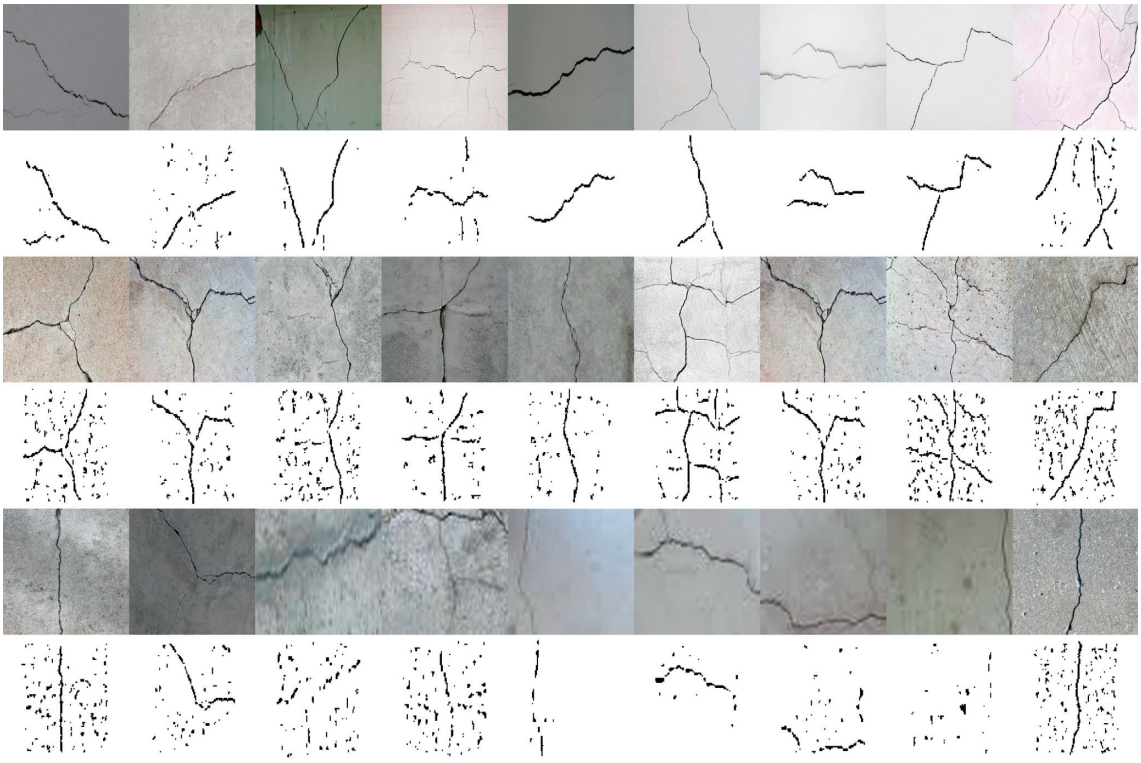


Figure 34: Percolation results using the enhanced percolation method with the linear stride.

