A Thesis

Entitled

Automated Pavement Distress Detection Using Advanced Image Processing Techniques

By

Yao Sun

Submitted as partial fulfillment of the requirements for

The Master of Science Degree in Engineering

_____

Advisor: Dr. Ezzatollah Salari

_____

College of Graduate Studies

The University of Toledo

December 2009

The University of Toledo

College of Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Yao Sun

ENTITLED Automated Pavement Distress Detection Using Advanced Image Processing Techniques

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering

_____

Thesis Advisor: Dr. Ezzatollah Salari

Recommendation concurred by

Committee

_____

Dr. Eddie Chou

On

_____

Dr. Mohsin Jamali

Final Examination

_____

Dean, College of Engineering

An Abstract of

Automated Pavement Distress Detection Using Advanced Image Processing Techniques

by

Yao Sun

Submitted as partial fulfillment of the requirements for

The Master Science Degree in Engineering

The University of Toledo

December 2009

In this thesis, a novel, fast and self-adaptive image processing method is proposed for the extraction and connection of break points of cracks in pavement images. The algorithm first finds the initial point of a crack and then determines the crack's classification into transverse and longitudinal types. Different search algorithms are used for different types of cracks. Then the algorithm traces along the crack pixels to find the break point and then connect the identified crack point to the nearest break point in the particular search area. The nearest point then becomes the new initial point and the algorithm continues the process until reaching the end of the crack. The

experimental results show that this connection algorithm is very effective in maximizing the accuracy of crack identification.

Acknowledgements

Firstly, I am heartily thankful to my advisor Dr. Ezzatollah Salari, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. His comments on chapter drafts are themselves a course in critical thought upon which I will always draw. His capacity to combine critique with an immediate empathy and commitment towards students will always inspire me.

I am also indebted to Professor Eddie Yein Juin Chou whose work inspired me to examine research adaptation. He firm belief that a better way can always be found has influenced my approach to this research. His advice on interpreting the case study results on an earlier draft directly contributed to this study. I cannot thank him enough.

Many thanks to Nikhil Katakam whose editing suggestions and precise sense of language contributed to the final copy. He provided decisive and energetic support during the write-up stage, clearing the path towards thesis completion in his solution-oriented way.

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter presents a concise introduction about the development of pavement crack detection. The outline of this thesis is also given in this chapter.

1.1 Background

Road networks play a vital part of our world nowadays because it makes an important contribution to society. People cannot do their business and activities easily without good road networks. Unfortunately, pavement systems deteriorate over time primarily due to fatigue. This deterioration to pavement increases with the fourth power of the axle load of the vehicles traveling on it [1]. Technically, early pavement deterioration contains four different types of crack: transverse crack, longitudinal crack, block crack and alligator crack. Potholes are formed making the road becomes more dangerous if these early deteriorations are left untreated. Rehabilitation treatments, such as fixing potholes will cost about 10 to 20 times more than the cost of resealing cracks [2]. Therefore, pavement detection and rating are so important to keep the cost of fixing the road deterioration low and keep the road networks in good condition.

The United States Department of Transportation (DOT) is a federal Cabinet department of the United States government who takes care of transportation systems.

Its mission is to “Serve the United States by ensuring a fast, safe, efficient, accessible and convenient transportation system that meets our vital national interests and enhances the quality of life of the American people, today and into the future.”[3] DOT needs to hire lots of people to inspect road networks in order to know which roads should be repaired. Usually, people watch the pavement with their eyes and rate them based on samples and experience. So we can clearly find the shortcoming of this method. First, it is very dangerous for people to watch cracks on the roads and takes a lot of time to watch and evaluate pavements. Also the labor cost can be very high. Finally, there could be a large differences between the actual condition and evaluation results because of the subjectivity of the evaluation process.

Currently, several companies offer solutions to the problem of monitoring road surface conditions. The solutions include CSIRO’s road crack detection vehicle, the PAVUE system by OPQ systems, Roadware’s Wisecrax crack detection system and Piccrack. With the help of the Roads and Traffic Authority of New South Wales, Australia the CSIRO produced a system called the RoadCrack system. The CSIRO’s road fault detection system uses a special modified van, with a series of scan line cameras under a skirt in the middle of the vehicle, to acquire road surface footage. The cameras are mounted perpendicular to the surface of the road, and other devices which create a series of lights provide constant illumination to the road surface. Roadware Incorporate has developed the WiseCrax system. High speed cameras are put on the back of a van

that can go up to 50 MPH. Video is recorded as a continuous series of non-overlapping, high contrast images 4.9 ft by 13 ft [4]. The cameras have the synchronized strobe lights to eliminate shadows from trees, buildings or any other overhead objects even in bright sunlight. Images are processed off-line overnight at the office workstation by a unique open architecture process using advanced image recognition software. Thus, WiseCrax helps to remove the subjectivity and drudgery from pavement evaluation and ensures more accurate, repeatable comparisons of road deterioration from year to year [4]. Piccrack is Samsung's pavement crack detection system which is similar to Roadware's WiseCrax crack system. It also processes the pavement images offline. The images are collected by the data collection subsystem. It can automatically measure pavement crack type, extent, and severity [5]. The PAVUE system is a new system that has been developed in Sweden and is used to detect road surface faults. The system uses a series of four cameras with the synchronised flashing lights to capture the road surface data. Reports are then generated offline with the road surface data [6]. Unfortunately, because of the commercial nature information on these systems is limited and few published papers and documents could be found. Figure 1-1 shows one of the WiseCrax system carriers Automatic Road Analyzer.

Figure 1-1 Roadware's Automatic Road Analyzer (ARAN)

1.2 Outline of Thesis

The demand for automated inspection, monitoring, and pattern recognition for transportation applications are increasing. This increasing demand is partly driven by the decreasing costs of imaging technologies. A typical inspection process, including pavement distress inspection, can be divided into three stages: preprocessing, segmentation, and classification and measurements. Preprocessing is used to improve the quality of the input image in order to facilitate the analysis and interpretation at subsequent stages. Important tasks in preprocessing can include filtering for noise removal, deblurring the image, and the highlighting of specific features, e.g., cracks on the pavement. Image segmentation is the process of dividing an image into meaningful

regions, such as objects of interest and background. The main parameters concerning pavement management are the pattern classification and measurement of various parameters from crack features. In this thesis, advanced image processing techniques are used to deal with pavement pictures. Although this method can obtain no more pavement parameters than the commercial systems, it will reduce the hardware cost.

In this thesis, several methods are used to detect cracks in the pavement image, for example, enhancement, threshold, dilation, erosion, and connection method as shown in Figure 1-2. For every given image, enhancement improves the contrast, making the conversion from grayscale image to binary image easier through thresholding method. Morphology methods help to make the cracks look better. Noise removal is a very common step for image processing. The break points connectively algorithm is the main part of this thesis. Image skeletonization is a step for rating the cracks.

The rest of this thesis is organized as follows: Chapter 2 is the literature review. Chapter 3 proposes crack image segmentation using fractal theory. Preprocessing, including closing operation and noise removal, is explained in Chapter 4. In Chapter 5, the break points connectivity algorithm is introduced. Chapter 6 presents several computer simulation results and analysis. Finally, conclusions are given in Chapter 7.

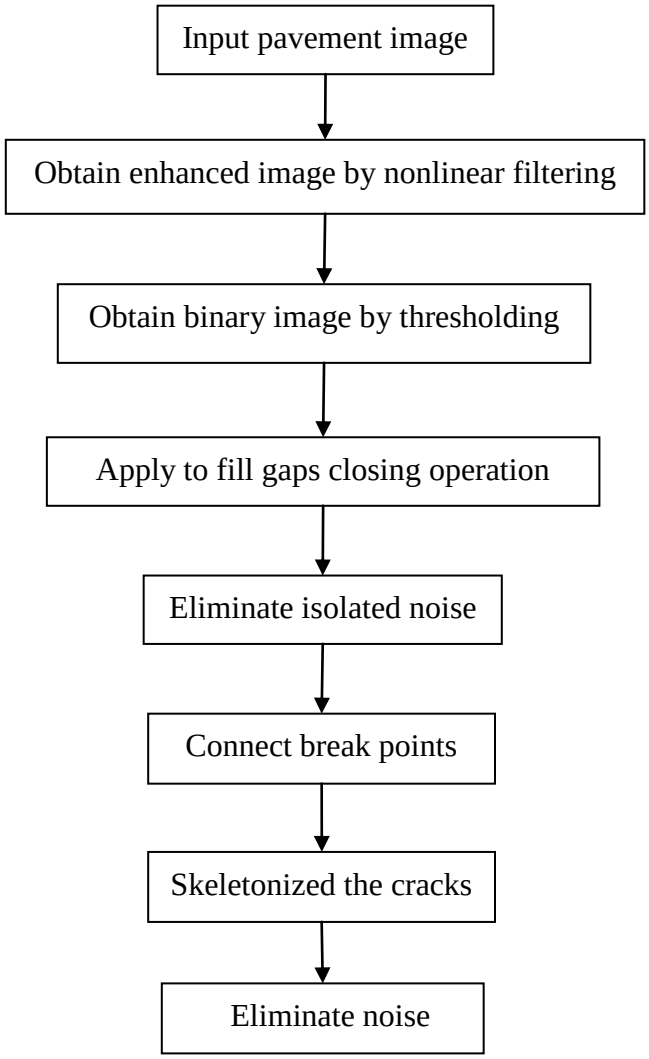


Figure 1-2 Processing steps

Chapter 2

Literature Review

This chapter introduces several related works about road inspection done by other people. These people proposed many aspects of road inspection by using different techniques. They make an effort to find a fast, objective, and relatively inexpensive automated road inspection method. It is hard to finish this thesis without their documents. Mature production of road inspection is developed by some companies. However, few published papers and documents about their production could be found.

Li, Chan and Lytton [7] proposed a method for detecting thin cracks on noisy pavement images. The mean width of a thin crack is less than 0.25 inch. This method is described below: First, the edges of a grey level image are extracted with the Sobel edge operators in the image statistics acquisition. The method will calculate the threshold level. After the pavement image is segmented into a binary image, this method will eliminate noisy spots, scan the crack segment, trace the boundary of the crack segment and determine the orientation of the segment. Finally, the length and width of the crack are calculated. The threshold T is calculated by the grey level of each pixel multiplied by its gradient and weighted with the corresponding gradient.

Some researchers have used wavelet transform methods as a crack detection tool.

The advantage of the wavelet transform is its multi-resolution property, which allows efficient identification of local features of the signal [8]. The wavelet transform has been successfully applied for crack localization in beam structures [9, 10]. The Lipschitz exponent is used to estimate the size of the crack [11]. Douka, Loutridis and Trochidis [12] proposed a method for estimating both the location and size of the crack by defining an intensity factor which relates the size of the crack to the coefficients of the wavelet transform. Although cracks in beam structures are different from those in pavement, we can use these methods as a source of reference in pavement crack detection.

Leontios, Douka and Athanasios [13] proposed an algorithm for crack detection called the Kurtosis Crack Detector (KCD). They analyzed the fundamental vibration mode of a cracked cantilever beam and estimated both the location and size of the crack. The location of the crack is determined by the abrupt change in the spatial variation of the analyzed response, while the size of the crack is related to the kurtosis estimate. The proposed technique forms a Kurtosis-based crack detector, which takes into account the non-Gaussianity of the vibration signal in order to efficiently detect both the location and the size of the crack. They found that the proposed technique is more robust against noise or measurement errors compared to other techniques such as the wavelet analysis.

Huang and Xu [14] proposed the crack cluster connection method. First, this method finds verified seeds of a crack and then connects the individual seeds into seed

clusters. Starting from one seed, a crack cluster grows by accepting adjacent seeds one at a time until no nearby seeds can be found.

Cheng et.al [15] proposed a novel pavement cracking detection algorithm based on fuzzy logic. First, they eliminate the non-uniform background intensity effect, which uses unsharp masking methods, by subtracting a blurred image from the original image and adding a positive constant to avoid negative values. Second, they determine the brightness membership function for the image in the difference domain which is the result of the first step. The brightness membership function denotes the degree of brightness possessed by the gray level. They transform the image from the difference domain to the brightness domain, creating a fuzzy image. Third, they transform the image from the brightness domain to the crack domain. The method is used as a transformation function which means that pixels having brightness less than a certain value are classified as crack pixels. Then, they check the connectivity of pixels in the crack domain because the crack has a certain length, and isolated darker pixels are considered as noise. Finally, they classify the type of cracks based on information from the four direction projections.

Li [16] proposed a robust and high-efficiency model for segmentation and distress statistics of massive pavement images which is based on multi-scale space. First, they used multi-scale based image segmentation to remove the uneven elimination. Then they used the multi-scale based distress statistics to obtain the pavement cracking index

of each image. Finally, they used the distress index based pavement distressed images to separate them from the massive pavement source images.

Zuo et.al [18] proposed a novel pavement image segmentation method based on fractal theory. This method is compared with the classical Sobel filter and Otsu method. Segmentation based on fractal properties provides better results with little influence from noise, fast calculating speed and high accuracy. Their experimental results demonstrate the fractal method can correctly identify tiny cracks even from noisy pavement images.

Li et.al. [19] proposed a pavement image thresholding method based on neighboring differential histogram statistics. The distressed pixels in pavement images are darker than their surroundings and continuous, and the thresholding value is strongly related to the image standard deviation. So the principle of this method is that if the number of the surrounding pixels are bigger than the object pixel in the grey-level image, then this pixel has bigger probability to be a crack pixel.

Yan et.al. [20] proposed a pavement crack detection method for the high-grade highway. This method reconstructs the median-filter algorithm with four structural elements to enhance the gray-scale pavement images, and combines the morphology-gradient operator with the morphology-close operator to extract the crack edges and fill the gaps of the cracks. After obtaining skeletons, the length and width of cracks are measured.

Chapter 3

Image Segmentation Techniques

This chapter introduces the pavement segmentation using the fractal theory. The pavement images need to be enhanced before applying thresholding. A nonlinear filter is used in this chapter in order to improve the contrast, which can make easily find the thresholding. The fractal thresholding helps to obtain useful binary pavement images, which is the foundation for the subsequent steps.

3.1 Image Improvement

The aim of preprocessing in pavement image inspection is to suppress the unwanted information from the image data and enhance the desired image features important for further processing. Preprocessing is an important step in the sense that, with an effective process, much of the subsequent analysis will be simplified. Due to non-uniform lighting or weather conditions, the contrast between distress and background is often very low. In addition, the image is often corrupted with noise and undesired features. Therefore, an image enhancement method capable of removing non-uniform background illumination effects and noises is required.

A promising technique would be to use a nonlinear filter which takes the mean and variance of local gray values into account. Other techniques, such as median filtering

can be used to reduce the noise while preserving much of the detail in the image. To remove the non-uniform background intensity effect, the following nonlinear filter is used, as show in (1),

$$f^* = Z(i,j) \times \left[f_{org}(i,j) - f_{blur}(i,j)\right] + m \quad (1)$$

where f^* , $f_{org}(i,j)$,and $f_{blur}(i,j)$ are respectively the filtered, original and blurred images of the pavement, m is the mean value of the original image, and $Z(i,j)$ is a local gain factor sensitive to local variations which is 1. The blurred image is obtained by convoluting a 9×9 low-pass filter [15] with the original image. Some low-pass filters can be used to calculate the blurred image such as ideal low-pass filter, Butterworth low-pass filter and Gaussian low-pass filter. After many experiments, the Gaussian low-pass spatial filter has been chosen because it avoids a bright ringing effect. The Gaussian transfer function, as shows in (2):

$$H(u,v) = \exp\left[-\frac{D^2(u,v)}{2\sigma_0^2}\right] \quad (2)$$

σ_0 is cut-off frequency. Thus, the blurred image can be obtained by (3):

$$f_{blur}(i,j) = f_{org}(i,j) * \mathcal{F}^{-1}[H(\Omega_i, \Omega_j)] \quad (3)$$

Figure 3-1 shows an original pavement image. Figure 3-2 shows the same image with contract enhanced by applying the nonlinear filter. Figure 3-3 and Figure 3-4 are the histograms of the previous two images respectively.