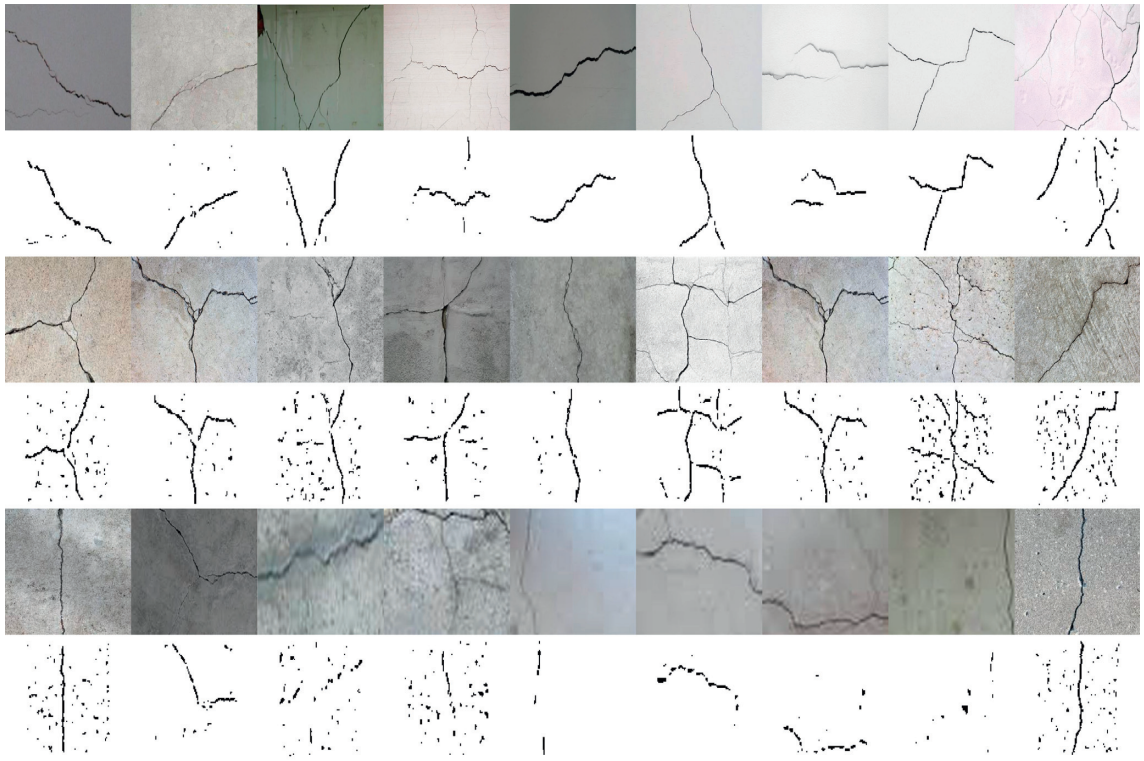


Figure 35: Percolation results using the enhanced percolation method with the quadratic stride.

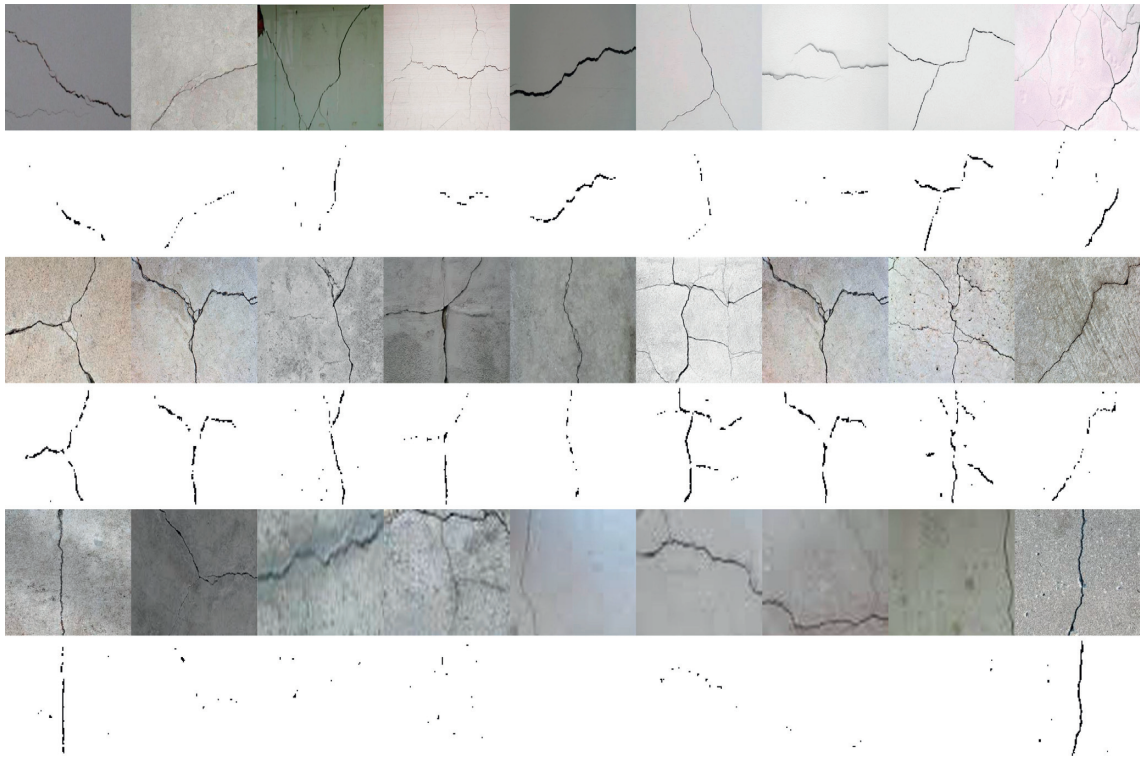


Figure 36: Percolation results using the enhanced percolation method with the cubic stride.

4. Conclusions

In this paper, an existing percolation method is presented, and three experiments are performed to illustrate its three limitations: boundary effects, unclear cracks, and eight neighbouring pixels' region. Based on these three limitations, we devise the enhanced percolation method by adding three improvements, which are image expansion, slacken stride, and four neighbouring pixels' region, respectively. The effect of each technique is demonstrated by the corresponding experiment. The enhanced percolation method is applied on the 200-dataset, and we compute the precision, recall, and F-1 for measuring the performance of the proposed method. By comparing with the classical crack detection methods, percolation using linear and quadratic strides performs better and can be used to detect concrete bridge cracks. To apply these two methods to detect cracks in real life, we still need to be careful about problems such as the light condition when shooting, the camera position, and angle. Dark light will lessen the difference of colour between cracks and backgrounds, and wrong-positioned cameras will make cracks lose their linear shape, both of which will cause detection errors. However, since the primary purpose of this paper is to propose an efficient method to detect concrete bridge cracks, solutions to these practical problems are omitted here.