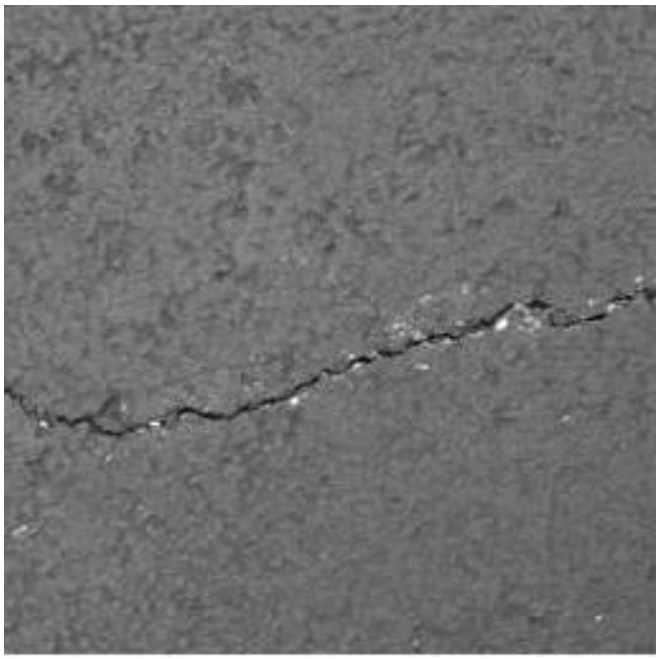


Figure 3-1 Original pavement image

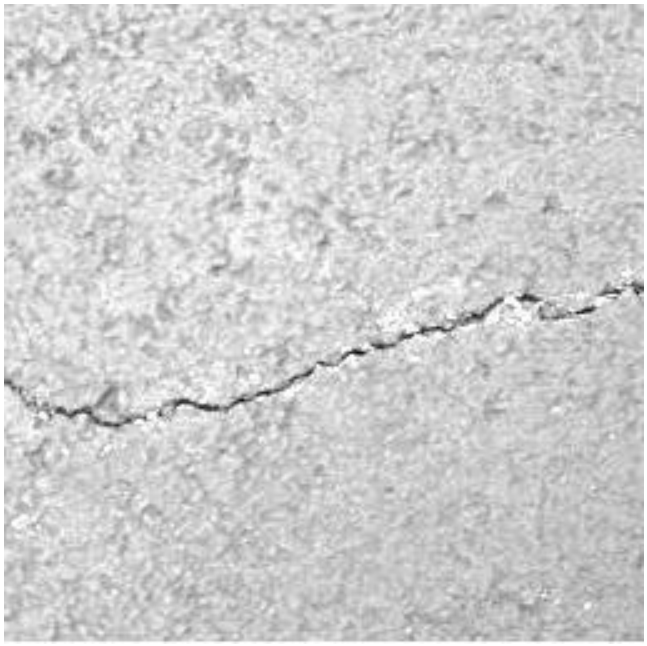


Figure 3-2 Image after improvement

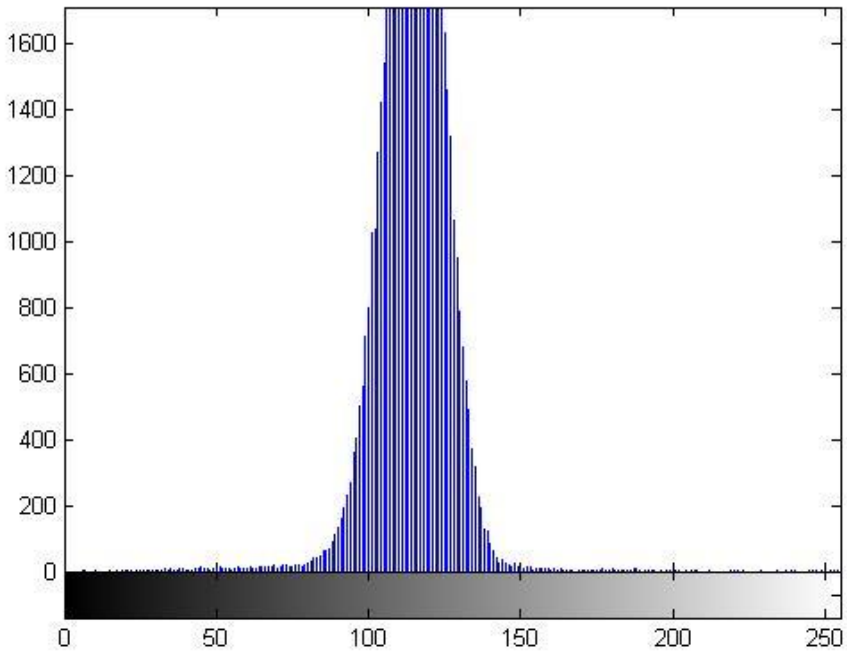


Figure 3-3 Histogram of original image

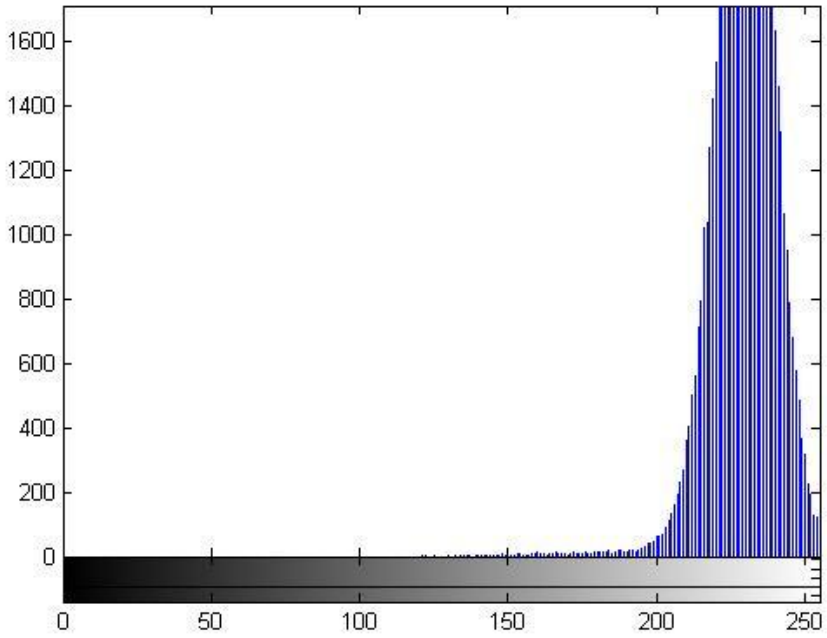


Figure 3-4 Histogram of improvement image

3.2 Thresholding Using Fractal Theory

Thresholding is a widely used technique for image segmentation and feature extraction. For a given image, most of thresholding techniques involve creating a histogram of the gray level values to be used to find the peaks that exist in the image. A threshold is then chosen according to the valley between these peaks or modes (usually two prominent peaks are assumed). Adaptive thresholding applies a different threshold to different regions of the image and results in better segmentation. Pavement cracks usually involve abrupt changes in the gray level of two adjacent regions of variant gray levels. With an appropriate threshold that is extracted from the block and lies somewhere between the means of the two regions, the block can be converted into a binary form.

In this thesis, a segmentation method based on fractal theory is used to segment the pavement image. A fractal is generally a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole, a property called self-similarity. The term was coined by Benoît Mandelbrot in 1975 and was derived from the Latin fractus meaning “broken” or “fractured.” A mathematical fractal is based on an equation that undergoes iteration, a form of feedback based on recursion [17]. Pavement crack image is composed of single crack which shows self-similarity. The property of self-similarity, or scaling, is one of the central

concepts of fractal geometry. Cracks are effectively represented by fractals and pavement crack images can be segmented by fractal method. It is noteworthy that the self-similarity of cracks only can exist on a small scale since there is no pure fractal image in the world.

First, the concept of upper surface $u(i,j,\epsilon)$ and lower surface $b(i,j,\epsilon)$ are introduced. Given, the gray level function $f(i,j) = u(i,j,0) = b(i,j,0)$, the upper surface for various values of ϵ and lower surface are defined as follows:

$$u(i,j,\epsilon+1) = \max\{u(i,j,\epsilon)+1, \max[u(m,n,\epsilon)]\} \quad (4)$$

$$|(m,n)-(i,j)| \leq 1$$

$$b(i,j,\epsilon+1) = \min\{b(i,j,\epsilon)+1, \min[b(m,n,\epsilon)]\} \quad (5)$$

$$|(m,n)-(i,j)| \leq 1$$

Where the point (m, n) is an immediate neighbor of (i, j). So the distance between point (m, n) and (i, j) is less than one. A covering blanket is defined by its upper surface and its lower surface. A point (x,y,f) is located such that $b(x,y,\epsilon) < f(x,y,\epsilon) < u(x,y,\epsilon)$. The volume of the blanket is computed from u and b by

$$V(\epsilon) = \sum_{i,j}(u(i,j,\epsilon) - b(i,j,\epsilon)) \quad (6)$$

As the surface area is measured with radius ϵ, the volume is divided by 2ϵ

$$A(\epsilon) = V(\epsilon)/2\epsilon \quad (7)$$

The area of a fractal surface behaves according to the expression:

$$A(\epsilon) = k\epsilon^{D} \quad (8)$$

After taking a logarithm of expression (8), we get the equation:

$$\log A(\epsilon) = D\log\epsilon + \log k \tag{9}$$

According to the expression (9), the slope D can be calculated by least-square fitting and the values of k are calculated.

When a pure fractal image is analyzed, the value of k is a constant. However, for images of different textures, the value of k changes on a scale, so the parameter k reflects the change of the surface area on different scales. Obviously, when the surface is smooth, the value of k is small. On the contrary, when the surface is rough, the value of k is bigger. Thus, we can set the parameter k as a local threshold to segment the image [18].

The algorithm can be presented as the following:

Step 1: Identify a window of size $s \times s$ in the image of $M \times N$ size and calculate every k of each window.

Step 2: Find the minimum of the k value such that $0 < \alpha < 1$ and β. Set the threshold $T = \alpha \times k_{min} + \beta$ to segment the image.

After many tests, $s = 3$, $\epsilon = 9$, $\alpha = 0.22$ and $\beta = 45$ are selected.

Figure 3-5 shows the pavement image after applying fractal thresholding base to Figure 3-2.

Figure 3-5 Binary image

Another example of using fractal thresholding is described. The pavement image after improvement is shown in Figure 3-6. The cracks are distinguished from pavement after applying fractal thresholding as shown in Figure 3-7. Only a few noise points are left in the image which can be removed in the following step.

In general, this chapter presents the first part of the pavement images processing. After enhancement, the gray scale image can be transformed to binary image by the fractal thresholding. However, some crack pixels are removed and some noise pixels remain irrespective of the thresholding method. Although there are many methods to compensate the crack and remove the noise, a good thresholding method is needed to obtain a clear binary image.

Figure 3-6 Image after improvement

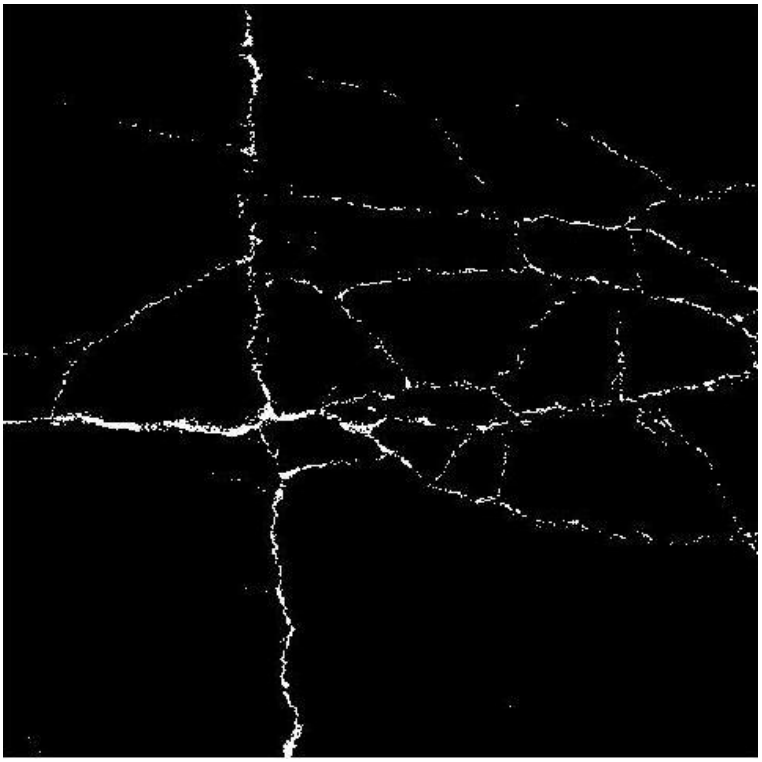


Figure 3-7 Binary image

Chapter 4

Mathematical Morphology and Noise Removal Techniques

This chapter introduces the mathematical morphology and noise removal techniques. The closing operation, which is a common mathematical morphology operator, is used to fill the gaps between cracks. Two noise removal methods are also compared in this chapter. The results show that median filtering is not efficient enough for noise removal.

4.1 Closing Operation Techniques

Mathematical morphology is a technique for analysis and processing of geometrical structures based on set theory, lattice theory, topology, and random functions. Mathematical morphology is an important tool for low-level image processing [21]. Most morphological transforms are constructed from elementary morphological operations such as dilation, erosion, hit-or-miss transform, morphological skeleton, filtering by reconstruction and granulometry. This operation is guided by structural elements. A structural element is a shape used to probe or interact with a given image with the purpose of drawing conclusions on how this shape fits or misses the shapes in the image [21].

Morphological transforms such as dilation and erosion are used in this thesis. The

dilation of A by the structuring element B is defined by (10):

$$A \oplus B = \{x | [(B^s)_x \cap A] \neq \emptyset\} \quad (10)$$

Where B^s denotes the symmetric of B.

The dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. Formula (10) can be written as (11) according to this explanation.

$$A \oplus B = \{x | [(B^s)_x \cap A] \subseteq A\} \quad (11)$$

The erosion of A by the structuring element B can be understood as the locus of points reached by the center of B when B moves inside A. The definition is given by (12):

$$A \ominus B = \{x | (B)_x \subseteq A\} \quad (12)$$

Dilation and erosion can be cascade connected because they are not inverse operations. Applying dilation before erosion is called a closing operation which can fill the holes in a binary image. The closing operation is a very important operation of mathematical morphology, and the definition is given by (13):

$$A \cdot B = (A \oplus B) \ominus B \quad (13)$$

After many tests, a morphological structure element "disk" for both dilation and erosion has been chosen. The structure element creates a flat, disk-shaped with the radius is 5. This structure element is described in Figure 4-1.

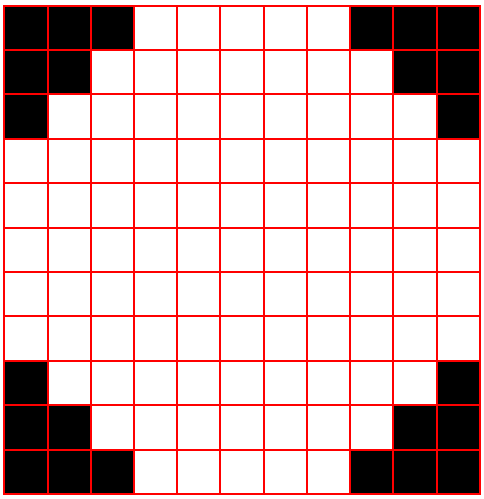


Figure 4-1 Disk structure element

4.2 Noise Removal

Noise reduction is one of the most significant steps of image processing along with crack detection. Median filtering is an example of many noise removal techniques. The median filter is a non-linear digital filtering technique, often used to remove noise from images or other signals. The idea is to examine a sample of the input and decide if it is representative of the signal. This filter is performed using a window consisting of an odd number of samples. The values in the window are sorted in a numerical order and the median value is selected as the output. The oldest sample is discarded, a new sample acquired, and the calculation repeats. [21]

Median filtering is one of the most commonly used techniques for noise removal. It usually is used on gray scale images. In this thesis, median filter is used on binary images following the same principle. In this process, a window 3×3 pixels is selecetd as shown in Figure 4-2.

$$\begin{matrix}0&0&1\\0&1&1\\0&0&0\end{matrix} \rightarrow 0\,0\,0\,0\,0\,0\,1\,1\,1 \rightarrow \begin{matrix}0&0&1\\0&0&1\\0&0&0\end{matrix}$$

Figure 4-2 A 3×3 window median filter

In the simulation result, a pavement image has almost no "salt and pepper" noise after applying median filter. Figure 4-3 shows the image after the closing operation. Figure 4-4 shows the image after the median filter is done. However, some crack pixels are also removed after the median filter. Comparing Fig 4-4 to Fig 4-3 shows three additional gaps have been generated after applying the median filter. This problem may influence the following steps.

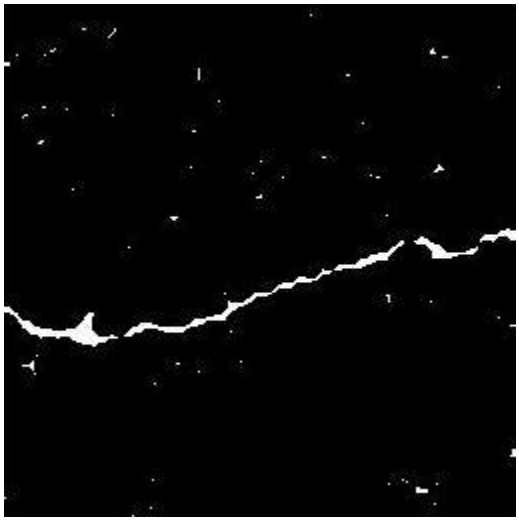


Figure 4-3 Image after closing operation

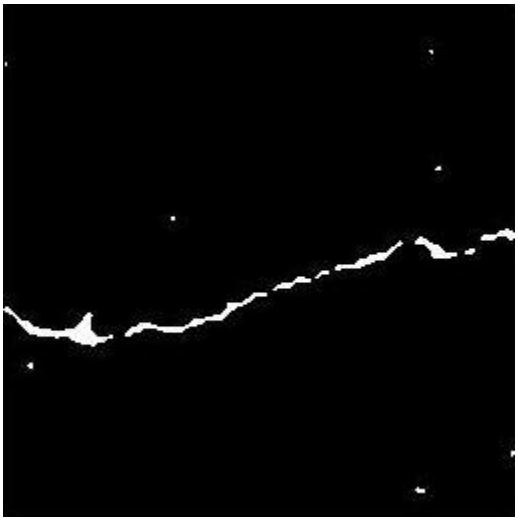


Figure 4-4 Image after median filter

Comparing Figure 4-3 and Figure 4-4 carefully, we can find that the disappeared crack pixels occur in the thin part of the crack. The median value has more probability to be a background pixel than an object pixel in the thin areas. As a result, these useful pixels are removed as noise pixels after median filtering. Figure 4-5 shows the process of crack pixel removal. As seen in Figure 4-5 (a), the original crack has oen pixel in the middle. Figure 4-5 (b) shows the filter window, and as seen in Figure 4-5 (c), the median value pixel has been changed to a background pixel. The gap generated after median filtering is shown in Figure 4-5 (d).

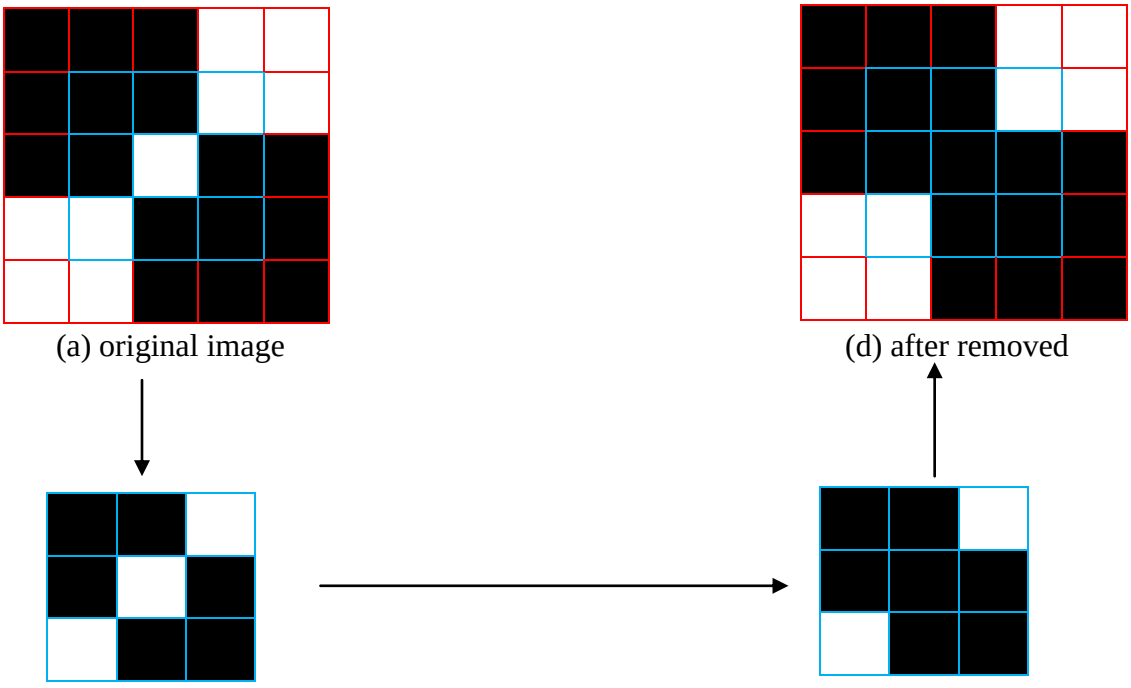


Figure 4-5 Crack pixel has been removed after median filter

Based on the problem of resulting from using the median filter, a new noise removed method is proposed. This method is designed according to the linearity of the cracks. Median filter does not consider the connectivity of the cracks, so some gaps are generated after median filtering. To remove isolated noise and retain crack pixels, this method checks neighboring pixels of every bright pixel. The algorithm for noise removal is summarized below:

$$if \quad f(i+2,j)=0 \ \cap \ f(i,j+2)=0 \ \cap \ f(i+2,j+2)=0 \ \cap$$

$$f(i-2,j)=0 \ \cap \ f(i,j-2)=0 \ \cap f(i-2,j-2)=0 \ \cap$$

$$f(i+2,j-2)=0 \ \cap f(i-2,j+2)=0$$

$then\ f(i,j) = 0$

The main idea of this method is to find adjacent background pixels for every bright pixel. Each bright pixel has to be complimented by bright neighboring pixels to exist. This method using 5×5 neighborhood yields better results. Figure 4-6 shows the result using this method. Although some noise points are still in the image, no crack gap is generated and almost no crack pixel is removed after using this method compare to using the median filter. The remaining noise points can be removed by the connected component method explained in the next chapter.

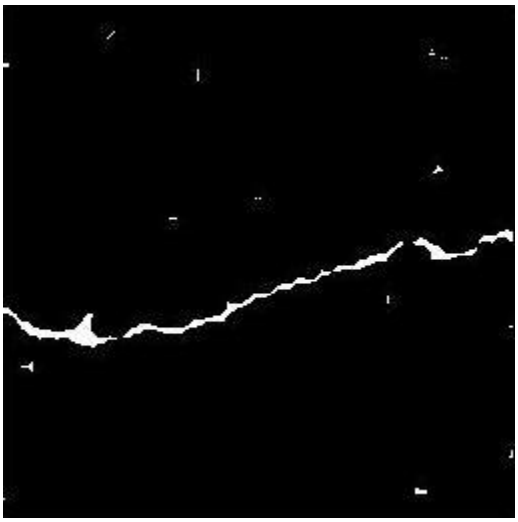


Figure 4-6 Noise removal

Another example of using this method is shown in Figure 4-7 to Figure 4-9. Figure 4-7 shows a pavement image after closing operation. Noise removal using a median filtering is shown in Figure 4-8. Not only are all the noise points removed, many crack pixels have also disappeared making it very hard to detect a crack in the image. Figure

4-9 shows the result of using the noise removal method introduced before. Crack information is retained in Figure 4-9 compared to Figure 4-8. Therefore, this method is better than median filtering for noise removal in images with cracks.

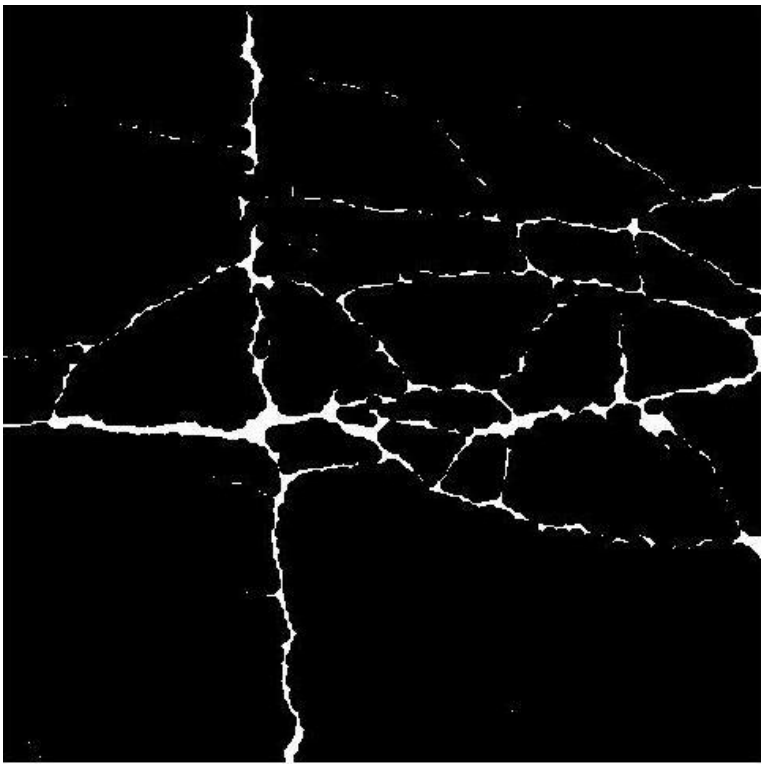


Figure 4-7 Image after closing operation

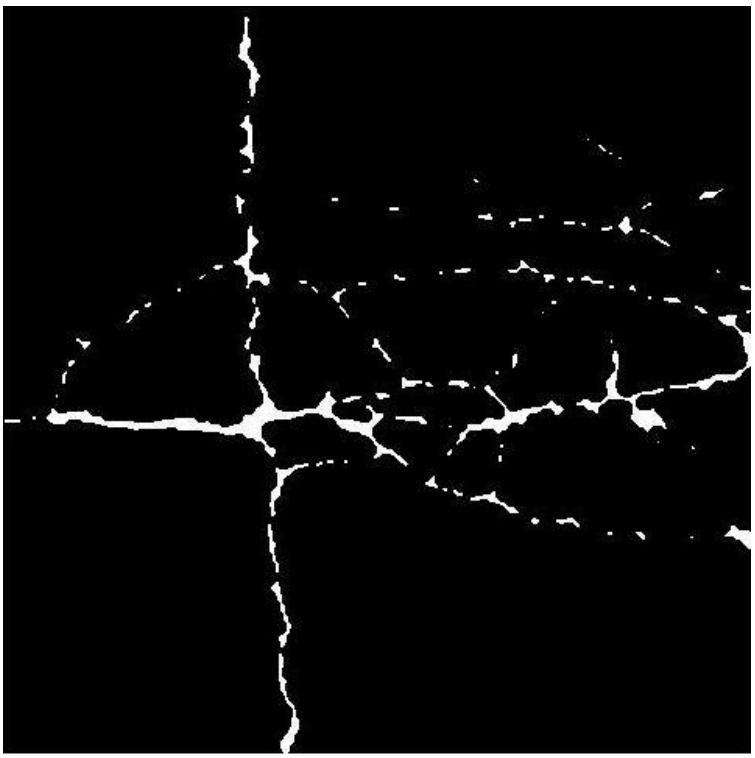


Figure 4-8 Image after median filter

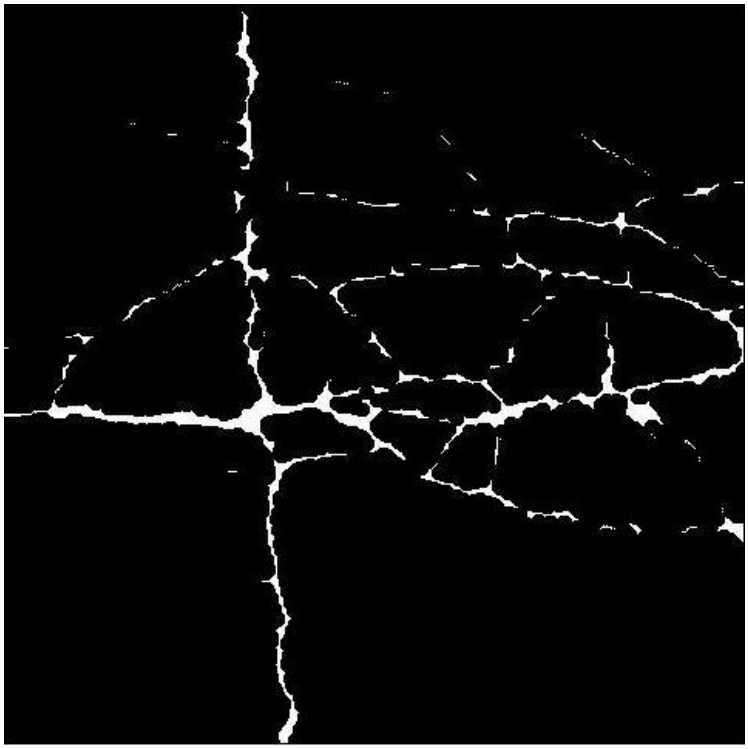


Figure 4-9 Noise removal

Chapter 5

Break Points Connectivity Algorithm

After preprocessing, the cracks have some break points which influence rating of the cracks. These break points are generated mainly because of discontinuity that occurred in the previous steps due to some crack pixels changing into background after thresholding or being removed after noise reduction. They may also be lost due to objects such as little stones in the cracks even before processing. In order to obtain the accurate rating results of the pavement images, a break points connectivity algorithm is proposed in this thesis. This algorithm mainly contains three steps: finding initial pixel of a crack, looking for the break points for every crack, and connecting the break points. Figure 5-1 shows the algorithm to verify the connectivity of break points.

The input is a binary image. This algorithm searches the initial pixel of transverse crack before longitudinal crack. The coordinate of two kinds of initial pixels are recorded in two different matrices respectively. This algorithm then searches the break points in horizontal direction from a transverse initial pixel. After finding a gap, this algorithm will search the neighbor crack pixel in a defined area and connects them. This algorithm searches the next break point along the same direction until the end of the crack or image. This process is repeated from the next initial pixel. After dealing with

all the transverse initial pixels, this algorithm searches the break points in vertical direction following the similar rule. The resultant output images are checked twice for connectivity in horizontal and vertical directions.

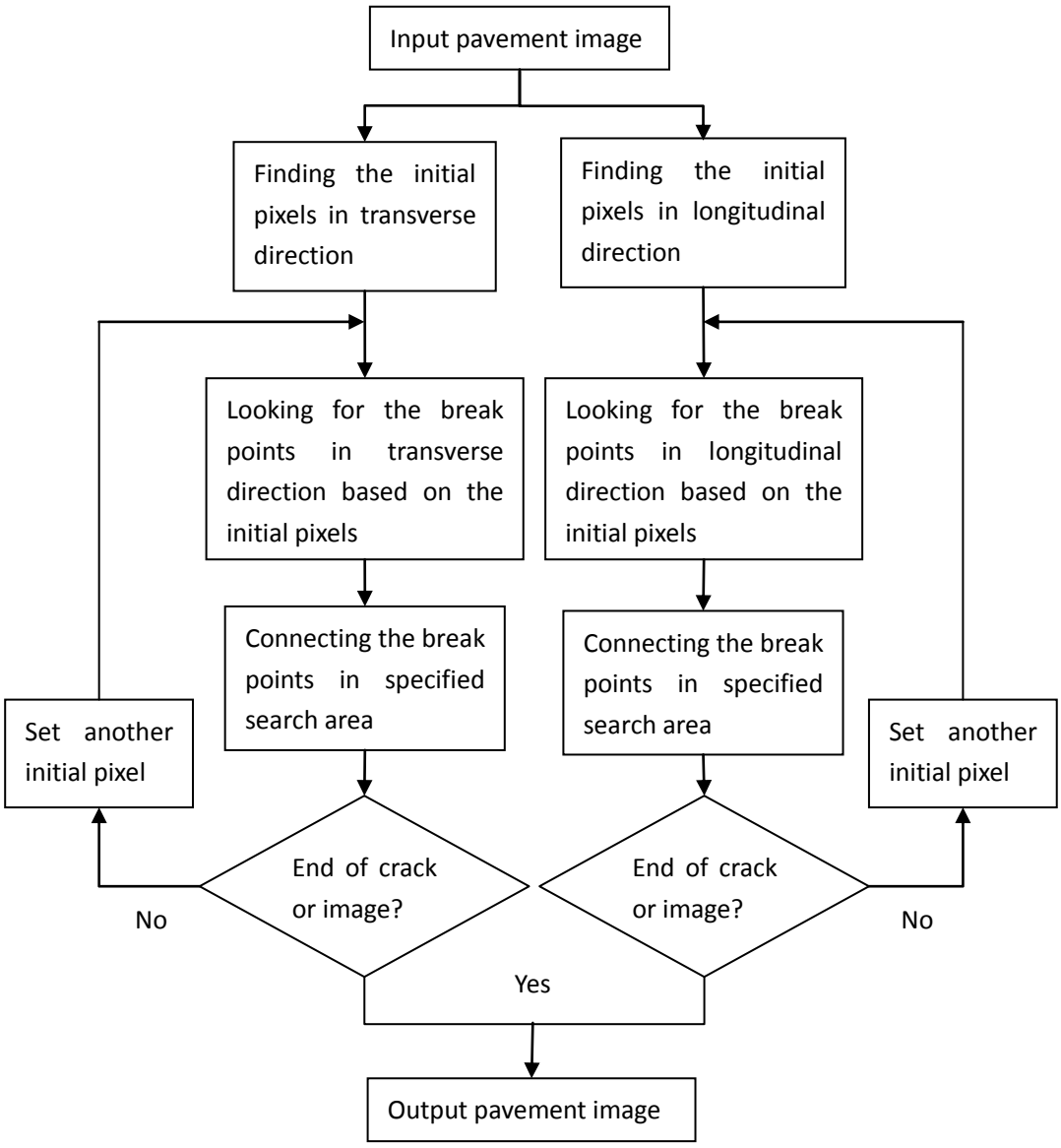


Figure 5-1 Break points connectivity algorithm processing steps

5.1 Finding the initial pixels

The first step of of break points connectivity algorithm is finding the initial pixels. A crack pixel need to be located so the break points could be found in the direction specified. An initial pixel has no use for the first pixel of a crack. However, the initial pixel should be on the left of a transverse crack and on top of a longitudinal crack in order to find as many break points as possible. Otherwise, some break points before the initial pixel cannot be found.