Even though the method has gained good performance, there are still several limitations to pay attention to, which will be the focus of our future work. Firstly, for damages owning similar colour and linear shape as cracks, the enhanced method cannot differentiate them very well. The solution to this is to turn to the convolutional neural network for help. We collect photos of different types of bridge damages, split them into different datasets, and train the network to classify those photos into different damages. The second limitation is that we just clarify how to detect cracks from bridge photos. However, for how to get those photos and what qualities, such as the camera position and light condition, we require for those photos if they want to be detected correctly, are not specified. Both of these limitations will be discussed further in our future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (51778194), the China Postdoctoral Science Foundation (2017M621282), and the Fundamental Research Funds for the Central Universities (HIT. NSRIF. 2019056).

References

[1] K. Golabi and R. Shepard, "Pontis: a system for maintenance optimization and improvement of US bridge networks," *Interfaces*, vol. 27, no. 1, pp. 71–88, 1997.

[2] S. Zhang and Y. Liu, "Damage detection of bridges monitored within one cluster based on the residual between the cumulative distribution functions of strain monitoring data," *Structural Health Monitoring*, vol. 19, no. 6, p. 1764, 2020.

[3] Q. Gao, Z. Dong, K. Cui, C. Liu, and Y. Liu, "Fatigue performance of profiled steel sheeting-concrete bridge decks subjected to vehicular loads," *Engineering Structures*, vol. 213, Article ID 110558, 2020.

[4] Z. Zhang, X. Liu, Y. Zhang, M. Zhou, and J. Chen, "Time interval of multiple crossings of the Wiener process and a fixed threshold in engineering," *Mechanical Systems and Signal Processing*, vol. 135, Article ID 106389, 2020.

[5] M. A. Saleem, A. Mirmiran, J. Xia, and K. Mackie, "Ultra-high-performance concrete bridge deck reinforced with high-strength steel," *ACI Structural Journal*, vol. 108, pp. 601–609, 2011.

[6] L. Jiang, J. Ye, and H. Zheng, "Collapse mechanism analysis of the FIU pedestrian bridge based on the improved structural vulnerability theory (ISVT)," *Engineering Failure Analysis*, vol. 104, pp. 1064–1075, 2019.

[7] https://www.miamiherald.com/news/local/community/miami-dade/article231428938.html.

[8] A. Mital, M. Govindaraju, and B. Subramani, "A comparison between manual and hybrid methods in parts inspection," *Integrated Manufacturing Systems*, vol. 9, no. 6, pp. 344–349, 1998.

[9] P. Rose, B. Aaron, D. E. Tamir, L. Lu, J. Hu, and H. Shi, *Supervised Computer-Vision-Based Sensing of Concrete Bridges for Crack-Detection and Assessment*, TRB 93rd Annual Meeting Compendium of Papers, Washington DC, USA, 2014.

[10] J.-K. Oh, G. Jang, S. Oh et al., "Bridge inspection robot system with machine vision," *Automation in Construction*, vol. 18, no. 7, pp. 929–941, 2009.

[11] T. Yamaguchi, S. Nakamura, R. Saegusa, and S. Hashimoto, "Image-based crack detection for real concrete surfaces," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 3, no. 1, pp. 128–135, 2008.

[12] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure," *Advanced Engineering Informatics*, vol. 29, no. 2, pp. 196–210, 2015.

[14] A. Mohan and S. Poobal, "Crack detection using image processing: a critical review and analysis," *Alexandria Engineering Journal*, vol. 57, no. 2, pp. 787–798, 2018.

[15] M. R. Jahanshahi, J. S. Kelly, S. F. Masri, and G. S. Sukhatme, "A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures," *Structure and Infrastructure Engineering*, vol. 5, no. 6, pp. 455–486, 2009.

[16] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.

[17] A. Ayenu-Prah and N. Attoh-Okine, "Evaluating pavement cracks with bidimensional empirical mode decomposition," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, Article ID 861701, 7 pages, 2008.

[18] S. K. Sinha and P. W. Fieguth, "Automated detection of cracks in buried concrete pipe images," *Automation in Construction*, vol. 15, no. 1, pp. 58–72, 2006.

[19] G. Li, S. He, and Y. Ju, "Image-based method for concrete bridge crack detection," *Journal of Information and Computational Science*, vol. 10, no. 8, pp. 2229–2236, 2013.

[20] R. S. Lim, H. M. La, 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, 2014.

[21] I. Sobel, *History and definition of the sobel operator*, SCRIBD, San Francisco, CA, USA, 2014.