Four different types of cracks are introduced in Chapter 1. Two types of cracks have been identified based on the direction of the crack: transverse and longitudinal. Any other directions can be attributed to these two directions, which can make the process easier. After preprocessing, binary pavement images contain bright pixels which are crack and noise pixels and dark pixels which are background pixels. The main idea of finding the initial pixels is to check the continuity of some bright pixels, so that noise pixels cannot be set as the initial pixel.

For every input pavement image, the algorithm searches the initial pixel for transverse cracks before the initial pixel for longitudinal cracks. Considering the location of the cracks, the pavement image is longitudinally divided into two equal segments. When the algorithm searches the transverse initial pixels, the initial pixel is found in each half. The pavement image is further divided into some small parts along the longitudinal direction. The size of each segment of the image is 20 pixels which is

an empirical value based on the distance between the transverse cracks. Figure 5-2 shows that all parts are divided in the pavement image for searching the initial pixel for transverse cracks. After the pavement image is divided into twelve parts as shown in Figure 5-2, the algorithm searches the initial pixel for transverse cracks in these twelve parts one by one. Each transverse crack can be found using this method. In other words, it is hard to miss the transverse cracks since this method guarantees finding all break points.

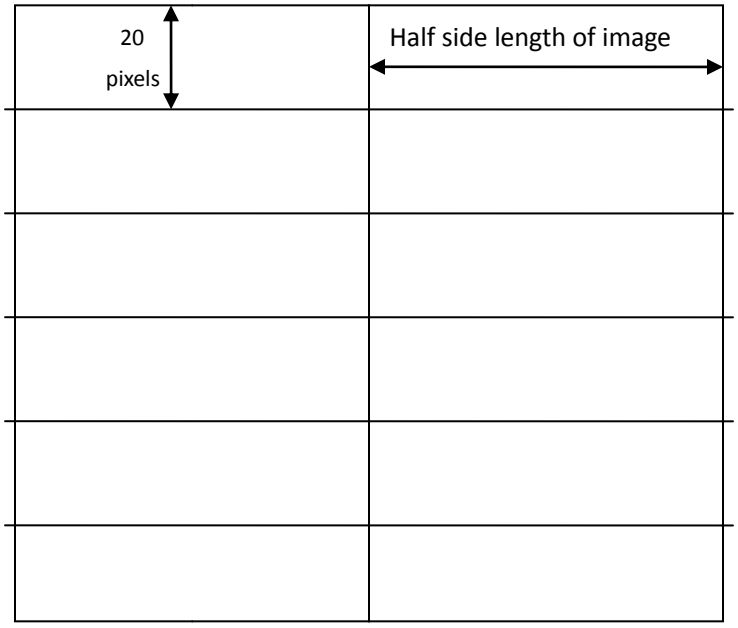


Figure 5-2 Divide pavement image for searching the transverse initial pixels

All the coordinates of bright pixels including crack and noise pixels are recorded in a matrix. The difference between crack and noise is continuity. Therefore, this

algorithm checks for pixels with in the distance of 10 pixels for every bright pixel. If more than one pixel is bright, the original bright pixel is set as the initial pixel and the search for initial pixels is stopped. Otherwise, the next bright pixel is checked until the algorithm finds an initial pixel. Figure 5-3 shows the searching method in one of the segments. We can see that this algorithm checks five pixels for one bright pixel in Figure 5-3. These five pixels are in the same row. The determined function is summarized below:

$$check\ the\ neighbor\ pixels\ of\ f(i,j)$$

$$if\ f(i-2,j+10)=1 \cup f(i-1,j+10)=1 \cup f(i,j+10)=1$$

$$\cup f(i+1,j+10)=1 \cup f(i+2,j+10)=1$$

$$then\ set\ f(i,j)\ as\ transverse\ initial\ pixel$$

$$Otherwise\ check\ the\ \ next\ bright\ pixel$$

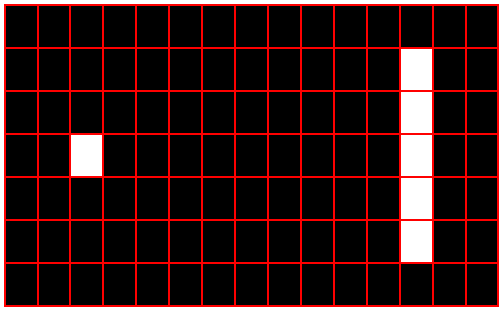


Figure 5-3 To determine the transverse initial pixel

The procedure is repeated in the longitudinal direction. The pavement image is

divided in the transverse direction into two equal halves and the algorithm searches for initial pixels in the longitudinal direction. The pavement image is further divided into some smaller segments along the transverse direction. The distance between each small segment is also 20 pixels. Figure 5-4 shows that all parts are divided in the pavement image for searching the longitudinal initial pixels.

This algorithm mainly deals with the transverse cracks and longitudinal cracks on the condition that the distance between cracks is not too less. A large area cannot be searched for initial pixels easily. Therefore, the pavement images will further divided into some smaller segment that can find all the cracks for block cracks or alligator cracks.

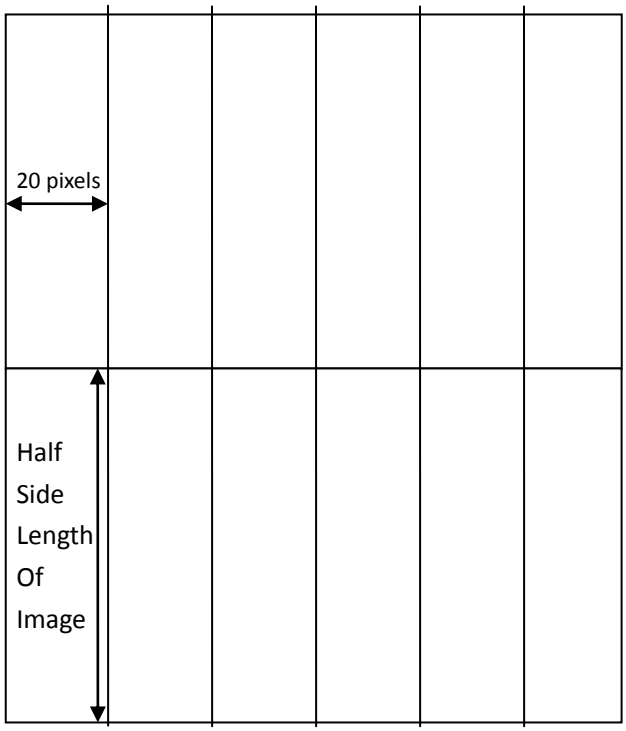


Figure 5-4 Divide pavement image for searching the longitudinal initial pixels

This algorithm also checks five pixels with the distance of 10 pixels for every bright pixel in each segment. Figure5-5 shows that these five pixels are below the bright pixel. The determine function is summarized below:

$$check\ the\ neighbor\ pixels\ of\ f(i,j)$$

$$if\ f(i+10,j-2)=1 \cup f(i+10,j-1)=1 \cup f(i+10,j)=1$$

$$\cup f(i+10,j+1)=1 \cup f(i+10,j+2)=1$$

$$then\ set\ f(i,j)\ as\ longitudinal\ initial\ pixel$$

$$Otherwise\ check\ the\ \ next\ bright\ pixel$$

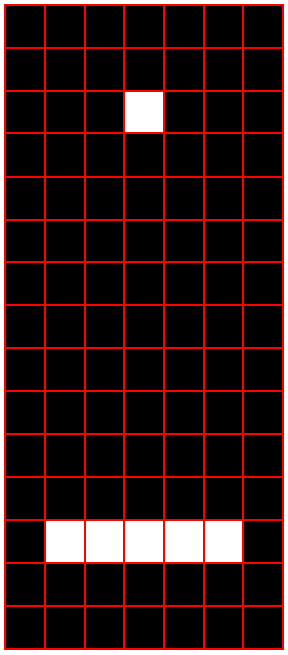


Figure 5-5 To determine the longitudinal initial pixel

Block crack and alligator cracks consist of a combination of transverse and

longitudinal cracks. Therefore, every crack can be found with this method. The coordinate of initial pixels found are recorded in a matrix. Searching the break points starting from these coordinates is the next step of this algorithm.

5.2 Finding the Break Points

The connectivity analysis of the crack pixels is based on a depth-first searching method. The method of searching the break points is composed of transverse searching and longitudinal searching based on two different initial pixels respectively. This method does not require checking every crack pixel, but only moving along the crack with special rule to find the break points. In the following, a procedure for finding the break points of the transversal cracks is described.

After finding the coordinates of the initial pixel of a transverse crack, three prioritized directions are defined, namely, the right, up, and down directions to denote the first, second, and third directions of the search, respectively. The basic rule of the searching method is to follow the bright pixels in the first direction from the initial pixel until there is no bright pixel in this direction. It will then continue along the second direction and if no bright pixel is found in this direction, scanning changes to the first direction to check for bright pixels. If a bright pixel exists, the algorithm will continue in the first direction again, otherwise, scanning takes place in the third direction. In

searching for transverse cracks, the high priority level of searching is followed by the second direction and then finally in the third direction with the lowest priority. If there is no bright pixel within 4-neighbors, this method will check 8-neighbors and start searching from the first direction again.

The search algorithm for connectivity analysis is summarized below:

1) *Start from an initial crack pixel.*
2) *Follow the crack pixels in three directions right, up and down until no crack pixel is found.*
3) *Check 8-neighbors of the pixel visited last*
4) *Determine the presence of either a break point or a column of break points.*
5) *Look for the nearest crack pixel in a specific search area.*
6) *Connect them and repeat the process for the entire image.*

In order to define the area to be searched in the following step, the break points are divided into two types: single pixel break point and multiple pixels break point. These two break points are explained in Case 1 and Case 2 as follows.

Case 1: Note that, when there is no bright pixel in any of the three directions, the method will then check the upper right and bottom right pixels, i.e., diagonal pixel elements. If neither pixel is bright, the final bright pixel is a break point. Otherwise, the search algorithm will continue in the first direction from one of the right pixels. Figure 5-6 shows the procedure of searching single pixel break point.

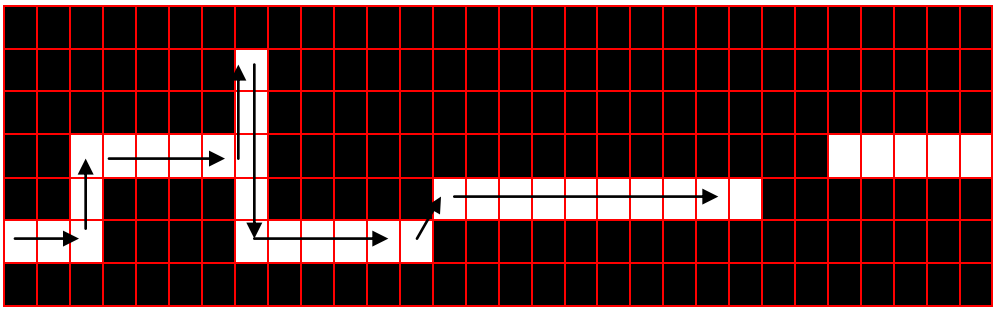


Figure 5-6 Single pixel break point for transverse cracks

Case 2: On the other hand, after continuing along the third direction, if there is no bright pixel in the next column, multiple pixels break point is obtained. Figure 5-7 shows the procedure of searching multiple pixels break point. If there is no bright pixel in the first direction after checking the third direction, this method will check every adjacent right pixel of the pixels in previous column and the diagonal pixel elements of this column. After finding a bright pixel, this algorithm starts searching from the first direction again. Otherwise, multiple pixels break point is obtained.

These two cases explain the configuration of the break points on pixel level clearly. The search areas are designed and directed towards the two cases. The search area for multiple pixels break point is larger than the single pixel break point because the width of multiple pixels break point corresponding to the cracks could be bigger. The details of connection method are presented as follows.

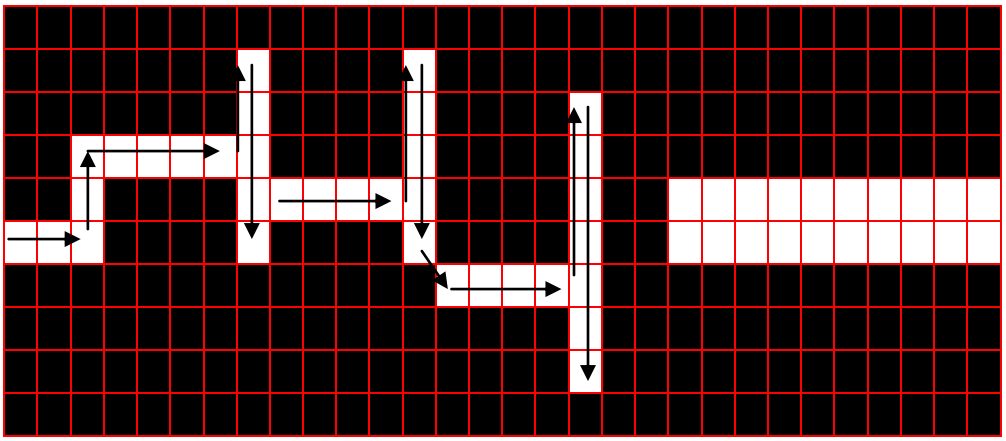


Figure 5-7 Multiple pixels break point for transverse cracks

The process for obtaining a longitudinal crack is similar to the transverse; however, the three prioritized search directions will change in the following way. The downward direction is crucial for this case; therefore, it will be the first direction with high priority to search for continuity. The second and the third directions are the right and left directions, respectively. The priority level is the same as the transversal cracks. Note that, the order of priority is very important and should be observed during the search process. We cannot use the same search method for both transversal and longitudinal cracks, because the tendency for transversal cracks is in the right direction, and the tendency of longitudinal crack is in the downward direction.

Two different types of break points are identified in longitudinal cracks. As mentioned above, the search directions are changed sequentially. Thus, the search way for longitudinal cracks is a little different from transverse cracks. Break points for

single pixel and multiple pixels for longitudinal cracks are shown in Figure 5-8 and Figure 5-9 respectively.

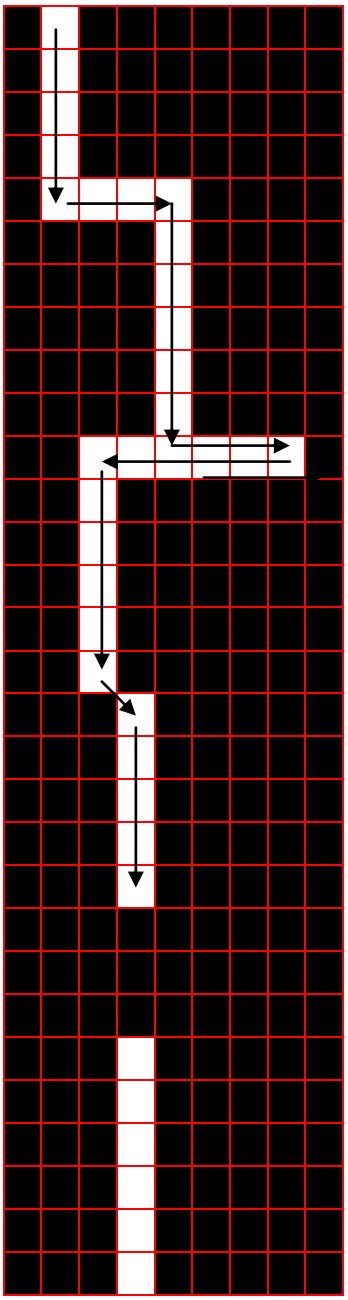


Figure 5-8 Single pixel break point for longitudinal cracks

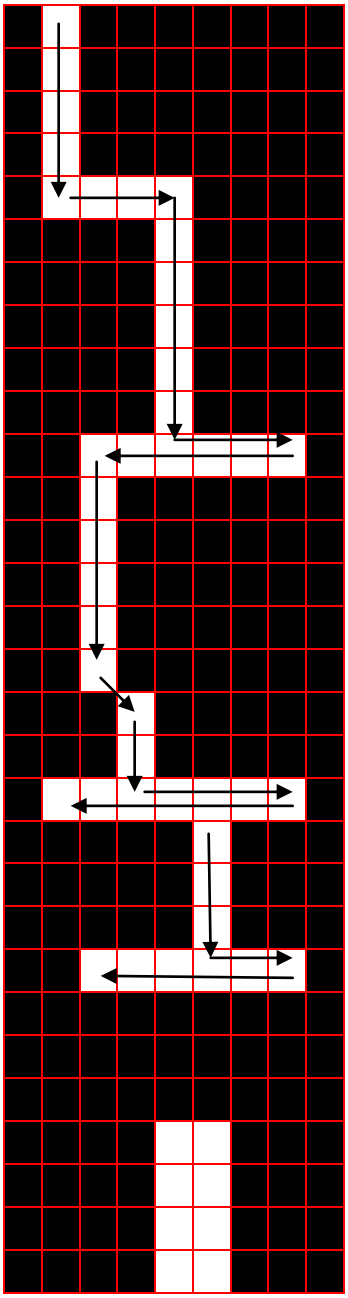


Figure 5-9 Multiple pixels break point for longitudinal cracks

5.3 Connecting the Break Points

After the break points are found, this algorithm continues to search for crack pixels. There are two steps to eliminate breaks in the cracks. First, this algorithm searches the bright pixels within a given area according to break points for single and multiple pixels. Then, this algorithm changes dark pixel to bright pixel in the break accordingly. This algorithm deals with the pavement images on pixel level in order to connect the break points smoothly. One loop of this algorithm is finished after connecting the first break point. Since a crack has more than one break point, this algorithm continues to search the next break point from another initial pixel until the end of the crack or the image. The last findable bright pixel in the given area is set as the initial pixel and the procedure is repeated.

The search areas for finding the nearest crack pixel would be different for the two types of break points in order to obtain a connected set of crack points. The algorithm will check pixels 4 rows above and below the initial point and the right 20 columns based on the single pixel break point. If a bright pixel is found in the search area, the algorithm changes the previous pixels in the row traversed before to a bright pixel. Then all the bright pixels are adjacent so that a break point is connected. Figure 5-10 shows the search area corresponding to Case 1. The set of pixels within blue frame is the search area. The white arrows show the direction of search for bright pixels. After finding the bright pixel, this algorithm changes the previous pixels to the bright pixel.

Figure 5-11 shows the procedure of this step.

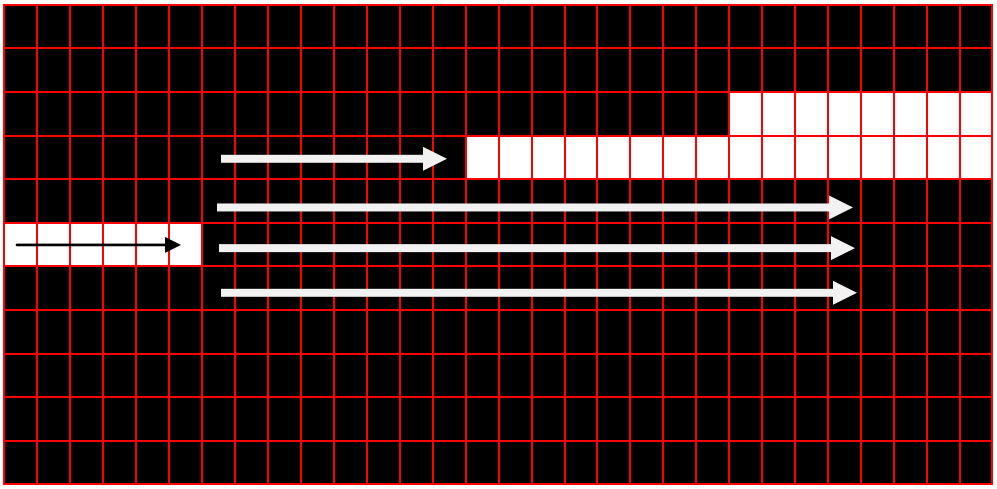


Figure 5-10 Search area 1

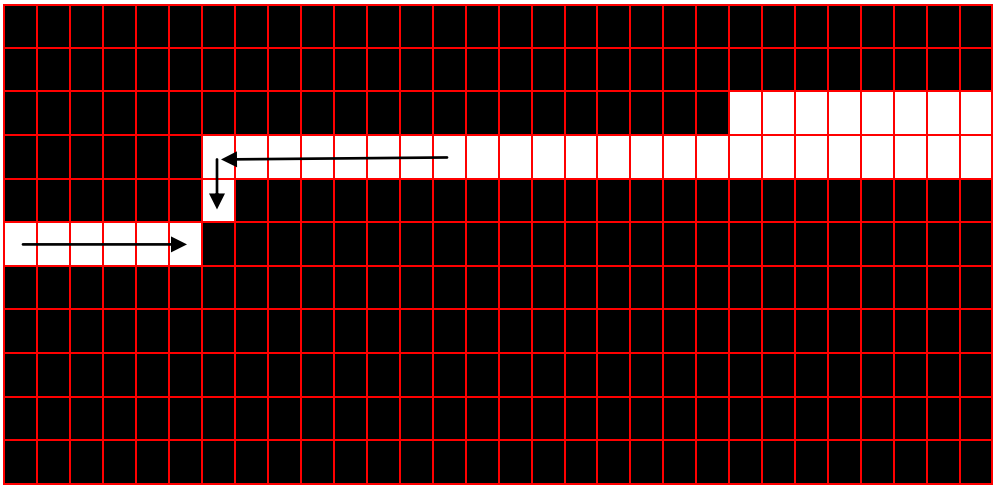


Figure 5-11 Connect the Single pixel break point

In a similar way, the algorithm finds a crack pixel in the search area and connects it to the previous break point by backtracking for the multiple pixels break point. The search area is expanded in this situation because this gap could be bigger than the single pixel break point. Figure 5-12 shows the part of the search area corresponding to Case 2.

Figure 5-13 shows the procedure to connect the break point.

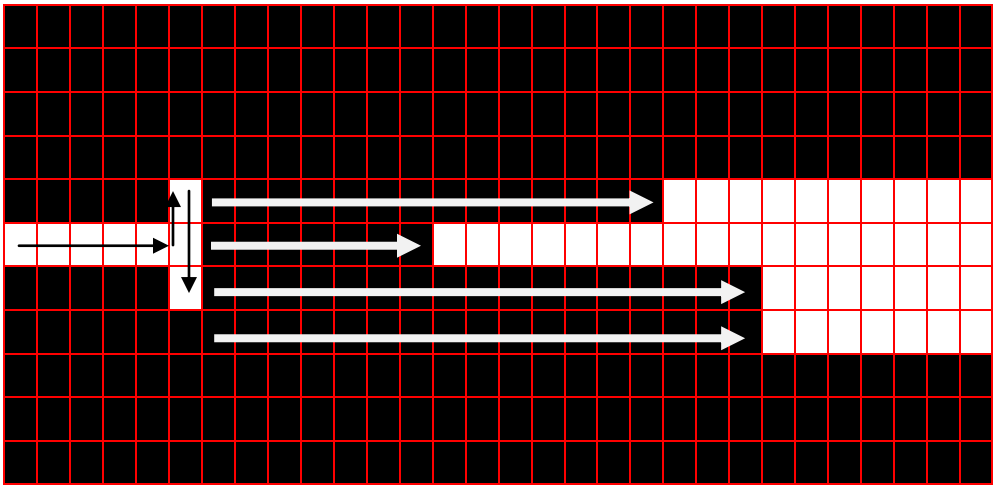


Figure 5-12 Search area 2

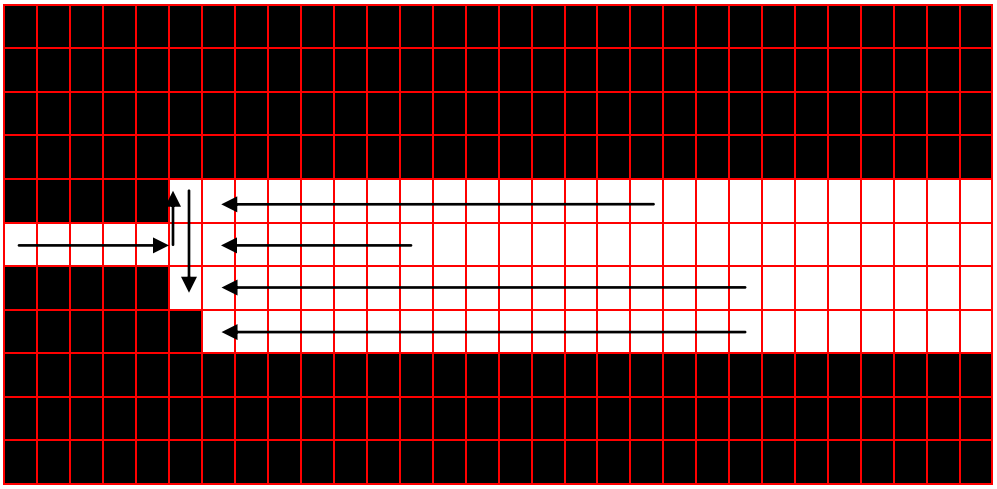


Figure 5-13 Connect the multiple pixels break point

The search area is also defined in the break points of longitudinal cracks. The dark pixels are also changed to the bright pixels in order to connect the break points following a similar procedure. After connecting one break point, this algorithm searches for the next break point in the longitudinal crack with the longitudinal search procedure.

A transverse crack with four gaps after noise removal is shown in Figure 5-14. After applying the break points connectivity algorithm, these four gaps are connected, as shown in Figure 5-15. Figure 5-16 compare the image before connecting the break points to the image after connecting on pixel level. A longitudinal crack with five gaps after noise removal is shown in Figure 5-17. Figure 5-18 shows that these five gaps are connected after applying the break points connectivity algorithm.

This chapter proposed a break points connectivity algorithm. This algorithm is based on depth-first search method and is modified according to the orientation of cracks. The remaining noise pixels are easily removed after skeletonization based on this algorithm.

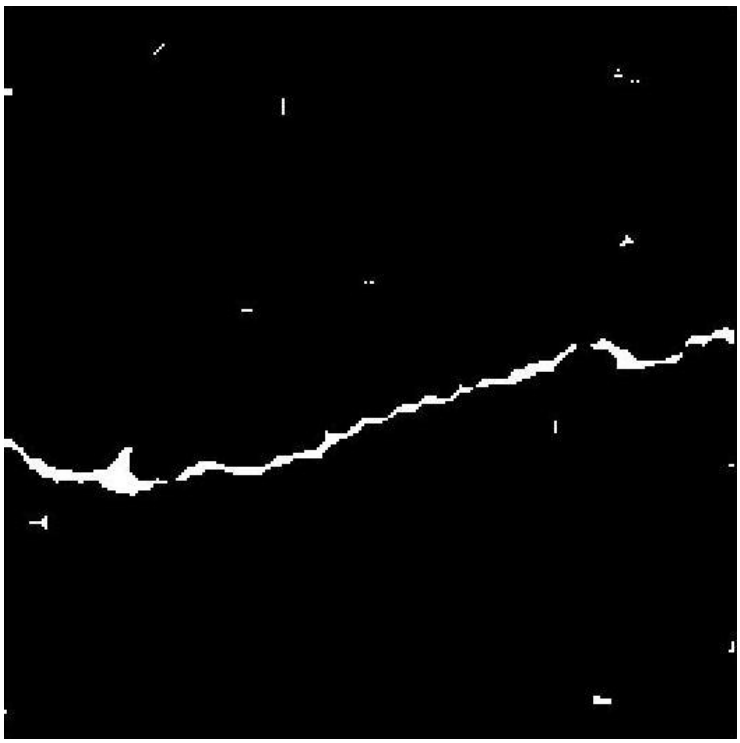


Figure 5-14 A transverse crack before connecting

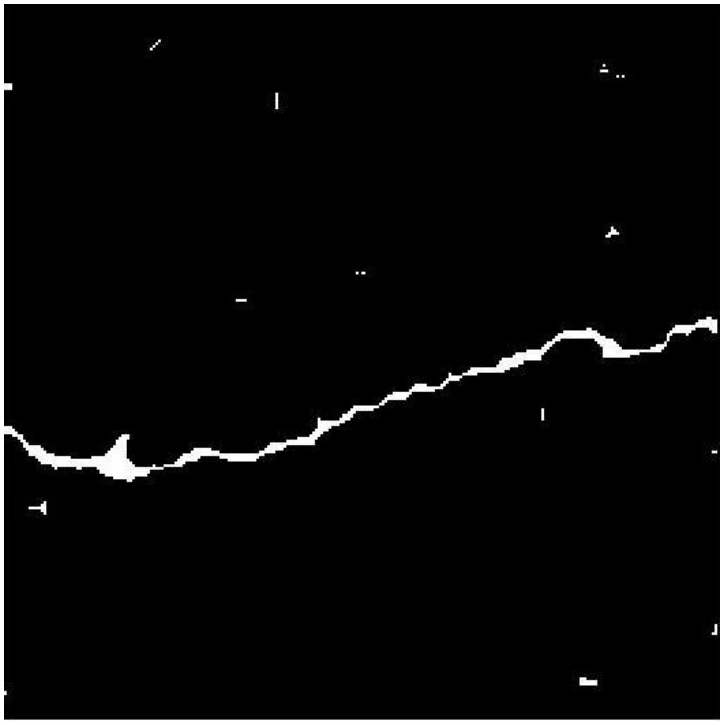
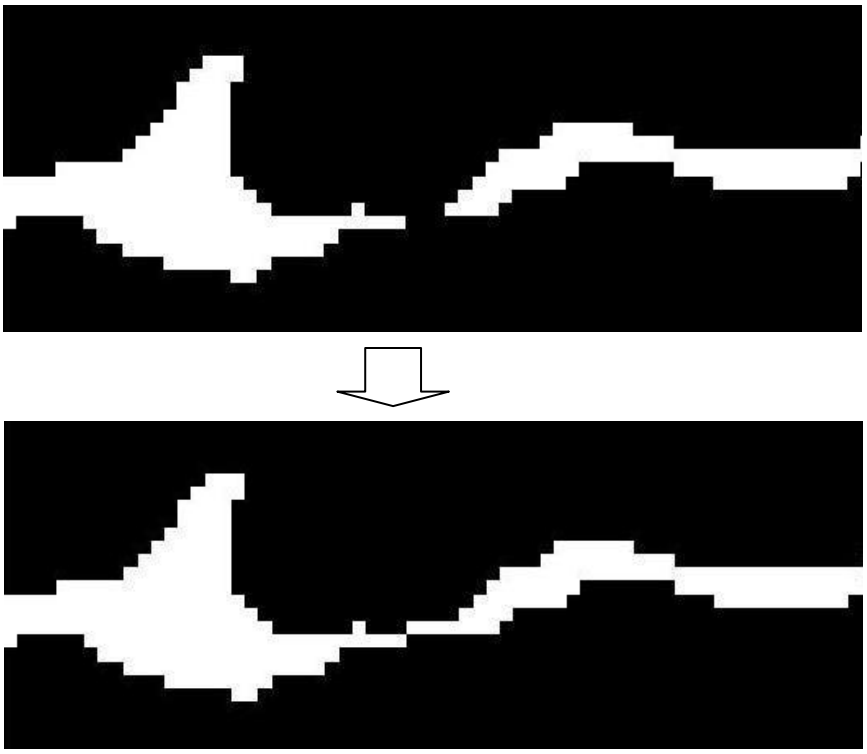


Figure 5-15 A transverse crack after connecting



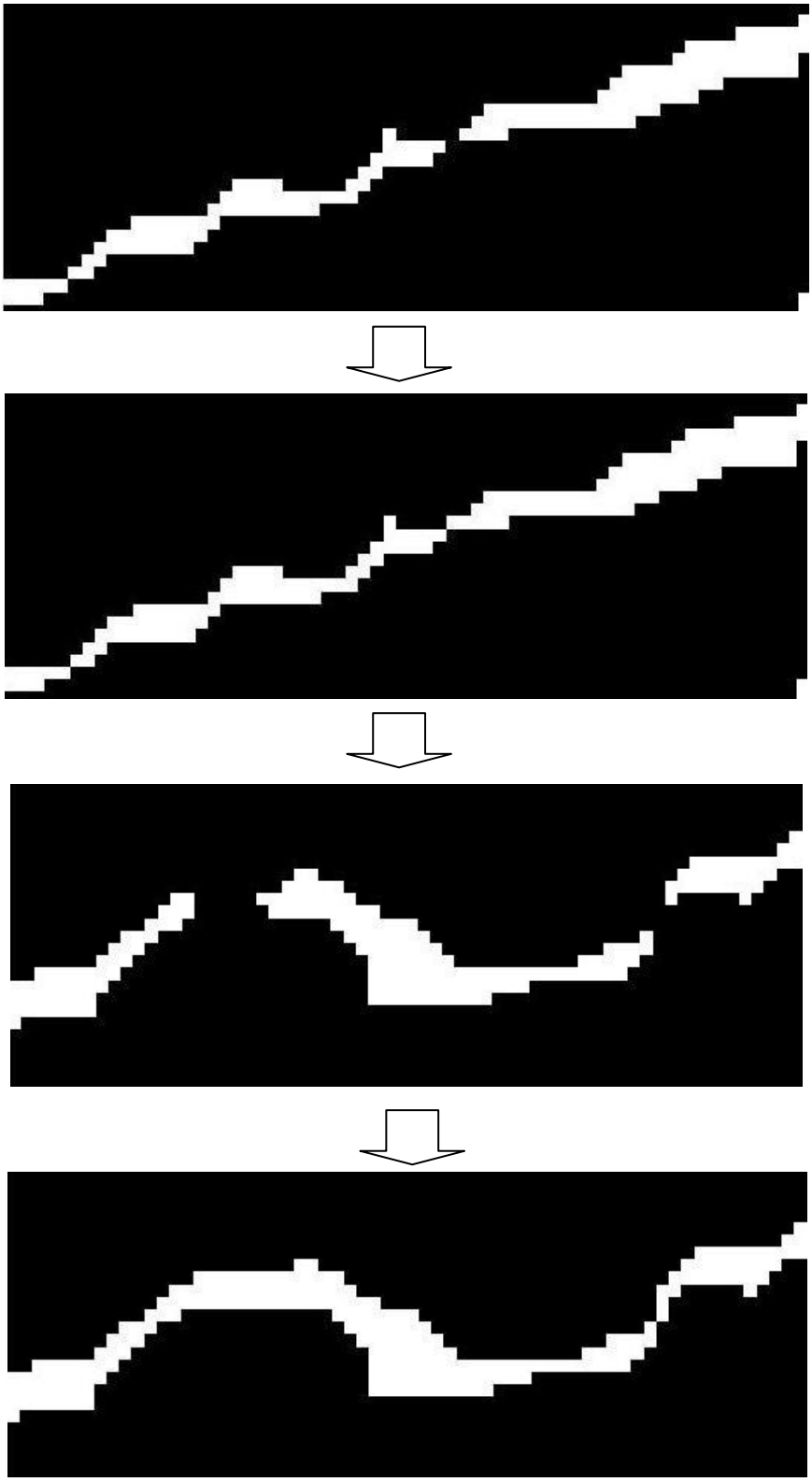


Figure 5-16 The break points connection on pixel level

Figure 5-17 A longitudinal crack before connecting

Figure 5-18 A longitudinal crack after connecting

Chapter 6

Simulation Results and Analysis

This chapter shows pavement images applying the algorithm introduced in this thesis by Matlab©. According to the process explained in the previous chapters, the process takes place in the following steps image enhancement, thresholding, closing operation, noise removal and break points connection. The results of Image skeletonization and connected components are shown in this chapter.