[22] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[23] Q. Pan, L. Zhang, G. Dai, and H. Zhang, "Two denoising methods by wavelet transform," *IEEE Transactions on Signal Processing*, vol. 47, no. 12, pp. 3401–3406, 1999.

[24] T. Yamaguchi and S. Hashimoto, "Automated crack detection for concrete surface image using percolation model and edge information," in *Proceedings of the IECON 2006—32nd Annual Conference on IEEE Industrial Electronics*, pp. 3355–3360, Paris, France, November 2006.

[25] T. Yamaguchi, S. Nakamura, and S. Hashimoto, "An efficient crack detection method using percolation-based image processing," in *Proceedings of the IEEE Conference on Industrial Electronics and Applications*, pp. 1875–1880, Singapore, Singapore, June 2008.

[26] T. Yamaguchi and S. Hashimoto, "Improved percolation-based method for crack detection in concrete surface images," in *Proceedings of the 2008 19th International Conference on Pattern Recognition*, pp. 1–4, Tampa, FL, USA, December 2008.

[27] T. Yamaguchi and S. Hashimoto, "Fast crack detection method for large-size concrete surface images using percolation-based image processing," *Machine Vision and Applications*, vol. 21, no. 5, pp. 797–809, 2010.

[28] Z. Qu, L.-D. Lin, Y. Guo, and N. Wang, "An improved algorithm for image crack detection based on percolation model," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 10, no. 2, pp. 214–221, 2015.

[29] Z. Qu, Y.-X. Chen, L. Liu, Y. Xie, and Q. Zhou, "The algorithm of concrete surface crack detection based on the genetic programming and percolation model," *IEEE Access*, vol. 7, pp. 57592–57603, 2019.

[30] Z. Qu, L. Bai, S. Q. An, F. R. Ju, and L. Liu, "Lining seam elimination algorithm and surface crack detection in concrete tunnel lining," *Journal of Electronic Imaging*, vol. 25, pp. 1–17, 2016.

[31] P. Prasanna, K. Dana, N. Gucunski, and B. Basily, "Computer-vision based crack detection and analysis," *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, 2012.

[32] D. Lattanzi and G. R. Miller, "Robust automated concrete damage detection algorithms for field applications," *Journal of Computing in Civil Engineering*, vol. 28, no. 2, pp. 253–262, 2012.

[33] I. H. Kim, H. Jeon, S. C. Baek, W. H. Hong, and H. J. Jung, "Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle," *Sensors*, vol. 18, pp. 1–14, 2018.

[34] W. R. L. Silva and D. S. Lucena, "Concrete cracks detection based on deep learning image classification," in *Proceedings of the 18th International Conference on Experimental Mechanics*, Brussels, Belgium, June 2018.

[35] L. F. Li, W. F. Ma, L. Li, and C. Lu, "Research on detection algorithm for bridge cracks based on deep learning," *Acta Automatica Sinica*, 2019, in Press.

[36] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction*, vol. 22, pp. 567–576, 2012.

[37] M. Cabaleiro, R. Lindenbergh, W. F. Gard, P. Arias, and J. W. G. Van de Kuilen, "Algorithm for automatic detection and analysis of cracks in timber beams from LiDAR data," *Construction and Building Materials*, vol. 130, pp. 41–53, 2017.

[38] I. Abdel-Qader, S. Pashaie-Rad, O. Abudayyeh, and S. Yehia, "PCA-based algorithm for unsupervised bridge crack detection," *Advances in Engineering Software*, vol. 37, no. 12, pp. 771–778, 2006.

[39] S.-N. Yu, J.-H. Jang, and C.-S. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Automation in Construction*, vol. 16, no. 3, pp. 255–261, 2007.

[40] Q. Li, Q. Zou, D. Zhang, and Q. Mao, "FoSA: F* Seed-growing Approach for crack-line detection from pavement images," *Image and Vision Computing*, vol. 29, no. 12, pp. 861–872, 2011.

[41] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.

[42] M. V. Madurwar, R. V. Ralegaonkar, and S. A. Mandavgane, "Application of agro-waste for sustainable construction materials: a review," *Construction and Building Materials*, vol. 38, pp. 872–878, 2013.

[43] Kobetičová, Klára, and R. Černý, "Ecotoxicology of building materials: a critical review of recent studies," *Journal of Cleaner Production*, vol. 165, pp. 500–508, 2017.

[44] C. Alberts and Y. Hjalmar, "Method for sealing cracks and cavities in different kinds of building constructions, such as building constructions in rock, concrete, brickwork and timber," US Patent No. 4,086,309, 1978.

[45] J. K. Brimacombe and K. Sorimachi, "Crack formation in the continuous casting of steel," *Metallursgical Transactions B*, vol. 8, no. 2, pp. 489–505, 1977.