Skeletonization is a specific morphological operation to the binary image. It removes pixels on the boundaries of objects but does not allow objects to break apart. The pixels remaining make up the image skeleton. Matlab©, is used to perform this operation but noise is still retained after skeletonization. Connected components method checks the connectivity of bright pixels. The bright points whose connectivity is less than 30 pixels are removed thereby removing the noise. Finally, the skeleton of the crack is obtained free of noise.

The crack classification is the final step in the process. The coordinates of bright pixels on the lines along horizontal and vertical directions are recorded. The maximum amount of bright pixels in a line should be the number of the cracks. The vertical coordinates and horizontal coordinates in the three lines are used to classify the cracks.

If the difference of vertical coordinates between the maximum and minimum is less than 45 pixels, then the crack is classified to the transverse crack. Similarly, if the difference of horizontal coordinates between the maximum and minimum is less than 45 pixels, then the crack is classified to the longitudinal crack.

6.1 The transverse crack

A pavement image with three transverse cracks is shown in Figure 6-1. The image after enhancement is shown in Figure 6-2. In the following, the images fractal thresholding is used. The binary image contains discontinuities in cracks and noise as shown in Figure 6-3. The discontinuities are filled after applying closing operation, as shown in Figure 6-4. Figure 6-5 and Figure 6-6 are two different methods of noise reduction. It is evident that the method explained in this thesis has a better result than the median filtering. Eight gaps on two cracks are connected after applying the connectivity algorithm, as shown in Figure 6-7. Figure 6-8 shows the skeletonization result. The noise points are removed after using the connected component method except a large patch of noise because of the connection of crack pixels, as shown in Figure 6-9.

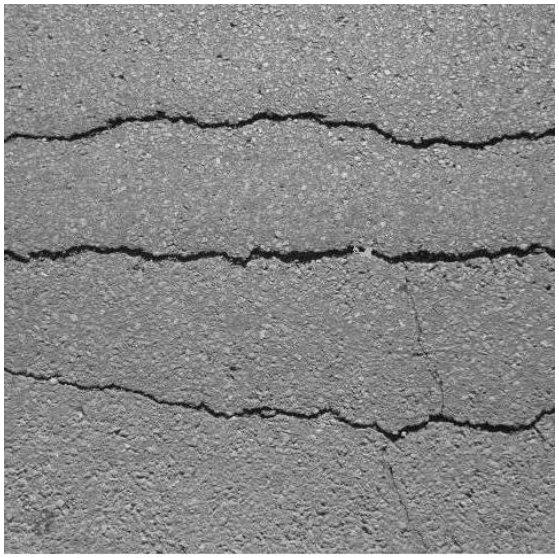


Figure 6-1 Original pavement image

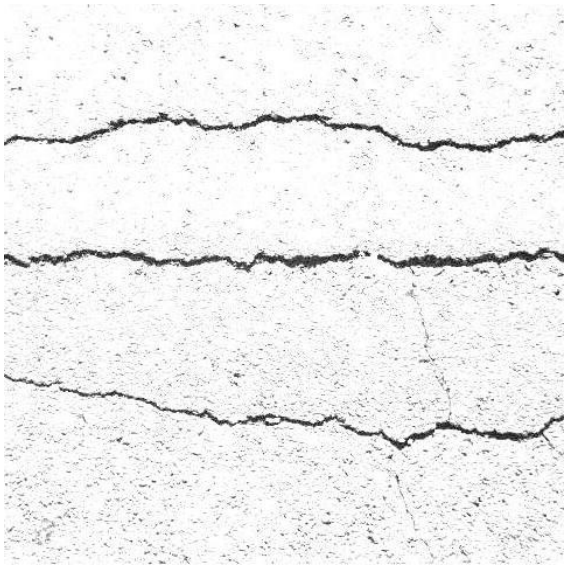


Figure 6-2 Enhancement image

In Figure 6-2, the background is changed to white, so that the cracks look more obvious. A large contrast between background and distress makes thresholding easier.

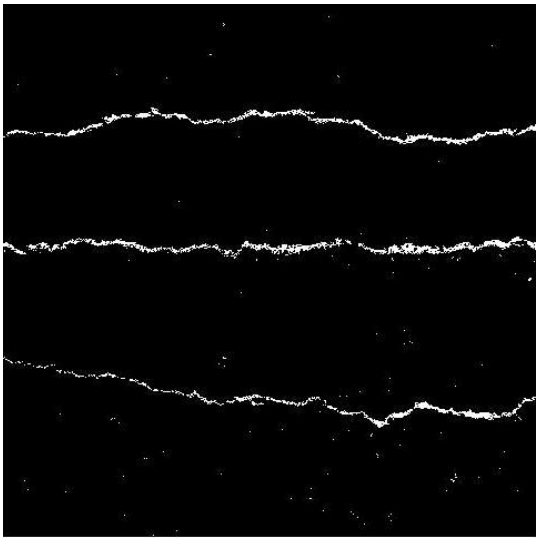


Figure 6-3 Binary image

The pavement image is transformed to binary image in Figure 6-3. The integrity of crack pixels are retained by fractal thresholding.

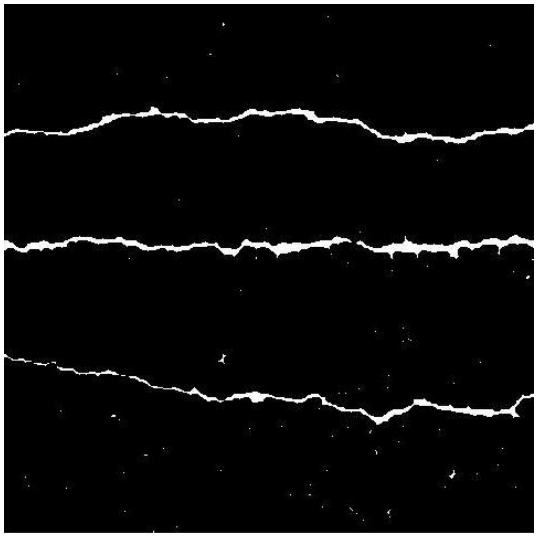


Figure 6-4 Image after closing operation

Compared to the binary image, Figure 6-4 shows that the cracks are smoothened since many gaps are filled by the closing operation.

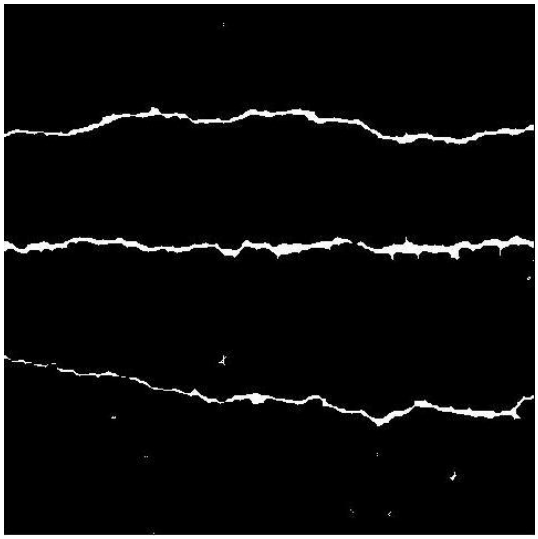


Figure 6-5 Noise reduction image

After noise reduction, many noise pixels are removed. However, the crack pixels are retained, as shown in Figure 6-5.

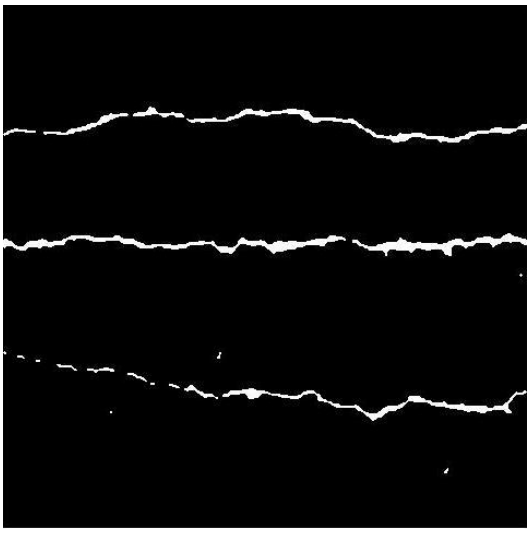


Figure 6-6 Median filter image

In Figure 6-6, the median filter shows a bad result of noise reduction because although the noise pixels are removed, many crack pixles have disappeared.

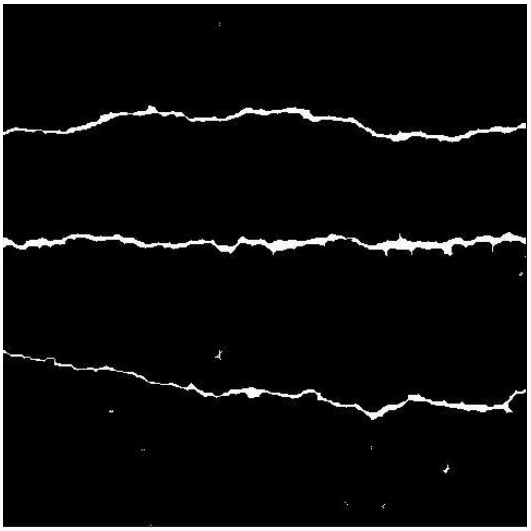


Figure 6-7 Connectivity image

As mentioned above, the gaps on cracks are connected after applying the connectivity algorithm. All the initial pixels are found according to Figure 6-7.

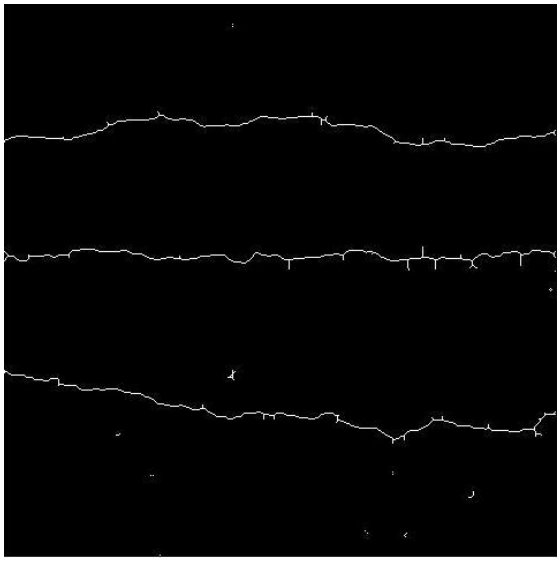


Figure 6-8 Skeleton image

After skeletonization, the noise points contain fewer noise pixels, as shown in Figure 6-8. As a result, these points are easily removed without tampering the cracks.

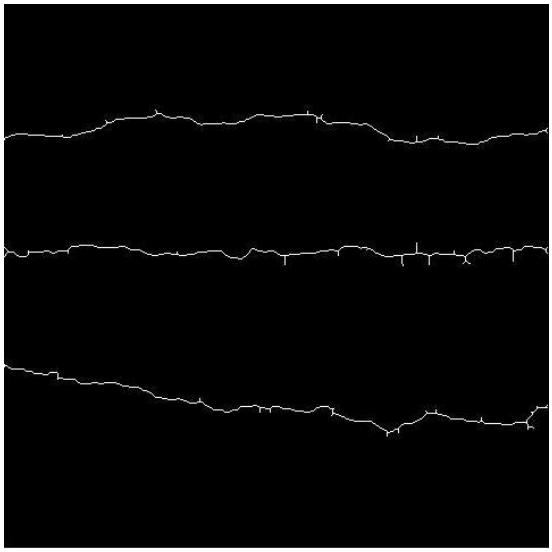


Figure 6-9 Image after connected component

A clear crack image is obtained in Figure 6-9. The final result for this pavement image is easily calculated based on this image.

6.2 The longitudinal crack

A pavement image with two longitudinal cracks is shown in Figure 6-10. The image after enhancement is shown in Figure 6-11. In the following images, fractal thresholding is used. The binary image contains discontinuities in cracks and noise, as shown in Figure 6-12. The discontinuities are filled after closing operation, as shown in Figure 6-13. Figure 6-14 and Figure 6-15 are two different methods of noise reduction. Three gaps on two cracks are connected after applying the connectivity algorithm, as shown in Figure 6-16. Figure 6-17 shows the skeletonization result. The noise points are removed after using the connected component method except one white patch

because of the connection of crack pixels, as shown in Figure 6-18.

Figure 6-10 Original pavement image

Figure 6-11 Enhancement image

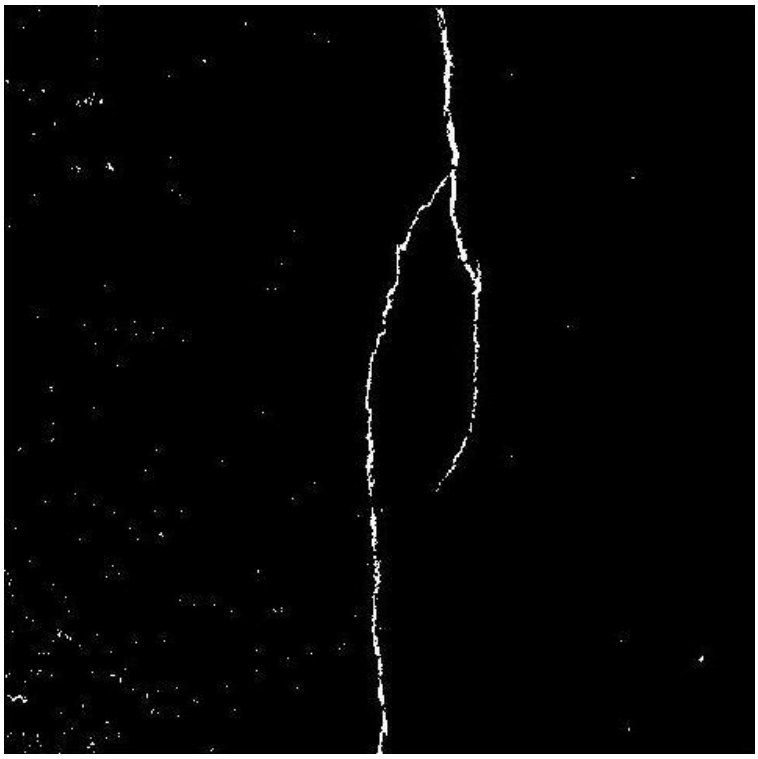


Figure 6-12 Binary image

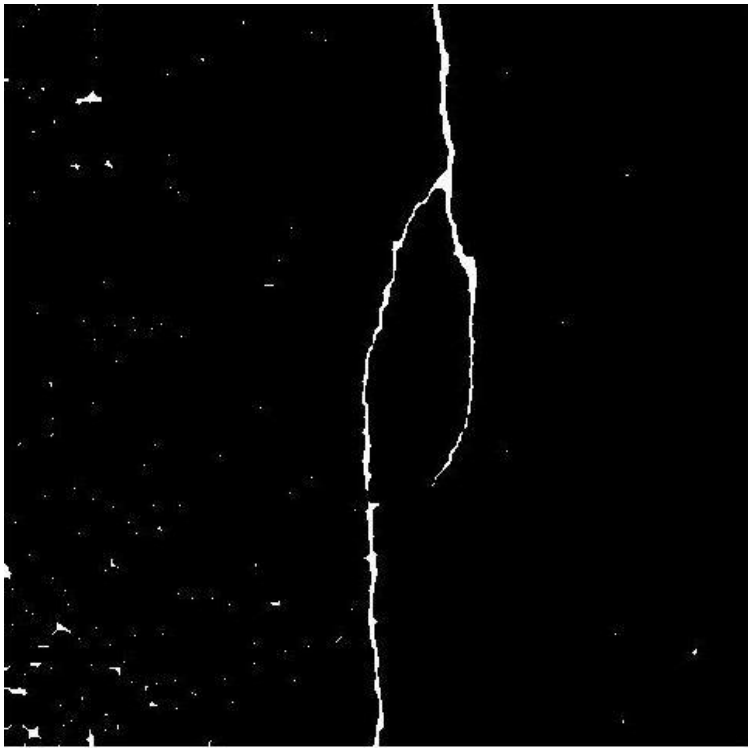


Figure 6-13 Image after closing operation

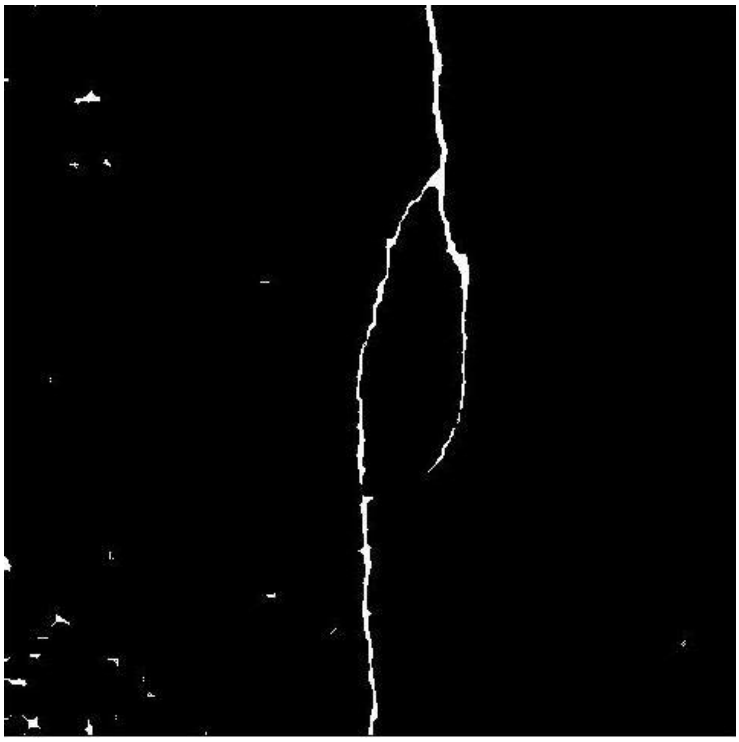


Figure 6-14 Noise reduction image

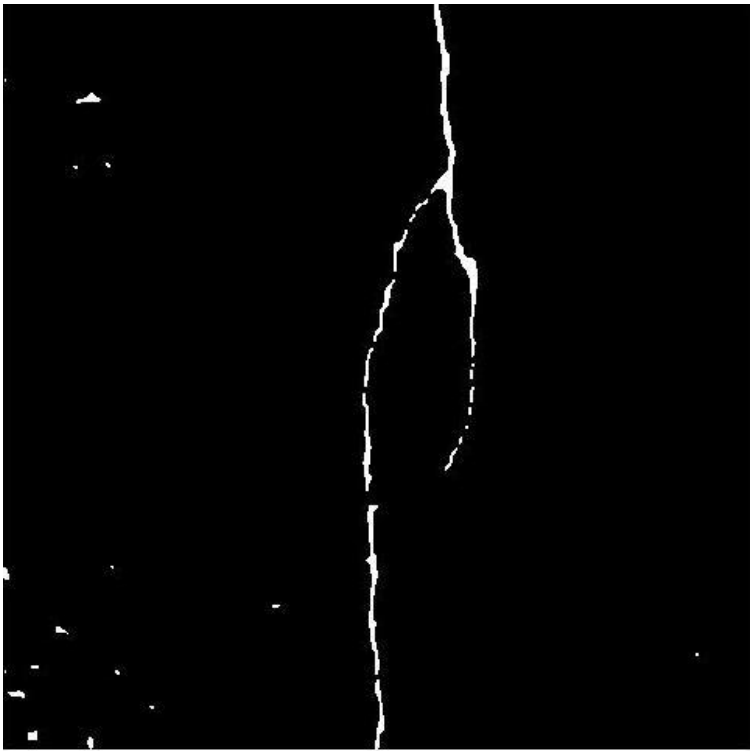


Figure 6-15 Median filter image

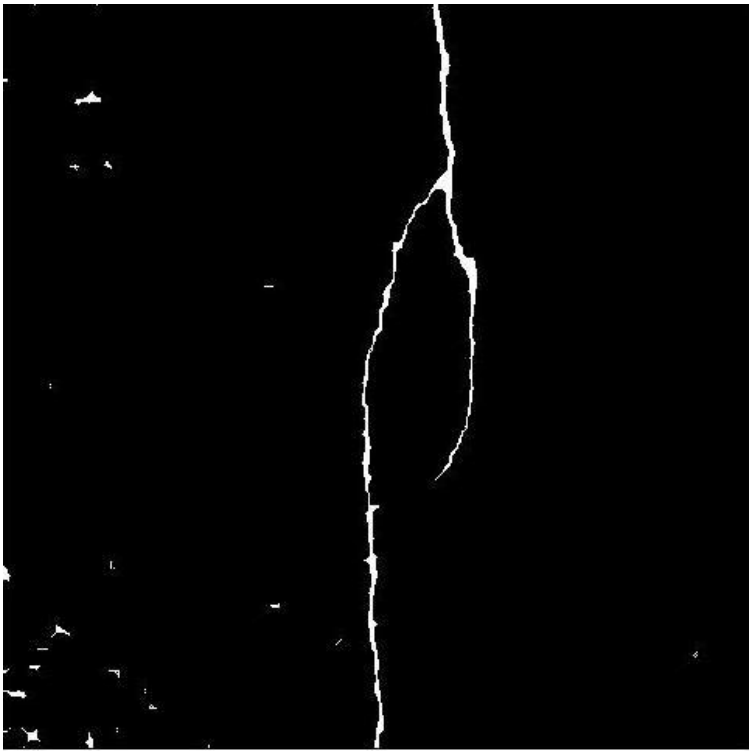


Figure 6-16 connectivity image

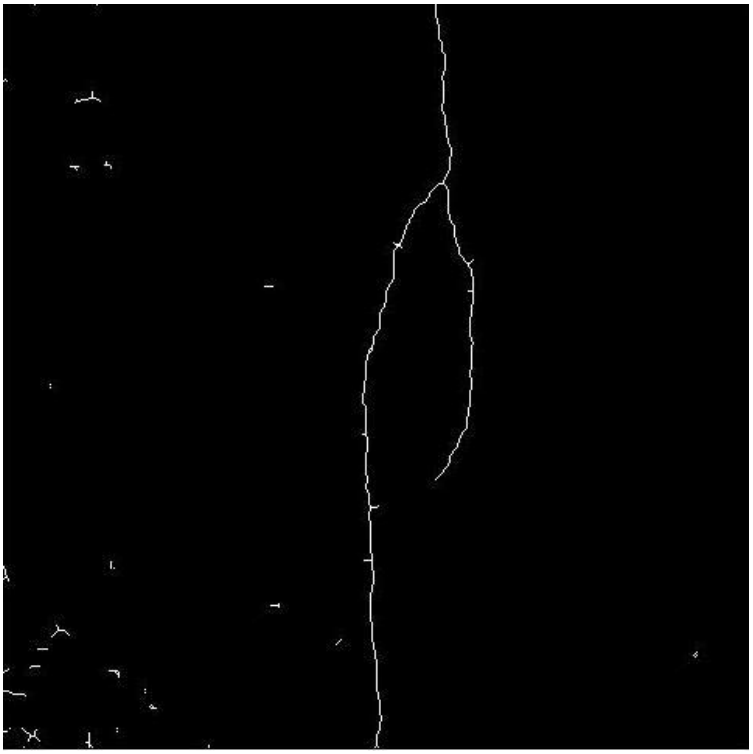


Figure 6-17 Skeleton image

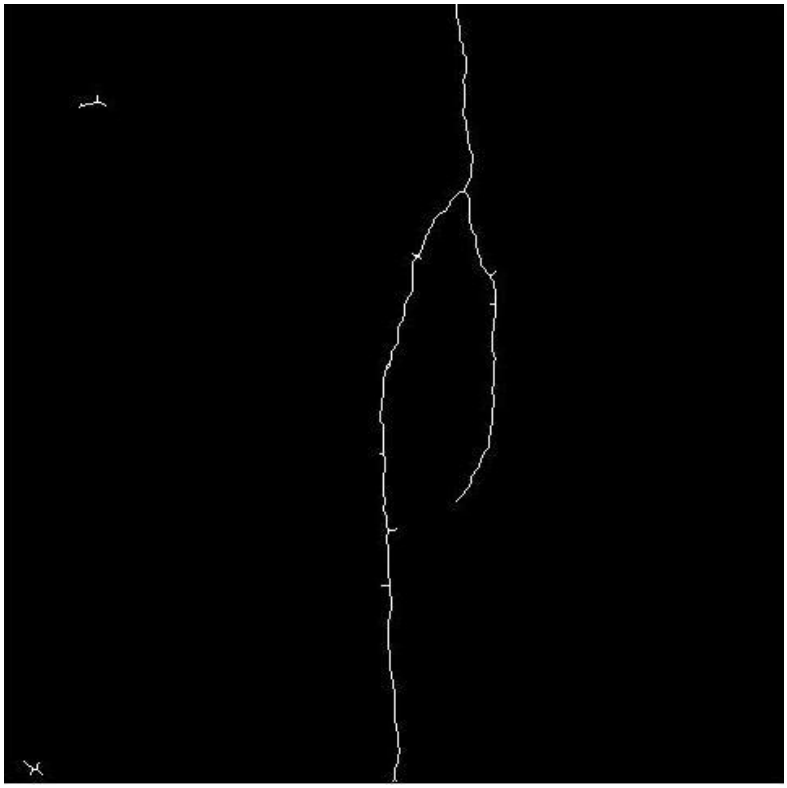


Figure 6-18 image after connected component

The final result for this pavement image is two longitudinal cracks.

Chapter 7

Conclusion and Further Work

A novel algorithm for the extraction of both transverse and longitudinal cracks from pavement images is presented in this thesis. The first step of the proposed method involves pre-processing which consists of enhancement, thresholding, morphological operations using dilation and erosion to fill in the discontinuities between cracks. Two noise reduction methods are compared in this thesis. The result shows that median filtering is ineffective for crack detection. One of the major components of the algorithm is the determination of break points and their connection to extract the crack features followed by skeletonization. The noise is easily removed by the connected component because the crack pixels have much longer connection than the noise pixels.

Experimental results clearly demonstrate that the method can effectively and efficiently extract the crack features from the pavement images. The results from the preprocessing steps give a good usable input for the break points connectivity algorithm.

Further work is needed to modify the break points connectivity algorithm so that it can be able to segregate and classify transverse and longitudinal cracks in block cracks and alligator cracks accordingly.

Reference

[1] Digital Systems Honors (1200), Research Proposal. Semi-automated Detection and Measurement of Pavement Defects. School of Computer Science and Software Engineering Monash University. Semester 1, 2004.

[2] http://www.vicroads.vic.gov.au/Home/

[3] http://www.dot.gov/about_dot.html#perfbudgplan

[4] http://www.roadware.com/_lib/pdf/datasheet.wisecrax.pdf

[5] Samsung, PicCrack User's Guide: Salt Lake City.

[6] http://www.opq.se/index.php/products

[7] L. Li, P. Chan and R. L. Lytton, "Detection of Thin Cracks on Noisy Pavement Images," *Transportation Research Record*, No. 1311, Pavement Management: Data Collection, Analysis, pp. 131-135, 1991.

[8] D. E. Newland, "Wavelet analysis of vibration," *J Vib Acoust*, vol, 116, pp.409–416, 1994.

[9] Q. Wang and X. Deng, "Damage detection with spatial wavelets," *Int J Solids Struct*, vol. 36, pp.3443–3468, 1999.

[10] S. Quek, Q. Wang, L. Zhang and K.Ang. "Sensitivity analysis of crack detection in beams by the wavelet technique," *J Mech Sci*, vol. 43, pp. 2899–2910, 2001.

[11] J. C. Hong, Y. Y. Kim, C. Lee and Y. W. Lee, “Damage detection using the Lipschitz exponent estimated by the wavelet transform,” *Int J Solids Struct*, vol. 39, pp.1803–16, 2002.

[12] E. Douka, S. Loutridis and A. Trochidis, “Crack identification in beams using wavelet analysis,” *Int J Solids Struct*, vol. 40, pp. 3557–3569, 2003.

[13] J. Leontios, H. E. Douka and T. Athanasios, “Crack detection in beams using kurtosis,” *Computers and Structures*, vol. 83, pp. 909–919, 2005.

[14] Y. Huang and B. Xu, “Automatic inspection of pavement cracking distress,” *Journal of Electronic Imaging*, vol. 15, pp. 13-17, 2006.

[15] H. D. Cheng, J. R. Chen, C. Glazier and Y. G. Hu, “Novel Approach to Pavment Cracking Detection Based on Fuzzy Set Theory,” *Journal of Computing in Civil Engineering*, vol. 13, no. 4, 1999.

[16] Q. Li and X. Liu, “A Model for Segmentation and Distress Statistic of Massive Pavement Images Based on Multi-scale Strategies,” presented at The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, part. B5, Beijing, 2008.

[17] B. B. Mandelbrot, “The Fractal Geometry of Nature,” W. H. Freeman, San Francisco, ISBN 0-7167-1186-9, 1982.

[18] Y. Zuo, G. Wang and C. Zuo, “A Novel Image Segmentation Method of Pavement Surface Cracks Based on Fractal,” presented at Theory 2008 International

Conference on Computational Intelligence and Security 978-0-7695-3508-1/08, 2008.

[19] Q. Li and X. Liu, "Novel Approach to Pavement Image Segmentation Based on Neighboring Difference Histogram Method," Congress on Image and Signal Processing, ISBN: 978-0-7695-3119-9, May, 2008.

[20] M. Yan, S. Bo, K. Xu and Y. He, "Pavement Crack Detection and Analysis for High-grade Highway," The Eighth International Conference on Electronic Measurement and Instruments, ISBN: 978-1-4244-1136-8, Xi'an, 2007.

[21] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," third edition, Prentice Hall, 2008.

Appendix- Matlab Code

```
clear all;
close all;
k=input('Enter the file name','s');
J=imread(k);
I=rgb2gray(J);
figure,imshow(I,[]);
title('Original Image');
[m,n] = size(I); % Image improvement
[f1,f2]=freqspace(size(I),'meshgrid');
D=100/size(I,1);
Hd=ones(9);      %9*9 low pass filter
r=f1.^2+f2.^2;
for i=1:9
      for j=1:9
          t=r(i,j)/(D*D);
          Hd(i,j)=exp(-t);
      end
end
Y=fft2(double(I));
Y=fftshift(Y);
Ya=convn(Y,Hd);
Ya=ifftshift(Ya);
Ia=ifft2(Ya);
I_rz=imresize(Ia,[m,n]); % blurr image
I_rz=uint8(I_rz);
If=imsubtract(I,I_rz);
b=mean2(I);
I_aa=b*ones(size(I));
I_aa=uint8(I_aa);
Ip=imadd(If,I_aa);
figure,imshow(Ip,[]);
title('enhancement image');
L = size(Ip);  % The fractal thresholding
Lseg=3;
max_row = floor(L(1)/Lseg);
```

```
max_col = floor(L(2)/Lseg);
seg1 = cell(max_row,max_col); %segment the image
r1 = 1;
for row = 1:max_row
    c1 = 1;
    for col = 1:max_col
        r2 = r1+Lseg-1;
        c2 = c1+Lseg-1;
        seg1(row,col) = {Ip(r1:r2,c1:c2,:)};
        c1 = c2 +1;
    end
    r1 = r2 +1;
end
kk=zeros(row,col);
for m=1:row
    for n=1:col
U=[0,0,0,0,0,0,0,0,0];
B=[0,0,0,0,0,0,0,0,0];
D1=zeros(size(Lseg,Lseg));
D2=zeros(size(Lseg,Lseg));
D3=zeros(size(Lseg,Lseg));
D4=zeros(size(Lseg,Lseg));
D5=zeros(size(Lseg,Lseg));
D6=zeros(size(Lseg,Lseg));
D7=zeros(size(Lseg,Lseg));
D8=zeros(size(Lseg,Lseg));
D9=zeros(size(Lseg,Lseg));
for i=1:Lseg
    for j=1:Lseg
        U(1)=seg1{m,n}(i,j); %calculate the upper surface and lower surface
        B(1)=seg1{m,n}(i,j);
        for k=1:8
            if (i-1==0)
                ax1=0;
            else
                ax1=seg1{m,n}(i-1,j);
            end
            if (i+1>3)
                ax2=0;
            else
                ax2=seg1{m,n}(i+1,j);
```

```
                end
                if (j-1==0)
                    ax3=0;
                else
                    ax3=seg1{m,n}(i,j-1);
                end
                if (j+1>3)
                    ax4=0;
                else
                    ax4=seg1{m,n}(i,j+1);
                end
                ax=max(ax1,ax2);
                bx=max(ax,ax4);
                cx=max(bx,ax3);
                dx=max(cx,U(k)+1);
                U(k+1)=dx;
                an=min(ax1,ax2);
                bn=min(an,ax4);
                cn=min(bn,ax3);
                dn=min(cn,B(k)+1);
                B(k+1)=dn;
            end
            D2(i,j)=U(2)-B(2);
            D3(i,j)=U(3)-B(3);
            D4(i,j)=U(4)-B(4);
            D5(i,j)=U(5)-B(5);
            D6(i,j)=U(6)-B(6);
            D7(i,j)=U(7)-B(7);
            D8(i,j)=U(8)-B(8);
            D9(i,j)=U(9)-B(9);
v2=0;
v3=0;
v4=0;
v5=0;
v6=0;
v7=0;
v8=0;
v9=0;
for i2=1:size(D2,1)    %calculate the D
    for j2=1:size(D2,2)
        v2=D2(i2,j2)+v2;
```

```
    end
end
for i3=1:size(D3,1)
    for j3=1:size(D3,2)
        v3=D3(i3,j3)+v3;
    end
end
for i4=1:size(D4,1)
    for j4=1:size(D4,2)
        v4=D4(i4,j4)+v4;
    end
end
for i5=1:size(D5,1)
    for j5=1:size(D5,2)
        v5=D5(i5,j5)+v5;
    end
end
for i6=1:size(D6,1)
    for j6=1:size(D6,2)
        v6=D6(i6,j6)+v6;
    end
end
for i7=1:size(D7,1)
    for j7=1:size(D7,2)
        v7=D7(i7,j7)+v7;
    end
end
for i8=1:size(D8,1)
    for j8=1:size(D8,2)
        v8=D8(i8,j8)+v8;
    end
end
for i9=1:size(D9,1)
    for j9=1:size(D9,2)
        v9=D9(i9,j9)+v9;
    end
end
a2=v2/4;
a3=v3/6;
a4=v4/8;
a5=v5/10;
```

```
a6=v6/12;
a7=v7/14;
a8=v8/16;
a9=v9/18;
aa=[log10(a2),log10(a3),log10(a4),log10(a5),log10(a6),log10(a7),log10(a8),log10(a9)];
ay=(log10(a2)+log10(a3)+log10(a4)+log10(a5)+log10(a6)+log10(a7)+log10(a8)+log10(
a9))/8;
at=(log10(2)+log10(3)+log10(4)+log10(5)+log10(6)+log10(7)+log10(8)+log10(9))/8;
x11=0;
x12=0;
for i0=1:8    % least-square fitting
      x11=(log10(i0+1)-at)*(aa(i0)-ay)+x11;
end

for j0=2:9
      x12=(log10(j0)-at)^2+x12;
end
x1=x11/x12;
x0=ay-x1*at;
kk(m,n)=10^x0;
      end
end
      end
end
kmin=min(min(kk));
ss=0.2247*kmin+40;
T=fix(ss);
Ib=zeros(size(Ip));
for i=1:size(Ip,1)
      for j=1:size(Ip,2)
            if Ip(i,j)>=T
                  Ib(i,j)=0;
            else
                  Ib(i,j)=255;
            end
      end
end
figure,imshow(Ib,[]);
title('Binary Image');
se=strel('disk',5);%closing operation
B=imdilate(Ib,se);
```

```
figure,imshow(B,[]);
title('Image after being dialated.');
se=strel('disk',5);
B_q = imerode(B,se);
figure,imshow(B_q,[]);
title('Image after being eroded');
B_c=B_q;
B_a=B_q;
[row,col]=find(B_a==255);%Noise removal
for i=1:size(row)
      for j=1:size(col)
            if (row(i)<size(B_a,1)-2 && col(j)<size(B_a,2)-2 && row(i)>2 && col(j)>2)
            if (B_a(row(i)+2,col(j))==0 && B_a(row(i),col(j)+2)==0 &&
B_a(row(i)+2,col(j)+2)==0 && B_a(row(i)-2,col(j))==0 && B_a(row(i),col(j)-2)==0
&& B_a(row(i)-2,col(j)-2)==0 && B_a(row(i)+2,col(j)-2)==0 &&
B_a(row(i)-2,col(j)+2)==0) %check the neighborhood
                  B_a(row(i),col(j))=0;
            end
            end
      end
end
B_e=B_a;
 [row_e,col_e]=find(B_e==255);
for i=1:size(B_e,2)
      B_e(i,size(B_e,2))=0;
      B_e(size(B_e,2),i)=0;
end
figure,imshow(B_e,[]);
title('image after remove noise');
X=medfilt2(B_q);
figure,imshow(X,[]);
title('Image after median filter');
ih1=zeros(1,1);%finding the transverse initial pixels
ih2=zeros(1,1);
jih=1;
for k=0:40:size(B_e,1) %segment the image
    tt3=0;
      tt4=0;
      for ii=1:size(row_e,1) %check the neighborhood
      tt1=0;
      tt2=0;
```

```
            if (row_e(ii)>k && row_e(ii)<k+20 && col_e(ii)<size(B_e,2)/2)
               if    (row_e(ii)+3<size(B_e,1) && col_e(ii)+11<size(B_e,2) &&
row_e(ii)-2>0)
                  while (tt1==0 && tt3==0)
                  if (B_e(row_e(ii)-2,col_e(ii)+10)==255 ||
B_e(row_e(ii)-1,col_e(ii)+10)==255 || B_e(row_e(ii),col_e(ii)+10)==255 ||
B_e(row_e(ii)+1,col_e(ii)+10)==255 || B_e(row_e(ii)+2,col_e(ii)+10)==255)
                        ih1(jih)=row_e(ii);
                        ih2(jih)=col_e(ii);
                        jih=jih+1;
                        tt1=1;
                        tt3=1;
                  else
                        tt1=1;
                  end
                  end
               end
            end
            if (row_e(ii)>k && row_e(ii)<k+20 && col_e(ii)>size(B_e,2)/2)
            if    (row_e(ii)+3<size(B_e,1) && col_e(ii)+11<size(B_e,2) &&
row_e(ii)-2>0 )
                  while (tt2==0 && tt4==0)
                  if (B_e(row_e(ii)-2,col_e(ii)+10)==255 ||
B_e(row_e(ii)-1,col_e(ii)+10)==255 || B_e(row_e(ii),col_e(ii)+10)==255 ||
B_e(row_e(ii)+1,col_e(ii)+10)==255 || B_e(row_e(ii)+2,col_e(ii)+10)==255)
                        ih1(jih)=row_e(ii);
                        ih2(jih)=col_e(ii);
                        jih=jih+1;
                        tt2=1;
                        tt4=1;
                  else
                        tt2=1;
                  end
                  end
            end
            end
      end
end
for l=1:size(ih1,2)    %search transverse break points
          cont_j_1=1;
      while (cont_j_1==1 && ih1(l)<425 && ih2(l)<425)
```

```
        cont_i=1;
while (cont_i==1)
        breakdown3=1;
while (breakdown3==1)
        breakdown1=1;
while (breakdown1==1)
        if (ih2(l)+1<480)
if (B_e(ih1(l),ih2(l)+1)~=0) %the first direction
        while (B_e(ih1(l),ih2(l))~=0 && ih2(l)<480)
                ih2(l)=ih2(l)+1;
        end
        ih3=ih1(l)-1; %set the searching pixel
        ih4=ih2(l)-1; %set the searching pixel
else
        ih3=ih1(l)-1; %set the searching pixel
        ih4=ih2(l);    %set the searching pixel
end
        end
if (B_e(ih3,ih4)~=0) %the second direction
        breakdown1=1;
        while (B_e(ih3,ih4)~=0 && ih3>2)
                ih3=ih3-1;
        end
        ih1(l)=ih3+1; %set the searching pixel
        ih2(l)=ih4;    %set the searching pixel
else
        breakdown1=0;
end
end
ih5=ih3+1;    %set the searching pixel
ih6=ih4;       %set the searching pixel
if (B_e(ih5+1,ih6)~=0)    %the third direction
        while (B_e(ih5,ih6)~=0 && ih5<479)
                ih5=ih5+1;
        end
        if (B_e(ih5-1,ih6+1)~=0)
        ih1(l)=ih5-1;    %set the searching pixel
        ih2(l)=ih6+1;    %set the searching pixel
        breakdown3=1;
        else                %check next column pixels
                for i=0:ih5-ih3-2
```

```
                    if (B_e(ih5-1-i,ih6+1)~=0)
                        ih1(l)=ih5-1-i;
                        ih2(l)=ih6+1;
                       breakdown3=1;
                       break
                    else
                        breakdown3=0;
                        case_i=1;
                    end
                end
            end
        elseif (B_e(ih5+1,ih6)==0)
            breakdown3=0;
            case_i=2;
        else
            breakdown3=0;
            case_i=2;
        end
        end
        if (case_i==2) %check diagonal pixels
            if (ih6~=size(B_e,1))
            if (B_e(ih5-1,ih6+1)~=0)
                cont_i=1;
                ih1(l)=ih5-1;
                ih2(l)=ih6+1;
            elseif (B_e(ih5+1,ih6+1)~=0)
                    cont_i=1;
                    ih1(l)=ih5+1;
                    ih2(l)=ih6+1;
            else
                A=1;
                cont_i=0;
            end
            else
                breakdown1=0;
                breakdown3=0;
                cont_i=0;
                cont_j_1=0;
            end
        elseif (case_i==1) %check diagonal pixels
            if (ih4~=size(B_e,1) && ih6~=size(B_e,1))
```

```
        if (B_e(ih3,ih4+1)~=0)
            cont_i=1;
            ih1(l)=ih3;
            ih2(l)=ih4+1;
        elseif (B_e(ih5,ih6+1)~=0)
            cont_i=1;
            ih1(l)=ih5;
            ih2(l)=ih6+1;
        else
            A=2;
            cont_i=0;
        end
        else
            breakdown1=0;
            breakdown3=0;
            cont_i=0;
            cont_j_1=0;
        end
else
        cont_i=0;
end
end
if (A==1)%single pixels break point connection
        if (B_e(ih5-1,ih6+2)~=0)    %search the bright pixels
            B_e(ih5-1,ih6+1)=255; %change the dark pixle to bright pixel
            ih1(l)=ih5-1;
            ih2(l)=ih6+2;
            cont_j_1=1;
        elseif (B_e(ih5-1,ih6+3)~=0)
            B_e(ih5-1,ih6+1)=255;
            B_e(ih5-1,ih6+2)=255;
            ih1(l)=ih5-1;
            ih2(l)=ih6+3;
            cont_j_1=1;
        elseif (B_e(ih5-1,ih6+4)~=0)
            B_e(ih5-1,ih6+1)=255;
            B_e(ih5-1,ih6+2)=255;
            B_e(ih5-1,ih6+3)=255;
            ih1(l)=ih5-1;
            ih2(l)=ih6+4;
            cont_j_1=1;
```

```
        elseif (B_e(ih5-1,ih6+5)~=0)
                B_e(ih5-1,ih6+1)=255;
                B_e(ih5-1,ih6+2)=255;
                B_e(ih5-1,ih6+3)=255;
                B_e(ih5-1,ih6+4)=255;
                ih1(l)=ih5-1;
                ih2(l)=ih6+5;
                cont_j_1=1;
        elseif (B_e(ih5-1,ih6+6)~=0)
                B_e(ih5-1,ih6+1)=255;
                B_e(ih5-1,ih6+2)=255;
                B_e(ih5-1,ih6+3)=255;
                B_e(ih5-1,ih6+4)=255;
                B_e(ih5-1,ih6+5)=255;
                ih1(l)=ih5-1;
                ih2(l)=ih6+6;
                cont_j_1=1;
    else
                cont_j_1=0;
     end
elseif (A==2) %multiple pixels break point connection
if (ih6+1<size(B_e,2))
    x=ih5-ih3-1;
    for i=1:x
        if (B_e(ih5-i,ih6+2)~=0)    %search the bright pixels
                B_e(ih5-i,ih6+1)=255; %change the dark pixle to bright pixel
                ih1(l)=ih5-i;
                ih2(l)=ih6+2;
                cont_j_1=1;
        elseif (B_e(ih5-i,ih6+3)~=0)
                B_e(ih5-i,ih6+1)=255;
                B_e(ih5-i,ih6+2)=255;
                ih1(l)=ih5-i;
                ih2(l)=ih6+3;
                cont_j_1=1;
        elseif (B_e(ih5-i,ih6+4)~=0)
                B_e(ih5-i,ih6+1)=255;
                B_e(ih5-i,ih6+2)=255;
                B_e(ih5-i,ih6+3)=255;
                ih1(l)=ih5-i;
                ih2(l)=ih6+4;
```

```
                cont_j_1=1;
            elseif (B_e(ih5-i,ih6+5)~=0)
                B_e(ih5-i,ih6+1)=255;
                B_e(ih5-i,ih6+2)=255;
                B_e(ih5-i,ih6+3)=255;
                B_e(ih5-i,ih6+4)=255;
                ih1(l)=ih5-i;
                ih2(l)=ih6+5;
                cont_j_1=1;
            elseif (B_e(ih5-i,ih6+6)~=0)
                B_e(ih5-i,ih6+1)=255;
                B_e(ih5-i,ih6+2)=255;
                B_e(ih5-i,ih6+3)=255;
                B_e(ih5-i,ih6+4)=255;
                B_e(ih5-i,ih6+5)=255;
                ih1(l)=ih5-i;
                ih2(l)=ih6+6;
                cont_j_1=1;
            end
        end
      else
            cont_j_1=0;
        end
    else
        cont_j_1=0;
    end
   end
    end
  end
figure,imshow(B_e,[]);
title('connecting image');
[row_w,col_w]=find(BW==1);%rate the cracks
D1=zeros(1,1); %scane line
j1=1;
for i=1:size(col_w,1)
    if col_w(i)==20;
        D1(j1)=row_w(i);
        j1=j1+1;
    end
end
D2=zeros(1,1); %scane line
```

```
j2=1;
for i=1:size(col_w,1)
      if col_w(i)==220;
            D2(j2)=row_w(i);
            j2=j2+1;
      end
end
D3=zeros(1,1); %scane line
j3=1;
for i=1:size(col_w,1)
      if col_w(i)==420;
            D3(j3)=row_w(i);
            j3=j3+1;
      end
end
for k1=1:size(D1,2)
      D1(k1);
end
for k2=1:size(D2,2)
      D2(k2);
end
for k3=1:size(D3,2)
      D3(k3);
end
ak=min(k1,k2);
aj=min(ak,k3);
for j=1:aj %compare the coordinates
      at=max(D1(j),D2(j));
      ab=max(at,D3(j));
      au=min(D1(j),D2(j));
      as=min(au,D3(j));
      if (ab-as<45) && (ab-as~=0) && (k1<5) && (k2<5) && (k3<5)
          disp('in a transverse line');
      end
end
D4=zeros(1,1); %scane line
j4=1;
for i=1:size(row_w,1)

      if row_w(i)==20;
            D4(j4)=col_w(i);
```

```
            j4=j4+1;

        end
end
D5=zeros(1,1); %scane line
j5=1;
for i=1:size(row_w,1)
        if row_w(i)==220;
            D5(j5)=col_w(i);
            j5=j5+1;
        end
end
D6=zeros(1,1); %scane line
j6=1;
for i=1:size(row_w,1)
        if row_w(i)==420;
            D6(j6)=col_w(i);
            j6=j6+1;
        end
end
for k4=1:size(D4,2)
        D4(k4);
end
for k5=1:size(D5,2)
        D5(k5);
end
for k6=1:size(D6,2)
        D6(k6);
end
akk=min(k4,k5);
ajj=min(akk,k6);
for j=1:ajj %compare the coordinates
        att=max(D4(j),D5(j));
        abb=max(att,D6(j));
        auu=min(D4(j),D5(j));
        ass=min(auu,D6(j));
        if (abb-ass<45) && (abb-ass~=0) && (k4<5) && (k5<5) && (k6<5)
            disp('in a longitudinal line');
        end
end
```